F. Borrelli, A. Bemporad, M. Morari

Predictive Control

for linear and hybrid systems

February 20, 2011

Cambridge

to Maryan, Federica and Marina
and all our families

Preface

Dynamic optimization has become a standard tool for decision making in a wide range of areas. The search for the most fuel-efficient strategy to send a rocket into orbit or the most economical way to start-up a chemical production facility can be expressed as dynamic optimization problems that are solved almost routinely nowadays.

The basis for these dynamic optimization problems is a dynamic model, for example,

$$x(k+1) = g(x(k), u(k)),\ \ x(0) = x_0$$

that describes the evolution of the state $x(k)$ with time, starting from the initial condition $x(0)$, as it is affected by the manipulated input $u(k)$. Here $g(x, u)$ is some nonlinear function. Throughout the book we are assuming that this discrete-time description, i.e., the model of the underlying system is available. The goal of the dynamic optimization procedure is to find the vector of manipulated inputs $U_N = [u(0)', ..., u(N-1)']'$ such that the objective function is optimized over some time horizon N, typically

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

The terms $q(x, u)$ and and $p(x)$ are referred to as the *stage cost* and *terminal cost*, respectively. Many practical problems can be put into this form and many algorithms and software packages are available to determine the optimal solution vector U_N^*, the *optimizer*. The various algorithms exploit the structure of the particular problem, e.g., linearity and convexity, so that even large problems described by complex models and involving many degrees of freedom can be solved efficiently and reliably.

One difficulty with this idea is that, in practice, the sequence of $u(0), u(1), ...$, which is obtained by this procedure cannot be simply applied. The model of the system predicting its evolution is usually inaccurate and the system may be affected by external disturbances that may cause its path to deviate significantly from the one that is predicted. Therefore, it is common practice to measure the state after some time period, say one time step, and to solve the dynamic optimization problem again, starting from the measured state $x(1)$ as the new initial condition. This *feedback* of the measurement information to the optimization endows the whole procedure with a *robustness* typical of closed-loop systems.

What we have described above is usually referred to as *Model Predictive Control (MPC)*, but other names like Open Loop Optimal Feedback and Reactive Scheduling have been used as well. Over the last 25 years MPC has evolved to dominate the process industry, where it has been employed for thousands of problems [217].

The popularity of MPC stems from the fact that the resulting operating strategy respects all the system and problem details, including interactions and constraints, which would be very hard to accomplish in any other way.

Indeed, often MPC is used for the regulatory control of large multivariable linear systems with constraints, where the objective function is not related to an economical objective, but is simply chosen in a mathematically convenient way, namely quadratic in the states and inputs, to yield a "good" closed-loop response. Again, there is no other controller design method available today for such systems, that provides constraint satisfaction and stability guarantees.

One limitation of MPC is that running the optimization algorithm online at each time step requires substantial time and computational resources, which are generally available in a large "slow" chemical production facility, but may not be available for the control of a system installed in an automobile that must be inexpensive and where the sampling time is in the range of milliseconds. In this case it would be desirable to have the result of the optimization pre-computed and stored for each x in the form of a look-up table or as an algebraic function $u(k) = f(x(k))$ which can be easily evaluated.

In other words, we want to determine the (generally nonlinear) feedback control law $f(x)$ that generates the optimal $u(k) = f(x(k))$ explicitly and not just implicitly as the result of an optimization problem. It requires the solution of the Bellman equation and has been a long standing problem in optimal control. A clean simple solution exists only in the case of linear systems with a quadratic objective function, where the optimal controller turns out to be a linear function of the state (Linear Quadratic Regulator, LQR). For all other cases a solution of the Bellman equation was considered prohibitive except for systems of dimension 2 or 3, where a look-up table can be generated by gridding the state space and solving the optimization problem off-line for each grid point.

generally, hybrid systems. The major new contribution of this book is to show how the nonlinear optimal feedback controller can be calculated efficiently for some important classes of systems, namely linear systems with constraints and switched linear systems or, more generally, hybrid systems. Traditionally, the design of feedback controllers for linear systems with constraints, for example anti-windup techniques, was *ad hoc* requiring both much experience and trial and error. Though significant progress has been achieved on anti-windup schemes over the last decade, these techniques deal with input constraints only and cannot be extended easily.

The classes of constrained linear systems and linear hybrid systems treated in this book cover many, if not most, practical problems. The new design techniques hold the promise to lead to better performance and a dramatic reduction in the required engineering effort.

The book is structured into four parts. Chapters marked with a star ($*$) are consider "advanced" material and can be skipped without compromising the understanding of the fundamental concepts presented in the book.

- In the first part of the book (Part I) we recall the main concepts and results of convex and discrete optimization.Our intent is to provide only the necessary background for the understanding of the rest of the book. The material of this part has been extracted from the following books and lecture notes: "Convex optimization" by Boyd and Vandenberghe [57], "Nonlinear Programming Theory and Algorithms" by Bazaraa, Sherali, Shetty [23], "LMIs in Control" by Scherer and Weiland [231] and "Lectures on Polytopes" by Ziegler [105].
  Continuous problems as well Integer and Mixed-Integer problems are presented in Chapter 1. Chapter 2 discusses the classical results of Lagrange duality. Since polyhedra are the fundamental geometric objects used in this book, in Chapter 3 we introduce the main definitions and the algorithms which describe standard operations on polyhedra. Part I closes with Chapter 4 where Linear and Quadratic programs are presented together with their properties and some fundamental results.
- The second part of the book (Part II) is a self-contained introduction to multi-parametric programming. In our framework, parametric programming is the main technique used to study and compute state feedback optimal control laws. In fact, we formulate the finite time optimal control problems as mathematical programs where the input sequence is the optimization vector. Depending on the dynamical model of the system, the nature of the constraints, and the cost function used, a different mathematical program is obtained. The current state of the dynamical system enters the cost function and the constraints as a parameter that affects the solution of the mathematical program. We study the structure of the solution as this parameter changes and we describe algorithms for solving multi-parametric linear, quadratic and mixed integer programs. They constitute the basic tools for computing the state feedback optimal control laws for these more complex systems in the same way as algorithms for solving the Riccati equation are the main tools for computing optimal controllers for linear systems. In Chapter 5 we introduce the concept of multiparametric programming and we recall the main results of nonlinear multiparametric programming. Then, in Chapter 6 we describe three algorithms for solving multiparametric linear programs (mp-LP), multipara-

metric quadratic programs (mp-QP) and multiparametric mixed-integer linear programs (mp-MILP).

- In the third part of the book (Part III) we introduce the general class of optimal control problem studied in the book. Chapter 7 contains the basic definitions and essential concepts. Chapter 8 presents standard results on Linear Quadratic Optimal Control while Chapter 9 studies unconstrained optimal control problems for linear systems with cost functions based on 1 and ∞ norms.
- In the fourth part of the book (Part IV) we focus on linear systems with polyhedral constraints on inputs and states. We study *finite time and infinite time* optimal control problems with cost functions based on 2, 1 and ∞ norms. in Chapter 10 we demonstrate that the solution to all these optimal control problems can be expressed as a *piecewise affine* state feedback law. Moreover, the optimal control law is continuous and the value function is convex and continuous. The results form a natural extension of the theory of the Linear Quadratic Regulator to constrained linear systems. Chapter 10 also presents a self-contained introduction to invariant set theory as well as dynamic programming for constrained systems.
  Chapter 11 presents the concept of Model Predictive Control. Classical feasibility and stability issues are shown through simple examples and explained by using invariant set methods. Finally we show how they can be solved with a proper choice of terminal constraints and cost function.
  Chapter 12 addresses the robustness of the optimal control laws. We discuss min-max control problems for uncertain linear systems with polyhedral constraints on inputs and states and present an approach to compute their state feedback solutions. Robustness is achieved against additive norm-bounded input disturbances and/or polyhedral parametric uncertainties in the state-space matrices.
  The result in Chapter 10 have important consequences for the implementation of MPC laws. Precomputing off-line the explicit piecewise affine feedback policy reduces the on-line computation for the receding horizon control law to a function evaluation, therefore avoiding the on-line solution of a mathematical program. In Chapter 13 the evaluation of the piecewise affine optimal feedback policy is carefully studied and we present algorithms to reduce its storage demands and computational complexity.
- In the fifth part of the book (Part V) we focus on linear hybrid systems. We give an introduction to the different formalisms used to model hybrid systems focusing on computation-oriented models (Chapter 14). In Chapter 15 we study finite time optimal control problems with cost functions based on 2, 1 and ∞ norms. The optimal control law is shown to be, in general, piecewise affine over non-convex and disconnected sets. Along with the analysis of the solution properties we present algorithms that efficiently compute the optimal control law for all the considered cases.

Berkeley *Francesco Borrelli*
Trento *Alberto Bemporad*
Zurich *Manfred Morari*
May 2010

Acknowledgements

Large part of the material presented in this book is extracted from the work of the authors Francesco Borrelli, Alberto Bemporad and Manfred Morari. Several sections have also been extracted from the PhD Theses of Miroslav Baric, Mato Baotic, Fabio Torrisi, Domenico Mignone, Pascal Grieder, Eric Kerrigan, Tobias Geyer and Frank J. Christophersen. A special tanks goes to Michal Kvasnica, Stefan Richter and Thomas Besselmann for their invaluable help on the examples construction.

Our wholehearted gratitude also goes to the several people who have careful read preliminary versions of the book and gave us suggestions on how to improve them. They include David Mayne, Per Olof Gutman and all the students of our class on Model Predictive Control (ME290J at UC Berkeley, 227-0221-00 at ETH).

Contents

Part I Basics on Optimization

1 Main Concepts 3
- 1.1 Optimization Problems 3
  - 1.1.1 Continuous Problems 4
  - 1.1.2 Integer and Mixed-Integer Problems 6
- 1.2 Convexity 7

2 Optimality Conditions 11
- 2.1 Introduction 11
- 2.2 Lagrange Duality Theory 12
  - 2.2.1 Strong Duality and Constraint Qualifications 14
  - 2.2.2 Certificate of Optimality 15
- 2.3 Complementary Slackness 16
- 2.4 Karush-Kuhn-Tucker Conditions 16
  - 2.4.1 KKT geometric interpretation 19

3 Polyhedra, Polytopes and Simplices 21
- 3.1 General Set Definitions and Operations 21
- 3.2 Polyhedra Definitions and Representations 23
- 3.3 Polytopal Complexes 26
  - 3.3.1 Functions on Polytopal Complexes 27
- 3.4 Basic Operations on Polytopes 28
  - 3.4.1 Convex Hull 28
  - 3.4.2 Envelope 29
  - 3.4.3 Vertex Enumeration 29
  - 3.4.4 Minimal Representation 29
  - 3.4.5 Chebychev Ball 31
  - 3.4.6 Projection 32
  - 3.4.7 Set-Difference 33
  - 3.4.8 Pontryagin Difference 34

3.4.9 Minkowski Sum 36
3.4.10 Polytope Union 37
3.4.11 Affine Mappings and Polyhedra 40
3.5 Operations on P-collections 41
3.5.1 Set-Difference 41
3.5.2 Polytope Covering 44
3.5.3 Union of P-collections 44

4 Linear and Quadratic Optimization 49
4.1 Linear Programming 49
4.1.1 Graphical Interpretation and Solutions Properties 49
4.1.2 Dual of LP 51
4.1.3 KKT condition for LP 51
4.1.4 Active Constraints and Degeneracies 52
4.1.5 Convex Piecewise Linear Optimization 55
4.2 Quadratic Programming 57
4.2.1 Graphical Interpretation and Solutions Properties 57
4.2.2 Dual of QP 58
4.2.3 KKT condition for QP 59
4.2.4 Active Constraints and Degeneracies 59
4.2.5 Constrained Least-Squares Problems 59
4.3 Mixed-Integer Optimization 60
4.3.1 Mixed Integer Linear Programming 60
4.3.2 Mixed Integer Quadratic Programming 61

Part II Multiparametric Programming

5 General Results for Multiparametric Nonlinear Programs. 67
5.1 General Results for Multiparametric Nonlinear Programs 71

6 Multiparametric Programming: a Geometric Approach . . . 81
6.1 Multiparametric Programs with Linear Constraints 82
6.1.1 Formulation 82
6.1.2 Definition of Critical Region 83
6.1.3 Reducing the Dimension of the Parameter Space 83
6.2 Multiparametric Linear Programming 85
6.2.1 Formulation 85
6.2.2 Critical Regions, Value Function and Optimizer: Local Properties 87
6.2.3 Nonunique Optimizer* 90
6.2.4 Propagation of the Set of Active Constraints 92
6.2.5 Value Function and Optimizer: Global Properties 96
6.2.6 mp-LP Algorithm 96
6.3 Multiparametric Quadratic Programming 101
6.3.1 Formulation 101
6.3.2 Critical Regions, Value Function and Optimizer: Local Properties102

6.3.3 Propagation of the Set of Active Constraints 106
6.3.4 Value Function and Optimizer: Global Properties 108
6.3.5 mp-QP Algorithm 112
6.4 Multiparametric Mixed-Integer Linear Programming 116
6.4.1 Formulation and Properties 116
6.4.2 Geometric Algorithm for mp-MILP 116
6.4.3 Theoretical Results 119
6.5 Multiparametric Mixed-Integer Quadratic Programming 121
6.5.1 Formulation and Properties 121
6.6 Literature Review 123

Part III Optimal Control

7 General Formulation and Discussion 127
7.1 Problem Formulation 127
7.2 Solution via Batch Approach 129
7.3 Solution via Recursive Approach 130
7.4 Optimal Control Problem with Infinite Horizon 133
7.4.1 Value Function Iteration 134
7.4.2 Receding Horizon Control 135
7.5 Lyapunov Stability 137
7.5.1 General Stability Conditions 137
7.5.2 Quadratic Lyapunov Functions for Linear Systems 140
7.5.3 $1/\infty$ Norm Lyapunov Functions for Linear Systems ... 141

8 Linear Quadratic Optimal Control 143
8.1 Solution via Batch Approach 144
8.2 Solution via Recursive Approach 145
8.3 Comparison Of The Two Approaches 147
8.4 Infinite Horizon Problem 148
8.5 Stability of the Infinite Horizon LQR 149

9 $1/\infty$ Norm Optimal Control 151
9.1 Solution via Batch Approach 152
9.2 Solution via Recursive Approach 155
9.3 Comparison Of The Two Approaches 157
9.4 Infinite Horizon Problem 158

Part IV Constrained Optimal Control of Linear Systems

10 Constrained Optimal Control 165
10.1 Invariant Sets 166
10.2 Constrained Optimal Control Problem Formulation 177
10.3 Feasible Solutions 179
10.4 State Feedback Solution, 2-Norm Case 183
10.4.1 Solution via Batch Approach 184

10.4.2 Solution via Recursive Approach . . . 187
10.4.3 Infinite Horizon Problem . . . 187
10.4.4 CLQR Algorithm . . . 189
10.4.5 Examples . . . 191
10.5 State Feedback Solution, 1-Norm and ∞-Norm Case . . . 194
10.5.1 Batch Approach . . . 194
10.5.2 Recursive Approach . . . 197
10.5.3 Example . . . 200
10.5.4 Infinite-Time Solution . . . 201
10.6 State-Feedback Solution, Minimum-Time Control . . . 202

11 Receding Horizon Control . . . 209
11.1 Introduction . . . 210
11.2 RHC Implementation . . . 211
11.3 RHC Main Issues . . . 218
11.3.1 Feasibility of RHC . . . 218
11.3.2 Stability of RHC . . . 221
11.4 State Feedback Solution of RHC, 2-Norm Case . . . 224
11.5 State Feedback Solution of RHC, $1, \infty$-Norm Case . . . 231
11.6 RHC Extensions . . . 235
11.7 Offset-Free Reference Tracking . . . 236
11.8 Literature Review . . . 244

12 Constrained Robust Optimal Control . . . 251
12.1 Introduction . . . 252
12.2 Robust Invariant Sets . . . 252
12.3 Problem Formulation . . . 265
12.4 Feasible Solutions . . . 269
12.5 State Feedback Solution, 1-Norm and ∞-Norm Case . . . 275
12.5.1 Batch Approach: Open-Loop Predictions . . . 277
12.5.2 Recursive Approach: Closed-Loop Predictions . . . 277
12.5.3 Solution to CROC-CL and CROC-OL via mp-MILP* . 279
12.6 Parametrizations of the Control Policies . . . 281
12.7 Example . . . 284
12.8 Robust Receding Horizon Control . . . 285
12.9 Literature Review . . . 287

13 On-line Control Computation . . . 289
13.1 Introduction . . . 290
13.2 Efficient On-Line Algorithms . . . 291
13.2.1 Efficient Implementation, $1, \infty$-Norm Case . . . 292
13.2.2 Efficient Implementation, 2-Norm Case . . . 294
13.3 Example . . . 300
13.3.1 CFTOC based on LP . . . 301
13.3.2 CFTOC based on QP . . . 302

13.4 Literature Review . . . 302

Part V Constrained Optimal Control of Hybrid Systems

14 Models of Hybrid Systems . . . 305
14.1 Hybrid models . . . 305
14.2 Piecewise Affine Systems . . . 306
14.2.1 Modeling Discontinuities . . . 307
14.2.2 Binary States, Inputs, and Outputs . . . 309
14.3 Discrete Hybrid Automata . . . 313
14.3.1 Switched Affine System (SAS) . . . 314
14.3.2 Event Generator (EG) . . . 314
14.3.3 Boolean Algebra . . . 316
14.3.4 Finite State Machine (*FSM*) . . . 316
14.3.5 Mode Selector . . . 318
14.3.6 DHA Trajectories . . . 318
14.4 Logic and Mixed-Integer Inequalities . . . 319
14.4.1 Transformation of Boolean Relations . . . 319
14.4.2 Translating DHA Components into Linear Mixed-Integer Relations320
14.5 Mixed Logical Dynamical Systems . . . 322
14.6 Model Equivalence . . . 323
14.7 The HYSDEL Modeling Language . . . 324
14.8 Literature Review . . . 329

15 Optimal Control of Hybrid Systems . . . 335
15.1 Problem Formulation . . . 336
15.2 Properties of the State Feedback Solution, 2-Norm Case . . . 338
15.3 Properties of the State Feedback Solution, $1, \infty$-Norm Case . . 344
15.4 Computation of the Optimal Control Input via Mixed Integer Programming346
15.5 State Feedback Solution via Batch Approach . . . 349
15.6 State Feedback Solution via Recursive Approach . . . 350
15.6.1 Preliminaries and Basic Steps . . . 350
15.6.2 Multiparametric Programming with Multiple Quadratic Functions353
15.6.3 Algorithmic Solution of the Bellman Equations . . . 356
15.6.4 Examples . . . 359
15.7 Discontinuous PWA systems . . . 361
15.8 Receding Horizon Control . . . 361
15.8.1 Stability and Feasibility Issues . . . 362
15.8.2 Examples . . . 363

A Polycover Algorithms . . . 367
A.1 Polycover: MILP formulation . . . 367
A.2 Polycover: Branch & Bound algorithm . . . 369

B Merging of P-collections 373
B.0.1 Preliminaries 373
B.1 Disjoint Optimal Complexity Reduction 375
B.1.1 Algorithm based on Branch and Bound 375
B.1.2 Branching Heuristics 377
B.2 Non-Disjoint Optimal Complexity Reduction 378
B.2.1 Boolean Calculus 378
B.2.2 Logic Minimization 378
B.2.3 Problem Formulation with Boolean Logic 379
B.2.4 Algorithm based on Logic Minimization 380
B.2.5 Multiple-Valued Logic Minimization 382
B.3 Local and Global Optimality 383
B.3.1 Derivation of Global Hyperplane Arrangement 383
B.3.2 Optimality of Algorithms 385

Bibliography

References 389

Symbols and Acronyms

Logic Operators and Functions

$A \Rightarrow B$	A implies B, i.e., if A is true then B must be true
$A \Leftrightarrow B$	A implies B and B implies A, i.e., A is true if and only if (iff) B is true

Sets

$\mathbb{R}$ ($\mathbb{R}_+$)	Set of (non-negative) real numbers
$\mathbb{N}$ ($\mathbb{N}_+$)	Set of (non-negative) integers
$\mathbb{R}^n$	Set of real vectors with n elements
$\mathbb{R}^{n \times m}$	Set of real matrices with n rows and m columns

Algebraic Operators

$[a, b]$	depending on the context: a closed interval in $\mathbb{R}$ or a row vector
A'	Transpose of matrix A
A^{-1}	Inverse of matrix A
$\det(A)$	Determinant of matrix A
$A \succ (\succeq) 0$	A symmetric positive (semi)definite matrix, $x'Ax > (\geq) 0,\ \forall x \neq 0$
$A \prec (\preceq) 0$	A symmetric negative (semi)definite matrix, $x'Ax < (\leq) 0,\ \forall x \neq 0$.
A_i	i-th row of matrix A
x_i	i-th element of vector x
$x \in \mathbb{R}^n,\ x > 0\ (x \geq 0)$	true iff $x_i > 0\ (x_i \geq 0)\ \forall\ i = 1, \dots, n$
$x \in \mathbb{R}^n,\ x < 0\ (x \leq 0)$	true iff $x_i < 0\ (x_i \leq 0)\ \forall\ i = 1, \dots, n$
$\lvert x \rvert,\ x \in \mathbb{R}$	absolute value of x
$\lVert x \rVert$	Any vector norm of x
$\lVert x \rVert_2$	Euclidian norm of vector $x \in \mathbb{R}^n$, $\lVert x \rVert_2 = \sqrt{\sum_{i=1}^n \lvert x_i \rvert^2}$
$\lVert x \rVert_1$	Sum of absolute elements of vector $x \in \mathbb{R}^n$, $\lVert x \rVert_1 := \sum_{i=1}^n \lvert x_i \rvert$
$\lVert x \rVert_\infty$	Largest absolute value of the vector $x \in \mathbb{R}^n$, $\lVert x \rVert_\infty := \max_{i \in \{1,\dots,n\}} \lvert x_i \rvert$
$\lVert S \rVert_\infty$	The matrix ∞-norm of $S \in \mathbb{C}^{m \times n_z}$,i.e., $\lVert S \rVert_\infty = \max_{1 \leq i \leq m} \sum_{j=1}^{n_z} \lvert s_{i,j} \rvert$
$\lVert S \rVert_1$	The matrix 1-norm of $S \in \mathbb{C}^{m \times n_z}$, i.e., $\lVert S \rVert_1 = \max_{1 \leq j \leq n_z} \sum_{i=1}^{m} \lvert s_{i,j} \rvert$

Set Operators and Functions

$\emptyset$	The empty set
$\lvert \mathcal{P} \rvert$	The cardinality of $\mathcal{P}$, i.e., the number of elements in $\mathcal{P}$
$\mathcal{P} \cap \mathcal{Q}$	Set intersection $\mathcal{P} \cap \mathcal{Q} = \{x\ : x \in \mathcal{P} \text{ and } x \in \mathcal{Q}\}$
$\mathcal{P} \cup \mathcal{Q}$	Set union $\mathcal{P} \cup \mathcal{Q} = \{x\ :\ x \in \mathcal{P} \text{ or } x \in \mathcal{Q}\}$
$\bigcup_{r \in \{1,\dots,R\}} \mathcal{P}_r$	Union of R sets $\mathcal{P}_r$, i.e. $\bigcup_{r \in \{1,\dots,R\}} \mathcal{P}_r = \{x\ :\ x \in \mathcal{P}_0 \text{ or } \dots \text{ or } x \in \mathcal{P}_R\}$
$\mathcal{P}^c$	Complement of the set $\mathcal{P}$, $\mathcal{P}^c = \{x\ :\ x \notin \mathcal{P}\}$
$\mathcal{P} \setminus \mathcal{Q}$	Set difference $\mathcal{P} \setminus \mathcal{Q} = \{x\ :\ x \in \mathcal{P} \text{ and } x \notin \mathcal{Q}\}$
$\mathcal{P} \subseteq \mathcal{Q}$	The set $\mathcal{P}$ is a subset of $\mathcal{Q}$, $x \in \mathcal{P} \Rightarrow\ x \in \mathcal{Q}$
$\mathcal{P} \subset \mathcal{Q}$	The set $\mathcal{P}$ is a strict subset of $\mathcal{Q}$, $x \in \mathcal{P} \Rightarrow\ x \in \mathcal{Q}$ and $\exists x \in (\mathcal{Q} \setminus \mathcal{P})$
$\mathcal{P} \supseteq \mathcal{Q}$	The set $\mathcal{P}$ is a superset of $\mathcal{Q}$
$\mathcal{P} \supset \mathcal{Q}$	The set $\mathcal{P}$ is a strict superset of $\mathcal{Q}$
$\mathcal{P} \ominus \mathcal{Q}$	Pontryagin difference $\mathcal{P} \ominus \mathcal{Q} = \{x\ :\ \ x + q \in \mathcal{P},\ \forall q \in \mathcal{Q}\}$
$\mathcal{P} \oplus \mathcal{Q}$	Minkowski sum $\mathcal{P} \oplus \mathcal{Q} = \{x + q\ :\ \ x \in \mathcal{P},\ q \in \mathcal{Q}\}$
$\partial \mathcal{P}$	The boundary of $\mathcal{P}$
$\text{int}(\mathcal{P})$	The interior of $\mathcal{P}$, i.e. $\text{int}(\mathcal{P}) = \mathcal{P} \setminus \partial \mathcal{P}$
$f(x)$	with abuse of notation denotes the value of the function f in x or the function f, $f:\ x \to f(x)$. The meaning will be clear from the context
$:$	"such that"

Others

I Identity matrix
1 Vector of ones, $\mathbf{1} = [1\ 1\ \dots\ 1]'$
0 Vector of zeros, $\mathbf{0} = [0\ 0\ \dots\ 0]'$

Acronyms

ARE	Algebraic Riccati Equation
BMI	Bilinear Matrix Inequality
CLQR	Constrained Linear Quadratic Regulator
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-QP	multi-parametric Quadratic Program
PWA	Piecewise Affine (See Definition 3.7)
PPWA	Piecewise Affine on Polyhedra (See Definition 3.8)
PWP	Piecewise Polynomial
PWQ	Piecewise Quadratic
QP	Quadratic Program(ming)
RHC	Receding Horizon Control
rhs	right-hand side
SDP	Semi Definite Program(ming)

Dynamical Systems

$x(k)$	Measurement/Estimation of state x at time k
x_k	Predicted value of state x at time k, given a measurement $x(0)$
x^+	Successor of vector x, i.e. if $x = x(k)$ then $x^+ = x(k+1)$
$\mathcal{F}_\infty$	Minimal robust positively invariant set
$\mathcal{O}_\infty$	Maximal robust positively invariant set
$\mathcal{C}_\infty$	Maximal robust control invariant set
$\mathcal{K}_\infty$	Maximal robust stabilizable set
$\mathcal{O}_\infty^{\mathrm{LQR}}$	Maximal positively invariant set $\mathcal{O}_\infty$ for LTI system under LQR feedback

Part I
Basics on Optimization

Chapter 1
Main Concepts

Abstract In this Chapter we introduce the main concepts and definitions of mathematical programming theory.

1.1 Optimization Problems

An optimization problem is generally formulated as

$$\begin{array}{ll} \inf_z & f(z) \\ \text{subj. to } & z \in S \subseteq Z \end{array} \tag{1.1}$$

where the vector z collects the decision variables, Z is the *domain* of the decision variables, $S \subseteq Z$ is the set of *feasible* or *admissible* decisions. The ~~function $f : Z \to \mathbb{R}$ assigns to each decision z a *cost* $f(z) \in \mathbb{R}$. We will~~ often use the following shorter form of problem (1.1)

$$\inf_{z \in S \subseteq Z} f(z) \tag{1.2}$$

Solving problem (1.2) means to compute the least possible cost J^*

$$J^* \triangleq \inf_{z \in S} f(z)$$

The number J^* is the *optimal value* of problem (1.2), i.e., the greatest lower bound of $f(z)$ over the set S:

$$f(z) \geq J^*,\ \forall z \in S \text{ and } (\exists \bar{z} \in S : f(\bar{z}) = J^* \text{ or } \forall \varepsilon > 0\ \exists z \in S :\ f(z) \leq J^* + \varepsilon)$$

If $J^* = -\infty$ we say that the problem is unbounded below. If the set S is empty then the problem is said to be *infeasible* and we set $J^* = +\infty$ by convention. If $S = Z$ the problem is said to be *unconstrained*.

In general, one is also interested in finding an *optimal solution*, that is in finding a decision whose associated cost equals the optimal value, i.e., $z^* \in S$ with $f(z^*) = J^*$. If such z^* exists, then we rewrite problem (1.2) as

$$J^* = \min_{z \in S} f(z) \tag{1.3}$$

and z^* is called *optimizer* or *global optimizer*. The set of all optimal solutions is denoted by

$$\text{argmin}_{z \in S} f(z) \triangleq \{z \in S : f(z) = J^*\} \,.$$

The problem of determining whether the set of feasible decisions is empty and, if not, to find a point which is feasible, is called *feasibility problem*.

1.1.1 Continuous Problems

In continuous optimization the problem domain Z is a subset of the finite-dimensional Euclidian vector-space $\mathbb{R}^s$ and the subset of admissible vectors is defined through a list of real-valued functions:

$$\begin{array}{lll} \inf_z & f(z) & \\ \text{subj. to} & g_i(z) \leq 0 & \text{for } i = 1, \ldots, m \\ & h_i(z) = 0 & \text{for } i = 1, \ldots, p \\ & z \in Z & \end{array} \tag{1.4}$$

where $f, g_1, \ldots, g_m, h_1, \ldots, h_p$ are real-valued functions defined over $\mathbb{R}^s$, i.e., $f : \mathbb{R}^s \to \mathbb{R}$, $g_i : \mathbb{R}^s \to \mathbb{R}$, $h_i : \mathbb{R}^s \to \mathbb{R}$. The domain Z is the intersection of the domains of the cost and constraint functions:

$$Z = \{z \in \mathbb{R}^s \ : z \in \text{dom } f, \ z \in \text{dom } g_i, \ i = 1, \ldots, m, \ z \in \text{dom } h_i, \ i = 1, \ldots, p\} \tag{1.5}$$

In the sequel we will consider the constraint $z \in Z$ implicit in the optimization problem and often omit it. Problem (1.4) is unconstrained if $m = p = 0$.

The inequalities $g_i(z) \leq 0$ are called *inequality constraints* and the equations $h_i(z) = 0$ are called *equality constraints*. A point $\bar{z} \in \mathbb{R}^s$ is *feasible* for problem (1.4) if: (i) it belongs to Z, (ii) it satisfies all inequality and equality constraints, i.e., $g_i(\bar{z}) \leq 0$, $i = 1, \ldots, m$, $h_i(\bar{z}) = 0$, $i = 1, \ldots, p$. The set of feasible vectors is

$$S = \{z \in \mathbb{R}^s : z \in Z, \ g_i(z) \leq 0, \ i = 1, \ldots, m, \ \ h_i(z) = 0, \ i = 1, \ldots, p\}. \tag{1.6}$$

Problem (1.4) is a continuous finite-dimensional optimization problem (since Z is a finite-dimensional Euclidian vector space). We will also refer to (1.4) as a *nonlinear mathematical program* or simply *nonlinear program*. Let J^* be the optimal value of problem (1.4). An optimizer, if it exists, is a feasible vector z^* with $f(z^*) = J^*$.

A feasible point $\bar{z}$ is *locally optimal* for problem (1.4) if there exists an $R \succ 0$ such that

$$\begin{array}{lll} f(\bar{z}) = \inf_z & f(z) & \\ \text{subj. to} & g_i(z) \leq 0 & \text{for } i = 1, \ldots, m \\ & h_i(z) = 0 & \text{for } i = 1, \ldots, p \\ & \|z - \bar{z}\| \leq R & \\ & z \in Z & \end{array} \tag{1.7}$$

Roughly speaking, this means that $\bar{z}$ is the minimizer of $f(z)$ in a feasible neighborhood of $\bar{z}$ defined by $\|z-\bar{z}\| \leq R$. The point $\bar{z}$ is called *local optimizer*.

1.1.1.1 Active, Inactive and Redundant Constraints

Consider a feasible point $\bar{z}$. We say that the i-th inequality constraint $g_i(z) \leq 0$ is *active* at $\bar{z}$ if $g_i(\bar{z}) = 0$. If $g_i(\bar{z}) < 0$ we say that the constraint $g_i(z) \leq 0$ is *inactive* at $\bar{z}$. Equality constraints are always active for all feasible points.

We say that a constraint is *redundant* if removing it from the list of constraints does not change the feasible set S. This implies that removing a redundant constraint from problem (1.4) does not change its solution.

1.1.1.2 Problems in Standard Forms

Optimization problems can be cast in several forms. In this book we use the form (1.4) where we adopt the convention to minimize the cost function and to have the right-hand side of the inequality and equality constraints equal to zero. Any problem in a different form (e.g., a maximization problem or a problem with "box constraints") can be transformed and arranged into this form. The interested reader is referred to Chapter 4 of [57] for a detailed discussion on transformations of optimization problems into different standard forms.

1.1.1.3 Eliminating Equality Constraints

Often in this book we will restrict our attention to problems without equality constraints, i.e., $p = 0$

$$\begin{array}{lll} \inf_z & f(z) & \\ \text{subj. to} & g_i(z) \leq 0 & \text{for } i = 1, \ldots, m \end{array} \tag{1.8}$$

The simplest way to remove equality constraints is to replace them with two inequalities for each equality, i.e., $h_i(z) = 0$ is replaced by $h_i(z) \leq 0$ and $-h_i(z) \leq 0$. Such a method, however, can lead to poor numerical conditioning and may ruin the efficiency and accuracy of a numerical solver.

If one can explicitly parameterize the solution of the equality constraint $h_i(z) = 0$, then the equality constraint can be *eliminated* from the problem. This process can be described in a simple way for linear equality constraints. Assume the equality constraints to be linear, $Az - b = 0$, with $A \in \mathbb{R}^{p \times s}$. If $Az = b$ is inconsistent then the problem is infeasible. The general solution of the equation $Az = b$ can be expressed as $z = Fx + x_0$ where F is a matrix of full rank whose spanned space coincides with the null space of the A matrix, i.e., $\mathcal{R}(F) = \mathcal{N}(A)$, $F \in \mathbb{R}^{s \times k}$, where k is the dimension of the null space of A. The variable $x \in \mathbb{R}^k$ is the new optimization variable and the original problem becomes

$$\begin{array}{lll} \inf_x & f(Fx + x_0) & \\ \text{subj. to} & g_i(Fx + x_0) \leq 0 & \text{for } i = 1, \ldots, m \end{array} \tag{1.9}$$

We want to point out that in some cases the elimination of equality constraints can make the problem harder to analyze and understand and can make a solver less efficient. In large problems it can destroy useful structural properties of the problem such as sparsity. Some advanced numerical solvers perform elimination automatically.

1.1.1.4 Problem Description

The functions f, g_i and h_i can be available in analytical form or can be described through an *oracle model* (also called "black box" or "subroutine" model). In an oracle model f, g_i and h_i are not known explicitly but can be evaluated by querying the oracle. Often the oracle consists of subroutines which, called with the argument z, return $f(z)$, $g_i(z)$ and $h_i(z)$ and their gradients $\nabla f(z)$, $\nabla g_i(z)$, $\nabla h_i(z)$. In the rest of the book we assume that analytical expressions of the cost and the constraints of the optimization problem are available.

1.1.2 Integer and Mixed-Integer Problems

If the decision set Z in the optimization problem (1.2) is finite, then the optimization problem is called *combinatorial* or *discrete*. If $Z \subseteq \{0, 1\}^s$, then the problem is said to be *integer*.

If Z is a subset of the Cartesian product of an integer set and a real Euclidian space, i.e., $Z \subseteq \{[z_c, z_b] : z_c \in \mathbb{R}^{s_c}, z_b \in \{0,1\}^{s_b}\}$, then the problem is said to be *mixed-integer*. The standard formulation of a *mixed-integer nonlinear program* is

$$\begin{array}{ll} \inf_{[z_c,z_b]} & f(z_c, z_b) \\ \text{subj. to} & g_i(z_c, z_b) \leq 0 \quad \text{for } i = 1, \ldots, m \\ & h_i(z_c, z_b) = 0 \quad \text{for } i = 1, \ldots, p \\ & z_c \in \mathbb{R}^{s_c},\ z_b \in \{0,1\}^{s_b} \\ & [z_c, z_b] \in Z \end{array} \tag{1.10}$$

where $f, g_1, \ldots, g_m, h_1, \ldots, h_p$ are real-valued functions defined over Z.

For combinatorial, integer and mixed-integer optimization problems all definitions introduced in the previous section apply.

1.2 Convexity

A set $S \in \mathbb{R}^s$ is *convex* if

$$\lambda z_1 + (1-\lambda)z_2 \in S \text{ for all } z_1, z_2 \in S, \lambda \in [0,1].$$

A function $f : S \to \mathbb{R}$ is convex if S is convex and

$$f(\lambda z_1 + (1-\lambda)z_2) \leq \lambda f(z_1) + (1-\lambda) f(z_2) \\ \text{for all } z_1, z_2 \in S, \lambda \in [0,1].$$

A function $f : S \to \mathbb{R}$ is *strictly convex* if S is convex and

$$f(\lambda z_1 + (1-\lambda)z_2) < \lambda f(z_1) + (1-\lambda) f(z_2) \\ \text{for all } z_1, z_2 \in S, \lambda \in (0,1).$$

A function $f : S \to \mathbb{R}$ is concave if S is convex and $-f$ is convex.

1.2.0.1 Operations preserving convexity

Various operations preserve convexity of functions and sets. A detailed list can be found in Chapter 3.2 of [57]. A few operations used in this book are reported below.

1. The intersection of an arbitrary number of convex sets is a convex set:

$$\text{if } S_1, S_2, \ldots, S_k \text{ are convex, then } S_1 \cap S_2 \cap \ldots \cap S_k \text{ is convex.}$$

This property extends to the intersection of an infinite number of sets:

$$\text{if } S_n \text{ is convex } forall n \in \mathbb{N}_+ \text{ then } \bigcap_{n\in\mathbb{N}_+} S_n \text{ is convex.}$$

The empty set is convex because it satisfies the definition of convexity.

2. The sub-level sets of a convex function f on S are convex:

$$\text{if } f(z) \text{ is convex then } S_\alpha \triangleq \{z \in S : f(z) \le \alpha\} \text{ is convex } \forall\alpha.$$

3. If $f_1, \dots, f_N$ are convex functions, then $\sum_{i=1}^N \alpha_i f_i$ is a convex function for all $\alpha_i \ge 0$, $i = 1, \dots, N$.
4. The composition of a convex function $f(z)$ with an affine map $z = Ax+b$ generates a convex function $f(Ax+b)$ of x:

$$\text{if } f(z) \text{ is convex then } f(Ax+b) \text{ is convex } on\{x : Ax+b \in \text{dom}(f)\}$$

5. Suppose $f(x) = h(g(x)) = h(g_1(x), \dots, g_k(x))$ with $h : \mathbb{R}^k \to R$, $g_i : \mathbb{R}^s \to R$. Then,
   a. f is convex if h is convex, h is nondecreasing in each argument, and g_i are convex,
   b. f is convex if h is convex, h is nonincreasing in each argument, and g_i are concave,
   c. f is concave if h is concave, h is nondecreasing in each argument, and g_i are concave.
6. The pointwise maximum of a set of convex functions is a convex function:

$$f_1(z), \dots, f_k(z) \text{ convex functions} \Rightarrow f(z) = \max\{f_1(z), \dots, f_k(z)\} \text{ is a convex function.}$$

1.2.0.2 Linear and quadratic convex functions

The following convex functions will be used extensively in this book:

1. A linear function $f(z) = c'z + d$ is both convex and concave.
2. A quadratic function $f(z) = z'Qz + 2r'z + s$ is convex if and only if $Q \succeq 0$.
3. A quadratic function $f(z) = z'Qz + 2r'z + s$ is strictly convex if and only if $Q \succ 0$.

1.2.0.3 Convex optimization problems

The standard optimization problem (1.4) is said to be *convex* if the cost function f is convex on Z *and* S is a convex set. A fundamental property of convex optimization problems is that local optimizers are also global optimizers. This is proven in the next proposition.

Proposition 1.1. *Consider a convex optimization problem and let $\bar{z}$ be a local optimizer. Then, $\bar{z}$ is a global optimizer.*

Proof: By hypothesis $\bar{z}$ is feasible and there exists R such that

$$f(\bar{z}) = \inf\{f(z) : g_i(z) \leq 0 \, i = 1, \ldots, m, \; h_i(z) = 0, \; i = 1, \ldots, p \; \|z - \bar{z}\| \leq R\}. \tag{1.11}$$

Now suppose that $\bar{z}$ is not globally optimal. Then, there exist a feasible y such that $f(y) < f(\bar{z})$, which implies that $\|y - \bar{z}\| > R$. Now consider the point z given by

$$z = (1-\theta)\bar{z} + \theta y, \quad \theta = \frac{R}{2\|y - \bar{z}\|}.$$

Then $\|z - \bar{z}\| = R/2 < R$ and by convexity of the feasible set z is feasible. By convexity of the cost function f

$$f(z) \leq (1-\theta) f(\bar{z}) + \theta f(y) < f(\bar{z}),$$

which contradicts (1.11). □

Proposition 1.1 does not make any statement about the existence of a solution to problem (1.4). It merely states that all local minima of problem (1.4) are also global minima. For this reason, convexity plays a central role in the solution of continuous optimization problems. It suffices to compute a local minimum to problem (1.4) to determine its global minimum. Convexity also plays a major role in most non-convex optimization problems which are solved by iterating between the solutions of convex sub-problems.

It is difficult to determine whether the feasible set S of the optimization problem (1.4) is convex or not except in special cases. For example, if the functions $g_1(z), \ldots, g_m(z)$ are convex and all the $h_i(z)$ (if any) are affine in z, then the feasible region S in (1.6) is an intersection of convex sets and therefore convex. Moreover there are non-convex problems which can be transformed into convex problems through a change of variables and manipulations on cost and constraints. The discussion of this topic goes beyond the scope of this overview on optimization. The interested reader is referred to [57].

Remark 1.1. With the exception of trivial cases, integer and mixed-integer optimization problems are always non-convex problems because $\{0, 1\}$ is not a convex set.

Chapter 2
Optimality Conditions

2.1 Introduction

In general, an analytical solution to problem (1.4), restated below, does not exist.

$$\begin{array}{lll} \inf_z & f(z) & \\ \text{subj. to} & g_i(z) \leq 0 & \text{for } i = 1, \ldots, m \\ & h_i(z) = 0 & \text{for } i = 1, \ldots, p \\ & z \in Z & \end{array} \tag{2.1}$$

Solutions are usually computed by recursive algorithms which start from an initial guess z_0 and at step k generate a point z_k such that the sequence $\{f(z_k)\}_{k=0,1,2,\ldots}$ converges to J^* as k increases. These algorithms recursively use and/or solve conditions for optimality, i.e., analytical conditions that a point z must satisfy in order to be an optimizer. For instance, for *convex, unconstrained* optimization problems with a *smooth* cost function the best known optimality criterion requires the gradient to vanish at the optimizer, i.e., z is an optimizer if and only if $\nabla f(z) = 0$.

Next we will briefly summarize necessary and sufficient optimality conditions for unconstrained optimization problems.

Necessary conditions (unconstrained problem)

Theorem 2.1. *Suppose that $f : \mathbb{R}^s \to \mathbb{R}$ is differentiable at $\bar{z}$. If there exists a vector d such that $\nabla f(\bar{z})'d < 0$, then there exists a $\delta > 0$ such that $f(\bar{z} + \lambda d) < f(\bar{z})$ for all $\lambda \in (0, \delta)$.*

The vector d in the theorem above is called *descent direction*. In a given point $\bar{z}$ a descent direction d satisfies the condition $\nabla f(\bar{z})'d < 0$. Theorem 2.1 states that if a descent direction exists at a point $\bar{z}$, then it is possible to move from $\bar{z}$ towards a new point $\tilde{z}$ whose associated cost $f(\tilde{z})$ is lower than $f(\bar{z})$. The smaller $\nabla f(\bar{z})'d$ is the smaller will be the cost at the new point $f(\tilde{z})$. The direction of *steepest descent* d_s at a given point

$\bar{z}$ is defined as the normalized direction where $\nabla f(\bar{z})' d_s < 0$ is minimized. The direction d_s of steepest descent is $d_s = -\frac{\nabla f(\bar{z})}{\|\nabla f(\bar{z})\|}$.
Two corollaries of Theorem 2.1 are stated next.

Corollary 2.1. *Suppose that $f : \mathbb{R}^s \to \mathbb{R}$ is differentiable at $\bar{z}$. If $\bar{z}$ is a local minimizer, then $\nabla f(\bar{z}) = 0$.*

Corollary 2.2. *Suppose that $f : \mathbb{R}^s \to \mathbb{R}$ is twice differentiable at $\bar{z}$. If $\bar{z}$ is a local minimizer, then $\nabla f(\bar{z}) = 0$ and the Hessian $\nabla^2 f(\bar{z})$ is positive semidefinite.*

Sufficient condition (unconstrained problem)

Theorem 2.2. *Suppose that $f : \mathbb{R}^s \to \mathbb{R}$ is twice differentiable at $\bar{z}$. If $\nabla f(\bar{z}) = 0$ and the Hessian of $f(z)$ at $\bar{z}$ is positive definite, then $\bar{z}$ is a local minimizer*

Necessary and sufficient condition (unconstrained problem)

Theorem 2.3. *Suppose that $f : \mathbb{R}^s \to \mathbb{R}$ is differentiable at $\bar{z}$. If f is convex, then $\bar{z}$ is a global minimizer if and only if $\nabla f(\bar{z}) = 0$.*

The proofs of the theorems presented above can be found in Chapters 4 and 8.6.1 of [23].

When the optimization is constrained and the cost function is not sufficiently smooth, the conditions for optimality become more complicated. The intent of this chapter is to give an overview of some important optimality criteria for constrained nonlinear optimization. The optimality conditions derived in this chapter will be the main building blocks for the theory developed later in this book.

2.2 Lagrange Duality Theory

Consider the nonlinear program (2.1). Let J^* be the optimal value. Denote by Z the domain of cost and constraints (1.5). Any feasible point $\bar{z}$ provides an upper bound to the optimal value $f(\bar{z}) \geq J^*$. Next we will show how to generate a lower bound on J^*.

Starting from the standard nonlinear program (2.1) we construct another problem with different variables and constraints. The original problem (2.1) will be called primal problem while the new one will be called *dual* problem. First, we augment the objective function with a weighted sum of the constraints. In this way the *Lagrange function* L is obtained

$$\begin{aligned} L(z,u,v) = f(z) + u_1 g_1(z) + \ldots + u_m g_m(z) + \\ + v_1 h_1(z) + \ldots + v_p h_p(z) \end{aligned} \tag{2.2}$$

where the scalars $u_1, \dots, u_m, v_1, \dots, v_p$ are real. We can write equation (2.2) in the compact form

$$L(z, u, v) \triangleq f(z) + u'g(z) + v'h(z), \tag{2.3}$$

where $u = [u_1, \dots, u_m]'$, $v = [v_1, \dots, v_p]'$ and $L : \mathbb{R}^s \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$. The components u_i and v_i are called Lagrange multipliers or dual variables. Note that the i-th dual variable u_i is associated with the i-th inequality constraint of problem (2.1), the i-th dual variable v_i is associated with the i-th equality constraint of problem (2.1).

Let z be a feasible point: for arbitrary vectors $u \geq 0$ and v we trivially obtain a lower bound on $f(z)$

$$L(z, u, v) \leq f(z). \tag{2.4}$$

We minimize both sides of equation (2.4)

$$\inf_{z \in Z,\ g(z) \leq 0,\ h(z)=0} L(z, u, v) \leq \inf_{z \in Z,\ g(z) \leq 0,\ h(z)=0} f(z) \tag{2.5}$$

in order to reconstruct the original problem on the right-hand side of the expression. Since for arbitrary $u \geq 0$ and v

$$\inf_{z \in Z,\ g(z) \leq 0,\ h(z)=0} L(z, u, v) \geq \inf_{z \in Z} L(z, u, v), \tag{2.6}$$

we obtain

$$\inf_{z \in Z} L(z, u, v) \leq \inf_{z \in Z,\ g(z) \leq 0,\ h(z)=0} f(z). \tag{2.7}$$

Equation (2.7) implies that for arbitrary $u \geq 0$ and v the solution to

$$\inf_{z} L(z, u, v), \tag{2.8}$$

provides us with a lower bound to the original problem. The "best" lower bound is obtained by maximizing problem (2.8) over the dual variables

$$\sup_{(u,v),\ u \geq 0} \inf_{z \in Z} L(z, u, v) \leq \inf_{z \in Z,\ g(z) \leq 0,\ h(z)=0} f(z).$$

Define the dual cost $\Theta(u, v)$ as follows

$$\Theta(u, v) \triangleq \inf_{z \in Z} L(z, u, v) \in [-\infty, +\infty]. \tag{2.9}$$

Then the Lagrange dual problem is defined as

$$\sup_{(u,v),\ u \geq 0} \Theta(u, v). \tag{2.10}$$

The dual cost $\Theta(u,v)$ is the optimal value of an **unconstrained optimization problem**. Problem (2.9) is called *Lagrange dual subproblem*. Only points (u,v) with $\Theta(u,v) > -\infty$ are interesting for the Lagrange dual problem. A point (u,v) will be called *dual feasible* if $u \geq 0$ and $\Theta(u,v) > -\infty$. $\Theta(u,v)$ is **always a concave function** since it is the pointwise infimum of a family of affine functions of (u,v). This implies that the *dual problem is a convex optimization problem (max of a concave function over a convex set) even if the original problem is not convex*. Therefore, it is much easier to solve the dual problem than the primal (which is in general non-convex). However, in general the solution to the dual problem is only a lower bound of the primal problem:

$$\sup_{(u,v),\ u\geq 0} \Theta(u,v) \leq \inf_{z\in Z,\ g(z)\leq 0,\ h(z)=0} f(z)$$

Such a property is called **weak duality**. In a simpler form, let J^* and d^* be the primal and dual optimal value, respectively,

$$J^* = \inf_{z\in Z,\ g(z)\leq 0,\ h(z)=0} f(z) \tag{2.11a}$$

$$d^* = \sup_{(u,v),\ u\geq 0} \Theta(u,v) \tag{2.11b}$$

then, we always have

$$J^* \geq d^* \tag{2.12}$$

and the difference $J^* - d^*$ is called **optimal duality gap**. The weak duality inequality (2.12) holds also when d^* and J^* are infinite. For example, if the primal problem is unbounded below, so that $J^* = -\infty$, we must have $d^* = -\infty$, i.e., the Lagrange dual problem is infeasible. Conversely, if the dual problem is unbounded above, so that $d^* = +\infty$, we must have $J^* = +\infty$, i.e., the primal problem is infeasible.

2.2.1 Strong Duality and Constraint Qualifications

If $d^* = J^*$, then the duality gap is zero and we say that **strong duality** holds:

$$\sup_{(u,v),\ u\geq 0} \Theta(u,v) = \inf_{z\in Z,\ g(z)\leq 0,\ h(z)=0} f(z) \tag{2.13}$$

This means that the best lower bound obtained by solving the dual problem coincides with the optimal cost of the primal problem. In general, strong duality does not hold, even for convex primal problems. **Constraint qualifications** are conditions on the constraint functions which imply strong duality for convex problems. A detailed discussion on constraints qualifications can be found in Chapter 5 of [23].

A well known simple constraint qualification is the "Slater's condition":

Definition 2.1 (Slater's condition). Consider problem (2.1). There exists $\hat{z} \in \mathbb{R}^s$ which belongs to the relative interior of the problem domain Z, which is feasible ($g(\hat{z}) \leq 0,\ h(\hat{z}) = 0$) and for which $g_j(\hat{z}) < 0$ for all j for which g_j is not an affine function.

Remark 2.1. Note that Slater's condition reduces to feasibility when all inequality constraints are linear.

Theorem 2.4 (Slater's theorem). *Consider the primal problem (2.11a) and its dual problem (2.11b). If the primal problem is convex and Slater's condition holds then $d^* > -\infty$ and $d^* = J^*$.*

Note that Slater's theorem states that Slater's condition implies strong duality for convex problems and that the dual optimal value is attained when $d^* > -\infty$.

2.2.2 Certificate of Optimality

Consider the (primal) optimization problem (2.1) and its dual (2.10). Any feasible point z give us information about an upper bound on the cost, i.e., $J^* \leq f(z)$. If we can find a dual feasible point (u, v) then we can establish a lower bound on the optimal value of the primal problem: $\Theta(u, v) \leq J^*$. In summary, without knowing the exact value of J^* we can give a bound on how suboptimal a given feasible point is. In fact, if z is primal feasible and (u, v) is dual feasible then $\Theta(u, v) \leq J^* \leq f(z)$. Therefore z is ε-suboptimal, with ε equal to primal-dual gap, i.e., $\varepsilon = f(z) - \Theta(u, v)$.

The optimal value of the primal (and dual) problems will lie in the same interval

$$J^*, d^* \in [\Theta(u, v), f(z)].$$

For this reason (u, v) is also called a *certificate* that proves the (sub)optimality of z. Optimization algorithms make extensive use of such criteria. Primal-Dual algorithms iteratively solve primal and dual problems and generate a sequence of primal and dual feasible points z_k, (u_k, v_k), $k \geq 0$ until a certain ε is reached. The condition

$$f(z_k) - \Theta(u_k, v_k) < \varepsilon,$$

for terminating the algorithm guarantees that when the algorithm terminates, z_k is ε-suboptimal. If strong duality holds the condition can be met for arbitrarily small tolerances ε.

2.3 Complementary Slackness

Consider the (primal) optimization problem (2.1) and its dual (2.10). Assume that strong duality holds. Suppose that z^* and $(u^*,\ v^*)$ are primal and dual feasible with zero duality gap (hence, they are primal and dual optimal):

$$f(z^*) = \Theta(u^*, v^*)$$

By definition of the dual problem, we have

$$f(z^*) = \inf_z \left(f(z) + u^{*\prime} g(z) + v^{*\prime} h(z) \right)$$

Therefore

$$f(z^*) \leq f(z^*) + u^{*\prime} g(z^*) + v^{*\prime} h(z^*) \tag{2.14}$$

and since $h(z^*) = 0$, $u^* \geq 0$ and $g(z^*) \leq 0$ we have

$$f(z^*) \leq f(z^*) + u^{*\prime} g(z^*) \leq f(z^*) \tag{2.15}$$

From the last equation we can conclude that $u^{*\prime} g(z^*) = \sum_{i=1}^m u_i^* g_i(z^*) = 0$ and since $u_i^* \geq 0$ and $g_i(z^*) \leq 0$, we have

$$u_i^* g_i(z^*) = 0,\ \ i = 1, \ldots, m \tag{2.16}$$

Conditions (2.16) are called **complementary slackness** conditions. Complementary slackness conditions can be interpreted as follows. If the i-th constraint of the primal problem is inactive at optimum ($g_i(z^*) < 0$) then the i-th dual optimizer has to be zero ($u_i^* = 0$). Vice versa, if the i-th dual optimizer is different from zero ($u_i^* > 0$), then the i-th constraint is active at the optimum ($g_i(z^*) = 0$).

Relation (2.16) implies that the inequality in (2.14) holds as equality

$$f(z^*) + \sum_i u_i^* g_i(z^*) + \sum_j v_j^* h_j(z^*) = \min_{z \in Z} \left(f(z) + \sum_i u_i^* g_i(z) + \sum_j v_j^* h_j(z) \right). \tag{2.17}$$

Therefore, complementary slackness implies that z^* is a minimizer of $L(z, u^*, v^*)$.

2.4 Karush-Kuhn-Tucker Conditions

Consider the (primal) optimization problem (2.1) and its dual (2.10). Assume that strong duality holds. Assume that the cost functions and constraint functions f, g_i, h_i are differentiable. Let z^* and $(u^*,\ v^*)$ be primal and dual optimal points, respectively. Complementary slackness implies that

z^* minimizes $L(z, u^*, v^*)$ under no constraints (equation (2.17)). Since f, g_i, h_i are differentiable, the gradient of $L(z, u^*, v^*)$ must be zero at z^*

$$\nabla f(z^*) + \sum_i u_i^* \nabla g_i(z^*) + \sum_j v_j^* \nabla h_j(z^*) = 0.$$

In summary, the primal and dual optimal pair z^*, (u^*, v^*) of an optimization problem with differentiable cost and constraints and zero duality gap, have to satisfy the following conditions:

$$\nabla f(z^*) + \sum_{i=1}^{m} u_i^* \nabla g_i(z^*) + \sum_{j=1}^{p} v_j^* \nabla h_i(z^*) = 0, \tag{2.18a}$$

$$u_i^* g_i(z^*) = 0, \quad i = 1, \dots, m \tag{2.18b}$$

$$u_i^* \geq 0, \quad i = 1, \dots, m \tag{2.18c}$$

$$g_i(z^*) \leq 0, \quad i = 1, \dots, m \tag{2.18d}$$

$$h_j(z^*) = 0 \quad j = 1, \dots, p \tag{2.18e}$$

where equations (2.18d)-(2.18e) are the primal feasibility conditions, equation (2.18c) is the dual feasibility condition and equations (2.18b) are the complementary slackness conditions.

Conditions (2.18a)-(2.18e) are called the *Karush-Kuhn-Tucker* (KKT) conditions. The KKT conditions are necessary conditions for any primal-dual optimal pair if strong duality holds and the cost and constraints are differentiable, i.e., any primal and dual optimal points z^*, (u^*, v^*) must satisfy the KKT conditions (2.18). If the primal problem is also convex then the KKT conditions are sufficient, i.e., a primal dual pair z^*, (u^*, v^*) which satisfies conditions (2.18a)-(2.18e) is a primal dual optimal pair with zero duality gap.

There are several theorems which characterize primal and dual optimal points z^* and (u^*, v^*) by using KKT conditions. They mainly differ on the type of constraint qualification chosen for characterizing strong duality. Next we report just two examples.

If a convex optimization problem with differentiable objective and constraint functions satisfies Slater's condition, then the KKT conditions provide necessary and sufficient conditions for optimality:

Theorem 2.5 (pag. 244 in [23]). *Consider problem (2.1) and let Z be a nonempty set of $\mathbb{R}^s$. Suppose that problem (2.1) is convex and that cost and constraints f, g_i and h_i are differentiable at a feasible z^*. If problem (2.1) satisfies Slater's condition then z^* is optimal if and only if there are (u^*, v^*) that, together with z^*, satisfy the KKT conditions (2.18).*

If a convex optimization problem with differentiable objective and constraint functions has linearly independent set of active constraints then the KKT conditions provide necessary and sufficient conditions for optimality:

Theorem 2.6 (Section 4.37 in [23]). *Consider problem (2.1) and let Z be a nonempty open set of $\mathbb{R}^s$. Let z^* be a feasible solution and $A = \{i : g_i(z^*) = 0\}$ be the set of active constraints at z^*. Suppose cost and constraints f, g_i are differentiable at z^* for all i and that h_i are continuously differentiable at z^* for all i. Further, suppose that $\nabla g_i(z^*)$ for $i \in A$ and $\nabla h_i(z^*)$ for $i = 1, \ldots, p$, are linearly independent. If z^*, (u^*, v^*) are primal and dual optimal points, then they satisfy the KKT conditions (2.18). In addition, if problem (2.1) is convex, then z^* is optimal if and only if there are (u^*, v^*) that, together with z^*, satisfy the KKT conditions (2.18).*

The KKT conditions play an important role in optimization. In a few special cases it is possible to solve the KKT conditions (and therefore, the optimization problem) analytically. Many algorithms for convex optimization are conceived as, or can be interpreted as, methods for solving the KKT conditions as Boyd and Vandenberghe observe in [57].

The following example [23] shows a convex problem where the KKT conditions are not fulfilled at the optimum. In particular, both the constraint qualifications of Theorem 2.6 and Slater's condition in Theorem 2.5 are violated.

Example 2.1. [23] Consider the convex optimization problem

$$\begin{array}{rl} \min & z_1 \\ \text{subj. to} & (z_1-1)^2 + (z_2-1)^2 \leq 1 \\ & (z_1-1)^2 + (z_2+1)^2 \leq 1 \end{array} \tag{2.19}$$

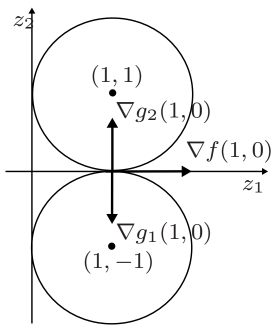


Fig. 2.1 Constraints, feasible set and gradients of Example 2.1

From the graphical interpretation in Figure 2.1 it is immediate that the feasible set is a single point $\bar{z} = [1, 0]'$. The optimization problem does not satisfy Slater's conditions and moreover $\bar{z}$ does not satisfy the constraint qualifications in Theorem 2.6. At the optimum $\bar{z}$ equations (2.18a) cannot be satisfied for any pair of real numbers u_1 and u_2.

2.4.1 KKT geometric interpretation

A geometric interpretation of the KKT conditions is depicted in Figure 2.2 for an optimization problem in two dimensions with inequality constraints and no equality constraints.

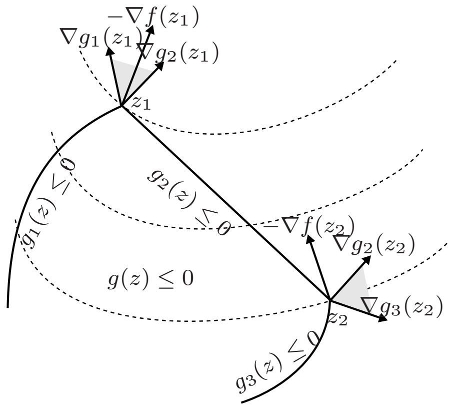


Fig. 2.2 Graphical interpretation of KKT conditions [23]

Equation (2.18a), can be rewritten as

$$-\nabla f(z) = \sum_{i\in A} u_i \nabla g_i(z), \quad u_i \geq 0. \tag{2.20}$$

where $A = \{1,2\}$ in z_1 and $A = \{2,3\}$ in z_2. This means that the negative gradient of the cost at the optimum $-\nabla f(z^*)$ (which represents the direction of steepest descent) has to belong to the cone spanned by the gradients of the active constraints ∇g_i, (since inactive constraints have the corresponding Lagrange variables equal to zero). In Figure 2.2, condition (2.20) is not satisfied at z_2. In fact, one can move within the set of feasible points $g(z) \leq 0$ and decrease f, which implies that z_2 is not optimal. At point z_1, on the other hand, the cost f can only decrease if some constraint is violated. Any movement in a feasible direction increases the cost. Conditions (2.20) are fulfilled and hence z_1 is optimal.

Chapter 3
Polyhedra, Polytopes and Simplices

Polyhedra will be the fundamental geometric objects used in this book. There is a vast body of literature related to polyhedra because they are important for a wide range of applications. In this chapter we introduce the main definitions and the algorithms which describe some operations on polyhedra. Most definitions given here are standard. For additional details the reader is referred to [270, 126, 102]. First, we recall a few basic general set definitions and operations.

3.1 General Set Definitions and Operations

An n**-dimensional ball** $\mathcal{B}(x_c, \rho)$ is the set $\mathcal{B}(x_c, \rho) = \{x \in \mathbb{R}^n : \|x - x_c\|_2 \leq \rho\}$. The vector x_c is the center of the ball and ρ is the radius.

Affine sets are sets described by the solutions of a system of linear equations:

$$\mathcal{F} = \{x \in \mathbb{R}^n \ : \ Ax = b, \ \text{ with } \ A \in \mathbb{R}^{m \times n}, \ b \in \mathbb{R}^m\}. \tag{3.1}$$

If $\mathcal{F}$ is an affine set and $\bar{x} \in \mathcal{F}$, then the translated set $\mathcal{V} = \{x - \bar{x} : x \in \mathcal{F}\}$ is a subspace.

The **affine combination** of a finite set of points $x^1, \ldots, x^k$ belonging to $\mathbb{R}^n$ is defined as the point $\lambda^1 x^1 + \ldots + \lambda^k x^k$ where $\sum_{i=1}^k \lambda^i = 1$.

The **affine hull** of $\mathcal{K} \subseteq \mathbb{R}^n$ is the set of all affine combinations of points in $\mathcal{K}$ and it is denoted as aff($\mathcal{K}$):

$$\text{aff}(\mathcal{K}) = \{\lambda^1 x^1 + \ldots + \lambda^k x^k \ : \ x^i \in \ \mathcal{K}, \ i = 1, \ldots, k, \ \sum_{i=1}^k \lambda^i = 1\} \tag{3.2}$$

The affine hull of $\mathcal{K}$ is the smallest affine set that contains $\mathcal{K}$, in the following sense: if $\mathcal{S}$ is any affine set with $\mathcal{K} \subseteq \mathcal{S}$, then aff($\mathcal{K}$)$\subseteq \mathcal{S}$.

The **dimension** of an affine set, affine combination or affine hull is the dimension of the largest ball of radius $\rho > 0$ included in the set.

Example 3.1. The set

$$\mathcal{F} = \{x \in \mathbb{R}^2 \ : \ x_1 + x_2 = 1\}$$

is an affine set in $\mathbb{R}^2$ of dimension one. The points $x^1 = [0, 1]'$ and $x^2 = [1, 0]'$ belong to the set $\mathcal{F}$. The point $\bar{x} = -0.2x^1 + 1.2x^2 = [1.2, -0.2]'$ is an affine combination of points x^1 and x^2. The affine hull of x^1 and x^2, aff$(\{x^1, x^2\})$, is the set $\mathcal{F}$.

Convex sets have been defined in Section 1.2

The **convex combination** of a finite set of points $x^1, \dots, x^k$ belonging to $\mathbb{R}^n$ is defined as the point $\lambda^1 x^1 + \dots + \lambda^k x^k$ where $\sum_{i=1}^k \lambda^i = 1$ and $\boxed{\lambda^i \geq 0,}$ $i = 1, \dots, k$

The **convex hull** of a set $\mathcal{K} \subseteq \mathbb{R}^n$ is the set of all convex combinations of points in $\mathcal{K}$ and it is denoted as conv$(\mathcal{K})$:

$$\text{conv}(\mathcal{K}) \triangleq \{\lambda^1 x^1 + \dots + \lambda^k x^k \ : \ x_i \in \mathcal{K}, \boxed{\lambda^i \geq 0}, \ i = 1, \dots, k, \ \sum_{i=1}^k \lambda^i = 1\}. \tag{3.3}$$

The convex hull of $\mathcal{K}$ is the smallest convex set that contains $\mathcal{K}$, in the following sense: if $\mathcal{S}$ is any convex set with $\mathcal{K} \subseteq \mathcal{S}$, then conv$(\mathcal{K}) \subseteq \mathcal{S}$.

Example 3.2. Consider three points $x^1 = [1, 1]'$, $x^2 = [1, 0]'$, $x^3 = [0, 1]'$ in $\mathbb{R}^2$. The point $\bar{x} = \lambda^1 x^1 + \lambda^2 x^2 + \lambda^3 x^3$ with $\lambda^1 = 0.2$, $\lambda^2 = 0.2$, $\lambda^3 = 0.6$ is $\bar{x} = [0.4, 0.8]'$ and it is a convex combination of the points $\{x^1, \ x^2, \ x^3\}$. The convex hull of $\{x^1, \ x^2, \ x^3\}$ is the triangle plotted in Figure 3.1. Note that any set in $\mathbb{R}^2$ strictly contained in the triangle and containing $\{x^1, \ x^2, \ x^3\}$ is non-convex.

A **cone** spanned by a finite set of points $\mathcal{K} = \{x^1, \dots, x^k\}$ is defined as

$$\text{cone}(\mathcal{K}) = \{\sum_{i=1}^k \lambda^i x^i, \ \lambda^i \geq 0, i = 1, \dots, k\}. \tag{3.4}$$

We define cone$(\mathcal{K}) = \{0\}$ if $\mathcal{K}$ is the empty set.

Example 3.3. Consider three points $x^1 = [1, 1, 1]'$, $x^2 = [1, 2, 1]'$, $x^3 = [1, 1, 2]'$ in $\mathbb{R}^3$. The cone spanned by of $\{x^1, \ x^2, \ x^3\}$ is an unbounded set. Its restriction to the box of size 10 is depicted in Figure 3.2.

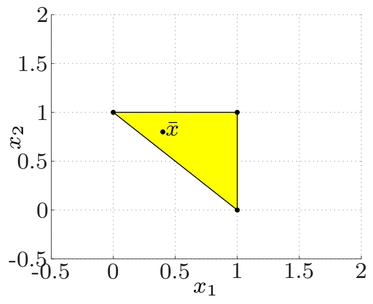


Fig. 3.1 Illustration of a convex hull of three points $x^1 = [1, 1]'$, $x^1 = [1, 0]'$, $x^3 = [0, 1]'$

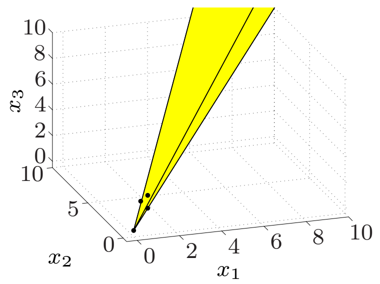


Fig. 3.2 Illustration of a cone spanned by the three points $x^1 = [1, 1, 1]'$, $x^2 = [1, 2, 1]'$, $x^3 = [1, 1, 2]'$

The **Minkowski sum** of two sets $\mathcal{P}, \mathcal{Q} \subseteq \mathbb{R}^n$ is defined as

$$\mathcal{P} \oplus \mathcal{Q} \triangleq \{x + y \ : \ x \in \mathcal{P}, \ y \in \mathcal{Q}\}. \tag{3.5}$$

By definition, any point in $\mathcal{P} \oplus \mathcal{Q}$ can be written as the sum of two points, one in $\mathcal{P}$ and one in $\mathcal{Q}$. For instance, the Minkowski sum of two balls ($\mathcal{P}$ and $\mathcal{Q}$) centered in the origin and of radius 1, is a ball ($\mathcal{P} \oplus \mathcal{Q}$) centered in the origin and of radius 2.

3.2 Polyhedra Definitions and Representations

In the following we give two definitions of a polyhedron. They are mathematically (but not algorithmically) equivalent. The proof of equivalence is not trivial and can be found in [105].

An *$\mathcal{H}$-polyhedron* $\mathcal{P}$ in $\mathbb{R}^n$ denotes an intersection of a finite set of closed halfspaces in $\mathbb{R}^n$:

$$\mathcal{P} = \{x \in \mathbb{R}^n : \ Ax \leq b\} \tag{3.6}$$

where $Ax \leq b$ is the usual shorthand form for a system of inequalities, namely $a_i x \leq b_i,\ i = 1, \ldots, m$, where $a_1, \ldots, a_m$ are the rows of A, and $b_1, \ldots, b_m$ are the components of b. In Figure 3.3 a two-dimensional $\mathcal{H}$-polyhedron is plotted.

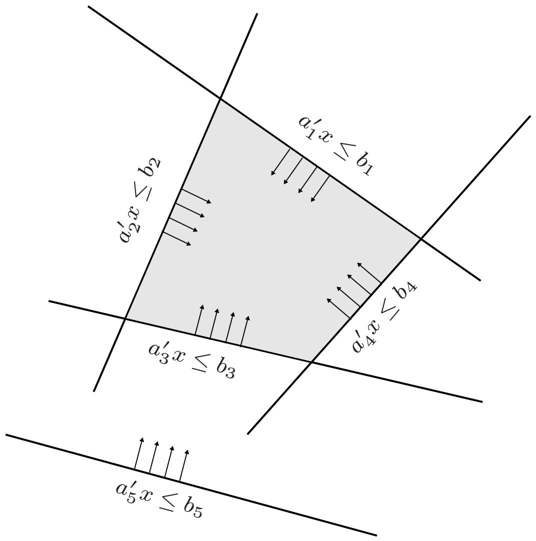


Fig. 3.3 $\mathcal{H}$-polyhedron

A $\mathcal{V}$-*polyhedron* $\mathcal{P}$ in $\mathbb{R}^n$ denotes the Minkowski sum of the convex hull of a finite set of points $\{V_1, \ldots, V_k\}$ of $\mathbb{R}^n$ and the cone generated by a finite set of vectors $\{y_1, \ldots, y_{k'}\}$ of $\mathbb{R}^n$:

$$\mathcal{P} = \mathrm{conv}(V) \oplus \mathrm{cone}(Y) \tag{3.7}$$

for some $V = [V_1, \ldots, V_k] \in \mathbb{R}^{n \times k}$, $Y = [y_1, \ldots, y_{k'}] \in \mathbb{R}^{n \times k'}$. The main theorem for polyhedra states that any $\mathcal{H}$-polyhedron is a $\mathcal{V}$-polyhedron and vice-versa [105](pag. 30).

An $\mathcal{H}$-*polytope* is a bounded $\mathcal{H}$-polyhedron (in the sense that it does not contain any ray $\{x + ty\ :\ t \geq 0\}$). A $\mathcal{V}$-*polytope* is a bounded $\mathcal{V}$-polyhedron

$$\mathcal{P} = \mathrm{conv}(V) \tag{3.8}$$

The main theorem for polytopes states that any $\mathcal{H}$-polytope is a $\mathcal{V}$-polytope and vice-versa [105](pag. 29).

The dimension of a polytope (polyhedron) $\mathcal{P}$ is the dimension of its affine hull and is denoted by $\dim(\mathcal{P})$. We say that a polytope $\mathcal{P} \subset \mathbb{R}^n$, $\mathcal{P} = \{x \in \mathbb{R}^n\ :\ P^x x \leq P^c\}$, is *full-dimensional* if $\dim(\mathcal{P}) = n$ or, equivalently, if it is possible to fit a nonempty n-dimensional ball in $\mathcal{P}$,

$$\exists x \in \mathbb{R}^n, \epsilon > 0 \ : \ \mathcal{B}(x,\epsilon) \subset \mathcal{P}, \tag{3.9}$$

or, equivalently,

$$\exists x \in \mathbb{R}^n, \ \epsilon > 0 \ : \|\delta\|_2 \leq \epsilon \Rightarrow P^x(x+\delta) \leq P^c. \tag{3.10}$$

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

$$\nexists x \in \mathbb{R}^n : P^x x \leq P^c. \tag{3.11}$$

Furthermore, if $\|P_i^x\|_2 = 1$, where P_i^x denotes the i-th row of a matrix P^x, we say that the polytope $\mathcal{P}$ is *normalized.*

Let $\mathcal{P}$ be a polyhedron. A linear inequality $c'z \leq c_0$ is said to be *valid* for $\mathcal{P}$ if it is satisfied for all points $z \in \mathcal{P}$. A *face* of $\mathcal{P}$ is any nonempty set of the form

$$\mathcal{F} = \mathcal{P} \cap \{z \in \mathbb{R}^s : \ c'z = c_0\} \tag{3.12}$$

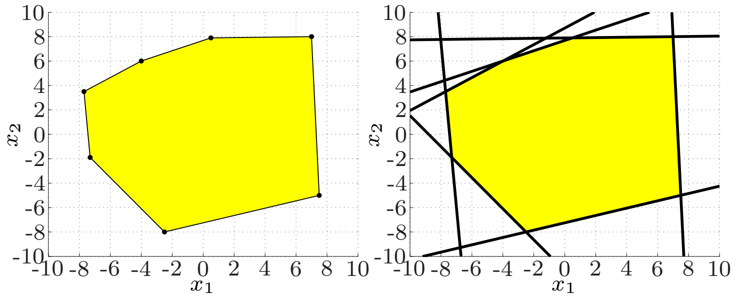
where $c'z \leq c_0$ is a *valid* inequality for $\mathcal{P}$. The *dimension* of a face is the dimension of its affine hull. For the valid inequality $0z \leq 0$ we get that $\mathcal{P}$ is a face of $\mathcal{P}$. All faces of $\mathcal{P}$ satisfying $\mathcal{F} \subset \mathcal{P}$ are called proper faces and have dimension less than dim($\mathcal{P}$). The faces of dimension 0,1, dim($\mathcal{P}$)-2 and dim($\mathcal{P}$)-1 are called *vertices*, *edges*, *ridges*, and *facets*, respectively. The next proposition summarizes basic facts about faces.

Proposition 3.1. *Let $\mathcal{P}$ be a polytope, V the set of all its vertices and $\mathcal{F}$ a face.*

1. *$\mathcal{P}$ is the convex hull of its vertices: $\mathcal{P}$=conv(V).*
2. *$\mathcal{F}$ is a polytope.*
3. *Every intersection of faces of $\mathcal{P}$ is a face of $\mathcal{P}$.*
4. *The faces of $\mathcal{F}$ are exactly the faces of $\mathcal{P}$ that are contained in $\mathcal{F}$.*
5. *The vertices of $\mathcal{F}$ are exactly the vertices of $\mathcal{P}$ that are contained in $\mathcal{F}$.*

A *d-simplex* is a polytope of $\mathbb{R}^d$ with $d+1$ vertices.

In this book we will work mostly with $\mathcal{H}$-polyhedra and $\mathcal{H}$-polytopes. This choice has important consequences for algorithms implementation and their complexity. As a simple example, a unit cube in $\mathbb{R}^d$ can be described through $2d$ equalities as an $\mathcal{H}$-polytope, but requires 2^d points in order to be described as a $\mathcal{V}$-polytope. We want to mention here that many efficient algorithms that work on polyhedra require both $\mathcal{H}$ and $\mathcal{V}$ representations. In Figure 3.4 the $\mathcal{H}$-representation and the $\mathcal{V}$-representation of the same polytope in two dimensions are plotted. Consider Figure 3.3 and notice that the fifth inequality can be removed without changing the polyhedron. Inequalities which can be removed without changing the polyhedron described by the original set are called *redundant.* The representation of an $\mathcal{H}$-polyhedron is *minimal* if it does not contain redundant inequalities. Detecting whether an inequality is redundant for an $\mathcal{H}$-polyhedron requires solving a linear program, as described in Section 3.4.4.


(a) $\mathcal{V}$-representation. The vertices $V_1^P, \dots, V_7^P$ are depicted as dots.

(b) $\mathcal{H}$-representation. The hyperplanes $P_i^x x = P_i^c$, $i = 1, \dots, 7$ are depicted as lines.

Fig. 3.4 Illustration of a polytope in $\mathcal{H}$- and $\mathcal{V}$- representation.

By definition a polyhedron is a closed set. In this book we will also work with sets which are not closed but whose closure is a polyhedron. For this reason the two following non-standard definitions will be useful.

Definition 3.1 (Open Polyhedron (Polytope)). A set $\mathcal{C} \subseteq \mathbb{R}^n$ is called *open polyhedron (polytope)* if it is open and its closure is a polyhedron (polytope).

Definition 3.2 (Neither Open nor Closed Polyhedron (Polytope)). A set $\mathcal{C} \subseteq \mathbb{R}^n$ is called *neither open nor closed polyhedron (polytope)* if it is neither open nor closed and its closure is a polyhedron (polytope).

3.3 Polytopal Complexes

According to our definition every polytope represents a convex, compact (i.e., bounded and closed) set. In this book we will 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 concepts.

Definition 3.3 (P-collection). A set $\mathcal{C} \subseteq \mathbb{R}^n$ is called a *P-collection* (in $\mathbb{R}^n$) if it is a collection of a finite number of n-dimensional polytopes, i.e.

$$\mathcal{C} = \{\mathcal{C}_i\}_{i=1}^{N_C}, \tag{3.13}$$

where $\mathcal{C}_i := \{x \in \mathbb{R}^n \ : \ C_i^x x \leq C_i^c\}$, $\dim(\mathcal{C}_i) = n$, $i = 1, \dots, N_C$, with $N_C < \infty$. $\square$

Definition 3.4 (Underlying Set). The *underlying set* of a P-collection $\mathcal{C} = \{\mathcal{C}_i\}_{i=1}^{N_C}$ is the

$$\underline{\mathcal{C}} := \bigcup_{\mathcal{P}\in\mathcal{C}} \mathcal{P} = \bigcup_{i=1}^{N_C} \mathcal{C}_i. \tag{3.14}$$

□

Example 3.4. A collection $\mathcal{R} = \{[-2,-1],[0,2],[2,4]\}$ is a P-collection in $\mathbb{R}^1$ with the underlying set $\underline{\mathcal{R}} = [-2,-1] \cup [0,4]$. As another example, $\mathcal{R} = \{[-2,0],[-1,1],[0,2]\}$ is a P-collection in $\mathbb{R}^1$ with underlying set $\underline{\mathcal{R}} = [-2,2]$. Clearly, polytopes that define a P-collection can overlap, while the underlying set can be disconnected and non-convex. □

Usually it is clear from the context if we are talking about the P-collection or referring to the underlying set of a P-collection. Therefore for simplicity, we use the same notation for both.

Definition 3.5 (Strict Polyhedral Partition). A collection of sets $\{\mathcal{C}_i\}_{i=1}^{N_C}$ is a *strict partition* of a set $\mathcal{C}$ if *(i)* $\bigcup_{i=1}^{N_C} \mathcal{C}_i = \mathcal{C}$ and *(ii)* $\mathcal{C}_i \cap \mathcal{C}_j = \emptyset$, $\forall i \neq j$. Moreover $\{\mathcal{C}_i\}_{i=1}^{N_C}$ is a *strict polyhedral partition* of a polyhedral set $\mathcal{C}$ if $\{\mathcal{C}_i\}_{i=1}^{N_C}$ is a strict partition of $\mathcal{C}$ and $\bar{\mathcal{C}}_i$ is a polyhedron for all i, where $\bar{\mathcal{C}}_i$ denotes the closure of the set $\mathcal{C}_i$.

Definition 3.6 (Polyhedral Partition). A collection of sets $\{\mathcal{C}_i\}_{i=1}^{N_C}$ is a *partition* of a set $\mathcal{C}$ if *(i)* $\bigcup_{i=1}^{N_C} \mathcal{C}_i = \mathcal{C}$ and *(ii)* $(\mathcal{C}_i \backslash \partial\mathcal{C}_i) \cap (\mathcal{C}_j \backslash \partial\mathcal{C}_j) = \emptyset$, $\forall i \neq j$. Moreover $\{\mathcal{C}_i\}_{i=1}^{N_C}$ is a *polyhedral partition* of a polyhedral set $\mathcal{C}$ if $\{\mathcal{C}_i\}_{i=1}^{N_C}$ is a partition of $\mathcal{C}$ and $\mathcal{C}_i$ is a polyhedron for all i. The set $\partial\mathcal{C}_j$ is the boundary of the set $\mathcal{C}_j$.

Note that in a *strict partition* some of the sets C_i must be open or neither open nor closed. In a *partition* all the sets may be closed and points on the closure of a particular set may also belong to one or several other sets. Also, note that a polyhedral partition is a special class of P-collection.

3.3.1 Functions on Polytopal Complexes

Definition 3.7. A function $h(\theta) : \Theta \to \mathbb{R}^k$, where $\Theta \subseteq \mathbb{R}^s$, is *piecewise affine (PWA)* if there exists a strict partition $R_1,\dots,R_N$ of Θ and $h(\theta) = H^i\theta + k^i$, $\forall \theta \in R_i$, $i = 1,\dots,N$.

Definition 3.8. A function $h(\theta) : \Theta \to \mathbb{R}^k$, where $\Theta \subseteq \mathbb{R}^s$, is *piecewise affine on polyhedra* (PPWA) if there exists a strict polyhedral partition $R_1,\dots,R_N$ of Θ and $h(\theta) = H^i\theta + k^i$, $\forall \theta \in R_i$, $i = 1,\dots,N$.

Definition 3.9. A function $h(\theta) : \Theta \to \mathbb{R}$, where $\Theta \subseteq \mathbb{R}^s$, is *piecewise quadratic (PWQ)* if there exists a strict partition R_1,…,R_N of Θ and $h(\theta) = \theta' H^i\theta + k^i\theta + l^i$, $\forall\theta \in R_i$, $i = 1,\dots,N$.

Definition 3.10. A function $h(\theta) : \Theta \to \mathbb{R}$, where $\Theta \subseteq \mathbb{R}^s$, is *piecewise quadratic on polyhedra* (PPWQ) if there exists a strict polyhedral partition R_1,…,R_N of Θ and $h(\theta) = \theta' H^i\theta + k^i\theta + l^i$, $\forall\theta \in R_i$, $i = 1,\dots,N$.

As long as the function $h(\theta)$ we are defining is continuous it is not important if the partition constituting the domain of the function is strict or not. If the function is discontinuous at points on the closure of a set, then this function can only be defined if the partition is strict. Otherwise we may obtain several conflicting definitions of function values (or set-valued functions). Therefore for the statement of theoretical results involving discontinuous functions we will always assume that the partition is strict. For notational convenience, however, when working with continuous functions we will make use of partitions rather than strict partitions.

3.4 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 [270, 126, 102]. All operations and functions described in this chapter are contained in the MPT toolbox [167, 166].

3.4.1 Convex Hull

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

$$\text{conv}(V) = \{x \in \mathbb{R}^n \ : x = \sum_{i=1}^{N_V} \alpha^i V^i,\ 0 \le \alpha^i \le 1,\ \sum_{i=1}^{N_V} \alpha^i = 1\}. \tag{3.15}$$

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

$$\text{conv}\left(\bigcup_{i=1}^{N_R} \mathcal{R}_i\right) := \{x \in \mathbb{R}^n \ : \ x = \sum_{i=1}^{N_R} \alpha^i x^i,\ x^i \in \mathcal{R}_i,\ 0 \le \alpha^i \le 1,\ \sum_{i=1}^{N_R} \alpha^i = 1\}. \tag{3.16}$$

An illustration of the convex hull operation is given in Figure 3.5. Construction of the convex hull of a set of polytopes is an expensive operation which is exponential in the number of facets of the original polytopes. An efficient software implementation is available in [101].

3.4.2 Envelope

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

$$\mathrm{env}(\mathcal{P}, \mathcal{Q}) = \{x \in \mathbb{R}^n \ : \ \bar{P}^x x \le \bar{P}^c, \ \bar{Q}^x x \le \bar{Q}^c\}, \tag{3.17}$$

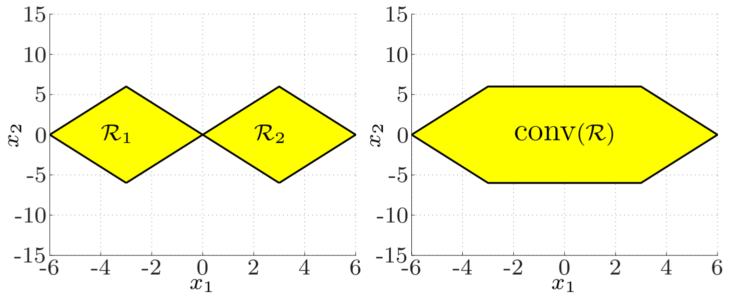
where $\bar{P}^x x \le \bar{P}^c$ is the subsystem of $P^x x \le P^c$ obtained by removing all the inequalities not valid for the polyhedron $\mathcal{Q}$, and $\bar{Q}^x x \le \bar{Q}^c$ is defined in a similar way with respect to $Q^x x \le Q^c$ and $\mathcal{P}$ [33]. In a similar fashion, the definition can be extended to the case of the envelope of a P-collection. An illustration of the envelope operation is depicted in Figure 3.6. The computation of the envelope is relatively cheap since it only requires the solution of one LP for each facet of $\mathcal{P}$ and $\mathcal{Q}$. Note that the envelope of two (or more) polytopes is not necessarily a bounded set (e.g. when $\mathcal{P} \cup \mathcal{Q}$ is shaped like a star).

3.4.3 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 of the convex hull operation and the algorithmic implementation is identical to a convex hull computation, i.e., given a set of extreme points $V = \{V_i\}_{i=1}^{N_V} = \mathrm{vert}(\mathcal{P})$ of a polytope $\mathcal{P}$ given in $\mathcal{H}$-representation it holds that $\mathcal{P} = \mathrm{conv}(V)$, where the operator vert denotes the vertex enumeration. The necessary computational effort is exponential in the number of input facets. Two different approaches for vertex enumeration are commonly used: the double description method [104] and reverse search [11]. An efficient implementation of the double description method is available in [101].

3.4.4 Minimal Representation

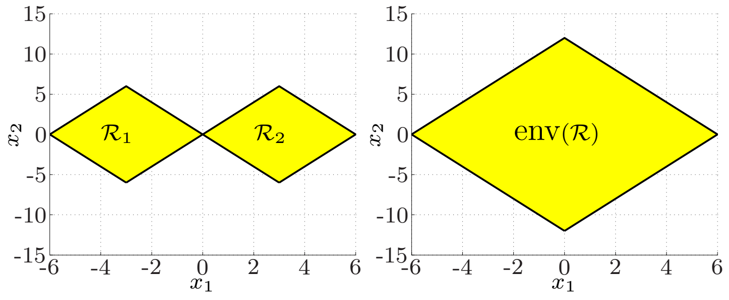
We say that a polytope $\mathcal{P} \subset \mathbb{R}^n$, $\mathcal{P} = \{x \in \mathbb{R}^n \ : \ P^x x \le P^c\}$ is in a *minimal representation* if the removal of any row in $P^x x \le P^c$ would change it (i.e., if there are no redundant constraints). The computation of the minimal



(a) P-collection $\mathcal{R} = \bigcup_{i\in\{1,2\}} \mathcal{R}_i$.

(b) Convex hull of $\mathcal{R}$.

Fig. 3.5 Illustration of the convex hull operation.



(a) P-collection $\mathcal{R} = \bigcup_{i\in\{1,2\}} \mathcal{R}_i$.

(b) Envelope env($\mathcal{R}$).

Fig. 3.6 Illustration of the envelope operation.

representation (henceforth referred to as *polytope reduction*) of polytopes is discussed in [102] and generally requires to solve one LP for each half-space defining the non-minimal representation of $\mathcal{P}$. We summarize this simple implementation of the polytope reduction in Algorithm 3.1. An improved algorithm for polytope reduction is discussed in [244] where the authors combine the 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. Note that 'unique' here means that for $\mathcal{P} := \{x \in \mathbb{R}^n \ : \ P^x x \leq P^c\}$ the matrix $[P^x P^c]$ consists of the unique set of row vectors, the rows order is irrelevant. 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   $\mathcal{P} = \{x \ : \ P^x x \le P^c\}$, with $P^x \in \mathbb{R}^{n_P \times n}$, $P^c \in \mathbb{R}^{n_P}$
 OUTPUT $\mathcal{Q} = \{x \ : \ Q^x x \le Q^c\} := minrep(\mathcal{P})$
   LET $\mathcal{I} \leftarrow \{1, \dots, n_P\}$
   FOR $i = 1$ TO $n_P$
   LET $\mathcal{I} \leftarrow \mathcal{I} \setminus \{i\}$
   LET $f^* \leftarrow \max_x P_i^x x, \quad subj. \ to \quad P^x_{(\mathcal{I})} x \le P^c_{(\mathcal{I})}, \ P_i^x x \le P_i^c + 1$
   IF $f^* > P_i^c$ THEN $\mathcal{I} \leftarrow \mathcal{I} \cup \{i\}$
END
LET $Q^x = P^x_{(\mathcal{I})}$, $Q^c = P^c_{(\mathcal{I})}$
```
□

Remark 3.1 (Full-dimensional polytopes). Throughout this book 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 book. For the same reason the MPT toolbox [167] only deals with full-dimensional polytopes. Polyhedra and lower-dimensional polytopes (with the exception of the empty polytope) are not considered. □

3.4.5 Chebychev Ball

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

$$\max_{x_c, R} R \tag{3.18a}$$

$$\text{subj. to} \quad P_i^x x_c + R\|P_i^x\|_2 \le P_i^c, \quad i = 1, \dots, n_P, \tag{3.18b}$$

where P_i^x denotes the i-th row of P^x. This can be proven as follows. Any point x of the ball can be written as $x = x_c + v$ where v is a vector of length less or equal to R. Therefore the center and radius of the Chebychev ball can be found by solving the following optimization problem

$$\max_{x_c, R} R \tag{3.19a}$$

$$\text{subj. to} \quad P_i^x (x_c + v) \le P_i^c, \quad \forall \ v \ \text{ such that } \|v\|_2 \le R, \ i = 1, \dots, n_P, \tag{3.19b}$$

Consider the i-th constraint

$$P_i^x(x_c + v) \le P_i^c, \quad \forall\, v \ \text{ such that } \|v\|_2 \le R$$

This can be written as

$$P_i^x x_c \le P_i^c - P_i^x v, \quad \forall\, v \ \text{ such that } \|v\|_2 \le R \tag{3.20}$$

Constraint (3.20) is satisfied $\forall\, v$ such that $\|v\|_2 \le R$ if and only if it is satisfied at $v = \frac{P_i^{x\prime}}{\|P_i^x\|_2} R$. Therefore we can rewrite the optimization problem (3.19) as the linear program (3.18).

If the radius obtained by solving (3.18) is $R = 0$, 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 one linear program. Furthermore, for a full-dimensional polytope we also get a point x_c that is in the strict interior of $\mathcal{P}$. However, the center of a Chebyshev Ball x_c in (3.18) is not necessarily a 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 involve the solution of Semi-Definite Programs (SDPs) and therefore they are more expensive than the Chebyshev Ball computation [57]. An illustration of the Chebyshev Ball is given in Figure 3.7.

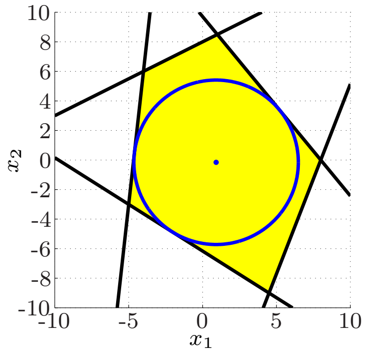


Fig. 3.7 Illustration of the Chebychev ball contained in a polytope $\mathcal{P}$.

3.4.6 Projection

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

$$\text{proj}_x(\mathcal{P}) := \{x \in \mathbb{R}^n \ : \ \exists y \in \mathbb{R}^m : P^x x + P^y y \le P^c\}. \tag{3.21}$$

An illustration of a projection operation is given in Figure 3.8. Current pro-

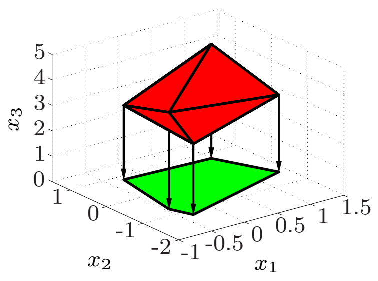


Fig. 3.8 Illustration of a projection of a 3-dimensional polytope $\mathcal{P}$ onto a plane.

jection methods that can operate in general dimensions can be grouped into four classes: Fourier elimination [75, 156], block elimination [14], vertex based approaches [103] and wrapping-based techniques [149]. For a good introduction to projection, we refer the reader to [149] and the references therein.

3.4.7 Set-Difference

The set-difference of two polytopes $\mathcal{Y}$ and $\mathcal{R}_0$

$$\mathcal{R} = \mathcal{Y} \setminus \mathcal{R}_0 := \{x \in \mathbb{R}^n \ : \ x \in \mathcal{Y}, x \notin \mathcal{R}_0\}, \tag{3.22}$$

in general, can be a nonconvex and disconnected set and can be described as a P-collection $\mathcal{R} = \bigcup_{i=1}^m \mathcal{R}_i$, where $\mathcal{Y} = \bigcup_{i=1}^m \mathcal{R}_i \bigcup (\mathcal{R}_0 \bigcap \mathcal{Y})$. The P-collection $\mathcal{R} = \bigcup_{i=1}^m \mathcal{R}_i$ can be computed by consecutively inverting the half-spaces defining $\mathcal{R}_0$ as described in the following Theorem 3.1.

Note that here we use the term P-collection in the dual context of both P-collection and its underlying set (cf. Definitions 3.3 and 3.4). The precise statement would say that $\underline{\mathcal{R}} = \mathcal{Y} \setminus \mathcal{R}_0$, where $\underline{\mathcal{R}}$ is underlying set of the P-collection $\mathcal{R} = \{\mathcal{R}_i\}_{i=1}^m$. However, whenever it is clear from context, we will use the former, more compact form.

Theorem 3.1. *[39] Let $\mathcal{Y} \subseteq \mathbb{R}^n$ be a polyhedron, $\mathcal{R}_0 \triangleq \{x \in \mathbb{R}^n : Ax \leq b\}$, and $\bar{\mathcal{R}}_0 \triangleq \{x \in \mathcal{Y} : Ax \leq b\} = \mathcal{R}_0 \bigcap \mathcal{Y}$, where $b \in \mathbb{R}^m$, $\mathcal{R}_0 \neq \emptyset$ and $Ax \leq b$ is a minimal representation of $\mathcal{R}_0$. Also let*

$$\mathcal{R}_i = \left\{ x \in \mathcal{Y} : \begin{array}{l} A^i x > b^i \\ A^j x \leq b^j, \forall j < i \end{array} \right\} i = 1, \ldots, m$$

Let $\mathcal{R} \triangleq \bigcup_{i=1}^m \mathcal{R}_i$. Then, $\mathcal{R}$ is a P-collection and $\{\bar{\mathcal{R}}_0, \mathcal{R}_1, \ldots, \mathcal{R}_m\}$ is a strict polyhedral partition of $\mathcal{Y}$.

Proof: (i) We want to prove that given an $x \in \mathcal{Y}$, then either x belongs to $\bar{\mathcal{R}}_0$ or to $\mathcal{R}_i$ for some i. If $x \in \bar{\mathcal{R}}_0$, we are done. Otherwise, there exists an index i such that $A^i x > b^i$. Let $i^* = \min_{i \le m}\{i : A^i x > b^i\}$. Then $x \in \mathcal{R}_{i^*}$, as $A^{i^*} x > b^{i^*}$ and $A^j x \le b^j$, $\forall j < i^*$, by definition of i^*.

(ii) Let $x \in \bar{\mathcal{R}}_0$. Then there does not exist any i such that $A^i x > b^i$, which implies that $x \notin \mathcal{R}_i$, $\forall i \le m$. Let $x \in \mathcal{R}_i$ and take $i > j$. Because $x \in \mathcal{R}_i$, by definition of $\mathcal{R}_i$ $(i > j)$ $A^j x \le b^j$, which implies that $x \notin \mathcal{R}_j$. □

As an illustration for the procedure proposed in Theorem 3.1 consider the two-dimensional case depicted in Figure 3.9(a). Here $\mathcal{X}$ is defined by the inequalities $\{x_1^- \le x_1 \le x_1^+,\ x_2^- \le x_2 \le x_2^+\}$, and $\mathcal{R}_0$ by the inequalities $\{g_1 \le 0,\ \ldots, g_5 \le 0\}$ 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 $g_1 \le 0$, the first set of the rest of the region $\mathcal{X} \backslash \mathcal{R}_0$ is given by $\mathcal{R}_1 = \{g_1 \ge 0,\ x_1 \ge x_1^-,\ x_2^- \le x_2 \le x_2^+\}$, which is obtained by reversing the sign of the inequality $g_1 \le 0$ and removing redundant constraints in $\mathcal{X}$ (see Figure 3.9(b)). Thus, by considering the rest of the inequalities we get the partition of the rest of the parameter space $\mathcal{X} \backslash \mathcal{R}_0 = \bigcup_{i=1}^{5} \mathcal{R}_i$, as reported in Figure 3.9(d).

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 $\mathcal{R}_i$, thus specifying if they are open or closed, we are not doing so in the algorithms described in this book nor in MPT [167, 166], cf. Remark 3.1. In computations, we will henceforth only consider the closure of the sets $\mathcal{R}_i$.

The set difference between two P-collections $\mathcal{P}$ and $\mathcal{Q}$ can be computed as described in [20, 125, 220].

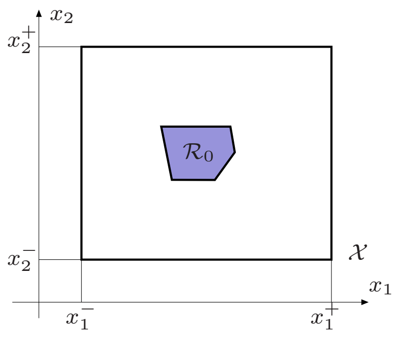
3.4.8 Pontryagin Difference

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

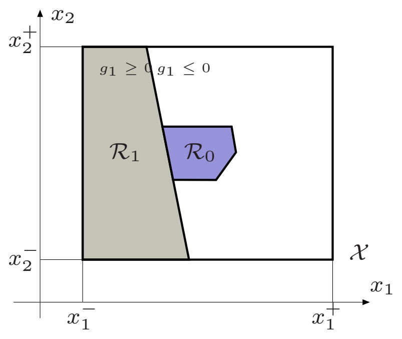
$$\mathcal{P} \ominus \mathcal{Q} := \{x \in \mathbb{R}^n \ :\ x + q \in \mathcal{P},\ \forall q \in \mathcal{Q}\}. \tag{3.23}$$

The Pontryagin difference can be efficiently computed for polytopes by solving a sequence of LPs as follows. Define the $\mathcal{P}$ and $\mathcal{Q}$ as

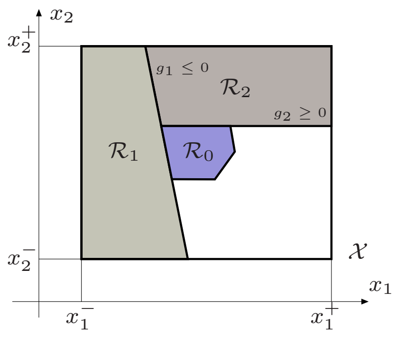
$$\mathcal{P} = \{y \in \mathbb{R}^n \ :\ , P^y y \le P^b\}, \qquad \mathcal{Q} = \{z \in \mathbb{R}^n :\ Q^z z \le Q^b\}, \tag{3.24}$$



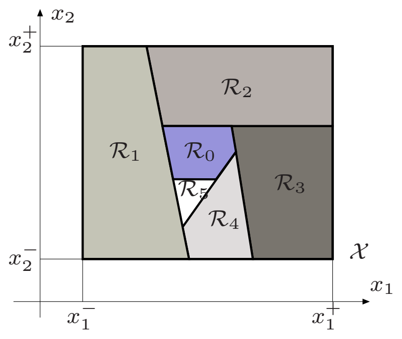
(a) Set of parameters $\mathcal{X}$ and initial set $\mathcal{R}_0$



(b) Partition of $\mathcal{X} \backslash \mathcal{R}_0$ - Step 1



(c) Partition of $\mathcal{X} \backslash \mathcal{R}_0$ - Step 2



(d) Final partition of $\mathcal{X} \backslash \mathcal{R}_0$

Fig. 3.9 Two dimensional example: partition of the rest of the space $\mathcal{X} \backslash \mathcal{R}_0$.

then

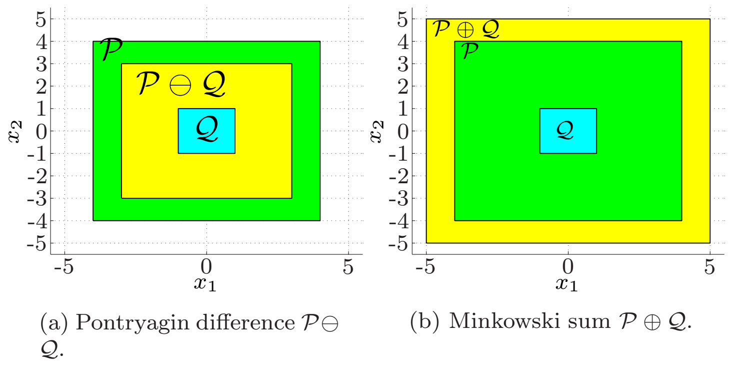
$$\mathcal{W} = \mathcal{P} \ominus \mathcal{Q} \tag{3.25a}$$

$$= \{x \in \mathbb{R}^n \ : \ P^y x \leq P^b - H(P^y, \mathcal{Q})\} \tag{3.25b}$$

where the i-th element of $H(P^y, \mathcal{Q})$ is

$$H_i(P^y, \mathcal{Q}) \triangleq \max_{x \in \mathcal{Q}} P_i^y x \tag{3.26}$$

and P_i^y is the i-th row of the matrix P^y. Note that For special cases (e.g. when $\mathcal{Q}$ is a hypercube), even more efficient computational methods exist [159]. An illustration of the Pontryagin difference is given in Figure 3.10a.



(a) Pontryagin difference $\mathcal{P} \ominus \mathcal{Q}$.

(b) Minkowski sum $\mathcal{P} \oplus \mathcal{Q}$.

Fig. 3.10 Illustration of the Pontryagin difference and Minkowski sum operations.

3.4.9 Minkowski Sum

The Minkowski sum of two polytopes $\mathcal{P}$ and $\mathcal{Q}$ is a polytope

$$\mathcal{P} \oplus \mathcal{Q} := \{x + q \in \mathbb{R}^n \ : \ x \in \mathcal{P}, \ q \in \mathcal{Q}\}. \tag{3.27}$$

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.

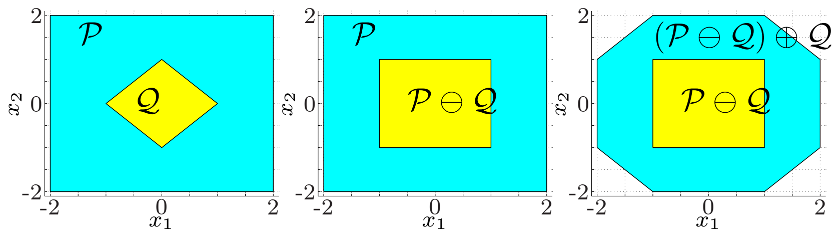
$$P = \{y \in \mathbb{R}^n \ : \ P^y y \leq P^c\}, \quad \mathcal{Q} = \{z \in \mathbb{R}^n \ : \ Q^z z \leq Q^c\},$$

it holds that

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

Both the projection and vertex enumeration based methods are implemented in the MPT toolbox [167]. An illustration of the Minkowski sum is given in Figure 3.10b.

Remark 3.3. The Minkowski sum is **not** the complement of the Pontryagin difference. For two polytopes $\mathcal{P}$ and $\mathcal{Q}$, it holds that $(\mathcal{P} \ominus \mathcal{Q}) \oplus \mathcal{Q} \subseteq \mathcal{P}$. This is illustrated in Figure 3.11.


$\mathcal{P}$
$\mathcal{Q}$
$\mathcal{P} \ominus \mathcal{Q}$
$(\mathcal{P} \ominus \mathcal{Q}) \oplus \mathcal{Q}$
x_1
x_2
-2
0
2

(a) Two polytopes $\mathcal{P}$ and $\mathcal{Q}$.

(b) Polytope $\mathcal{P}$ and Pontryagin difference $\mathcal{P} \ominus \mathcal{Q}$.

(c) Polytope $\mathcal{P} \ominus \mathcal{Q}$ and the set $(\mathcal{P} \ominus \mathcal{Q}) \oplus \mathcal{Q}$.

Fig. 3.11 Illustration that $(\mathcal{P} \ominus \mathcal{Q}) \oplus \mathcal{Q} \subseteq \mathcal{P}$.

3.4.10 Polytope Union

Consider the following basic problem in polyhedral computation: given two polyhedra $\mathcal{P} \subset \mathbb{R}^n$ and $\mathcal{Q} \subset \mathbb{R}^n$, decide whether their union is convex, and, if so, compute it. There are three classes of algorithms for the given problem depending on their representation: (1) $\mathcal{P}$ and $\mathcal{Q}$ are given in $\mathcal{H}$-representation, (2) $\mathcal{P}$ and $\mathcal{Q}$ are given in $\mathcal{V}$-representation, (3) both $\mathcal{H}$- and $\mathcal{V}$-representations are available for $\mathcal{P}$ and $\mathcal{Q}$. Next we present an algorithm for case (1). Cases (2), case (3) and the computational complexity of all three cases are discussed in [33].

The following basic lemma, whose proof naturally follows from the definition of convexity, will be used in the sequel to prove our main results.

Lemma 3.2. *Let $\mathcal{P}$ and $\mathcal{Q}$ be convex polyhedra with $\mathcal{V}$-representations conv$(V) \oplus$ cone(R) and conv$(W) \oplus$ cone(S), respectively. Then $\mathcal{P} \cup \mathcal{Q}$ is convex if and only if $\mathcal{P} \cup \mathcal{Q}$ is a convex polyhedron with $\mathcal{V}$-representation conv$(V \cup W) \oplus$ cone$(R \cup S)$.*

Recall the definition of envelope in Section 3.4.2. By definition, it is easy to see that

$$\mathcal{P} \cup \mathcal{Q} \subseteq \text{env}(\mathcal{P}, \mathcal{Q}). \tag{3.28}$$

Note that the set env$(\mathcal{P}, \mathcal{Q})$ does not depend on the $\mathcal{H}$-representations of $\mathcal{P}$ and $\mathcal{Q}$.

Theorem 3.2. *$\mathcal{P} \cup \mathcal{Q}$ is convex $\Leftrightarrow$ $\mathcal{P} \cup \mathcal{Q} = \text{env}(\mathcal{P}, \mathcal{Q})$.*

Proof: Theorem 3.2 The "$\Leftarrow$" part is trivial, as $\text{env}(\mathcal{P}, \mathcal{Q})$ is a convex set by construction. In order to prove the "$\Rightarrow$" part, assume $\mathcal{K} \triangleq \mathcal{P} \cup \mathcal{Q}$ is convex and without loss of generality $\mathcal{P}$, $\mathcal{Q}$ are full dimensional. The case $\dim(\mathcal{P}) = \dim(\mathcal{Q}) < n$ can be reduced to the case of full dimensional polyhedra in the common embedding subspace. In the case $\dim(\mathcal{P}) < \dim(\mathcal{Q})$, $\mathcal{P} \cup \mathcal{Q}$ convex implies $\mathcal{P} \subset \mathcal{Q}$, and therefore the proof is trivial.

Let $\text{conv}(V) \oplus \text{cone}(R)$ and $\text{conv}(W) \oplus \text{cone}(S)$ be $\mathcal{V}$-representations of $\mathcal{P}$ and $\mathcal{Q}$, respectively. By Lemma 3.2, $\mathcal{K} = \text{conv}(V \cup W) + \text{cone}(R \cup S)$ is a polyhedron, and has an H-representation. As, by (3.28), $\mathcal{K} \subseteq \text{env}(\mathcal{P}, \mathcal{Q})$, it is enough to prove $\text{env}(\mathcal{P}, \mathcal{Q}) \subseteq \mathcal{K}$, by showing that all the inequalities in the unique minimal representation are already in the inequalities representing $\text{env}(\mathcal{P}, \mathcal{Q})$. Suppose there is a facet inequality $r'x \leq s$ for $\mathcal{K}$ that is missing in the H-representation of $\text{env}(\mathcal{P}, \mathcal{Q})$. Let $H = \{x \in \mathbb{R}^n \ : \ r'x = s\}$. Since the inequality is missing in the H-representations of $\mathcal{P}$ and $\mathcal{Q}$, $\dim(\mathcal{P} \cap H) \leq n-2$ and $\dim(\mathcal{Q} \cap H) \leq n-2$ because it is valid for $\mathcal{P}$ and is not in $\text{env}(\mathcal{P}, \mathcal{Q})$. This implies that the facet $\mathcal{K} \cap H$ of $\mathcal{K}$ cannot be the union of two convex sets $\mathcal{P} \cap H$ and $\mathcal{Q} \cap H$, because they have smaller dimensions than $\mathcal{K} \cap H$. This contradicts $\mathcal{K} = \mathcal{P} \cup \mathcal{Q}$. □

Theorem 3.2 can be naturally generalized to the union of k polytopes. Theorem 3.2 represents a result for convexity recognition of the union of two $\mathcal{H}$-polyhedra. It also leads to an algorithm for checking convexity of the union, and for generating an $\mathcal{H}$-representation of the union when it is convex. Such algorithm is presented next.

Algorithm 3.4.1

```
1  Construct $\mathrm{env}(\mathcal{P},\mathcal{Q})$ by removing non-valid constraints
   (see Fig. 3.6),
   let $\tilde{A}x \le \tilde{\alpha}$, $\tilde{B}x \le \tilde{\beta}$ be the set of removed constraints, and
   let $\mathrm{env}(\mathcal{P},\mathcal{Q}) = \{x : \ Cx \le \gamma\}$ the resulting envelope;
2  Remove from $\mathrm{env}(\mathcal{P},\mathcal{Q})$ possible duplicates
   $(B_j, \beta_j) = (\sigma A_i, \sigma\alpha_i)$, $\sigma > 0$;
3  for each pair $\tilde{A}_i x \le \tilde{\alpha}_i$, $\tilde{B}_j x \le \tilde{\beta}_j$ do
4     Determine $\epsilon^*$ by solving the linear program (LP)

         $\epsilon^* = \max_{(x,\epsilon)} \ \epsilon$
               subj. to  $\tilde{A}_i x \ge \tilde{\alpha}_i + \epsilon$
                         $\tilde{B}_j x \ge \tilde{\beta}_j + \epsilon$
                         $Cx \le \gamma$;

       /* $\epsilon^* = -\infty$ if the LP is infeasible,
       $\epsilon^* = \infty$ if the LP is unbounded */
5     if $\epsilon^* > 0$, stop ; return nonconvex;
6  endfor ;
7  return $\mathrm{env}(\mathcal{P},\mathcal{Q})$.    /* $\mathcal{P}\cup\mathcal{Q}$ is convex. */
```

Note that if $\epsilon^* = 0$ for each i, j as defined in step 3, then by Theorem 3.2 the union is convex and equals $\mathrm{env}(\mathcal{P},\mathcal{Q})$. On the other hand, $\epsilon^* > 0$ indicates the existence of a point $x \in \mathrm{env}(\mathcal{P},\mathcal{Q})$ outside $\mathcal{P}\cup\mathcal{Q}$.

For recognizing convexity and computing the union of k polyhedra, the test can be modified by checking each k-tuple of removed constraints. Let $\tilde{m}_1$, …, $\tilde{m}_k$ be the number of removed constrains from the polyhedra $\mathcal{P}_1, \dots, \mathcal{P}_k$, respectively. Then similarly to step 4, $\prod_{i=1}^{k} \tilde{m}_i$ linear programs need to be solved in the general case.

Algorithm 3.4.1 provides an H-representation for $\mathcal{P}\cup\mathcal{Q}$ (step 1). We prove in next Proposition 3.2 that such representation is *minimal*, i.e. it contains no redundant inequalities.

Proposition 3.2. *If $\mathcal{P}$ and $\mathcal{Q}$ are given by minimal H-representation and are full-dimensional then Algorithm 3.4.1 outputs a minimal H-representation of $\mathcal{P}\cup\mathcal{Q}$.*

Proof: Suppose $\mathcal{P}$ and $\mathcal{Q}$ are n-dimensional and given in minimal $\mathcal{H}$-representation. Take any inequality T given by the algorithm. We may assume it comes from the representation of $\mathcal{P}$. By the minimality and full-dimensionality, it is a facet inequality for $\mathcal{P}$. By definition, the facet F determined by T contains n affinely independent points of $\mathcal{P}$. Since these points are also in $\mathcal{P} \cup \mathcal{Q}$, T is a facet inequality for $\mathcal{P} \cup \mathcal{Q}$. By the step 2 of the algorithm, there is no other inequality in the output that determines the same facet F. Therefore the output is minimal. □

Note that the full dimensionality of $\mathcal{P}$ and $\mathcal{Q}$ are necessary in Proposition 3.2.

3.4.11 Affine Mappings and Polyhedra

This section deals with the composition of affine mappings and polyhedra. Consider a polyhedron $\mathcal{P} = \{x \in \mathbb{R}^n \;:\; P^x x \leq P^c\}$, with $P^x \in \mathbb{R}^{n_P \times n}$ and an affine mapping $f(z)$

$$f: \; z \in \mathbb{R}^m \mapsto Az + b, \; A \in \mathbb{R}^{m_A \times m}, \; b \in \mathbb{R}^{m_A} \tag{3.29}$$

Let $m_A = n$, we define the composition of $\mathcal{P}$ and f as the following polyhedron

$$\mathcal{P} \circ f \triangleq \{z \in \mathbb{R}^m \;:\; P^x f(z) \leq P^c\} = \{z \in \mathbb{R}^m \;:\; P^x A z \leq P^c - P^x b\} \tag{3.30}$$

Let $m = n$, we define the composition of f and $\mathcal{P}$ as the following polyhedron

$$f \circ \mathcal{P} \triangleq \{y \in \mathbb{R}^{m_A} \;:\; y = Ax + b \; \forall x \in \mathbb{R}^n, \; P^x x \leq P^c\} \tag{3.31}$$

The polyhedron $f \circ \mathcal{P}$ in (3.31) can be computed as follows. Let us write $\mathcal{P}$ in $\mathcal{V}$-representation

$$\mathcal{P} = \text{conv}(V), \tag{3.32}$$

and let us map the set of vertices $V = \{V_1, \ldots, V_k\}$ through the transformation f. Because the transformation is affine, the set $f \circ \mathcal{P}$ is simply the convex hull of the transformed vertices

$$f \circ \mathcal{P} = \text{conv}(F), \; F = \{AV_1 + b, \ldots, AV_k + b\}. \tag{3.33}$$

The polyhedron $f \circ \mathcal{P}$ in (3.31) can be computed immediately if $m_A = m = n$ and A is invertible. In this case, from the definition in (3.31), $x = A^{-1}y - A^{-1}b$ and therefore

$$f \circ \mathcal{P} = \{y \in \mathbb{R}^{m_A} \;:\; P^x A^{-1} y \leq P^c + P^x A^{-1} b\} \tag{3.34}$$

Vertex enumeration can be avoided even if A is not invertible and $m_A \geq m = m$ by using a QR decomposition of the matrix A.

Remark 3.4. Often in the literature the symbol "$\circ$" is not used for linear maps $f = Az$. Therefore, $A\mathcal{P}$ refers to the operation $A \circ \mathcal{P}$ and $\mathcal{P}A$ refers to the operation $\mathcal{P} \circ A$.

3.5 Operations on P-collections

This section covers some results and algorithms which are specific to operations with P-collections. P-collections are unions of polytopes (see Definition 3.3) and therefore the set of states contained in a P-collection can be represented in an infinite number of ways, i.e. the P-collection representation is not unique. For example, one can subdivide any polytope $\mathcal{P}$ into a number of smaller polytopes whose union is a P-collection which covers $\mathcal{P}$. Note that the complexity of all subsequent computations depends strongly on the number of polytopes representing a P-collection. The smaller the cardinality of a P-collection, the more efficient the computations. The reader is referred to [220, 219] for proofs and comments on computational efficiency.

3.5.1 Set-Difference

The first two results given here show how the set difference of a P-collection and a P-collection (or polyhedron) may be computed.

Lemma 3.3. *Let $\mathcal{C} \triangleq \bigcup_{j\in\{1,\dots,J\}} \mathcal{C}_j$ be a P-collection, where all the $\mathcal{C}_j$, $j \in \{1,\dots,J\}$, are nonempty polyhedra. If $\mathcal{S}$ is a nonempty polyhedron, then $\mathcal{C} \setminus \mathcal{S} = \bigcup_{j\in\{1,\dots,J\}} (\mathcal{C}_j \setminus \mathcal{S})$ is a P-collection.*

Lemma 3.4. *Let the sets $\mathcal{C} \triangleq \bigcup_{j\in\{1,\dots,J\}} \mathcal{C}_j$ and $\mathcal{D} \triangleq \bigcup_{y=1,\dots,Y} \mathcal{D}_y$ be P-collections, where all the $\mathcal{C}_j$, $j \in \{1,\dots,J\}$, and $\mathcal{D}_y$, $y \in \{1,\dots,Y\}$, are nonempty polyhedra. If $\mathcal{E}_0 \triangleq \mathcal{C}$ and $\mathcal{E}_y \triangleq \mathcal{E}_{y-1} \setminus \mathcal{D}_y$, $y \in \{1,\dots,Y\}$ then $\mathcal{C} \setminus \mathcal{D} = \mathcal{E}_Y$ is a P-collection.*

That $\mathcal{C} \subseteq \mathcal{D}$ can be easily verified since $\mathcal{C} \subseteq \mathcal{D} \Leftrightarrow \mathcal{C} \setminus \mathcal{D} = \emptyset$, similarly $\mathcal{C} = \mathcal{D}$ is also easily verified since

$$\mathcal{C} = \mathcal{D} \Leftrightarrow (\mathcal{C} \setminus \mathcal{D} = \emptyset \text{ and } \mathcal{D} \setminus \mathcal{C} = \emptyset)$$

Next, an algorithm for computing the Pontryagin difference of a P-collection and a polytope is presented. If $\mathcal{S}$ and $\mathcal{B}$ are two subsets of $\mathbb{R}^n$ then $\mathcal{S} \ominus \mathcal{B} = [\mathcal{S}^c \oplus (-\mathcal{B})]^c$ where $(\cdot)^c$ denotes the set complement. The following algorithm implements the computation of the Pontryagin difference of a P-collection $\mathcal{C} \triangleq \cup_{j\in\{1,\dots,J\}} \mathcal{C}_j$, where $\mathcal{C}_j, j \in \{1,\dots,J\}$ are polytopes in $\mathbb{R}^n$, and a polytope $\mathcal{B} \subset \mathbb{R}^n$.

Algorithm 3.5 (Pontryagin Difference for P-collections, $\mathcal{C} \ominus \mathcal{B}$)

```
1. Input: P-collection $\mathcal{C}$, polytope $\mathcal{B}$;
2. $\mathcal{H} \triangleq \mathrm{env}(\mathcal{C})$ (or $\mathcal{H} \triangleq \mathrm{conv}(\mathcal{C})$);
3. $\mathcal{D} \triangleq \mathcal{H} \ominus \mathcal{B}$;
4. $\mathcal{E} \triangleq \mathcal{H} \setminus \mathcal{C}$;
```

5. $\mathcal{F} \triangleq \mathcal{E} \oplus (-\mathcal{B})$;
6. $\mathcal{G} \triangleq \mathcal{D} \setminus \mathcal{F}$;
7. Output: P-collection $\mathcal{G} \triangleq \mathcal{C} \ominus \mathcal{B}$.

Remark 3.5. Note that $\mathcal{H}$ in Step 2 of Algorithm 3.5 can be any convex set containing the P-collection $\mathcal{C}$. The computation of $\mathcal{H}$ is generally more efficient if the envelope operation is used instead of convex hull.

Remark 3.6. It is important to note that $(\bigcup_{j\in\{1,\dots,J\}} \mathcal{C}_j)\ominus\mathcal{B} \neq \bigcup_{j\in\{1,\dots,J\}}(\mathcal{C}_j\ominus\mathcal{B})$, where $\mathcal{B}$ and $\mathcal{C}_j$ are polyhedra; hence, the relatively high computational effort of computing the Pontryagin difference of a P-collection and a polytope.

Theorem 3.3 (Computation of Pontryagin Difference, [219]). *The output of Algorithm 3.5 is $\mathcal{G} = \mathcal{C} \ominus \mathcal{B}$.*

Proof: It holds by definition that

$$\begin{aligned}\mathcal{D} &\triangleq \mathcal{H} \ominus \mathcal{B} = \{x \ : \ x + w \in \mathcal{H}, \ \forall w \in \mathcal{B}\},\\ \mathcal{E} &\triangleq \mathcal{H} \setminus \mathcal{C} = \{x \ : \ x \in \mathcal{H} \text{ and } x \notin \mathcal{C}\}.\end{aligned}$$

By the definition of the Minkowski sum:

$$\begin{aligned}\mathcal{F} \triangleq \mathcal{E} \oplus (-\mathcal{B}) &= \{x \ : \ x = z + w, \ z \in \mathcal{E}, w \in (-\mathcal{B})\}\\ &= \{x \ : \ \exists w \in (-\mathcal{B}), \text{ s.t. } x + w \in \mathcal{E}\}.\end{aligned}$$

By definition of the set difference:

$$\begin{aligned}\mathcal{D} \setminus \mathcal{F} &\triangleq \{x : \ x \in \mathcal{D} \text{ and } x \notin \mathcal{F}\}\\ &= \{x \in \mathcal{D} \ : \ \nexists\, w \in \mathcal{B} \text{ s.t. } x + w \in \mathcal{E}\}\\ &= \{x \in \mathcal{D} \ : \ x + w \notin \mathcal{E}, \ \forall w \in \mathcal{B}\}.\end{aligned}$$

From the definition of the set $\mathcal{D}$:

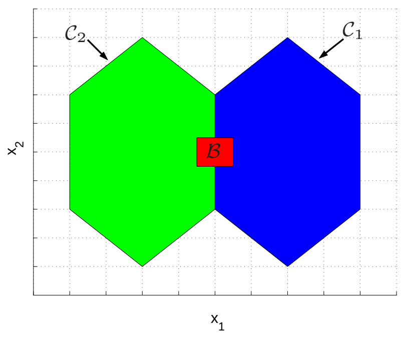
$$\mathcal{D} \setminus \mathcal{F} = \{x \ : \ x + w \in \mathcal{H} \text{ and } x + w \notin \mathcal{E}, \ \forall w \in \mathcal{B}\}$$

and from the definition of the set $\mathcal{E}$ and because $\mathcal{C} \subseteq \mathcal{H}$:

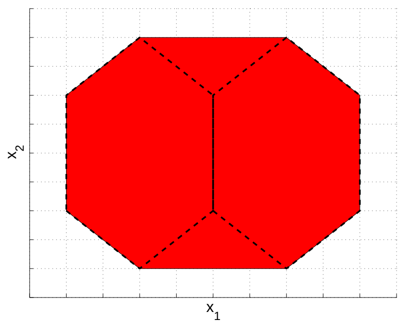
$$\begin{aligned}\mathcal{D} \setminus \mathcal{F} &= \{x \ : \ x + w \in \mathcal{H} \text{ and } \ (x + w \notin \mathcal{H} \text{ or } x + w \in \mathcal{C}) \ \forall w \in \mathcal{B}\}\\ &= \{x \ : \ x + w \in \mathcal{C}, \ \forall w \in \mathcal{B}\}\\ &= \mathcal{C} \ominus \mathcal{B}.\end{aligned}$$

□

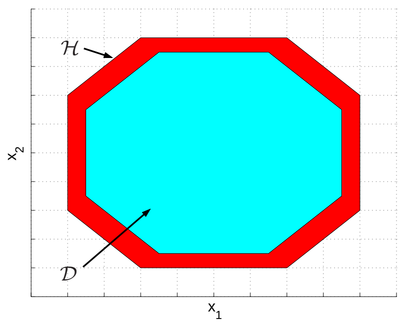
Algorithm 3.5 is illustrated on a sample P-collection in Figures 3.12(a) to 3.12(f).



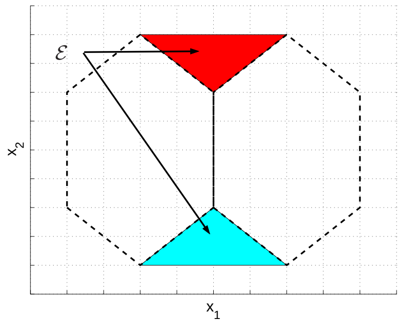
(a) $\bigcup_{j\in\{1,\dots,J\}} \mathcal{C}_j$ and $\mathcal{B}$.



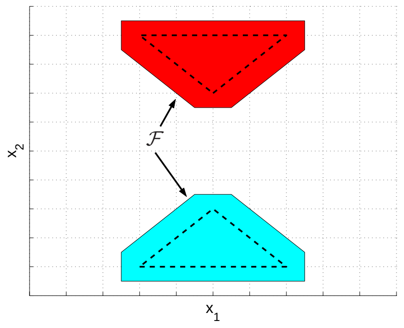
(b) $\mathcal{H} = \text{conv}(\mathcal{C})$.



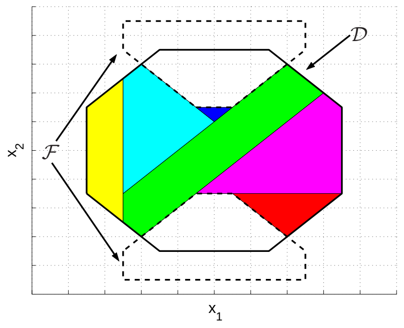
(c) $\mathcal{D} = \mathcal{H} \ominus \mathcal{B}$.



(d) $\mathcal{E} = \mathcal{H} \setminus \mathcal{C}$.



(e) $\mathcal{F} = \mathcal{E} \oplus (-\mathcal{B})$.



(f) $\mathcal{G} = \mathcal{D} \setminus \mathcal{F}$.

Fig. 3.12 Graphical illustration of Algorithm 3.5.

Remark 3.7. It should be noted that Algorithm 3.5 for computation of the Pontryagin difference is conceptually similar to the algorithm proposed in

[237, 158]. However, in general the envelope operation employed in step 2 significantly reduces the number of sets obtained at step 4, which in turn results in fewer Minkowski set additions. Since the computation of a Minkowski set addition is expensive, a runtime improvement can be expected.

3.5.2 Polytope Covering

The problem of checking if some $\mathcal{P}$ polytope is covered with the union of other polytopes, i.e. a P-collection $\mathcal{Q} := \cup_i \mathcal{Q}_i$ is discussed in this section. We consider two related problems:

polycover: Check if $\mathcal{P} \subseteq \mathcal{Q}$, and
regiondiff: Compute P-collection $\mathcal{R} = \mathcal{P} \setminus \mathcal{Q}$.

Clearly, **polycover** is just a special case of **regiondiff**, where the resulting P-collection $\mathcal{R} = \emptyset$. Also, it is straightforward to extend the above problems to the case where both $\mathcal{P}$ and $\mathcal{Q}$ are both P-collections.

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

$$\mathcal{P} \subseteq \mathcal{Q} \quad \Leftrightarrow \quad \mathcal{P} = \cup_i (\mathcal{P} \cap \mathcal{Q}_i).$$

Therefore, we could create $\mathcal{R}_i = \mathcal{Q}_i \cap \mathcal{P}$, for $i = 1, \ldots, N_Q$ and then compute the union of the collection of polytopes $\{\mathcal{R}_i\}$ by using the polyunion algorithm for computing the convex union of $\mathcal{H}$-polyhedra reported discussed in Section 3.4.10. If polyunion succeeds (i.e., the union is a convex set) and the resulting polytope is equal to $\mathcal{P}$ then $\mathcal{P}$ is covered by $\mathcal{Q}$, otherwise it is not. However, this approach is computationally very expensive. In the Appendix A two alternative approaches for checking if $\mathcal{P} \subseteq (\bigcup_{i=1}^{N_Q} \mathcal{Q}_i)$ are presented.

3.5.3 Union of P-collections

Consider a P-collection $\mathcal{P} = \{\mathcal{P}_i\}_{i=1}^{p}$. We study the problem of finding a minimal representation on $\mathcal{P}$ by merging one or more polyhedra belonging to the P-collection. Clearly one could use the polyunion algorithm presented in Section 3.4.10 for all possible subsets of the P-collection and solve the problem by comparing all solutions. However this approach is not computationally efficient.

Typically, merging problems are formally defined by using the definitions of “hyperplane arrangements” and “markings”. Next we introduce such definitions and formally define thre classes of merging problems. We refer the reader to the Appendix B for more details on the topic.

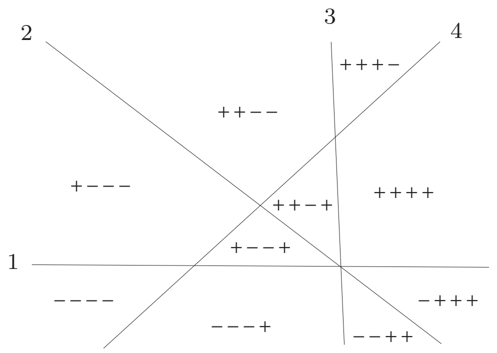


Fig. 3.13 Arrangement of four hyperplanes (lines) in $\mathcal{R} = \mathbb{R}^2$ with markings $m \in M(\mathcal{R})$

3.5.3.1 Hyperplane Arrangements and Markings

Let A be a collection of n distinct hyperplanes $\{H_i\}_{i=1,\dots,n}$ in the d-dimensional Euclidian space $\mathbb{R}^d$, where each hyperplane is given by a linear equality $H_i = \{x \in \mathbb{R}^d \ : \ a_i'x = b_i\}$. We say that the hyperplanes of A are in *general position*, if there exists no pair of parallel hyperplanes, and if any point of $\mathbb{R}^d$ belongs at most to d hyperplanes. Let $\text{SV} : \mathbb{R}^d \to \{-,+\}^n$ be the simplified sign vector defined as

$$\text{SV}_i(x) = \begin{cases} - & \text{if } a_i'x \le b_i, \\ + & \text{if } a_i'x > b_i \end{cases} \quad \text{for } i \in \{1,2,\dots,n\}\,. \tag{3.35}$$

Note that in general, the sign vector is defined such that its image is $\{-,0,+\}$, where the '0' element corresponds to $a_i'x = b_i$. Cells with '0' markings are lower-dimensional.

Consider the set $\mathcal{P}_m = \{x \in \mathbb{R}^d \ : \ \text{SV}(x) = m\}$ for a given sign vector m. This set is called a *cell* of the arrangement and is a polyhedron as it is defined by linear inequalities. We will refer to m as the *marking* of the polyhedron (or cell) $\mathcal{P}_m$ in the *hyperplane arrangement* A (see Fig. 3.13). Let $M(\mathcal{R})$ be the image of the function $\text{SV}(x)$ for $x \in \mathcal{R} \subseteq \mathbb{R}^d$, namely the collection of all the possible markings of all the points in $\mathcal{R}$.

Let the '$*$' element extend the sign vector in the sense that it denotes the union of cells, where the associated hyperplane is not a facet of the associated polyhedron $\mathcal{P}_m$. As an example, consider in Fig. 3.13 the two polyhedra with the markings $m_1 = ----$ and $m_2 = +---$. Then, $m = *---$ is equivalent to $\{m_1, m_2\}$ and refers to $\mathcal{P}_{m_1} \cup \mathcal{P}_{m_2}$.

The cell enumeration problem in a hyperplane arrangement amounts to enumerate all the elements of the set $M(\mathcal{R})$. Let $\#M(\mathcal{R})$ be the number of cells identified by $M(\mathcal{R})$. Buck's formula [65] defines the upper bound

$$|M| \leq \sum_{i=0}^{d} \binom{n}{i} = O(n^d), \tag{3.36}$$

with the equality satisfied if the hyperplanes are in general position and $\mathcal{R} = \mathbb{R}^d$.

The cell enumeration problem admits an optimal solution with time and space complexity $O(n^d)$ [92]. An alternative approach based on reverse search was presented in [12], improved in [98] and implemented in [97]. Reverse search is an exhaustive search technique that can be considered as a special graph search. This search technique has been used to design efficient algorithms for various enumeration problems such as enumeration of all spanning trees and cells in hyperplane arrangements.

3.5.3.2 Standard Merging Problems

Consider the following assumption

Assumption 3.1 *The polyhedra of the P-collectionare cells in a (global) hyperplane arrangement, of which the markings are available.*

and pose the following three problems.

Problem 3.1 (Disjoint Optimal Complexity Reduction). Given an initial set of polyhedra $\{\mathcal{P}_i\}_{i=1}^{p}$ satisfying Assumption 3.1, the *disjoint* optimal complexity reduction problem amounts to derive a new set of polyhedra $\{\mathcal{Q}_i\}_{i=1,\ldots,q}$ with the following properties: (i) the union of the new polyhedra is equal to the union of the original ones, i.e. $(\bigcup_{i=1}^{q} \mathcal{Q}_i) = (\bigcup_{i=1}^{p} \mathcal{P}_i)$, (ii) q is minimal, i.e. there exists no set $\{\mathcal{Q}_i\}_{i=1,\ldots,q}$ with a smaller number of polyhedra, (iii) the new polyhedra are mutually disjoint, i.e. $\mathcal{Q}_i \neq \mathcal{Q}_j$ for all $i, j \in \{1, \ldots, q\}$, $i \neq j$, and (iv) the new polyhedra are formed as unions of the old ones, i.e. for each $\mathcal{Q}_j$, $j \in \{1, \ldots, q\}$, there exists an index set $\mathcal{I} \subseteq \{1, \ldots, p\}$, such that $\mathcal{Q}_j = \bigcup_{i \in \mathcal{I}} \mathcal{P}_i$.

This problem is equivalent to an optimal merging problem. Next, we remove Requirements (iii) and (iv) thus allowing for overlaps in the resulting polyhedra.

Problem 3.2 (Non-Disjoint Optimal Complexity Reduction). Given an initial set of polyhedra $\{\mathcal{P}_i\}_{i=1,\ldots,p}$ satisfying Assumption 3.1, the *non-disjoint* optimal complexity reduction problem amounts to derive a new set of polyhedra $\{\mathcal{Q}_i\}_{i=1,\ldots,q}$ with Properties (i) and (ii) as in Problem 3.1.

Strictly speaking, the second problem is not a merging problem, but a more general optimal set covering problem, which is equivalent to logic minimization frequently used in digital circuit design. Nevertheless, we will sometimes use the term merging instead of complexity reduction.

Next, the assumption that the original polyhedra are cells in a hyperplane arrangement and that markings are available are removed, but require additionally that each polyhedron is represented with a minimal number of facets. This problem can be considered as the general non-disjoint optimal complexity reduction problem for PWA functions.

Problem 3.3 (General Non-Disjoint Optimal Complexity Reduction). Given an initial set of polyhedra $\{\mathcal{P}_i\}_{i=1,\dots,p}$ where Assumption 3.1 is *not* required to hold, the *general* non-disjoint optimal complexity reduction problem amounts to derive a new set of polyhedra $\{\mathcal{Q}_i\}_{i=1,\dots,q}$ with Properties (i) and (ii) as in Problem 3.1, and (iii) the number of facets for each $\mathcal{Q}_i$ being minimal.

All three tasks are non-trivial, as the union of polyhedra is in general non-convex, and because we are aiming at deriving the optimal solution, more specifically, the set of polyhedra with the minimal cardinality. Indeed, the problems are $\mathcal{NP}$-hard (see [77] and references therein). As a direct consequence, fast algorithms are unlikely to exist leaving us either with rather long computation times or suboptimal solutions.

Our interest in this problem will be clear later in this book (Section 10.2) when computing the PWA state-feedback control law to optimal control problems. Once the PWA state-feedback control law has been derived, the memory requirement and the on-line computation time are linear in the number of polyhedra of the feedback law when using standard brute force search. Therefore, we will be interested in the problem of finding a *minimal representation* of piecewise affine (PWA) systems, or more specifically, for a given PWA system, we solve the problem of deriving a PWA system, that is both equivalent to the former and minimal in the number of regions. This is done by associating with different feedback law a different color, and we collect the polyhedra with the same color. Then, for a given color, we try to merge the corresponding P-collection by solving one of the three problem described above. If the number of polyhedra with the same affine dynamic is large, the number of possible polyhedral combinations for merging explodes. As most of these unions are not convex or even not connected and thus cannot be merged, trying all combinations using standard techniques based on *linear programming* (LP) is prohibitive. Furthermore, our objective here is not only to reduce the number of polyhedra but rather to find the minimal and thus optimal number of disjoint polyhedra. This problem is known to be $\mathcal{NP}$-hard, and to the best of our knowledge, it is still an open problem. In Appendix B algorithms for the solutions of the three posed problems are presented.

Chapter 4
Linear and Quadratic Optimization

4.1 Linear Programming

When the cost and the constraints of the continuous optimization problem (1.4) are affine, then the problem is called *linear program* (LP). The most general form of a linear program is

$$\begin{array}{ll} \inf_z & c'z \\ \text{subj. to} & Gz \leq W \end{array} \tag{4.1}$$

where $G \in \mathbb{R}^{m \times s}$, $W \in \mathbb{R}^m$. Linear programs are convex optimization problems.

Two other common forms of linear programs include both equality and inequality constraints:

$$\begin{array}{ll} \inf_z & c'z \\ \text{subj. to} & Gz \leq W \\ & G_{eq}z = W_{eq} \end{array} \tag{4.2}$$

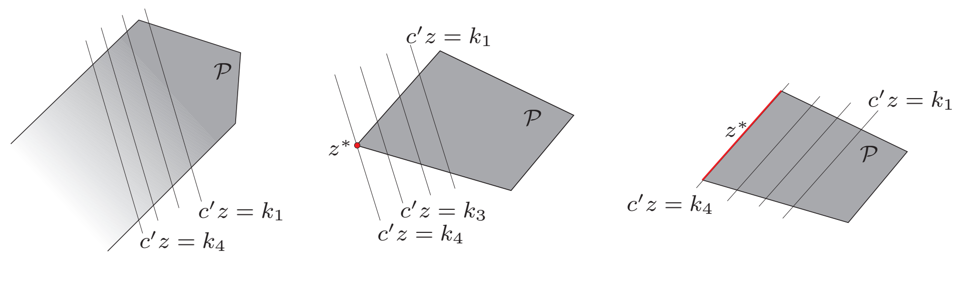
where $G_{eq} \in \mathbb{R}^{p \times s}$, $W_{eq} \in \mathbb{R}^p$, or only equality constraints and positive variables:

$$\begin{array}{ll} \inf_z & c'z \\ \text{subj. to} & G_{eq}z = W_{eq} \\ & z \geq 0 \end{array} \tag{4.3}$$

By standard and simple manipulations [57](p. 146) it is always possible to convert one of the three forms (4.1), (4.2) and (4.3) into the other.

4.1.1 Graphical Interpretation and Solutions Properties

Let $\mathcal{P}$ be the feasible set (1.6) of problem (4.1). As $Z = \mathbb{R}^s$, this implies that $\mathcal{P}$ is a polyhedron defined by the inequality constraints in (4.1). If $\mathcal{P}$ is empty,



(a) Linear Program - Case 1 (b) Linear Program - Case 2 (c) Linear Program - Case 3

Fig. 4.1 Graphical Interpretation of the Linear Program Solution, $k_i < k_{i-1}$

then the problem is infeasible. We will assume for the following discussion that $\mathcal{P}$ is not empty. Denote by J^* the optimal value and by Z^* the set of optimizers of problem (4.1)

$$Z^* = \text{argmin}_{z \in \mathcal{P}} \;\; c'z$$

Three cases can occur.

Case 1. The LP solution is unbounded, i.e., $J^* = -\infty$.
Case 2. The LP solution is bounded, i.e., $J^* > -\infty$ and the optimizer is unique. Z^* is a singleton.
Case 3. The LP solution is bounded and there are multiple optima. Z^* is an uncountable subset of $\mathbb{R}^s$ which can be bounded or unbounded.

The two dimensional geometric interpretation of the three cases discussed above is depicted in Figure 4.1. The level curves of the cost function $c'z$ are represented by the parallel lines. All points z belonging both to the line $c'z = k_i$ and to the polyhedron $\mathcal{P}$ are feasible points with an associated cost k_i, with $k_i < k_{i-1}$. Solving (4.1) amounts to finding a feasible z which belongs to the level curve with the smallest cost k_i. Since the gradient of the cost is c', the direction of steepest descent is $-c'$.

Case 1 is depicted in Figure 4.1(a). The feasible set $\mathcal{P}$ is unbounded. One can move in the direction of steepest descent $-c$ and be always feasible, thus decreasing the cost to $-\infty$. Case 2 is depicted in Figure 4.1(b). The optimizer is unique and it coincides with one of the vertices of the feasible polyhedron. Case 3 is depicted in Figure 4.1(c). The whole bold facet of the feasible polyhedron $\mathcal{P}$ is optimal, i.e., the cost for any point z belonging to the facet equals the optimal value J^*. In general, the optimal facet will be a facet of the polyhedron $\mathcal{P}$ parallel to the hyperplane $c'z = 0$.

From the analysis above we can conclude that the optimizers of any bounded LP always lie on the boundary of the feasible polyhedron $\mathcal{P}$.

4.1.2 Dual of LP

Consider the LP (4.1)

$$\begin{array}{ll} \inf_z & c'z \\ \text{subj. to} & Gz \leq W \end{array} \tag{4.4}$$

with $z \in \mathbb{R}^s$ and $G \in \mathbb{R}^{m \times s}$.

The Lagrange function as defined in (2.3) is

$$L(z,u) = c'z + u'(Gz - W).$$

The dual cost is

$$\Theta(u) = \inf_z L(z,u) = \inf_z (c' + u'G)z - u'W = \begin{cases} -u'W & \text{if} - G'u = c \\ -\infty & \text{if} - G'u \neq c \end{cases}$$

Since we are interested only in cases where Θ is finite, from the relation above we conclude that the dual problem is

$$\begin{array}{ll} \sup_u & -u'W \\ \text{subj. to} & -G'u = c \\ & u \geq 0 \end{array} \tag{4.5}$$

which can be rewritten as

$$\begin{array}{ll} \inf_u & W'u \\ \text{subj. to} & G'u = -c \\ & u \geq 0 \end{array} \tag{4.6}$$

4.1.3 KKT condition for LP

The KKT conditions (2.18a)-(2.18e) for the LP (4.1) become

$$G'u = -c, \tag{4.7a}$$
$$(G_j z - W_j)u_j = 0, \tag{4.7b}$$
$$u \geq 0, \tag{4.7c}$$
$$Gz \leq W \tag{4.7d}$$

which are: primal feasibility (4.7d), dual feasibility (4.7a) and (4.7c) and slackness complementary conditions (4.7b).

4.1.4 Active Constraints and Degeneracies

For the topics covered in this book the number of constraints active at the optimum deserves particular attention.

Consider the LP (4.1). Let $J \triangleq \{1, \ldots, m\}$ be the set of constraint indices. For any $A \subseteq J$, let G_A and W_A be the submatrices of G and W, respectively, comprising the rows indexed by A and denote with G_j and W_j the j-th row of G and W, respectively. Let z be a feasible point and consider the set of active and inactive constraints at z:

$$\begin{aligned} A(z) &\triangleq \{j \in J : G_j z = W_j\} \\ NA(z) &\triangleq \{j \in J : G_j z < W_j\}. \end{aligned} \tag{4.8}$$

The cardinality of $A(z)$ depends on the cases discussed in Section (4.1.1) and on the shape of the feasible set.

Case 1 - Unbounded solution. Since a minimizer is not defined in this case, the number of active constraints is not defined.

Case 2 - Bounded solution, unique optimum. The number of active constraints can be any number between s and m (included).
Note that even if the description of the feasible polyhedron $\mathcal{P}$ is minimal, then the cardinality of $A(z^*)$ can be greater than s. For instance this is the case if the optimum is the vertex of a pyramid with a rectangular base in three dimensions. Also, if the description of the feasible polyhedron $\mathcal{P}$ is not minimal, i.e., $Gz \leq W$ contains redundant constraints, the cardinality of $A(z^*)$ can be higher than s. Such a case is depicted in Figure 4.2 where constraints $1, 3, 5, 6$ are active at the optimum.

Case 3 - Bounded solution, multiple optima. The number of active constraints can be any number between 1 and m (included). Note that if the description of the polyhedron $\mathcal{P}$ is minimal, the cardinality of $A(z^*)$ can be smaller than s at all points $z^* \in Z^*$ contained in the interior of the optimal facet.

This discussion on the constraints that are active at the optimum is fundamental throughout this book. For this reason we introduce the following concept of primal degeneracy.

Definition 4.1. The LP (4.1) is said to be primal degenerate if there exists a $z^* \in Z^*$ such that the number of active constraints at z^* is greater than the number of variables s.

Figure 4.2 depicts a case of primal degeneracy.

Definition 4.2. The LP (4.1) is said to be dual degenerate if its dual problem is primal degenerate.

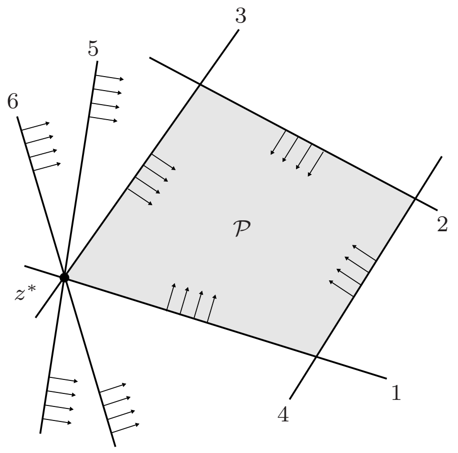


Fig. 4.2 Primal Degeneracy in a Linear Program.

For the above discussion we conclude that if the primal problem has multiple optima, then the dual problem is primal degenerate (i.e., the primal problem is dual degenerate). The converse is not always true. In order to show this observation and better understand primal and dual degeneracy we discuss a simple example next.

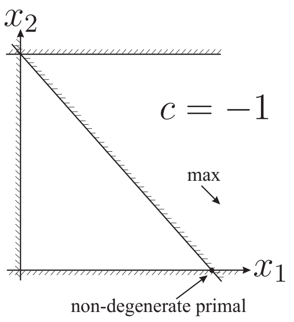
Example 4.1. **Primal and dual degeneracies**
Consider the following pair of primal and dual LPs

$$\begin{array}{ll} \text{Primal} & \text{Dual} \\ \max\ [1\ \ c]x & \min\ [1\ \ 1]y \\ \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} x \leq \begin{bmatrix} 1 \\ 1 \end{bmatrix},\ x \geq 0 & \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} y \geq \begin{bmatrix} 1 \\ c \end{bmatrix},\ y \geq 0 \end{array} \tag{4.9}$$

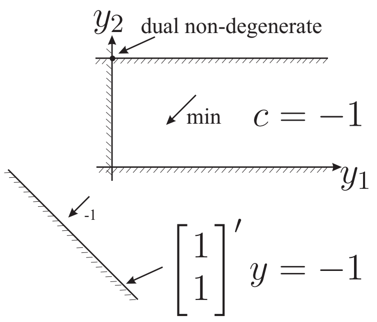
where c is a parameter.

Case 1. Figure 4.3 depicts the situation for $c = -1$. We see that both the primal and the dual problem exhibit no degeneracies for $c = -1$. There is a unique optimizer for the primal LP at $(1, 0)$, which is defined by the intersection of two active constraints, exactly as many as the number of decision variables. The same holds true in the dual space, where the optimal dual variable is $(0, 1)$.

Case 2. The situation changes when $c = 1$, as portrayed in Figure 4.4. Consider the two solutions for the primal LP denoted with 1 and 2 in Figure 4.4(a) and referred to as “basic” solutions. Basic solution 1 is primal non-degenerate, since it is defined by exactly as many active constraints as there are variables. Basic solution 2 is primal degenerate, since it is defined by three active constraints, i.e., more than two. Any convex combination

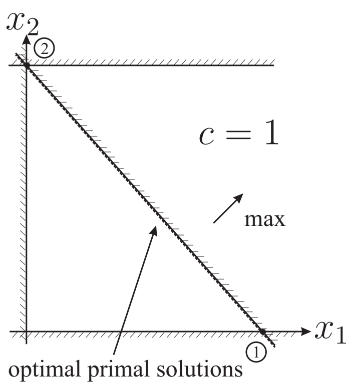


(a) Case 1 - Primal LP

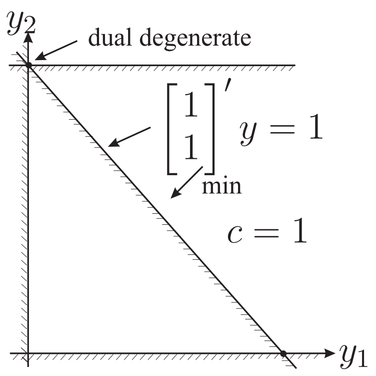


(b) Case 1 - Dual LP

Fig. 4.3 LP with no Primal or Dual Degeneracy



(a) Case 2 - Primal LP



(b) Case 2 - Dual LP

Fig. 4.4 LP with Primal and Dual Degeneracy

of optimal solutions 1 and 2 is also optimal. This continuum of optimal solutions in the primal problem corresponds to a degenerate solution in the dual space, hence the primal problem is dual-degenerate. That is, the dual problem is primal-degenerate. Both basic solutions correspond to a degenerate solution point in the dual space, as seen on Figure 4.4(b). In conclusion, Figures 4.4(a) and 4.4(b) show an example of a primal problem with multiple optima and the corresponding dual problem being primal degenerate.

Case 3. We want to show that the statement "if the dual problem is primal degenerate then the primal problem has multiple optima" is, in general, not true. Consider case 2 and switch dual and primal problems, i.e.,call the "dual problem" primal problem and the "primal problem" dual problem (this can be done since the dual of the dual problem is the primal problem). Then, we have a dual problem which is primal degenerate in solution 2 while the primal problem does not present multiple optima.

4.1.5 Convex Piecewise Linear Optimization

Consider a continuous and convex piecewise affine function $J : \mathcal{R} \subseteq \mathbb{R}^s \to \mathbb{R}$:

$$J(z) = c_i' z + d_i \text{ for } [\,z\,] \in \mathcal{R}_i, \tag{4.10}$$

where $\{\mathcal{R}_i\}_{i=1}^p$ are polyhedral sets with disjoint interiors, $\mathcal{R} \triangleq \bigcup_{i=1}^p \mathcal{R}_i$ is a polyhedron and $c_i \in \mathbb{R}^s$, $d_i \in \mathbb{R}$. Then, $J(z)$ can be rewritten as [230]

$$J(z) = \max_{i=1,\dots,k} \{c_i' z + d_i\} \;\forall\; z \in \mathcal{R} \tag{4.11}$$

Consider the following optimization problem

$$\begin{array}{rll} J^* = \min_z & J(z) \\ & \text{subj. to } Gz \le W \\ & z \in \mathcal{R} \end{array} \tag{4.12}$$

where the cost function has the form (4.10).

By using the equivalence between the form (4.10) and (4.11) of continuous and convex piecewise affine functions, it can be easily proven that the optimization problem (4.12) with $J(z)$ defined in (4.10) can be solved by the following linear program [57]:

$$\begin{array}{rll} J^* = \min_{z,\varepsilon} & \varepsilon \\ & \text{subj. to } Gz \le W \\ & \quad c_i' z + d_i \le \varepsilon, \quad i = 1,\dots,k \\ & \quad z \in \mathcal{R} \end{array} \tag{4.13}$$

The previous result can be extended to the sum of continuous and convex piecewise affine functions. Let $J : \mathcal{R} \subseteq \mathbb{R}^s \to \mathbb{R}$ be defined as:

$$J(z) = \sum_{j=1}^{r} J^j(z) \tag{4.14}$$

with

$$J^j(z) = \max_{i=1,\dots,k^j} \{c_i^{j\,\prime} z + d_i^j\} \;\forall\; z \in \mathcal{R} \tag{4.15}$$

Then the following optimization problem

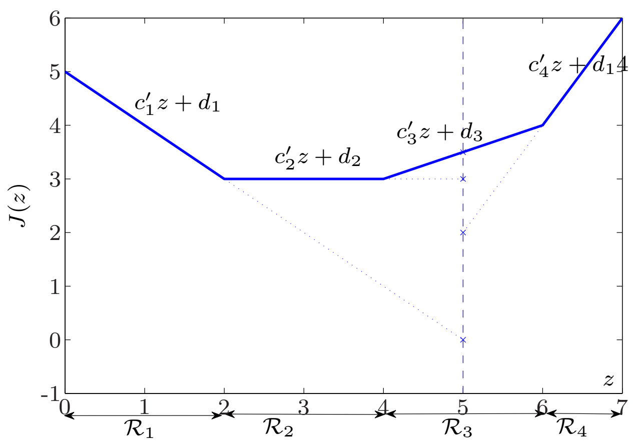


Fig. 4.5 Convex PWA Function Described as the Max of Affine Functions

$$
\begin{array}{rl}
J^* = \min_z & J(z) \\
& \text{subj. to } Gz \leq W \\
& z \in \mathcal{R}
\end{array}
\tag{4.16}
$$

where the cost function has the form (4.14) can be solved by the following linear program:

$$
\begin{array}{rll}
J^* = \min_{z,\varepsilon^1,\ldots,\varepsilon^r} & \varepsilon^1 + \cdots + \varepsilon^r & \\
\text{subj. to} & Gz \leq W & \\
& c_i^{1\prime} z + d_i^1 \leq \varepsilon^1, & i = 1, \ldots, k^1 \\
& c_i^{2\prime} z + d_i^2 \leq \varepsilon^2, & i = 1, \ldots, k^2 \\
& \vdots & \\
& c_i^{r\prime} z + d_i^r \leq \varepsilon^r, & i = 1, \ldots, k^r \\
& z \in \mathcal{R} &
\end{array}
\tag{4.17}
$$

Remark 4.1. Note that the results of this section can be immediately applied to the minimization of one or infinity norms. Note that for any $y \in \mathbb{R}$, $|y| = \max\ \{y, -y\}$. Therefore for any $Q \in \mathbb{R}^{k \times s}$ and $p \in \mathbb{R}^k$:

$$\|Qz - p\|_\infty = \max\{{Q_1}'z + p_1, -{Q_1}'z - p_1, \ldots, {Q_k}'z + p_k, -{Q_k}'z - p_k\}$$

and

$$\|Qz - p\|_1 = \sum_{i=1}^{k} |{Q_i}'z + p_i| = \sum_{i=1}^{k} \max\{{Q_i}'z + p_i, -{Q_i}'z - p_i\}$$

4.2 Quadratic Programming

The continuous optimization problem (1.4) is called *quadratic program* (QP) if the constraint functions are affine and the cost function is a convex quadratic function. In this book we will use the form:

$$\begin{array}{ll} \min_z & \frac{1}{2}z'Hz + q'z + r \\ \text{subj. to} & Gz \leq W \end{array} \tag{4.18}$$

where $z \in \mathbb{R}^s$, $H = H' \succ 0 \in \mathbb{R}^{s \times s}$, $q \in \mathbb{R}^s$, $G \in \mathbb{R}^{m \times s}$. In (4.18) the constant term can be omitted if one is only interested in the optimizer.

Other QP forms often include equality and inequality constraints.

$$\begin{array}{ll} \min_z & \frac{1}{2}z'Hz + q'z + r \\ \text{subj. to} & Gz \leq W \\ & G_{eq}z = W_{eq} \end{array} \tag{4.19}$$

4.2.1 Graphical Interpretation and Solutions Properties

Let $\mathcal{P}$ be the feasible set (1.6) of problem (4.18). As $Z = \mathbb{R}^s$, this implies that $\mathcal{P}$ is a polyhedron defined by the inequality constraints in (4.18). The two dimensional geometric interpretation is depicted in Figure 4.6. The level curves of the cost function $\frac{1}{2}z'Hz + q'z + r$ are represented by the ellipsoids. All the points z belonging both to the ellipsoid $\frac{1}{2}z'Hz + q'z + r = k_i$ and to the polyhedron $\mathcal{P}$ are feasible points with an associated cost k_i. The smaller is the ellipsoid, the smaller is its cost k_i. Solving (4.18) amounts to finding a feasible z which belongs the level curve with the smallest cost k_i. Since H is strictly positive definite, the QP (4.18) cannot have multiple optima nor unbounded solutions. If $\mathcal{P}$ is not empty the optimizer is unique. Two cases can occur if $\mathcal{P}$ is not empty:

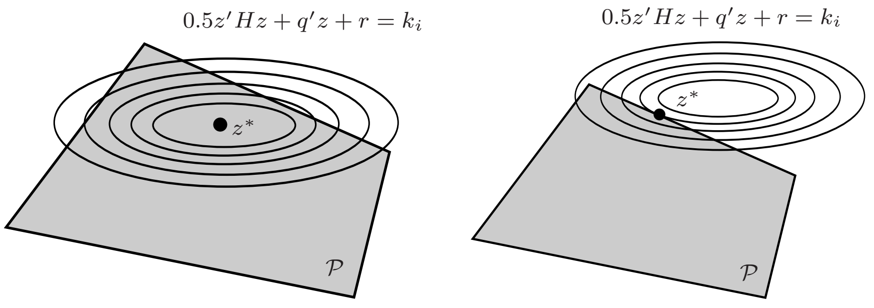
Case 1. The optimizer lies strictly inside the feasible polyhedron (Figure 4.6(a)).

Case 2. The optimizer lies on the boundary of the feasible polyhedron (Figure 4.6(b)).

In case 1 the QP (4.18) is unconstrained and we can find the minimizer by setting the gradient equal to zero

$$Hz^* + q = 0. \tag{4.20}$$

Since $H \succ 0$ we obtain $z^* = -H^{-1}q$.



(a) Quadratic Program - Case 1 (b) Quadratic Program - Case 2

Fig. 4.6 Graphical Interpretation of the Quadratic Program Solution

4.2.2 Dual of QP

Consider the QP (4.18)

$$\begin{array}{ll}\min_z & \frac{1}{2}z'Hz + q'z \\ \text{subj. to} & Gz \leq W\end{array}$$

The Lagrange function as defined in (2.3) is

$$L(z,u) = \{\frac{1}{2}z'Hz + q'z + u'(Gz - W)\}$$

The the dual cost is

$$\Theta(u) = \min_z \{\frac{1}{2}z'Hz + q'z + u'(Gz - W)\} \tag{4.21}$$

and the dual problem is

$$\max_{u \geq 0} \min_z \{\frac{1}{2}z'Hz + q'z + u'(Gz - W)\}. \tag{4.22}$$

For a given u the Lagrange function $\frac{1}{2}z'Hz + q'z + u'(Gz - W)$ is convex. Therefore it is necessary and sufficient for optimality that the gradient is zero

$$Hz + q + G'u = 0.$$

From the equation above we can derive $z = -H^{-1}(q + G'u)$ and substituting this in equation (4.21) we obtain:

$$\Theta(u) = -\frac{1}{2}u'(GH^{-1}G')u - u'(W + GH^{-1}q) - \frac{1}{2}q'H^{-1}q \tag{4.23}$$

By using (4.23) the dual problem (4.22) can be rewritten as:

$$\begin{array}{ll}\min_u & \frac{1}{2}u'(GH^{-1}G')u + u'(W + GH^{-1}q) + \frac{1}{2}q'H^{-1}q \\ \text{subj. to } u \geq 0 & \end{array} \tag{4.24}$$

4.2.3 KKT condition for QP

Consider the QP (4.18). Then, $\nabla f(z) = Hz + q$, $g_i(z) = G_i'z - W_i$ (where G_i' is the i-th row of G), $\nabla g_i(z) = G_i$. The KKT conditions become

$$Hz + q + G'u = 0 \tag{4.25a}$$
$$u_i(G_i'z - W_i) = 0 \tag{4.25b}$$
$$u \geq 0 \tag{4.25c}$$
$$Gz - W \leq 0 \tag{4.25d}$$

4.2.4 Active Constraints and Degeneracies

Consider the definition of active set $A(z)$ in (4.8). The cardinality of $A(z^*)$ depends on the cases discussed in Section (4.2.1).

Case 1. $A(z^*) = \{\emptyset\}$.
Case 2. $A(z^*)$ is a nonempty subset of $\{1, \ldots, m\}$.

We define primal and dual degeneracy as in the LP case.

Definition 4.3. The QP (4.18) is said to be primal degenerate if there exists a $z^* \in Z^*$ such that the number of active constraints at z^* is greater than the number of variables s.

Definition 4.4. The QP (4.18) is said to be dual degenerate if its dual problem is primal degenerate.

As we have assumed the Hessian H to be strictly positive definite, the primal problem has a unique optimum. However, as discussed in Section 4.1.4 the dual problem could still be primal degenerate. By considering the dual problem (4.24) it is clear that dual degeneracy cannot occur as there can be at most m active constraints on m variables.

4.2.5 Constrained Least-Squares Problems

The problem of minimizing the convex quadratic function

$$\|Az - b\|_2^2 = z'A'Az - 2b'Az + b'b \tag{4.26}$$

is an (unconstrained) QP. It arises in many fields and has many names, e.g., linear regression or least-squares approximation. From (4.20) we find the minimizer

$$z^* = (A'A)^{-1}A'b \triangleq A^\dagger b$$

When linear inequality constraints are added, the problem is called constrained linear regression or *constrained least-squares*, and there is no longer a simple analytical solution. As an example we can consider regression with lower and upper bounds on the variables, i.e.,

$$\begin{array}{ll} \min_z & \|Az - b\|_2^2 \\ \text{subj. to} & l_i \le z_i \le u_i, \quad i = 1, \dots, n, \end{array} \tag{4.27}$$

which is a QP. In chapter 6.3.1 we will show how to compute an analytical solution to the constrained least-squares problem.

4.3 Mixed-Integer Optimization

As discussed in Section 1.1.2, if the decision set Z in the optimization problem (1.2) is the Cartesian product of an integer set and a real Euclidian space, i.e., $Z \subseteq \{[z_c, z_b] : z_c \in \mathbb{R}^{s_c}, z_b \in \{0,1\}^{s_b}\}$, then the optimization problem is said to be *mixed-integer*. In this section Mixed Integer Linear Programming (MILP) and Mixed Integer Quadratic Programming (MIQP) are introduced.

4.3.1 Mixed Integer Linear Programming

When the cost and the constraints of the optimization problem (1.10) are affine, then the problem is called *mixed integer linear program* (MILP). The form of an MILP used in this book is

$$\begin{array}{ll} \inf_{[z_c,z_b]} & c_c'z_c + c_b'z_b + d \\ \text{subj. to} & G_c z_c + G_b z_b \le W \\ & z_c \in \mathbb{R}^{s_c}, \ z_b \in \{0,1\}^{s_b} \end{array} \tag{4.28}$$

where $G_c \in \mathbb{R}^{m \times s_c}$, $G_b \in \mathbb{R}^{m \times s_b}$, $W \in \mathbb{R}^m$. MILPs with equality and inequality constraints take the form:

$$\begin{array}{ll} \inf_{[z_c,z_b]} & c_c'z_c + c_b'z_b + d \\ \text{subj. to} & G_c z_c + G_b z_b \le W \\ & G_{eq,c} z_c + G_{eq,b} z_b = W_{eq} \\ & z_c \in \mathbb{R}^{s_c}, \ z_b \in \{0,1\}^{s_b} \end{array} \tag{4.29}$$

where $G_{eq,b} \in \mathbb{R}^{p \times s_b}$, $W_{eq} \in \mathbb{R}^p$. Mixed integer linear programs are nonconvex optimization problems, in general. Often the term d is omitted from the cost since it does not affect the optimizer, but d has to be considered when computing the optimal value. Form (4.29) can always be translated into the form (4.28) by standard and simple manipulations.

For a fixed integer value $\bar{z}_b$ of z_b, the MILP (4.28) becomes a linear program:

$$\begin{array}{ll} \inf_{z_c} & c_c' z_c + (c'b\bar{z}_b + d) \\ \text{subj. to} & G_c z_c \leq W - G_b \bar{z}_b \\ & z_c \in \mathbb{R}^{s_c} \end{array} \tag{4.30}$$

Therefore, the simplest way to interpret and solve an MILP is to enumerate the 2^{s_b} integer values of the variable z_b and solve the corresponding LPs. By comparing the 2^{s_b} optimal costs one can find the optimizer and the optimal cost of the MILP (4.29). Although this approach is not used in practice, it gives a simple way for proving what is stated next.

Let $\mathcal{P}_{\hat{z}_b}$ be the feasible set (1.6) of problem (4.28) for a fixed $z_b = \hat{z}_b$. The cost is an affine function defined over $\mathbb{R}^{s_c}$ and $\mathcal{P}_{\hat{z}_b}$ is a polyhedron defined by the inequality constraints

$$G_c z_c \leq W - G_b \hat{z}_b \tag{4.31}$$

Denote by J^* the optimal value and by Z^* the set of optimizers of problem (4.28)

$$Z^* = \arg\min_{z_b \in \{0,1\}^{s_b}, z_c \in \mathcal{P}_{z_b}} \; c_c' z_c$$

If $\mathcal{P}_{z_b}$ is empty for all z_b, then problem (4.28) is infeasible. Five cases can occur if $\mathcal{P}_{z_b}$ is not empty for at last one $z_b \in \{0,1\}^{s_b}$:

Case 1. The MILP solution is unbounded, i.e., $J^* = -\infty$.
Case 2. The MILP solution is bounded, i.e., $J^* > -\infty$ and the optimizer is unique. Z^* is a singleton.
Case 3. The MILP solution is bounded and there are infinitely many optima corresponding to the same integer value. Z^* is the Cartesian product of an infinite dimensional subset of $\mathbb{R}^s$ and an integer number z_b^*.
Case 4. The MILP solution is bounded and there are finitely many optima corresponding to different integer values. Z^* is a finite set of optimizers $\{(z_{1,c}^*, z_{1,b}^*), \ldots, (z_{N,c}^*, z_{N,b}^*)\}$.
Case 5. The union of Case 3 and Case 4.

4.3.2 Mixed Integer Quadratic Programming

When the cost of the optimization problem (1.10) is quadratic and the constraints are affine, then the problem is called *mixed integer quadratic program* (MIQP). The most general form of an MIQP is

$$\begin{array}{ll} \inf_{[z_c,z_b]} & \frac{1}{2}z'Hz+q'z+r \\ \text{subj. to} & G_c z_c + G_b z_b \leq W \\ & G_{eq,c} z_c + G_{eq,b} z_b = W_{eq} \\ & z_c \in \mathbb{R}^{s_c},\ \ z_b \in \{0,1\}^{s_b} \\ & z=[z_c,z_b],\, s=s_c+s_d \end{array} \tag{4.32}$$

where $H \succeq 0 \in \mathbb{R}^{s\times s}$, $G_c \in \mathbb{R}^{m\times s_c}$, $G_b \in \mathbb{R}^{m\times s_b}$, $W \in \mathbb{R}^m$, $G_{eq,c} \in \mathbb{R}^{p\times s_c}$, $G_{eq,b} \in \mathbb{R}^{p\times s_b}$, $W_{eq} \in \mathbb{R}^p$. Mixed integer quadratic programs are nonconvex optimization problems, in general. Often the term d is omitted from the cost since it does not affect the optimizer, but d has to be considered when computing the optimal value. In this book we will often use the form of MIQP with inequality constraints only

$$\begin{array}{ll} \inf_{[z_c,z_b]} & \frac{1}{2}z'Hz+q'z+r \\ \text{subj. to} & G_c z_c + G_b z_b \leq W \\ & z_c \in \mathbb{R}^{s_c},\ \ z_b \in \{0,1\}^{s_b} \\ & z=[z_c,z_b],\, s=s_c+s_d \end{array} \tag{4.33}$$

The general form can always be translated into the form with inequality constraints only by standard and simple manipulations.

For a fixed integer value $\bar{z}_b$ of z_b, the MIQP (4.32) becomes a quadratic program:

$$\begin{array}{ll} \inf_{[z_c]} & \frac{1}{2}z_c'H_c z_c+q_c'z+k \\ \text{subj. to} & G_c z_c \leq W - G_b\bar{z}_b \\ & G_{eq,c} z_c = W_{eq} - G_{eq,b}\bar{z}_b \\ & z_c \in \mathbb{R}^{s_c} \end{array} \tag{4.34}$$

Therefore the simplest way to interpret and solve an MIQP is to enumerate all the 2^{s_b} integer values of the variable z_b and solve the corresponding QPs. By comparing the 2^{s_b} optimal costs one can derive the optimizer and the optimal cost of the MIQP (4.32). Although this approach is not used in practice, it gives a simple way for proving what is stated next. Let $\mathcal{P}_{\hat{z}_b}$ be the feasible set (1.6) of problem (4.33) for a fixed $z_b = \hat{z}_b$. The cost is a quadratic function defined over $\mathbb{R}^{s_c}$ and $\mathcal{P}_{\hat{z}_b}$ is a polyhedron defined by the inequality constraints

$$G_c z_c \leq W - G_b\hat{z}_b \tag{4.35}$$

Denote by J^* the optimal value and by Z^* the set of optimizers of problem (4.28)

$$Z^* = \arg\min_{z_b\in\{0,1\}^{s_b}, z_c\in\mathcal{P}_{z_b}} \ \frac{1}{2}z_c'H_c z_c + q_c'z$$

If $\mathcal{P}_{z_b}$ is empty for all z_b, then the problem (4.33) is infeasible. Five cases can occur if $\mathcal{P}_{z_b}$ is not empty for at last one $z_b \in \{0,1\}^{s_b}$:

Case 1. The MIQP solution is unbounded, i.e., $J^* = -\infty$. This cannot happen if $H \succ 0$.

Case 2. The MIQP solution is bounded, i.e., $J^* > -\infty$ and the optimizer is unique. Z^* is a singleton.

Case 3. The MIQP solution is bounded and there are infinitely many optima corresponding to the same integer value. Z^* is the Cartesian product of an infinite dimensional subset of $\mathbb{R}^s$ and an integer number z_b^*. This cannot happen if $H \succ 0$.

Case 4. The MIQP solution is bounded and there are finitely many optima corresponding to different integer values. Z^* is a finite set of optimizers $\{(z_{1,c}^*, z_{1,b}^*), \dots, (z_{N,c}^*, z_{N,b}^*)\}$.

Case 5. The union of Case 3 and Case 4.

Part II
Multiparametric Programming

Chapter 5
General Results for Multiparametric Nonlinear Programs.

In this chapter we introduce the concept of multiparametric programming and we recall the main results of nonlinear multiparametric programming. Then, in Chapter 6 we describe three algorithms for solving multiparametric linear programs (mp-LP), multiparametric quadratic programs (mp-QP) and multiparametric mixed-integer linear programs (mp-MILP).

Consider the mathematical program

$$\begin{aligned} J^*(x) = \inf_z \ & J(z,x) \\ \text{subj. to } & g(z,x) \leq 0 \end{aligned}$$

where z is the optimization vector and x is a vector of parameters. We are interested in studying the behavior of the value function $J^*(x)$ and the optimizer $z^*(x)$ as we vary the parameter x.

The operations research community has addressed parameter variations in mathematical programs at two levels: *sensitivity analysis*, which characterizes the change of the solution with respect to small perturbations of the parameters, and *parametric programming*, where the characterization of the solution for a full range of parameter values is sought. More precisely, programs where x is scalar are referred to as *parametric programs*, while programs where x is a vector are referred to as *multiparametric programs*.

There are several reasons to look for efficient solvers of multiparametric programs. Typically, mathematical programs are affected by uncertainties due to factors that are either unknown or that will be decided later. Parametric programming systematically subdivides the space of parameters into characteristic regions, which depict the feasibility and corresponding performance as a function of the uncertain parameters, and hence provide the decision maker with a complete map of various outcomes.

Our interest in multiparametric programming arises from the field of system theory and optimal control. For discrete-time dynamical systems, finite time constrained optimal control problems can be formulated as mathematical programs where the cost function and the constraints are functions of the initial state of the dynamical system. In particular, Zadeh and Whalen [267] appear to have been the first ones to express the optimal control problem for constrained discrete-time linear systems as a linear program. We can interpret the initial state as a parameter. By using multiparametric programming we can characterize and compute the solution of the optimal control problem explicitly as a function of the initial state.

We are further motivated by the *model predictive control* (MPC) technique. MPC is very popular in the process industry for the automatic regulation of process units under operating constraints, and has attracted a considerable research effort in the last two decades. MPC requires an optimal control problem to be solved on-line in order to compute the next command action. This mathematical program depends on the current sensor measurements. The computation effort can be moved off-line by solving multiparametric programs, where the control inputs are the optimization variables and the measurements are the parameters. The solution of the parametric program problem is a *control law* describing the control inputs as function of the measurements. Model Predictive Control and its multiparametric solution are discussed in Chapter 11.

In the following will present several examples that illustrate the parametric programming problem and hint at some of the issues that need to be addressed by the solvers.

Example 5.1. Consider the parametric quadratic program

$$\begin{aligned} J^*(x) = \min_z \; & J(z,x) = \tfrac{1}{2}z^2 + 2xz \\ \text{subj. to } & z \le 1 + x, \end{aligned}$$

where $x \in \mathbb{R}$. Our goals are:

1. to find $z^*(x) = \arg\min_z J(z,x)$,
2. to find all x for which the problem has a solution, and
3. to compute the value function $J^*(x)$.

The Lagrangian function is

$$L(z,x,u) = \frac{1}{2}z^2 + 2xz + u(z - x - 1)$$

and the KKT conditions are (see Section 4.2.3 for KKT conditions for quadratic programs)

$$z + 2x + u = 0 \tag{5.1a}$$
$$u(z - x - 1) = 0 \tag{5.1b}$$
$$u \ge 0 \tag{5.1c}$$
$$z - x - 1 \le 0 \tag{5.1d}$$

Consider (5.1) and the two strictly complementary cases:

$$\begin{array}{ll} \text{A.} & \begin{array}{l} z + 2x + u = 0 \\ z - x - 1 = 0 \\ u > 0 \end{array} \;\Rightarrow \begin{cases} z^* = x + 1 \\ J^* = \frac{5}{2}x^2 + 3x + \frac{1}{2} \\ x < -\frac{1}{3} \end{cases} \\[2ex] \text{B.} & \begin{array}{l} z + 2x + u = 0 \\ z - x - 1 \le 0 \\ u = 0 \end{array} \;\Rightarrow \begin{cases} z^* = -2x \\ J^* = -2x^2 \\ x \ge -\frac{1}{3} \end{cases} \end{array} \tag{5.2}$$

This solution is depicted in Figure 5.1.

The above simple procedure, which required nothing but the solution of the KKT conditions, yielded the optimizer $z^*(x)$ and the value function $J^*(x)$ for all values of the parameter x. The set of admissible parameters values was divided into two *critical regions*, defined by $x < -\frac{1}{3}$ and $x \ge -\frac{1}{3}$. In the

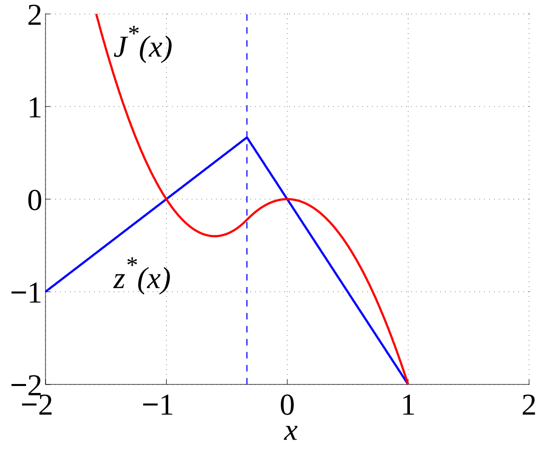


Fig. 5.1 Example 5.1: Optimizer $z^*(x)$ and value function $J^*(x)$ as a function of the parameter x.

region $x < -\frac{1}{3}$ the inequality constraint is active and the Lagrange multiplier is greater than zero, in the other region $x \geq -\frac{1}{3}$ the Lagrange multiplier is equal to zero and the inequality constraint is not active (with the exception of the boundary point $-\frac{1}{3}$). In general when there are more than one inequality constraints a critical region is defined by the set of inequalities that are active in the region. Throughout a critical region the conditions for optimality derived from the KKT conditions do not change. For our example, in each critical region the optimizer $z^*(x)$ is affine and the value function $J^*(x)$ is quadratic. Thus, considering all x, $z^*(x)$ is piecewise affine and $J^*(x)$ is piecewise quadratic. Both $z^*(x)$ and $J^*(x)$ are continuous, but $z^*(x)$ is not continuously differentiable.

In much of this book we will be interested in two questions: how to find the value function $J^*(x)$ and the optimizer $z^*(x)$ and what are their structural properties, e.g., continuity, differentiability and convexity. Such questions have been addressed for general nonlinear multiparametric programming by several authors in the past (see [17] and references therein), by making use of quite involved mathematical theory based on the continuity of point-to-set maps. The concept of point-to-set maps will *not* be extensively used in this book. However, it represents a key element for a rigorous mathematical description of the properties of a nonlinear multiparametric program and hence a few key theoretical results for nonlinear multiparametric programs based on the point-to-set map formalism will be recalled in this chapter.

5.1 General Results for Multiparametric Nonlinear Programs

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

$$\begin{aligned} J^*(x) = \inf_z \; & J(z,x) \\ \text{subj. to } & g(z,x) \le 0 \end{aligned} \tag{5.3}$$

where $z \in \mathcal{Z} \subseteq \mathbb{R}^s$ is the optimization vector, $x \in \mathcal{X} \subseteq \mathbb{R}^n$ is the parameter vector, $f : \mathbb{R}^s \times \mathbb{R}^n \to \mathbb{R}$ is the cost function and $g : \mathbb{R}^s \times \mathbb{R}^n \to \mathbb{R}^{n_g}$ are the constraints. We denote by $g_i(z,x)$ the i-th component of the vector-valued function $g(x,z)$.

A small perturbation of the parameter x in the mathematical program (5.3) can cause a variety of results. 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 R the point-to-set map which assigns to a parameter $x \in \mathcal{X}$ the (possibly empty) *set* $R(x)$ *of feasible variables* $z \in \mathcal{Z}$, $R : \mathcal{X} \mapsto 2^{\mathcal{Z}}$

$$R(x) = \{z \in \mathcal{Z} : \; g(z,x) \le 0\}, \tag{5.4}$$

by $\mathcal{K}^*$ the set of feasible parameters

$$\mathcal{K}^* = \{x \in \mathcal{X} : \; R(x) \neq \emptyset\}, \tag{5.5}$$

by $J^*(x)$ the real-valued function that expresses the dependence of the minimum value of the objective function over $\mathcal{K}^*$ on x

$$J^*(x) = \inf_z \{J(z,x) : \; z \in R(x)\}, \tag{5.6}$$

and by $Z^*(x)$ the point-to-set map which assigns the (possibly empty) set of optimizers $z^* \in 2^{\mathcal{Z}}$ to a parameter $x \in \mathcal{X}$

$$Z^*(x) = \{z \in R(x) : \; J(z,x) \le J^*(x)\}. \tag{5.7}$$

$J^*(x)$ will be referred to as optimal value function or simply *value function*, $Z^*(x)$ will be referred to as *optimal set*. If $Z^*(x)$ is a singleton for all x, then $z^*(x) \triangleq Z^*(x)$ will be called *optimizer function*. We remark that R and Z^* are set-valued functions. As discussed in the notation section, with abuse of notation $J^*(x)$ and $Z^*(x)$ will denote both the functions and the value of the functions at the point x. The context will make clear which is being discussed

The book by Bank and coauthors [17] and Chapter 2 of [100] describe conditions under which the solution of the nonlinear multiparametric program (5.3) is locally well behaved and establish properties of the optimal value function and of the optimal set. The description of such conditions requires the definition of continuity of point-to-set maps. Before introducing this concept we will show through two simple examples that continuity of the constraints $g_i(z,x)$ with respect to z and x is not enough to imply any regularity of the value function and the optimizer function.

Example 5.2 ([17] page 12).
Consider the following problem:

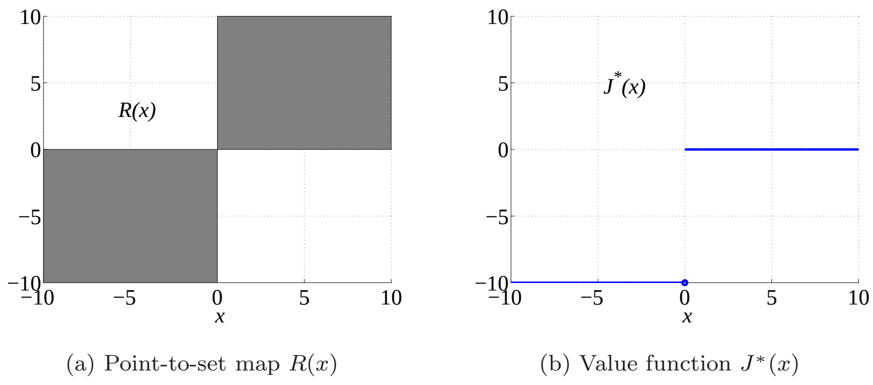
$$\begin{array}{rl} J^*(x) = \min_z & x^2z^2 - 2x(1-x)z \\ \text{subj. to} & z \geq 0 \\ & 0 \leq x \leq 1 \end{array} \tag{5.8}$$

Cost and constraints are continuous and continuously differentiable. For $0 < x \leq 1$ the optimizer function is $z^* = (1-x)/x$ and the value function is $J^*(x) = -(1-x)^2$. For $x = 0$, the value function is $J^*(x) = 0$ while the optimal set is $Z^* = \{z \in \mathbf{R} : \ z \geq 0\}$. Thus, the value function is *discontinuous* at 0 and the optimal set is single-valued for all $0 < x \leq 1$ and set-valued for $x = 0$.

Example 5.3. Consider the following problem:

$$\begin{array}{rl} J^*(x) = \inf\limits_z & z \\ \text{subj. to} & zx \geq 0 \\ & -10 \leq z \leq 10 \\ & -10 \leq x \leq 10 \end{array} \tag{5.9}$$

where $z \in \mathbb{R}$ and $x \in \mathbb{R}$. For each fixed x the set of feasible z is a segment. The point-to-set map $R(x)$ is plotted in Figure 5.2a. The function $g_1 : (z,x) \mapsto zx$ is continuous. Nevertheless, the value function $J^*(x) = z^*(x)$ has a discontinuity at the origin as can be seen in Figure 5.2b.



(a) Point-to-set map $R(x)$ (b) Value function $J^*(x)$

Fig. 5.2 Example 5.3: Solution.

Example 5.4. Consider the following problem:

$$\begin{array}{rl} J^*(x) = \inf\limits_{z_1,z_2} & -z_1 \\ \text{subj. to} & g_1(z_1,z_2)+x \leq 0 \\ & g_2(z_1,z_2)+x \leq 0 \end{array} \tag{5.10}$$

where examples of the functions $g_1(z_1,z_2)$ and $g_2(z_1,z_2)$ are plotted in Figures 5.3(a)–5.3(c). Figures 5.3(a)–5.3(c) also depict the point-to-set map $R(x) = \{[z_1,z_2] \in \mathbb{R}^2 | g_1(z_1,z_2)+x \leq 0,\ g_2(z_1,z_2)+x \leq 0\}$ for three fixed x. Starting from $x = \bar{x}_1$, as x increases, the domain of feasibility in the space z_1, z_2 shrinks; at the beginning it is connected (Figure 5.3(a)), then it becomes disconnected (Figure 5.3(b)) and eventually connected again (Figure 5.3(c)). No matter how smooth one chooses the functions g_1 and g_2, the value function $J^*(x) = -z_1^*(x)$ will have a discontinuity at $x = \bar{x}_3$.

Examples 5.2, 5.3, 5.4 show the case of simple and smooth constraints which lead to discontinuous behavior of value function and optimal set. We anticipate here the main causes:

- in example 5.2 the feasible vector space $\mathcal{Z}$ is unbounded ($z \geq 0$),
- in examples 5.3 and 5.4 the feasible point-to-set map $R(x)$ (defined in (5.4)) is discontinuous, as precisely defined below.

In the next sections we discuss both cases in detail.

Continuity of Point-to-Set Maps

Consider a point-to-set map $R : x \in \mathcal{X} \mapsto R(x) \in 2^{\mathcal{Z}}$. We give the following definitions of open and closed map according to Hogan [140]:

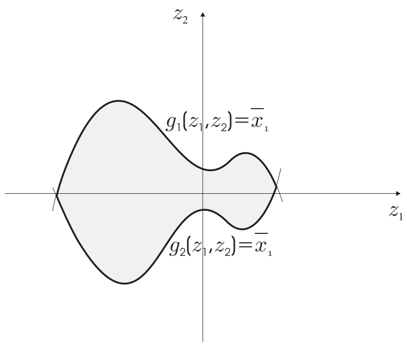
Definition 5.1. The point-to-set map $R(x)$ is open at a point $\bar{x} \in \mathcal{K}^*$ if for all sequences $\{x^k\} \subset \mathcal{K}^*$ with $x^k \to \bar{x}$ and for all $\bar{z} \in R(\bar{x})$ there exists an integer m and a sequence $\{z^k\} \in \mathcal{Z}$ such that $z^k \in R(x^k)$ for $k \geq m$ and $z^k \to \bar{z}$

Definition 5.2. The point-to-set map $R(x)$ is closed at a point $\bar{x} \in \mathcal{K}^*$ if for each pair of sequences $\{x^k\} \subset \mathcal{K}^*$, and $z^k \in R(x^k)$ with the properties

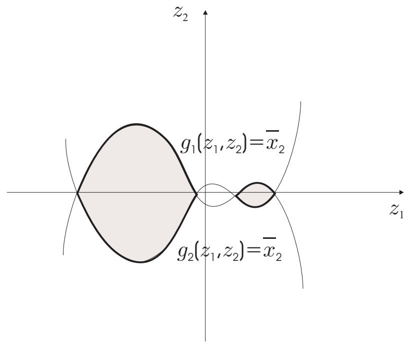
$$x^k \to \bar{x},\ z^k \to \bar{z}.$$

it follows that $\bar{z} \in R(\bar{x})$.

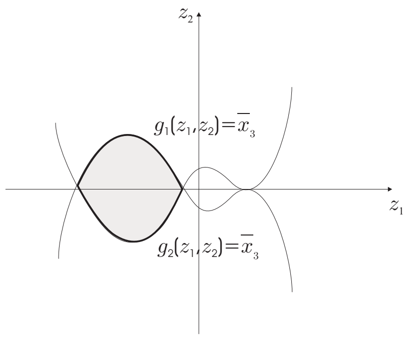
We define the continuity of a point-to-set map according to Hogan [140] as follows:



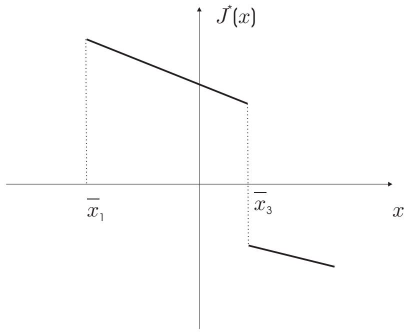
(a) Set $R(\bar{x}_1)$ shaded in gray.



(b) Set $R(\bar{x}_2)$ shaded in gray.



(c) Set $R(\bar{x}_3)$ shaded in gray.



(d) Value function $J^*(x)$

Fig. 5.3 Example 5.4: problem (5.10). (a)-(c) Projections of the point-to-set map $R(x)$ for three values of the parameter x: $\bar{x}_1 < \bar{x}_2 < \bar{x}_3$; (d) Value function $J^*(x)$.

Definition 5.3. The point-to-set map $R(x)$ is continuous at a point $\bar{x}$ in $\mathcal{K}^*$ if it is both open and closed at $\bar{x}$. $R(x)$ is continuous in $\mathcal{K}^*$ if $R(x)$ is continuous at any point x in $\mathcal{K}^*$.

The definitions above are illustrated through two examples.

Example 5.5. Consider

$$R(x) = \{z \in \mathbb{R} : \ z \in [0,1] \text{ if } x < 1,\ z \in [0, 0.5] \text{ if } x \geq 1\}$$

The point-to-set map $R(x)$ is plotted in Figure 5.4. It easy to see that $R(x)$ is not closed but open. In fact, if one considers a sequence $\{x^k\}$ that converges to $\bar{x} = 1$ from the left and extracts the sequence $\{z^k\}$ plotted in Figure 5.4 converging to $\bar{z} = 0.75$, then $\bar{z} \notin R(\bar{x})$ since $R(1) = [0, 0.5]$.

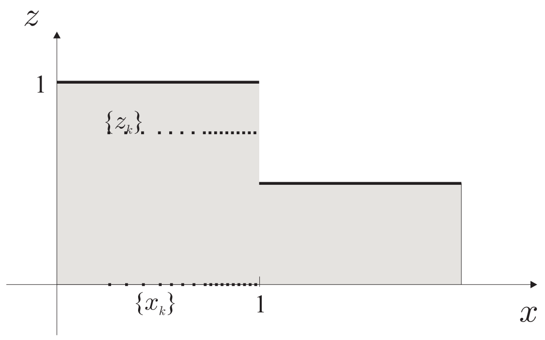


Fig. 5.4 Example 5.5: Open and not closed point-to-set map $R(x)$

Example 5.6. Consider

$$R(x) = \{z \in \mathbb{R} \ : \ z \in [0,1] \quad \text{if } x \le 1, \ z \in [0,0.5] \text{ if } x > 1\}$$

The point-to-set map $R(x)$ is plotted in Figure 5.5. It easy to verify that $R(x)$ is closed but not open. Choose $\bar{z} = 0.75 \in R(\bar{x})$. Then, for any sequence $\{x^k\}$ that converges to $\bar{x} = 1$ from the right, one is not able to construct a sequence $\{z^k\} \in \mathcal{Z}$ such that $z^k \in R(x^k)$ for and $z^k \to \bar{z}$. In fact, such sequence z^k will always be bounded between 0 and 0.5.

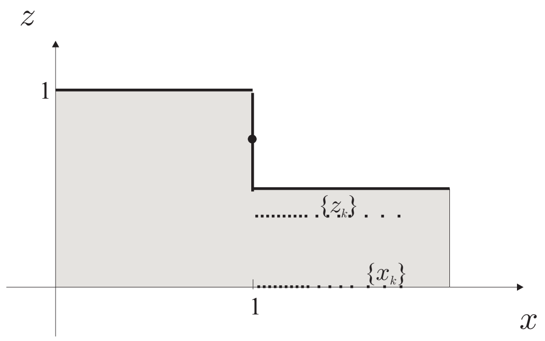


Fig. 5.5 Example 5.6: Closed and not open point-to-set map $R(x)$

Remark 5.1. We remark that "upper semicontinuous" and "lower semicontinuous" definitions of point-to-set map are sometimes preferred to open and closed definitions [43, page 109]. In [17, page 25] nine different definitions for the continuity of point-to-set maps are introduced and compared. We will not give any details on this subject and refer the interested reader to [17, page 25].

The examples above are only illustrative. In general, it is difficult to test if a set is closed or open by applying the definitions. Several authors have proposed sufficient conditions on g_i which imply the continuity of $R(x)$. In the following we introduce a theorem which summarizes the main results of [227, 84, 140, 43, 17].

Theorem 5.1. *If $\mathcal{Z}$ is convex, if each component $g_i(z, x)$ of $g(z, x)$ is continuous on $\mathcal{Z} \times \bar{x}$ and convex in z for each fixed $x \in \mathcal{X}$ and if there exists a $\bar{z}$ such that $g(\bar{z}, \bar{x}) < 0$, then $R(x)$ is continuous at $\bar{x}$.*

The proof is given in [140, Theorems 10 and 12]. An equivalent proof can be also derived from [17, Theorem 3.1.1 and Theorem 3.1.6]. $\square$

Remark 5.2. Note that convexity in z for each x is not enough to imply the continuity of $R(x)$ everywhere in $\mathcal{K}^*$. In [17, Example 3.3.1 on page 53] an example illustrating this is presented. We remark that in Example 5.3 the origin does not satisfy the last hypothesis of Theorem 5.1.

Remark 5.3. If the assumptions of Theorem 7.1 hold at each $\bar{x} \in \mathcal{X}$ then one can extract a continuous single-valued function (often called "continuous selection") $r : \mathcal{X} \mapsto \mathbb{R}$ such that $r(x) \in R(x)$, $\forall x \in \mathcal{X}$, provided that $\mathcal{Z}$ is finite-dimensional. Note that convexity of $R(x)$ is a critical assumption [17, Corollary 2.3.1]. The following example shows a point-to-set map $R(x)$ not convex for a fixed x which is continuous but has no continuous selection [17, pag. 29]. Let Λ be the unit disk in $\mathbb{R}^2$, define $R(x)$ as

$$x \in \Lambda \mapsto R(x) \triangleq \{z \in \Lambda : \ \|z - x\|_2 \geq \frac{1}{2}\} \tag{5.11}$$

It can be shown that the point-to-set map $R(x)$ in (5.11) is continuous according to Definition 5.3. In fact, for a fixed $\bar{x} \in \Lambda$ the set $R(\bar{x})$ is the set of points in the unit disk outside the disk centered in $\bar{x}$ and of radius 0.5 (next called the *half disk*); small perturbations of $\bar{x}$ yield small translations of the half disk inside the unit disk for all $\bar{x} \in \Lambda$. However $R(x)$ has no continuous selection. Assume that there exists a continuous selection $r : x \in \Lambda \mapsto r(x) \in \Lambda$. Then, there exists a point x^* such that $x^* = r(x^*)$. Since $r(x) \in R(x)$, $\forall x \in \Lambda$, there exists a point x^* such that $x^* \in R(x^*)$. This is not possible since for all $x^* \in \Lambda$, $x^* \notin R(x^*)$ (recall that $R(x^*)$ is set of points in the unit disk outside the disk centered in x^* and of radius 0.5).

Remark 5.4. Let Λ be the unit disk in $\mathbb{R}$, define $R(x)$ as

$$x \in \Lambda \mapsto R(x) \triangleq \{z \in \Lambda : \ |z - x| \geq \frac{1}{2}\}. \tag{5.12}$$

$R(x)$ is closed and not open and it has no continuous selection.

Remark 5.5. Based on [17, Theorem 3.2.1-(I) and Theorem 3.3.3], the hypotheses of Theorem 5.1 can be relaxed for affine $g_i(z, x)$. In fact, affine functions are weakly analytic functions according to [17, definition on page 47]. Therefore, we can state that if $\mathcal{Z}$ is convex, if each component $g_i(z,x)$ of $g(z,x)$ is an affine function, then $R(x)$ is continuous at $\bar{x}$ for all $\bar{x} \in \mathcal{K}^*$.

Properties of Value Function and Optimal Set

Consider the following definition

Definition 5.4. A point-to-set map $R(x)$ is said to be *uniformly compact* near $\bar{x}$ if there exist a neighborhood N of $\bar{x}$ such that the closure of the set $\bigcup_{x\in N} R(x)$ is compact.

Now we are ready to state the two main theorems on the continuity of the value function and of the optimizer function.

Theorem 5.2. *[140, Theorem 7] Consider problem (5.3)–(5.4). If $R(x)$ is a continuous point-to-set map at $\bar{x}$ and uniformly compact near $\bar{x}$ and if J is continuous on $\bar{x} \times R(\bar{x})$, then J^* is continuous at $\bar{x}$.*

Theorem 5.3. *[140, Corollary 8.1] Consider problem (5.3)–(5.4). If $R(x)$ is a continuous point-to-set map at $\bar{x}$, J is continuous on $\bar{x} \times R(\bar{x})$, Z^* is nonempty and uniformly compact near $\bar{x}$, and $Z^*(\bar{x})$ is single valued, then Z^* is continuous at $\bar{x}$.*

Remark 5.6. Equivalent results of Theorems 5.2 and 5.3 can be found in [43, page 116] and [17, Chapter 4.2].

Example 5.7 (Example 5.2 revisited). Consider Example 5.2. The feasible map $R(x)$ is unbounded and therefore it does not satisfy the assumptions of Theorem 5.2 (since it is not uniformly compact). Modify Example 5.2 as follows:

$$\begin{array}{rl} J^*(x) = \min_z & x^2z^2 - 2x(1-x)z \\ \text{subj. to} & 0 \le z \le M \\ & 0 \le x \le 1 \end{array} \tag{5.13}$$

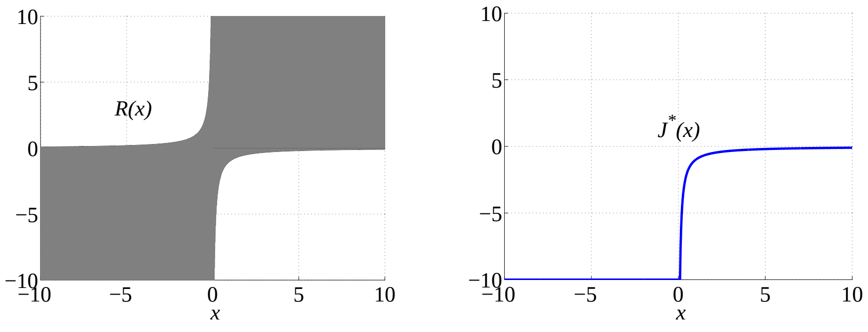
with $M \ge 0$. The solution can be computed immediately. For $1/(1+M) < x \le 1$ the optimizer function is $z^* = (1-x)/x$ and the value function is $J^*(x) = -(1-x)^2$. For $0 < x \le 1/(1+M)$, the value function is $J^*(x) = x^2M^2 - 2x(1-x) * M$ and the optimizer function is $z^* = M$. For $x = 0$, the value function is $J^*(x) = 0$ while the optimal set is $Z^* = \{z \in \mathbf{R} :\ 0 \le z \le M\}$.

No matter how large we choose M, the value function and the optimal set are continuous for all $x \in [0,1]$.

Example 5.8 (Example 5.3 revisited). Consider Example 5.3. The feasible map $R(x)$ is not continuous at $x = 0$ and therefore it does not satisfy the assumptions of Theorem 5.2. Modify Example 5.3 as follows:

$$\begin{aligned} J^*(x) = \inf_z \; & z \\ \text{subj. to } & zx \geq -\varepsilon \\ & -10 \leq z \leq 10 \\ & -10 \leq x \leq 10 \end{aligned} \tag{5.14}$$

where $\varepsilon > 0$. The value function and the optimal set are depicted in Figure 5.6 for $\varepsilon = 1$. No matter how small we choose ε, the value function and the optimal set are continuous for all $x \in [-10, 10]$



(a) Point-to-set map $R(x)$ (b) Value function $J^*(x)$

Fig. 5.6 Example 5.8: Solution.

The following corollaries consider special classes of parametric problems.

Corollary 5.1 (mp-LP). *Consider the special case of the multiparametric program (5.3). where the objective and the constraints are linear*

$$\begin{aligned} J^*(x) = \min_z \; & c'z \\ \textit{subj. to } & Gz \leq W + Sx, \end{aligned} \tag{5.15}$$

and assume that there exists an $\bar{x}$ and a finite $z^*(\bar{x})$. Then, $\mathcal{K}^*$ is a nonempty polyhedron, $J^*(x)$ is a continuous and convex function in $\mathcal{K}^*$ and the optimal set $Z^*(x)$ is a continuous point-to-set map in $\mathcal{K}^*$.

Proof: See Theorem 5.5.1 in [17] and the bottom of page 138 in [17]. $\square$

Corollary 5.2 (mp-QP). *Consider the special case of the multiparametric program (5.3). where the objective is quadratic and the constraints are linear*

$$\begin{aligned} J^*(x) = \min_z \quad & \tfrac{1}{2} z'Hz + z'F \\ \textit{subj. to } & Gz \leq W + Sx, \end{aligned} \tag{5.16}$$

and assume that $H \succ 0$ *and that there exists* $(\bar{z}, \bar{x})$ *such that* $G\bar{z} \leq W + S\bar{x}$*. Then,* $\mathcal{K}^*$ *is a nonempty polyhedron,* $J^*(x)$ *is a continuous and convex function in* $\mathcal{K}^*$ *and the optimizer function* $z^*(x)$ *is continuous in* $\mathcal{K}^*$*.*

Proof: See Theorem 5.5.1 in [17] and the bottom of page 138 in [17]. □

Remark 5.7. We remark that Corollary 5.1 requires the existence of a *finite* optimum $z^*(\bar{x})$. This is implicitly guaranteed in the mp-QP case since in Corollary 5.2 the matrix H is assumed to be strictly positive definite. Moreover, the existence of a *finite* optimum $z^*(\bar{x})$ guarantees that $J^*(x)$ is finite for all x in $\mathcal{K}^*$. This has been proven in [106, page 178, Theorem 1] for the mp-LP case and it is immediate to prove for the mp-QP case.

Remark 5.8. Both Corollary 5.1 (mp-LP) and Corollary 5.2 (mp-QP) could be formulated stronger: J^* and Z^* are even Lipschitz-continuous. J^* is also piecewise affine (mp-LP) or piecewise quadratic (mp-QP), and for the mp-QP $z^*(x)$ is piecewise affine. For the linear case, Lipschitz continuity is known from Walkup-Wets [259] as a consequence of Hoffman's theorem. For the quadratic case, Lipschitz continuity follows from Robinson [226], as e.g. shown in Klatte-Thiere [162]. The "piecewise" properties are consequences of local stability analysis of parametric optimization, e.g. [100, 17, 183] and are the main focus of the next chapter.

Chapter 6
Multiparametric Programming: a Geometric Approach

In this chapter we will concentrate on multiparametric linear programs (mp-LP), multiparametric quadratic programs (mp-QP) and multiparametric mixed-integer linear programs (mp-MILP).

The main idea of the multiparametric algorithms presented in this chapter is to construct a critical region in a neighborhood of a given parameter, by using necessary and sufficient conditions for optimality, and then to recursively explore the parameter space outside such a region. For this reason the methods are classified as "geometric". All the algorithms are easy to implement once standard solvers are available: linear programming, quadratic programming and mixed-integer linear programming for solving mp-LP, mp-QP and mp-MILP, respectively. A literature review is presented in Section 6.6.

6.1 Multiparametric Programs with Linear Constraints

6.1.1 Formulation

Consider the multiparametric program

$$\begin{aligned} J^*(x) = \min_{z} \; & J(z,x) \\ \text{subj. to } & Gz \le W + Sx, \end{aligned} \tag{6.1}$$

where $z \in \mathbb{R}^s$ are the optimization variables, $x \in \mathbb{R}^n$ is the vector of parameters, $J(z,x) : \mathbb{R}^{s+n} \to \mathbb{R}$ is the objective function and $G \in \mathbb{R}^{m \times s}$, $W \in \mathbb{R}^m$, and $S \in \mathbb{R}^{m \times n}$. Given a closed and bounded polyhedral set $\mathcal{K} \subset \mathbb{R}^n$ of parameters,

$$\mathcal{K} \triangleq \{x \in \mathbb{R}^n : \; Tx \le N\}, \tag{6.2}$$

we denote by $\mathcal{K}^* \subseteq \mathcal{K}$ the region of parameters $x \in \mathcal{K}$ such that (6.1) is feasible:

$$\mathcal{K}^* = \{x \in \mathcal{K} : \exists z \text{ satisfying } Gz \le W + Sx\} \tag{6.3}$$

In this book we assume that

1. constraint $x \in \mathcal{K}$ to be included into the constraints $Gz \le W + Sx$,
2. the polytope $\mathcal{K}$ is full dimensional. Otherwise we can reformulate the problem with a smaller set of parameters such that $\mathcal{K}$ becomes full dimensional.
3. S has full column rank. Otherwise we can again reformulate the problem in a smaller set of parameters.

Proposition 6.1. *Consider the multiparametric problem (6.1). If the domain of $J(z,x)$ is $\mathbb{R}^{s+n}$ then $\mathcal{K}^*$ is a polytope.*

Proof: $\mathcal{K}^*$ is the projection of the set $Gz - Sx \le W$ on the x space intersected with the polytope $\mathcal{K}$. □

For any given $\bar{x} \in \mathcal{K}^*$, $J^*(\bar{x})$ denotes the minimum value of the objective function in problem (6.1) for $x = \bar{x}$. The function $J^* : \mathcal{K}^* \to \mathbb{R}$ expresses the dependence of the minimum value of the objective function on x, $J^*(x)$ is called value function. The set-valued function $Z^* : \mathcal{K}^* \to 2^{\mathbb{R}^s}$, where $2^{\mathbb{R}^s}$ is the set of subsets of $\mathbb{R}^s$, describes for any fixed $x \in \mathcal{K}^*$ the set $Z^*(x)$ of optimizers $z^*(x)$ yielding $J^*(x)$.

We aim to determine the feasible region $\mathcal{K}^* \subseteq \mathcal{K}$ of parameters, the expression of the value function $J^*(x)$ and the expression of one of the optimizers $z^*(x) \in Z^*(x)$.

6.1.2 Definition of Critical Region

Consider the multiparametric program (6.1). Let $I \triangleq \{1, \ldots, m\}$ be the set of constraint indices. For any $A \subseteq I$, let G_A and S_A be the submatrices of G and S, respectively, comprising the rows indexed by A and denote with G_j, S_j and W_j the j-th row of G, S and W, respectively. We define CR_A as the set of parameters x for which the same set A of constraints is active at the optimum. More formally we have the following definitions.

Definition 6.1. The optimal partition of I at x is the partition $(A(x), NA(x))$ where

$$\begin{aligned} A(x) &\triangleq \{j \in I : G_j z^*(x) - S_j x = W_j \text{ for all } z^*(x) \in Z^*(x)\} \\ NA(x) &\triangleq \{j \in I : \text{exists } z^*(x) \in Z^*(x) \text{ satisfying } G_j z^*(x) - S_j x < W_j\}. \end{aligned}$$

It is clear that $(A(x), NA(x))$ are disjoint and their union is I.

Definition 6.2. Consider a set $A \subseteq I$. The critical region associated with the set of active constraints A is defined as

$$CR_A \triangleq \{x \in \mathcal{K}^* : A(x) = A\} \tag{6.4}$$

The set CR_A is the set of all parameters x such that the constraints indexed by A are active at the optimum of problem (6.1). Our first objective is to work with full dimensional critical regions. For this reason, next we discuss the case when the dimension of the parameter space can be reduced.

6.1.3 Reducing the Dimension of the Parameter Space

It may happen that the set of inequality constraints in (6.1) contains some “hidden” or “implicit” equality constraints as the following example shows.

Example 6.1.

$$\begin{array}{ll} & \min_z \ J(z,x) \\ \text{subj. to} & \left\{\begin{array}{l} z_1+z_2 \le 9-x_1-x_2 \\ z_1-z_2 \le 1-x_1-x_2 \\ z_1+z_2 \le 7+x_1+x_2 \\ z_1-z_2 \le -1+x_1+x_2 \\ \quad -z_1 \le -4 \\ \quad -z_2 \le -4 \\ \quad\quad z_1 \le 20-x_2 \end{array}\right. \end{array} \tag{6.5}$$

where $\mathcal{K} = \{[x_1, x_2]' \in \mathbb{R}^2 : \ -100 \le x_1 \le 100, \ -100 \le x_2 \le 100\}$. The reader can check that *all* feasible values of x_1, x_2, z_1, z_2 satisfy

$$\begin{array}{l} z_1+z_2 = 9-x_1-x_2 \\ z_1-z_2 = 1-x_1-x_2 \\ z_1+z_2 = 7+x_1+x_2 \\ z_1-z_2 = -1+x_1+x_2 \\ z_1 = 4 \\ z_2 = 4 \\ z_1 \le 20-x_2 \end{array} \tag{6.6}$$

where we have identified many of the inequalities to be hidden equalities. This can be simplified to

$$\begin{array}{l} x_1+x_2 = 1 \\ x_2 \le 16 \end{array} \tag{6.7}$$

Thus

$$\mathcal{K}^* = \left\{ [x_1, x_2]' \in \mathbb{R}^2 : \begin{array}{r} x_1+x_2 = 1 \\ -100 \le x_1 \le 100 \\ -100 \le x_2 \le 16. \end{array} \right\} \tag{6.8}$$

The example shows that the polyhedron $\mathcal{K}^*$ is contained in a lower dimensional subspace of $\mathcal{K}$, namely a line segment in $\mathbb{R}^2$.

Our goal is to identify the hidden equality constraints (as in (6.6)) and use them to reduce the dimension of the parameter space (as in (6.7)) for which the multiparametric program needs to be solved.

Definition 6.3. A *hidden equality* of the polyhedron $\mathcal{C} = \{\xi \in \mathbb{R}^s : \ B\xi \le v\}$ is an inequality $B_i\xi \le v_i$ such that $B_i\bar{\xi} = v_i \ \ \forall \bar{\xi} \in \mathcal{C}$

To find hidden equalities we need to solve

$$\begin{array}{l} v_i^* \triangleq \min \ B_i\xi \\ \quad \text{subj. to } B\xi \le v \end{array}$$

for all constraints $i = 1, \ldots, m$. If $v_i^* = v_i$, then $B_i\xi = v_i$ is a hidden equality.

We can apply this procedure with $\xi = \left[\begin{smallmatrix} x \\ z \end{smallmatrix}\right]$ to identify the hidden equalities in the set of inequalities in (6.1)

$$Gz \leq W + Sx$$

to obtain

$$G_{nh}z \leq W_{nh} + S_{nh}x \tag{6.9a}$$
$$G_h z = W_h + S_h x \tag{6.9b}$$

where we have partitioned G, W and S to reflect the hidden equalities. In order to find the equality constraints involving only the parameter x that allow us to reduce the dimension of x, we need to project the equalities (6.9b) onto the parameter space. Let the singular value decomposition of G_h be

$$G_h = [U_1 \; U_2] \, \Sigma \begin{bmatrix} V_1' \\ V_2' \end{bmatrix}$$

where the columns of U_2 are the singular vectors associated with the zero singular values. Then the matrix U_2' defines the projection of (6.9b) onto the parameter space, i.e.

$$U_2' G_h z = 0 = U_2' W_h + U_2' S_h x. \tag{6.10}$$

We can use this set of equalities to replace the parameter $x \in \mathbb{R}^n$ with a set of $n' = n - \text{rank}(U_1' S_h)$ new parameters in (6.9a) which simplifies the parametric program (6.1).

In the rest of this book we always assume that the multiparametric program has been preprocessed using the ideas of this section so that the number of independent parameters is reduced as much as possible and $\mathcal{K}^*$ is full dimensional.

6.2 Multiparametric Linear Programming

6.2.1 Formulation

Consider the special case of the multiparametric program (6.1) where the objective is linear

$$\begin{array}{rl} J^*(x) = & \min_z c'z \\ & \text{subj. to } Gz \leq W + Sx, \end{array} \tag{6.11}$$

All the variables were defined in Section 6.1.1. Our goal is to find the value function $J^*(x)$ and an optimizer function $z^*(x)$ for $x \in \mathcal{K}^*$. Note that $\mathcal{K}^*$ can be determined as discussed in Proposition 6.1. As suggested through Example 5.1 our search for these functions proceeds by partitioning the set of feasible parameters into critical regions. This is shown through a simple example next.

Example 6.2. Consider the parametric linear program

$$\begin{aligned} J^*(x) = \min_z \;& z+1 \\ \text{subj. to } & z \geq 1+x \\ & z \geq 0 \end{aligned}$$

where $z \in \mathbb{R}$ and $x \in \mathbb{R}$. Our goals are:

1. to find $z^*(x) = \arg\min_{z,\ z\geq 0,\ z\geq 1+x} z+1$,
2. to find all x for which the problem has a solution, and
3. to compute the value function $J^*(x)$.

The Lagrangian function is

$$L(z,x,u_1,u_2) = z + u_1(-z+x+1) + u_2(-z)$$

and the KKT conditions are (see Section 4.1.3 for KKT conditions for linear programs)

$$\begin{aligned} -u_1 - u_2 &= -1 && \text{(6.12a)} \\ u_1(-z+x+1) &= 0 && \text{(6.12b)} \\ u_2(-z) &= 0 && \text{(6.12c)} \\ u_1 &\geq 0 && \text{(6.12d)} \\ u_2 &\geq 0 && \text{(6.12e)} \\ -z+x+1 &\leq 0 && \text{(6.12f)} \\ -z &\leq 0 && \text{(6.12g)} \end{aligned}$$

Consider (6.12) and the three complementary cases:

$$\begin{array}{ll}
\text{A.} \begin{array}{l} u_1+u_2=1 \\ -z+x+1=0 \\ -z<0 \\ u_1>0 \\ u_2=0 \end{array} & \Rightarrow \begin{cases} z^* = 1+x \\ u_1^* = -1,\ u_2^* = 0 \\ J^* = 2+x \\ x > -1 \end{cases} \\[2ex]
\text{B.} \begin{array}{l} u_1+u_2=1 \\ -z+x+1<0 \\ -z=0 \\ u_1=0 \\ u_2>0 \end{array} & \Rightarrow \begin{cases} z^* = 0 \\ u_1^* = 0,\ u_2^* = -1 \\ J^* = 1 \\ x < -1 \end{cases} \\[2ex]
\text{C.} \begin{array}{l} u_1+u_2=1 \\ -z+x+1=0 \\ -z=0 \\ u_1\geq 0 \\ u_2\geq 0 \end{array} & \Rightarrow \begin{cases} z^* = 0 \\ u_1^* \geq 0,\ u_2 \geq 0,\ u_1^*+u_2^* = 1 \\ J^* = 1 \\ x = -1 \end{cases}
\end{array} \qquad (6.13)$$

This solution is depicted in Figure 6.1.

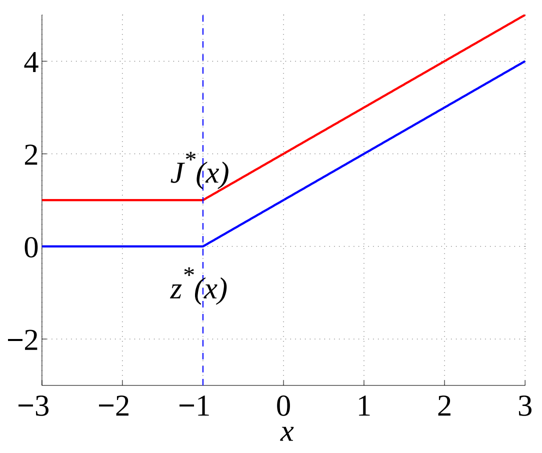


Fig. 6.1 Example 6.2: Optimizer $z^*(x)$ and value function $J^*(x)$ as a function of the parameter x.

The above simple procedure, which required nothing but the solution of the KKT conditions, yielded the optimizer $z^*(x)$ and the value function $J^*(x)$ for all values of the parameter x. The set of admissible parameters values was divided into three *critical regions*, defined by $x < -1$, $x > -1$ and $x = -1$. In the region $x < -1$ the first inequality constraint is active $(z = 1 + x)$ and the Lagrange multiplier u_1 is greater than zero, in the second region $x > -1$ the second inequality constraint is active $(z = 0)$ and the Lagrange multiplier u_2 is greater than zero. In the third region $x = -1$ both constraints are active and the Lagrange multipliers belong to the set $u_1^* \geq 0,\ u_2 \geq 0,\ u_1^* + u_2^* = 1$. Throughout a critical region the conditions for optimality derived from the KKT conditions do not change. For our example, in each critical region the optimizer $z^*(x)$ is affine and the value function $J^*(x)$ is also affine.

6.2.2 Critical Regions, Value Function and Optimizer: Local Properties

Consider the Definition 6.2 of critical region. In this section we show that critical regions of mp-LP are polyhedra. We use primal feasibility to derive the $\mathcal{H}$-polyhedral representation of the critical regions, the complementary slackness conditions to compute an optimizer $z^*(x)$, and the dual problem of (6.11) to derive the optimal value function $J^*(x)$ inside each critical region.

Consider the dual problem of (6.11):

$$\begin{aligned} \min_u \ & (W+Sx)'u \\ \text{subj. to } & G'u = -c \\ & u \geq 0. \end{aligned} \tag{6.14}$$

The dual feasibility (DF), the complementary slackness (CS) and the primal feasibility (PF), conditions for problems (6.11), (6.14) are

$$\text{DF:} \quad G'u = -c,\ u \geq 0 \tag{6.15a}$$
$$\text{CS:} \quad (G_j z - W_j - S_j x)u_j = 0,\ \forall j \in I \tag{6.15b}$$
$$\text{PF:} \quad Gz \leq W + Sx \tag{6.15c}$$

Let us assume that we have determined the optimal partition $(A, NA) \triangleq (A(x^*), NA(x^*))$ for some $x^* \in \mathcal{K}^*$. Let us calculate the critical region CR_A. The PF condition (6.15c) can be rewritten as

$$G_A z^*(x) - S_A x = W_A \tag{6.16a}$$
$$G_{NA} z^*(x) - S_{NA} x < W_{NA}. \tag{6.16b}$$

for all $x \in CR_A$. Let $l \triangleq \operatorname{rank} G_A$ and consider the QR decomposition of G_A

$$G_A = Q \begin{bmatrix} U_1 & U_2 \\ \mathbf{0}_{|A|-l\times l} & \mathbf{0}_{|A|-l\times |A|-l} \end{bmatrix}$$

where $Q \in \mathbb{R}^{|A|\times|A|}$ is a unitary matrix, $U_1 \in \mathbb{R}^{l\times l}$ is a full-rank square matrix and $U_2 \in \mathbb{R}^{l\times|A|-l}$. Let $\begin{bmatrix} P \\ D \end{bmatrix} \triangleq -Q^{-1}S_A$ and $\begin{bmatrix} q \\ r \end{bmatrix} \triangleq Q^{-1}W_A$. From (6.16a) we obtain

$$\begin{bmatrix} U_1 & U_2 & P \\ \mathbf{0}_{|A|-l\times l} & \mathbf{0}_{|A|-l\times|A|-l} & D \end{bmatrix} \begin{bmatrix} z_1^*(x) \\ z_2^*(x) \\ x \end{bmatrix} = \begin{bmatrix} q \\ r \end{bmatrix}. \tag{6.17}$$

We partition (6.16b) accordingly

$$\begin{bmatrix} E & F \end{bmatrix} \begin{bmatrix} z_1^*(x) \\ z_2^*(x) \end{bmatrix} - S_{NA} x < W_{NA} \tag{6.18}$$

Calculating $z_1^*(x)$ from (6.17) we obtain

$$z_1^*(x) = U_1^{-1}(-U_2 z_2^*(x) - Px + q), \tag{6.19}$$

which substituted in (6.18) gives:

$$(F - EU_1^{-1}U_2)z_2^*(x) + (S_{NA} - EU_1^{-1}P)x < W_{NA} - EU_1^{-1}q. \tag{6.20}$$

The critical region CR_A can be equivalently written as:

$$CR_A = \left\{ x \in \mathcal{P}_x \colon \exists z_2 \text{ such that } \begin{bmatrix} z_2 \\ x \end{bmatrix} \in \mathcal{P}_{z_2 x} \right\}, \tag{6.21}$$

where:

$$\mathcal{P}_x := \{x \colon Dx = r\}, \tag{6.22}$$

$$\mathcal{P}_{z_2 x} := \left\{ \begin{bmatrix} z_2 \\ x \end{bmatrix} : (F - EU_1^{-1}U_2)z_2^*(x) + (S_{NA} - EU_1^{-1}P)x < W_{NA} - EU_1^{-1}q \right\} \tag{6.23}$$

In other words:

$$CR_A = \mathcal{P}_x \cap \mathrm{proj}_x(\mathcal{P}_{z_2 x}). \tag{6.24}$$

We can now state some fundamental properties of critical regions, value function and optimizer inside a critical region.

Theorem 6.1. *Let $(A, NA) \triangleq (A(x^*), NA(x^*))$ for some $x^* \in \mathcal{K}^*$*

i) CR_A is an open polyhedron of dimension d where $d = n - \mathrm{rank}\begin{bmatrix} G_A \ S_A \end{bmatrix} + \mathrm{rank}\, G_A$. If $d = 0$ then $CR_A = \{x^*\}$.
ii) If $\mathrm{rank}\, G_A = s$ (recall that $z \in \mathbb{R}^s$) then the optimizer $z^*(x)$ is unique and given by an affine function of the state inside CR_A, i.e., $z^*(x) = F_i x + G_i$ for all $x \in CR_A$.
iii) If the optimizer is not unique in CR_A then $Z^*(x)$ is an open polyhedron for all $x \in CR_A$.
iv) $J^*(x)$ is an affine function of the state, i.e., $J^*(x) = c_i x + d_i$ for all $x \in CR_A$.

Proof of *(i)*
Polytope $\mathcal{P}_{z_2 x}$ (6.23) is open and nonempty, therefore it is full-dimensional in the (z_2, x) space and $\dim \mathrm{proj}_x(\mathcal{P}_{z_2 x}) = n$. Also,

$$\dim \mathcal{P}_x = n - \mathrm{rank}\, D = n - (\mathrm{rank}\begin{bmatrix} G_A \ S_A \end{bmatrix} - \mathrm{rank}\, G_A)$$

Since the intersection of $\mathcal{P}_x$ and $\mathrm{proj}_x(\mathcal{P}_{z_2 x})$ is nonempty (it contains at least the point x^*) we can conclude that

$$\dim CR_A = n - \mathrm{rank}\begin{bmatrix} G_A \ S_A \end{bmatrix} + \mathrm{rank}\, G_A.$$

Since we assumed that the set $\mathcal{K}$ in (6.2) is bounded, then CR_A is bounded. This implies that CR_A is an open polytope since it is the intersection of an open polytope and the subspace $Dx = r$. In general, if we allow $\mathcal{K}$ in (6.2) to be unbounded, then the critical region CR_A can be unbounded.

Proof of *(ii)*
Consider (6.19) and recall that $l \triangleq \mathrm{rank}\, G_A$. If $l = s$, then the primal optimizer is unique U_2 is an empty matrix and

$$z^*(x) = z_1^*(x) = U_1^{-1}(-Px + q). \tag{6.25}$$

Proof of *(iii)*
If the primal optimizer is not unique in CR_A then $Z^*(x)$ in CR_A is the following point-to-set map: $Z^*(x) = \{[z_1,\ z_2] : z_2 \in \mathcal{P}_{z_2x},\ U_1z_1 + U_2z_2 + Px = q)\}$. For a fixed $\bar{x} \in CR_A$, $Z^*(\bar{x})$ is an open polyhedron in the $z = [z_1,\ z_2]$ space. Moreover the set $Z^* = \{[z_1,\ z_2,\ x] : [z_2, x] \in \mathcal{P}_{z_2x},\ U_1z_1 + U_2z_2 + Px = q\}$ is an open polyhedron in the $[z,\ x]$ space.

Proof of *(iv)*
Consider the dual problem (6.14) and one of its optimizer u_0^* for $x = x^*$. By definition of a critical region u_0^* remains optimal for all $x \in CR_{A(x^*)}$. Therefore the value function in $CR_{A(x^*)}$ is

$$J^*(x) = (W + Sx)'u_0^* \tag{6.26}$$

which is an affine function of x on CR_A. □

Remark 6.1. If the optimizer is unique then the computation of CR_A in (6.24) does not require the projection of the set $\mathcal{P}_{z_2x}$ in (6.23). In fact, U_2 and F are empty matrices and

$$CR_A = \mathcal{P}_{z_2x} = \left\{x : Dx = r,\ (S_{NA} - EU^{-1}P)x < W_{NA} - EU^{-1}q\right\}. \tag{6.27}$$

Remark 6.2. Consider (6.17). If the LP is not primal degenerate for all $x \in CR_A$ then $\operatorname{rank} G_A = \operatorname{rank}\left[G_A\ S_A\right] = s$ and therefore D and r are empty. If the LP is primal degenerate in CR_A then we distinguish two cases:
Case 1. Matrix D is the null matrix and r is the null vector. Then we have a full-dimensional primal degenerate critical region $CR_{A(x_0)}$.
Case 2. The rank of D is $p > 0$. Then, $CR_{A(x_0)}$ has dimension $n - p < n = \dim(\mathcal{K}^*)$. By Theorem 6.1 and the assumption that $\mathcal{K}^*$ is full dimensional, we conclude that $CR_{A(x_0)}$ is an $(n-p)$-dimensional face of another critical region $CR_{A'}$ for some combination $A' \supset A(x_0)$.
Note that Case 2 occurs only if the parameter vector x^* in Theorem 6.1 lies on the face of two or more neighboring full-dimensional critical regions, while Case 1 occurs when a full-dimensional set of parameters makes the LP problem (6.11) primal degenerate.

6.2.3 Nonunique Optimizer*

If $Z^*(x)$ is not a singleton, then the projection of the set $\mathcal{P}_{z_2x}$ in (6.23) is required in order to compute the critical region $CR_{A(x^*)}$ (see Section 6.1.1 for a discussion of polyhedra projection). Moreover, if one needs to determine one possible optimizer $z^*(\cdot)$ in the dual degenerate region $CR_{A(x^*)}$ the following simple method can be used. Choose a particular optimizer which

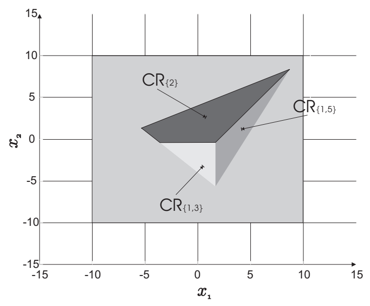


Fig. 6.2 Example 6.3: Polyhedral partition of the parameter space corresponding to the solution.

lies on a vertex of the feasible set, i.e., determine set $\widehat{A}(x^*) \supset A(x^*)$ of active constraints for which $\text{rank}(G_{\widehat{A}(x^*)}) = s$, and compute a subset $\widehat{CR}_{\widehat{A}(x^*)}$ of the dual degenerate critical region (namely, the subset of parameters x such that only the constraints $\widehat{A}(x^*)$ are active at the optimizer, which is not a critical region in the sense of Definition 6.1). Within $\widehat{CR}_{\widehat{A}(x^*)}$, the piecewise linear expression of an optimizers $z^*(x)$ is available from (6.25). The algorithm proceeds by exploring the space surrounding $\widehat{CR}_{\widehat{A}(x^*)}$ until $CR_{A(x^*)}$ is covered. The arbitrariness in choosing an optimizer leads to different ways of partitioning $CR_{\{A(x^*)\}}$, where the partitions, in general, may overlap. Nevertheless, in each region a unique optimizer is defined. The storing of overlapping regions can be avoided by intersecting each new region (inside the dual degenerate region) with the current partition computed so far. This procedure is illustrated in the following example.

Example 6.3. Consider the following mp-LP reported in [106, page 152]

$$\begin{array}{ll} \min & -2z_1 - z_2 \\ \text{subj. to} & \left\{ \begin{array}{r} z_1 + 3z_2 \leq 9 - 2x_1 + x_2 \\ 2z_1 + z_2 \leq 8 + x_1 - 2x_2 \\ z_1 \leq 4 + x_1 + x_2 \\ -z_1 \leq 0 \\ -z_2 \leq 0 \end{array} \right. \end{array} \tag{6.28}$$

where $\mathcal{K}$ is given by:

$$\begin{array}{c} -10 \leq x_1 \leq 10 \\ -10 \leq x_2 \leq 10. \end{array} \tag{6.29}$$

The solution is represented in Figure 6.2 and the critical regions are listed in Table 6.1.

The critical region $CR_{\{2\}}$ is related to a dual degenerate solution with non-unique optimizers. The analytical expression of $CR_{\{2\}}$ is obtained by projecting the $\mathcal{H}$-polyhedron

Region	Optimizer	Value Function
$CR_{\{2\}}$	not single valued	$-x_1 + 2x_1 - 8$
$CR_{\{1,5\}}$	$z_1^* = -2x_1 + x_2 + 9,\ z_2^* = 0$	$4x_1 - 2x_2 - 18$
$CR_{\{1,3\}}$	$z_1^* = x_1 + x_2 + 4,\ z_2^* = -x_1 + 5/3$	$-x_1 - 2x_2 - 29/3$

Table 6.1 Example 6.3: Critical regions and corresponding optimal value.

$$\begin{array}{ll} z_1 + 3z_2 + 2x_1 - x_2 & < 9 \\ 2z_1 + z_2 - x_1 + 2x_2 & = 8 \\ z_1 - x_1 - x_2 & < 4 \\ -z_1 & < 0 \\ -z_2 & < 0 \end{array} \tag{6.30}$$

on the parameter space to obtain:

$$\overline{CR}_{\{2\}} = \left\{ [x_1, x_2] : \begin{array}{ll} 2.5x_1 - 2x_2 & \leq 5 \\ -0.5x_1 + x_2 & \leq 4 \\ -12x_2 & \leq 5 \\ -x_1 - x_2 & \leq 4 \end{array} \right\} \tag{6.31}$$

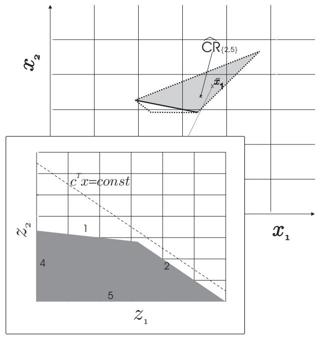
For all $x \in CR_{\{2\}}$, only one constraint is active at the optimum, which makes the optimizer not unique.

Figures 6.3 and 6.4 show two possible ways of covering $CR_{\{2\}}$. The generation of overlapping regions is avoided by intersecting each new region with the current partition computed so far, as shown in Figure 6.5 where $\widetilde{CR}_{\{2,4\}}$, $\widetilde{CR}_{\{2,1\}}$ represents the intersected critical region. In Figure 6.3, the regions are overlapping, and in Figure 6.5 artificial cuts are introduced at the boundaries inside the degenerate critical region $CR_{\{2\}}$. On the contrary, no artificial cuts are introduced in Figure 6.4.

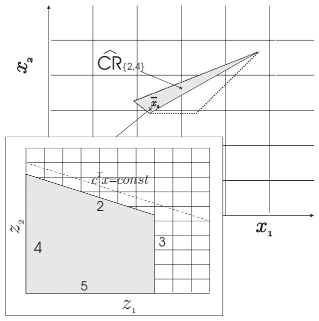
6.2.4 Propagation of the Set of Active Constraints

The objective of this section is to briefly describe the propagation of the set of active constraints when moving from one full-dimensional critical region to a neighboring full-dimensional critical region. We will use a simple example in order to illustrate the main points.

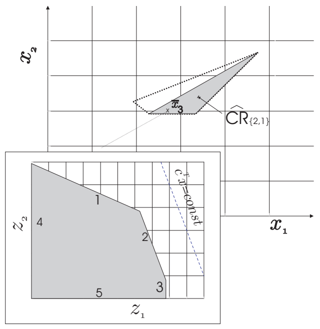
Example 6.4. Consider the mp-LP problem



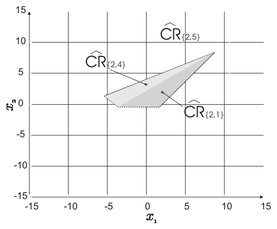
(a) First region $\widehat{CR}_{\{2,5\}} \subset CR_{\{2\}}$, and below the feasible set in the z-space corresponding to $\bar{x}_1 \in \widehat{CR}_{\{2,5\}}$



(b) Second region $\widehat{CR}_{\{2,4\}} \subset CR_{\{2\}}$, and below the feasible set in the z-space corresponding to $\bar{x}_2 \in \widehat{CR}_{\{2,4\}}$



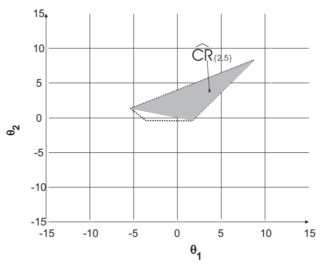
(c) Third region $\widehat{CR}_{\{2,1\}} \subset CR_{\{2\}}$, and below the feasible set in the z-space corresponding to $\bar{x}_3 \in \widehat{CR}_{\{2,1\}}$



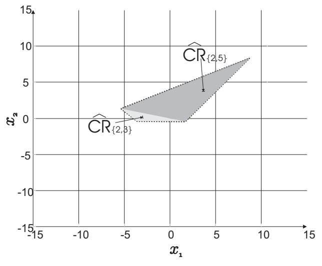
(d) Final partition of $CR_{\{2\}}$. Note that the region $\widehat{CR}_{\{2,5\}}$ is hidden by region $\widehat{CR}_{\{2,4\}}$ and region $\widehat{CR}_{\{2,1\}}$

Fig. 6.3 Example 6.3: A possible sub-partitioning of the degenerate region CR_2 where the regions $\widehat{CR}_{\{2,5\}}$ and $\widehat{CR}_{\{2,4\}}$ and $\widehat{CR}_{\{2,1\}}$ are overlapping. Note that below each picture the feasible set and the level set of the value function in the z-space is depicted for a particular choice of the parameter x indicated by a point marked with $\times$.

$$
\begin{array}{ll}
\min & z_1 + z_2 + z_3 + z_4 \\
\text{subj. to} & \left\{
\begin{array}{r}
-z_1 + z_5 \leq 0 \\
-z_1 - z_5 \leq 0 \\
-z_2 + z_6 \leq 0 \\
-z_2 - z_6 \leq 0 \\
-z_3 \leq x_1 + x_2 \\
-z_3 - z_5 \leq x_2 \\
-z_3 \leq -x_1 - x_2 \\
-z_3 + z_5 \leq -x_2 \\
-z_4 - z_5 \leq x_1 + 2x_2 \\
-z_4 - z_5 - z_6 \leq x_2 \\
-z_4 + z_5 \leq -1x_1 - 2x_2 \\
-z_4 + z_5 + z_6 \leq -x_2 \\
z_5 \leq 1 \\
-z_5 \leq 1 \\
z_6 \leq 1
\end{array}
\right.
\end{array}
\qquad (6.32)
$$



(a) First region $\widehat{CR}_{\{2,5\}} \subset CR_{\{2\}}$



(b) Second region $\widehat{CR}_{\{2,3\}} \subset CR_{\{2\}}$

Fig. 6.4 A possible solution to Example 6.3 where the regions $\widehat{CR}_{\{2,5\}}$ and $\widehat{CR}_{\{2,3\}}$ are non-overlapping

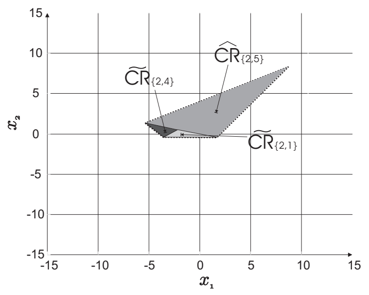


Fig. 6.5 A possible solution for Example 6.3 where $\widetilde{CR}_{\{2,4\}}$ is obtained by intersecting $\widehat{CR}_{\{2,4\}}$ with the complement of $\widehat{CR}_{\{2,5\}}$, and $\widetilde{CR}_{\{2,1\}}$ by intersecting $\widehat{CR}_{\{2,1\}}$ with the complement of $\widehat{CR}_{\{2,5\}}$ and $\widehat{CR}_{\{2,4\}}$

where $\mathcal{K}$ is given by

$$\begin{array}{c} -2.5 \leq x_1 \leq 2.5 \\ -2.5 \leq x_2 \leq 2.5. \end{array} \tag{6.33}$$

A solution to the mp-LP problem is shown in Figure 6.6 and the constraints which are active in each associated critical region are reported in Table 6.2. Clearly, as $z \in \mathbb{R}^6$, $CR6 = CR_{\{1,3,6,7,9,10,11,12\}}$ and $CR11 = CR_{\{2,4,5,8,9,10,11,12\}}$ are primal degenerate full-dimensional critical regions.

By observing Figure 6.6 and Table 6.2 we can notice the following. Under no primal and dual degeneracy, *(i)* full critical regions are described by a set of active constraint of dimension n, *(ii)* two neighboring full dimensional critical regions CR_{A_i} and CR_{A_j} have A_i and A_j differing only in one constraint, *(iii)* CR_{A_i} and CR_{A_j} will share a facet which is a primal degenerate critical region CR_{A_p} of dimension $n-1$ with $A_p = A_i \cup A_j$. In Example 6.4, CR1 ad CR13 are two full dimensional and neighboring critical regions and the corresponding

Critical Region	Value function
CR1=$CR_{\{2,3,4,7,10,11\}}$	$2x_1$+$3x_2$
CR2=$CR_{\{1,3,4,5,9,13\}}$	-$2x_1$-$3x_2$
CR3=$CR_{\{1,3,4,6,9,13\}}$	-x_1-$3x_2$-1
CR4=$CR_{\{1,3,6,9,10,13\}}$	-$2x_2$-1
CR5=$CR_{\{1,2,3,7,10,11\}}$	x_1
CR6=$CR_{\{1,3,6,7,9,10,11,12\}}$	x_1
CR7=$CR_{\{1,3,7,10,11,15\}}$	x_1
CR8=$CR_{\{1,3,4,5,9,12\}}$	-$2x_1$-$3x_2$
CR9=$CR_{\{2,4,8,11,12,14\}}$	$2x_2$-1
CR10=$CR_{\{1,2,4,5,9,12\}}$	-x_1
CR11=$CR_{\{2,4,5,8,9,10,11,12\}}$	-x_1
CR12=$CR_{\{2,4,5,9,12,16\}}$	-x_1
CR13=$CR_{\{2,3,4,7,11,14\}}$	$2x_1$+$3x_2$
CR14=$CR_{\{2,3,4,8,11,14\}}$	x_1+$3x_2$-1

Table 6.2 Example 6.4: Critical regions and corresponding value function.

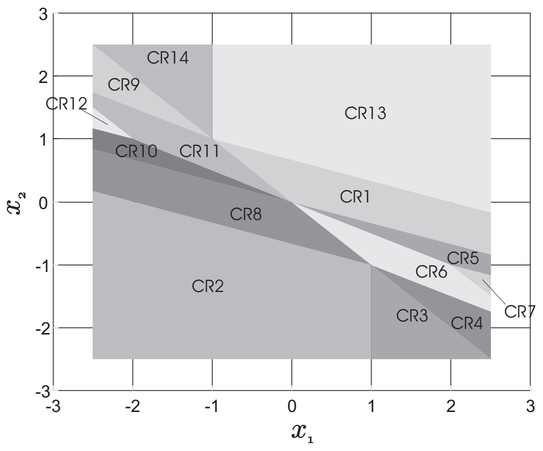


Fig. 6.6 Polyhedral partition of the parameter space corresponding to the solution of Example 6.4

active set differs only in one constraint (constraint 10 in CR1 and constraint 14 in CR13). They share a facet CR15 which is a one dimensional critical region (an open line), CR15=$CR_{\{2,3,4,8,10,11,14\}}$. We remark that the solution depicted in Figure 6.6 and detailed in Table 6.2 contains only full dimensional critical regions.

If primal and/or dual degeneracy occur, then things become more complex. In particular in the case of primal degeneracy, it might happen that full dimensional critical regions are described by more than s active constraints (CR6 or CR11 in Example 6.4). In case of dual degeneracy, it might happen that full dimensional critical regions are described by less than s active constraints ($CR_{\{2\}}$ in Example 6.3).

6.2.5 Value Function and Optimizer: Global Properties

In this section we discuss global properties of value function $J^*(x)$, optimizer $z^*(x)$, and of the set $\mathcal{K}^*$.

Theorem 6.2. *Assume that for a fixed $x^0 \in \mathcal{K}$ there exists a finite optimal solution $z^*(x_0)$ of (6.11). Then, for all $x \in \mathcal{K}$, (6.11) has either a finite optimum or no feasible solution.*

Proof: Consider the mp-LP (6.11) and assume by contradiction that there exist $x_0 \in \mathcal{K}$ and $\bar{x} \in \mathcal{K}$ with a finite optimal solution $z^*(x_0)$ and an unbounded solution $z^*(\bar{x})$. Then, the dual problem (6.14) for $x = \bar{x}$ is infeasible. This implies that the dual problem will be infeasible for all real vectors x since x enters only in the cost function. This contradicts the hypothesis since the dual problem (6.14) for $x = x_0$ has a finite optimal solution. □

Theorem 6.3. *The set of all parameters x such that the LP (6.11) has a finite optimal solution $z^*(x)$ equals $\mathcal{K}^*$.*

Proof: It follows directly from from Proposition 6.1 and Theorem 6.2. □

Note that from Definition 6.3 $\mathcal{K}^*$ is the set of feasible paraments. However the LP (6.11) might be unbounded from some $x \in \mathcal{K}^*$. Theorem 6.3 excludes this case.

The following Theorem 6.4 summarizes the properties enjoyed by the multiparametric solution.

Theorem 6.4. *The function $J^*(\cdot)$ is convex and piecewise affine over $\mathcal{K}^*$. If the optimizer $z^*(x)$ is unique for all $x \in \mathcal{K}^*$, then the optimizer function $z^* : \mathcal{K}^* \rightarrow \mathbb{R}^s$ is continuous and piecewise affine. Otherwise it is always possible to define a continuous and piecewise affine optimizer function z^* such that $z^*(x) \in Z^*(x)$ for all $x \in \mathcal{K}^*$.*

Remark 6.3. In Theorem 6.4, the piecewise affine property of optimizer and value function follows immediately from Theorem 6.1 and from the enumeration of all possible combinations of active constraints sets. Convexity of $J^*(\cdot)$ and continuity of $Z^*(x)$ follows from standard results on multiparametric programs (see Theorem 5.2 and Corollary 5.1). In the presence of multiple optima, the proof of existence of a continuous and piecewise affine optimizer function z^* such that $z^*(x) \in Z^*(x)$ for all $z \in \mathcal{K}^*$ is more involved and we refer the reader to [106, p. 180].

6.2.6 mp-LP Algorithm

The goal of an mp-LP algorithm is to determine the partition of $\mathcal{K}^*$ into critical regions CR_{A_i}, and to find the expression of the functions $J^*(\cdot)$ and

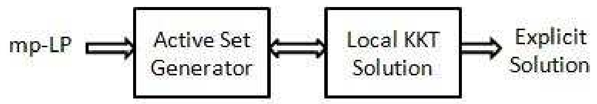


Fig. 6.7 Generic mp-LP algorithm

$z^*(\cdot)$ for each critical region. Figure 6.7 sketches the two main components of a generic mp-LP algorithm. The active set generator computes the set of active constraints A_i. The KKT solver computes CR_{A_i} and the expression of $J^*(\cdot)$ and $z^*(\cdot)$ in CR_{A_i} as explained in Theorem 6.1. The active set generator is the critical part. In principle, one could simply generate all the possible combinations of active sets. However, in many problems only a few active constraints sets generate full-dimensional critical regions inside the region of interest $\mathcal{K}$. Therefore, the goal is to design an active set generator algorithm which computes only the active sets A_i with associated full-dimensional critical regions covering only $\mathcal{K}^*$. This avoids the combinatorial explosion of a complete enumeration. Next we will describe one possible implementation of a mp-LP algorithm.

In order to start solving the mp-LP problem, we need an initial vector x_0 inside the polyhedral set $\mathcal{K}^*$ of feasible parameters. A possible choice for x_0 is the Chebychev center (see Section 3.4.5) of $\mathcal{K}^*$, i.e., x_0 solving the following LP:

$$\begin{array}{ll}\max_{x,z,\epsilon} & \epsilon \\ \text{subj. to} & T_i x + \epsilon\|T_i\|_2 \leq N_i, \quad i = 1,\dots,n_T \\ & Gz - Sx \leq W\end{array} \tag{6.34}$$

where n_T is the number of rows T_i of the matrix T defining the set $\mathcal{K}$ in (6.2). If $\epsilon \leq 0$, then the LP problem (6.11) is infeasible for all x in the interior of $\mathcal{K}$. Otherwise, we solve the primal and dual problems (6.11), (6.14) for $x = x_0$. Let z_0^* and u_0^* be the optimizers of the primal and the dual problem, respectively. The value z_0^* defines the following optimal partition

$$\begin{array}{r}\mathrm{A}(x_0) \triangleq \{j \in I : G_j z_0^* - S_j x_0 - W_j = 0\} \\ \mathrm{NA}(x_0) \triangleq \{j \in I : G_j z_0^* - S_j x_0 - W_j < 0\}\end{array} \tag{6.35}$$

and consequently the critical region $CR_{A(x_0)}$. Once the critical region $CR_{A(x_0)}$ has been defined, the rest of the space $R^{\text{rest}} = \mathcal{K} \backslash CR_{A(x_0)}$ has to be explored and new critical regions generated. An approach for generating a polyhedral partition $\{R_1, \dots, R_{n_rest}\}$ of the rest of the space R^{rest} is described in Theorem 3.1 in Section 3.4.7. The procedure proposed in Theorem 3.1 for partitioning the set of parameters allows one to recursively explore the parameter space. Such an iterative procedure terminates after a finite time, as the number of possible combinations of active constraints decreases with each iteration. Two following issues need to be considered:

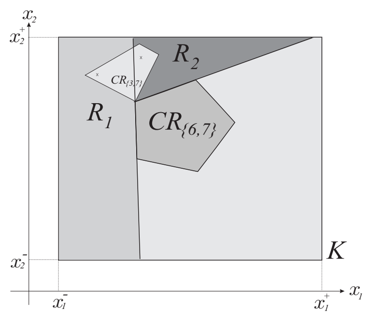


Fig. 6.8 Example of a critical region explored twice

1. The partitioning in Theorem 3.1 defines new polyhedral regions R_k to be explored that are not related to the critical regions which still need to be determined. This may split some of the critical regions, due to the artificial cuts induced by Theorem 3.1. Post-processing can be used to join cut critical regions [39]. As an example, in Figure 6.8 the critical region $CR_{\{3,7\}}$ is discovered twice, one part during the exploration of R_1 and the second part during the exploration of R_2. Although algorithms exist for convexity recognition and computation of the union of polyhedra, the post-processing operation is computationally expensive. Therefore, it is more efficient not to intersect the critical region obtained by (6.27) with halfspaces generated by Theorem 3.1, which is only used to drive the exploration of the parameter space. Then, no post processing is needed to join subpartitioned critical regions. On the other hand, some critical regions may appear more than once. Duplicates can be uniquely identified by the set of active constraints $A(x)$ and can be easily eliminated. To this aim, in the implementation of the algorithm we keep a list of all the critical regions which have already been generated in order to avoid duplicates. In Figure 6.8 the critical region $CR_{\{3,7\}}$ is discovered twice but stored only once.
2. If case 2 occurs in Remark 6.2 to avoid further recursion of the algorithm not producing any full-dimensional critical region, and therefore lengthen the number of steps required to determine the solution to the mp-LP, we perturb the parameter x_0 by a random vector $\epsilon \in \mathbb{R}^n$, where

$$\|\epsilon\|_2 < \min_i\{\tfrac{|T_i x_0 - N_i|}{\sqrt{T_i T_i{}'}}\}, \tag{6.36}$$

$\|\cdot\|_2$ denotes the standard Eucledian norm and $R_k = \{x : Tx \le N\}$ is the polyhedral region where we are looking for a new critical region. Equation (6.36) ensures that the perturbed vector $x_0 = x_0 + \varepsilon$ is still contained in R_k.

Based on the above discussion, the mp-LP solver can be summarized in the following recursive Algorithm 6.2.1. Note that the algorithm generates a partition of the state space which is not strict. The algorithm could be modified to store the critical regions as defined in (6.4) which are open sets, instead of storing their closure. In this case the algorithm has to explore and store all the critical regions that are not full-dimensional in order to generate a strict polyhedral partition of the set of feasible parameters. From a practical point of view such procedure is not necessary since the value function and the optimizer are continuous functions of x.

Algorithm 6.2.1

```
Input: Matrices $c$, $G$, $W$, $S$ of the mp-LP (6.11) and set $\mathcal{K}$ in (6.2)
Output: Multiparametric solution to the mp-LP (6.11)
1     execute partition($\mathcal{K}$);
2     end.
  procedure partition($Y$)
3         let $x_0 \in Y$ and $\epsilon$ the solution to the LP (6.34);
4         if $\epsilon \leq 0$ then exit; (no full dimensional CR is in $Y$)
5         solve the LP (6.11), (6.14) for $x = x_0$;
6         if the optimizer is not unique, then choose $z_0^*$ with
  $\text{rank}(A(x_0)) = s$, where $A(x_0)$ is the set of active constraints defined
  in (6.35) endif
7         let $A(x_0)$ be the set of active constraints as in (6.35);
8         let $U_1, U_2, P, D$ matrices as in (6.17). Note that $U_2$ is empty
  from step 6
9         determine $z^*(x)$ from (6.25) and $CR_{A(x_0)}$ from (6.27);
10        choose $z_0^*$ as one of the possible optimizers;
11        let $J^*(x)$ as in (6.26) for $x = x_0$;
12        Define and partition the rest of the region as in Theorem 3.1;
13        For each new sub-region $R_i$, partition($R_i$); end procedure.
```

Remark 6.4. As remarked in Section 6.2.2, if rank(D) > 0 in step *8*, the region $CR_{A(x_0)}$ is not full-dimensional. To avoid further recursion in the algorithm which does not produce any full-dimensional critical region, after computing U, P, D if $D \neq 0$ one should compute a random vector $\epsilon \in \mathbb{R}^n$ satisfying (6.36) and such that the LP (6.11) is feasible for $x_0 + \epsilon$ and then repeat step *7* with $x_0 \leftarrow x_0 + \epsilon$.

Remark 6.5. Note that step *6* can be easily executed by using an active set method for solving the LP (6.11) for $x = x_0$. Note also that primal basic solutions are needed only to define optimizers in a dual degenerate critical region. As remarked in the previous section, if one is only interested in characterizing the dual degenerate critical region, without characterizing one of the possible optimizer function $x^*(\cdot)$, step *6* can be avoided and instead of executing steps *8*-*9* one can compute $CR_{A(x_0)}$ by projecting (6.24) on $\mathcal{K}$ (note that $A(x_0)$ has to be the set of active constraints as defined in (6.1)).

Remark 6.6. The algorithm determines the partition of $\mathcal{K}$ recursively. After the first critical region is found, the rest of the region in $\mathcal{K}$ is partitioned into polyhedral sets $\{R_i\}$ as in Theorem 3.1. By using the same method, each set R_i is further partitioned, and so on.

6.3 Multiparametric Quadratic Programming

6.3.1 Formulation

In this section we investigate multiparametric quadratic programs (mp-QP), a special case of the multiparametric program (6.1) where the objective is a quadratic function

$$\begin{array}{rl} J^*(x) = \min_z & J(z,x) = \frac{1}{2}z'Hz \\ & \text{subj. to } Gz \leq W + Sx, \end{array} \tag{6.37}$$

All the variables were defined in Section 6.1.1. We assume $H \succ 0$. Our goal is to find the value function $J^*(x)$ and the optimizer function $z^*(x)$ in $\mathcal{K}^*$. Note that $\mathcal{K}^*$ can be determined as discussed in Proposition 6.1. As suggested through Example 5.1 our search for these functions proceeds by partitioning the set of feasible parameters into critical regions. Note that the more general problem with $J(z,x) = z'Hz + x'Fz$ can always be transformed in the mp-QP (6.37) by using the variable substitution $\tilde{z} \triangleq z + H^{-1}F'x$.

As in the previous sections, we denote with the subscript j the j-th row of a matrix or j-th element of a vector. Also, $J \triangleq \{1,\ldots,m\}$ is the set of constraint indices and for any $A \subseteq J$, G_A, W_A and S_A are the submatrices of G, W and S, respectively, consisting of the rows indexed by A. Without loss of generality we will assume that $\mathcal{K}^*$ is full dimensional (if it is not, then the procedure described in Section 6.1.3 can be used to obtain a full dimensional $\mathcal{K}^*$ in a reduced parameter space).

Next we reconsider the simple Example 5.1 of Chapter 5.

Example 6.5. Consider the parametric quadratic program

$$\begin{array}{l} J^*(x) = \min_z J(z,x) = \frac{1}{2}z^2 + 2xz \\ \quad \text{subj. to } z \leq 1 + x, \end{array}$$

where $z \in \mathbb{R}$ and $x \in \mathbb{R}$. Our goals are:

1. to find $z^*(x) = \arg\min_{z,\ z \leq 1+x} J(z,x)$,
2. to find all x for which the problem has a solution, and
3. to compute the value function $J^*(x)$.

The Lagrangian function is

$$L(z,x,u) = \frac{1}{2}z^2 + 2xz + u(z - x - 1)$$

and the KKT conditions are (see Section 4.2.3 for KKT conditions for quadratic programs)

$$z + 2x + u = 0 \tag{6.38a}$$
$$u(z - x - 1) = 0 \tag{6.38b}$$
$$u \geq 0 \tag{6.38c}$$
$$z - x - 1 \leq 0 \tag{6.38d}$$

Consider (6.38) and the two strictly complementary cases:

$$\begin{array}{ll}
\text{A.} & \begin{array}{l} z + 2x + u = 0 \\ z - x - 1 = 0 \\ u > 0 \end{array} \Rightarrow \begin{cases} z^* = x + 1 \\ u^* = -3x - 1 \\ J^* = \frac{5}{2}x^2 + 3x + \frac{1}{2} \\ x < -\frac{1}{3} \end{cases} \\
\text{B.} & \begin{array}{l} z + 2x + u = 0 \\ z - x - 1 \leq 0 \\ u = 0 \end{array} \Rightarrow \begin{cases} z^* = -2x \\ u^* = 0 \\ J^* = -2x^2 \\ x \geq -\frac{1}{3} \end{cases}
\end{array} \tag{6.39}$$

This solution is depicted in Figure 5.1 of Chapter 5. The above simple procedure, which required nothing but the solution of the KKT conditions, yielded the optimizer $z^*(x)$ and the value function $J^*(x)$ for all values of the parameter x. The set of admissible parameters values was divided into two *critical regions*, defined by $x < -\frac{1}{3}$ and $x \geq -\frac{1}{3}$. In the region $x < -\frac{1}{3}$ the inequality constraint is active and the Lagrange multiplier is greater than zero, in the other region $x \geq -\frac{1}{3}$ the Lagrange multiplier is equal to zero and the inequality constraint is not active (with the exception of the boundary point $-\frac{1}{3}$). Note that for $x = -\frac{1}{3}$ the inequality constraint $z - x - 1 \leq 0$ is active at z^* and the Lagrange multiplier is equal to zero. In this case the constraint is called *weakly active* at z^*.

Throughout a critical region the conditions for optimality derived from the KKT conditions do not change. In each critical region the optimizer $z^*(x)$ is affine and the value function $J^*(x)$ is quadratic. Both $z^*(x)$ and $J^*(x)$ are continuous.

6.3.2 Critical Regions, Value Function and Optimizer: Local Properties

Consider the definition of critical region given in Section 6.1.2. In this section we show that critical regions of mp-QP are polyhedra. We use the KKT conditions (Section 2.4) to derive the $\mathcal{H}$-polyhedral representation of the critical regions and to compute the optimizer function $z^*(x)$ and the optimal value function $J^*(x)$ inside each critical region.

The following theorem introduces fundamental properties of critical regions and value function and optimizer inside a critical region.

Theorem 6.5. *Let* $(A, NA) \triangleq (A(\bar{x}), NA(\bar{x}))$ *for some* $\bar{x} \in \mathcal{K}^*$*, Then*

i) the closure of CR_A *is a polyhedron.*
ii) $z^*(x)$ *is an affine function of the state inside* CR_A*, i.e.,* $z^*(x) = F_i x + G_i$ *for all* $x \in CR_A$*.*
iii) $J^*(x)$ *is a quadratic function of the state inside* CR_A*, i.e.,* $J^*(x) = x' M_i x + c_i x + d_i$ *for all* $x \in CR_A$

Proof: The first-order Karush-Kuhn-Tucker (KKT) optimality conditions (see Section 4.2.3) for the mp-QP are given by

$$
\begin{aligned}
& Hz^* + G'u^* = 0,\ u \in \mathbb{R}^m, && (6.40a)\\
& u_i^*(G_i z^* - W_i - S_i x) = 0,\ i = 1, \dots, m, && (6.40b)\\
& u^* \geq 0, && (6.40c)\\
& Gz^* \leq W + Sx, && (6.40d)
\end{aligned}
$$

We solve (6.40a) for z^*

$$z^* = -H^{-1}G'u^* \tag{6.41}$$

and substitute the result into (6.40b) to obtain the complementary slackness condition

$$u^*(-GH^{-1}G'u^* - W - Sx) = 0. \tag{6.42}$$

Let u_{NA}^* and u_A^* denote the Lagrange multipliers corresponding to inactive and active constraints, respectively. For inactive constraints $u_{NA}^* = 0$. For active constraints:

$$(-G_A H^{-1} G_A{}')u_A^* - W_A - S_A x = 0, \tag{6.43}$$

We distinguish two cases.
Case 1: The rows of G_A are linearly independent. This implies that $(G_A H^{-1} G_A{}')$ is a square full rank matrix and therefore

$$u_A^* = -(G_A H^{-1} G_A{}')^{-1}(W_A + S_A x) \tag{6.44}$$

where G_A, W_A, S_A correspond to the set of active constraints A. Thus u^* is an affine function of x. We can substitute u_A^* from (6.44) into (6.41) to obtain

$$z^* = H^{-1}G_A{}'(G_A H^{-1} G_A{}')^{-1}(W_A + S_A x) \tag{6.45}$$

and note that z^* is also an affine function of x. $J^*(x) = \frac{1}{2} z^*(x)' H z^*(x)$ and therefore is a quadratic function of x. The critical region CR_A is computed by substituting z^* from (6.41) in the primal feasibility conditions (6.40d)

$$\mathcal{P}_p \triangleq \{x : GH^{-1}G_A{}'(G_A H^{-1} G_A{}')^{-1}(W_A + S_A x) < W + Sx\} \tag{6.46}$$

and the Lagrange multipliers in (6.44) in the dual feasibility conditions (6.40c)

$$\mathcal{P}_d \triangleq \{x : -(G_A H^{-1} {G_A}')^{-1}(W_A + S_A x) \geq 0\} \tag{6.47}$$

In conclusion, the critical region CR_A is the intersection of $\mathcal{P}_p$ and $\mathcal{P}_d$:

$$CR_A = \{x : \ x \in \mathcal{P}_p, \ x \in \mathcal{P}_d\} \tag{6.48}$$

Obviously, the closure of CR_A is a polyhedron in the x-space.

The polyhedron $\mathcal{P}_p$ is open and non empty (it contains at least the point $\bar{x}$). Therefore it is full-dimensional in the x space. This implies that $\dim CR_A = \dim \mathcal{P}_d$.

Case 2: The rows of G_A are not linearly independent. For instance, this happens when more than s constraints are active at the optimizer $z^*(\bar{x}) \in \mathbb{R}^s$, i.e., in a case of *primal degeneracy*. In this case the vector of Lagrange multipliers u_0^* might not be uniquely defined, as the dual problem of (6.37) is not strictly convex. Note that *dual degeneracy* and nonuniqueness of $z^*(\bar{x})$ cannot occur, as $H \succ 0$.

First, consider a homogenous solution $\bar{u} \neq 0$ to (6.43):

$$(-G_A H^{-1} {G_A}')\bar{u} = 0. \tag{6.49}$$

Next we prove that $G_A H^{-1} \bar{u} = 0$. From (6.49), $G_A \sqrt{H^{-1}} \bar{y} = 0$ with $\bar{y} = \sqrt{H^{-1}} G'_A \bar{u}$. Therefore $\bar{y} = \sqrt{H^{-1}} {G_A}' \bar{u} \in \text{Null}(G_A \sqrt{H^{-1}})$. Recall that the null space of a linear transformation represented by the matrix M, Null(M), equals the space orthogonal to Span(M'). Therefore we have $\bar{y} = \sqrt{H^{-1}} {G_A}' \bar{u} \in \text{Span}(\sqrt{H^{-1}} {G_A}')^{\perp}$. A vector $\bar{y} \in \text{Span}(\sqrt{H^{-1}} {G_A}')$ and $\bar{y} \in \text{Span}(\sqrt{H^{-1}} {G_A}')^{\perp}$ can only be the zero vector:

$$H^{-1} {G_A}' \bar{u} = 0. \tag{6.50}$$

Let $l \triangleq \operatorname{rank} G_A$ and consider the QR decomposition of $-G_A H^{-1} G'_A$

$$-G_A H^{-1} G'_A = Q \begin{bmatrix} U_1 & U_2 \\ \mathbf{0}_{|A|-l \times l} & \mathbf{0}_{|A|-l \times |A|-l} \end{bmatrix}$$

where $Q \in \mathbb{R}^{|A| \times |A|}$ is a unitary matrix, $U_1 \in \mathbb{R}^{l \times l}$ is a full-rank square matrix and $U_2 \in \mathbb{R}^{l \times |A|-l}$. Let $\begin{bmatrix} P \\ D \end{bmatrix} \triangleq -Q^{-1} S_A$ and $\begin{bmatrix} q \\ r \end{bmatrix} \triangleq Q^{-1} W_A$. From (6.43) we obtain

$$\begin{bmatrix} U_1 & U_2 & P \\ \mathbf{0}_{|A|-l \times l} & \mathbf{0}_{|A|-l \times |A|-l} & D \end{bmatrix} \begin{bmatrix} u^*_{A,1} \\ u^*_{A,2} \\ x \end{bmatrix} = \begin{bmatrix} q \\ r \end{bmatrix}. \tag{6.51}$$

Computing $u^*_{A,1}$ from (6.51) we obtain:

$$u^*_{A,1} = U_1^{-1}(-U_2 u^*_{A,2} - Px + q) \tag{6.52}$$

In conclusion, we can substitute $u_A^* = \left[u_{A,1}^*; u_{A,2}^*\right]$ from (6.52) into (6.41) to obtain

$$z^* = H^{-1}{G_A}' \begin{bmatrix} U_1^{-1}(U_2 u_{A,2}^* + Px - q) \\ u_{A,2}^* \end{bmatrix} \tag{6.53}$$

Next we prove that the terms with $u_{A,2}^*$ in(6.53) disappear. Note that the vector $\begin{bmatrix} U_1^{-1}(U_2 u_{A,2}^*) \\ u_{A,2}^* \end{bmatrix}$ is a homogenous solution to (6.43) and (6.51):

$$(-G_A H^{-1}{G_A}') \begin{bmatrix} U_1^{-1}(U_2 u_{A,2}^*) \\ u_{A,2}^* \end{bmatrix} = 0,$$

and therefore from (6.49)-(6.50) we have

$$H^{-1}{G_A}' \begin{bmatrix} U_1^{-1}(U_2 u_{A,2}^*) \\ u_{A,2}^* \end{bmatrix} = 0. \tag{6.54}$$

Combining (6.53) and (6.54) we obtain

$$z^* = H^{-1}{G_A}' \begin{bmatrix} U_1^{-1}(U_2 u_{A,2}^*) \\ u_{A,2}^* \end{bmatrix} + H^{-1}{G_{A,1}}' U_1^{-1}(Px-q) = H^{-1}{G_{A,1}}' U_1^{-1}(Px-q) \tag{6.55}$$

where we have partitioned G_A as $G_A = [G_{A,1}; G_{A,2}]$ and $G_{A,1} \in \mathbb{R}^{l \times s}$.

Note that z^* is an affine function of x. $J^*(x) = \frac{1}{2} z^*(x)' H z^*(x)$ and therefore is a quadratic function of x.

The critical region CR_A is computed by substituting z^* from (6.55) in the primal feasibility conditions (6.40d)

$$\mathcal{P}_p \triangleq \{x : G(H^{-1}{G_{A,1}}' U_1^{-1}(Px - q)) < W + Sx\} \tag{6.56}$$

and the Lagrange multipliers in (6.52) in the dual feasibility conditions (6.40c)

$$\mathcal{P}_{u_{A,2}^*,x} \triangleq \{[\ u_{A,2}^*, x] : U_1^{-1}(-U_2 u_{A,2}^* - Px + q) \geq 0,\ \ u_{A,2}^* \geq 0\} \tag{6.57}$$

The critical region CR_A is the intersection of the sets $Dx = r$, $\mathcal{P}_p$ and $\text{proj}_x(\mathcal{P}_{u_{A,2}^*,x})$:

$$CR_A = \{x :\ \ Dx = r,\ \ x \in \mathcal{P}_p,\ \ x \in \text{proj}_x(\mathcal{P}_{u_{A,2}^*,x})\} \tag{6.58}$$

The closure of CR_A is a polyhedron in the x-space. □

Remark 6.7. In general, the critical region polyhedron (6.46)-(6.48) is open on facets arising from primal feasibility and closed on facets arising from dual feasibility.

Remark 6.8. If D in (6.51) is nonzero, then from (6.58), CR_A is a lower dimensional region, which, in general, corresponds to a common boundary between two or more full-dimensional regions. Typically, the following definition is introduced

Definition 6.4. For a given set of active constraints A we say that the *Linear Independence Constraint Qualification*(LICQ) holds if the rows of G_A are linearly independent.

Under LICQ case 2 in Theorem 6.5 cannot occur.

6.3.3 Propagation of the Set of Active Constraints

The objective of this section is to briefly describe the propagation of the set of active constraints when moving from one full-dimensional critical region to a neighboring full-dimensional critical region. We will use a simple example in order to illustrate the main points.

Example 6.6. Consider the mp-QP problem

$$\begin{array}{rl} J^*(x) = \min\limits_z & \frac{1}{2} z' H z + x' F z \\ & \text{subj. to } Gz \le W + Sx, \end{array} \tag{6.59}$$

with

$$H = \begin{bmatrix} 8.18 & -3.00 & 5.36 \\ -3.00 & 5.00 & -3.00 \\ 5.36 & -3.00 & 10.90 \end{bmatrix}, \quad F = \begin{bmatrix} 0.60 & 0.00 & 1.80 \\ 5.54 & -3.00 & 8.44 \end{bmatrix} \tag{6.60}$$

and

$$G = \begin{bmatrix} 1.00 & -1.00 & 1.00 \\ -1.00 & 1.00 & -1.00 \\ 0.00 & 0.00 & -0.30 \\ -1.00 & 0.00 & -1.00 \\ 0.00 & 0.00 & 0.30 \\ 1.00 & 0.00 & 1.00 \\ -0.30 & 0.00 & -0.60 \\ 0.00 & -1.00 & 0.00 \\ 0.30 & 0.00 & 0.60 \\ 0.00 & 1.00 & 0.00 \\ -1.00 & 0.00 & 0.00 \\ 1.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & -1.00 \\ 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 1.00 \\ 0.00 & 0.00 & -1.00 \\ 0.00 & 0.00 & 1.00 \end{bmatrix}, \quad E = \begin{bmatrix} 0.00 & -1.00 \\ 0.00 & 1.00 \\ 1.00 & 0.60 \\ 0.00 & 1.00 \\ -1.00 & -0.60 \\ 0.00 & -1.00 \\ 1.00 & 0.90 \\ 0.00 & 0.00 \\ -1.00 & -0.90 \\ 0.00 & 0.00 \\ 0.00 & 0.00 \\ 0.00 & 0.00 \\ 1.00 & 0.30 \\ 0.00 & 1.00 \\ -1.00 & -0.30 \\ 0.00 & -1.00 \\ 0.00 & 0.00 \\ 0.00 & 0.00 \end{bmatrix}, \quad W = \begin{bmatrix} 0.50 \\ 0.50 \\ 8.00 \\ 8.00 \\ 8.00 \\ 8.00 \\ 8.00 \\ 8.00 \\ 8.00 \\ 8.00 \\ 0.50 \\ 0.50 \\ 8.00 \\ 8.00 \\ 8.00 \\ 8.00 \\ 0.50 \\ 0.50 \end{bmatrix} \tag{6.61}$$

where $\mathcal{K}$ is given by

$$\begin{array}{c} -8 \le x_1 \le 8 \\ -8 \le x_2 \le 8. \end{array} \tag{6.62}$$

A solution to the mp-QP problem is shown in Figure 6.9 and the constraints which are active in each associated critical region are reported in Table 6.3. Figure 6.9 and Table 6.3 report only full dimensional critical regions.

Since A_1 is empty in CR1 from equations (6.46) and (6.47) we can conclude that the facets of CR1 are facets of primal feasibility and therefore do not

Critical Region	Active Constraints
CR1	{}
CR2	{1}
CR3	{2}
CR4	{11}
CR5	{12}
CR6	{17}
CR7	{18}
CR8	{1,11}
CR9	{1,18}
CR10	{2,12}
CR11	{2,17}
CR12	{11,17}
CR13	{12,18}
CR14	{1,11,17}
CR15	{1,11,18}
CR16	{2,12,17}
CR17	{2,12,18}

Table 6.3 Example 6.6: Critical regions and corresponding set of active constraints.

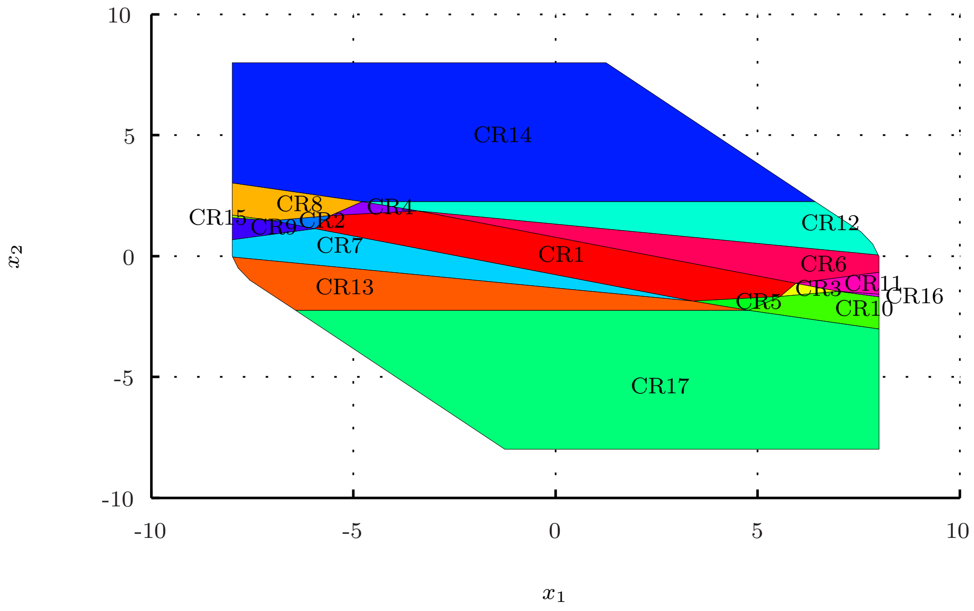


Fig. 6.9 Polyhedral partition of the parameter space corresponding to the solution of Example 6.6

belong to CR1. In general, as discussed in Remark 6.7 critical regions are open on facets arising from primal feasibility and closed on facets arising from dual feasibility. Next we focus on the closure of the critical regions.

By observing Figure 6.9 and Table 6.3 we notice that as we move away from region CR1 (corresponding to no active constraints), the number of active constraints increases. In particular, for any two neighboring full dimensional critical regions CR_{A_i} and CR_{A_j} have $A_i \subset A_j$ and $|A_i| = |A_j| - 1$ or $A_j \subset A_i$ and $|A_i| = |A_j| + 1$. This means that as one moves from one full dimensional region to a neighboring full dimensional region, one constraint is either included to the list of active constraints or removed from it. This happens for instance when moving from CR1 to CR6 to CR11 to CR16 to CR10 to CR5 to CR13 to CR17.

Things become more complex if LICQ does not hold everywhere in $\mathcal{K}^*$. In particular, there might exist two neighboring full dimensional critical regions CR_{A_i} and CR_{A_j} where A_i and A_j do not share any constraint and with $|A_i| = |A_j|$. Also, CR_{A_i} might have multiple neighboring region on the same facet. In other words, it can happen that the intersection of the closures of two adjacent full-dimensional critical regions is a not a facet of both regions but only a subset of it. We refer the reader to [243] for more details on such a degenerate condition.

6.3.4 Value Function and Optimizer: Global Properties

The convexity of the value function $J^*(x)$ and the continuity of the solution $z^*(x)$ easily follow from the general results on multiparametric programming as proven in Corollary 5.2. In the following we give a simple proof based on the linearity result of Theorem 6.5 together with a proof of the piecewise linearity of the solution $z^*(x)$. We remark that Proposition 6.1 proves that the set of feasible parameters $\mathcal{K}^* \subseteq \mathcal{K}$ of an mp-QP is a polyhedron.

Theorem 6.6. *Consider the multiparametric quadratic program (6.37) and let $H \succ 0$. Then, the optimizer $z^*(x) : \mathcal{K}^* \to \mathbb{R}^s$ is continuous and piecewise affine on polyhedra, in particular it is affine in each critical region, and the optimal solution $J^*(x) : \mathcal{K}^* \to \mathbb{R}$ is continuous, convex and piecewise quadratic on polyhedra.*

Proof: We first prove convexity of $J^*(x)$. Take generic $x_1, x_2 \in \mathcal{K}^*$, and let $J^*(x_1)$, $J^*(x_2)$ and z_1, z_2 the corresponding optimal values and minimizers. By optimality of $J^*(x_\alpha)$, $J^*(x_\alpha) \leq \frac{1}{2}z_\alpha' H z_\alpha$, and hence $J^*(x_\alpha) - \frac{1}{2}[\alpha z_1' H z_1 + (1-\alpha) z_2' H z_2] \leq \frac{1}{2}z_\alpha' H z_\alpha - \frac{1}{2}[\alpha z_1' H z_1 + (1-\alpha) z_2' H z_2] = \frac{1}{2}[\alpha^2 z_1' H z_1 + (1-\alpha)^2 z_2' H z_2 + 2\alpha(1-\alpha) z_2' H z_1 - \alpha z_1' H z_1 - (1-\alpha) z_2' H z_2] = -\frac{1}{2}\alpha(1-\alpha)(z_1 - z_2)' H (z_1 - z_2) \leq 0$, i.e. $J^*(\alpha x_1 + (1-\alpha)x_2) \leq \alpha J^*(x_1) + (1-\alpha) J^*(x_2)$, $\forall x_1, x_2 \in \mathcal{K}$, $\forall \alpha \in [0,1]$, which proves the convexity of $J^*(x)$ on $\mathcal{K}^*$. Within the closed polyhedral regions CR_i in $\mathcal{K}^*$ the solution $z^*(x)$ is affine (6.45) by Theorem 6.5. The boundary between two regions belongs to both closed regions. Since $H \succ 0$, the optimum is unique, and hence the solution must be continuous across the boundary. Therefore $z^*(x) : \mathcal{K}^* \to \mathbb{R}^s$ is continuous and piecewise affine on polyhedra. The fact that $J^*(x)$ is continuous and piecewise quadratic follows trivially. □

6.3.4.1 Continuous Differentiability of the Value Function*

Let $J^*(x)$ be the convex and piecewise quadratic value function in (6.37):

$$J^*(x) = q_i(x) \triangleq x'Q_ix + T_i'x + V_i, \quad \text{if } x \in CR_i, \; i = 1, \dots, N_n. \tag{6.63}$$

Through the next three theorems we will establish that if the mp-QP problem (6.37) is not degenerate then $J^*(x)$ is a $C^{(1)}$ function.

Theorem 6.7. *Assume that the mp-QP problem (6.37) is not degenerate. Consider the value function $J^*(x)$ in (6.63) and let CR_i, CR_j be the closure of two neighboring critical regions corresponding to the set of active constraints A_i and A_j, respectively, then*

$$Q_i - Q_j \preceq 0 \textit{ or } Q_i - Q_j \succeq 0 \textit{ and } Q_i \neq Q_j \tag{6.64}$$

and

$$Q_i - Q_j \preceq 0 \textit{ iff } A_i \subset A_j, \tag{6.65}$$

Proof: Let CR_i and CR_j be the closure of two neighboring critical regions and A_i and A_j be the corresponding sets of active constraints at the optimum of QP (6.37). 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., $Q_i - Q_j \preceq 0$ 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 $G_{A_i}z^* = W_{A_i} + S_{A_i}x$ transforms Problem (6.37) into a QP in a lower dimensional space.

For the unconstrained case we have $z^* = 0$ and $J_z^*(x) = 0$. Consequently

$$q_i(x) = 0. \tag{6.66}$$

For the constrained case, from equation (6.45) we obtain

$$q_j(x) = \frac{1}{2}x'(S_{A_j}'\varGamma^{-1}S_{A_j})x + W_{A_j}'\varGamma^{-1}S_{A_j}x + \frac{1}{2}W_{A_j}'\varGamma^{-1}W_{A_j}, \tag{6.67}$$

where $\varGamma = G_{A_j}H^{-1}\tilde{G}_{A_j}'$, $\varGamma = \varGamma' \succ 0$. The difference of the quadratic terms of $q_i(x)$ and $q_j(x)$ gives

$$Q_i - Q_j = -\frac{1}{2}S_{A_j}'\varGamma^{-1}S_{A_j} \preceq 0. \tag{6.68}$$

What is left to prove is $Q_i \neq Q_j$. We will prove this by showing that $Q_i = Q_j$ if and only if $CR_i = CR_j$. From (6.46)-(6.47) the polyhedron CR_j where the set of active constraints A_j is constant is defined as

$$CR_j = \{x \; : GH^{-1}G'_{A_j}\Gamma^{-1}(W_{A_j}+S_{A_j}x) \leq W+Sx, \; -\Gamma^{-1}(W_{A_j}+S_{A_j}x) \geq 0\}. \tag{6.69}$$

From (6.68) we conclude that $Q_i = Q_j$ if and only if $S_{A_j} = 0$. The continuity of $J_z^*(x)$ implies that $q_i(x) - q_j(x) = 0$ on the common facet of CR_i and CR_j. Therefore, by comparing (6.66) and (6.67), we see that $S_{A_j} = 0$ implies $W_{A_j} = 0$. Finally, for $S_{A_j} = 0$ and $W_{A_j} = 0$, from (6.69) it follows that $CR_i = CR_j = \{x \; : \; 0 \leq W + Sx\}$. □

The following property of convex piecewise quadratic functions was proved in [230]:

Theorem 6.8. *Consider the value function $J^*(x)$ in (6.63) satisfying (6.64) and its quadratic expression $q_i(x)$ and $q_j(x)$ on two neighboring polyhedra CR_i, CR_j then*

$$q_i(x) = q_j(x) + (a'x - b)(\gamma a'x - \bar{b}), \tag{6.70}$$

where $\gamma \in \mathbb{R}/\{0\}$.

Equation (6.70) states that the functions $q_i(x)$ and $q_j(x)$ in two neighboring regions CR_i, CR_j of a convex piecewise quadratic function on polyhedra satisfying (6.64) either intersect on two parallel hyperplanes: $a'x - b$ and $\gamma a'x - \bar{b}$ if $\bar{b} \neq \gamma b$ (see Figure 6.10(a)) or are tangent in one hyperplane: $a'x - b$ if $\bar{b} = \gamma b$ (see Figure 6.10(b)). We will prove next that if the mp-QP problem (6.37) is not degenerate then $J^*(x)$ is a $C^{(1)}$ function by showing that the case depicted in Figure 6.10(a) is not consistent with Theorem 6.7. In fact, Figure 6.10(a) depicts the case $Q_i - Q_j \preceq 0$, that implies $A_i \subset A_j$ by Theorem 6.7. However $q_j(0) < q_i(0)$ and from the definitions of q_i and q_j this contradicts the fact that $A_i \subset A_j$.

Theorem 6.9. *Assume that the mp-QP problem (6.37) is not degenerate, then the value function $J^*(x)$ in (6.63) is $C^{(1)}$.*

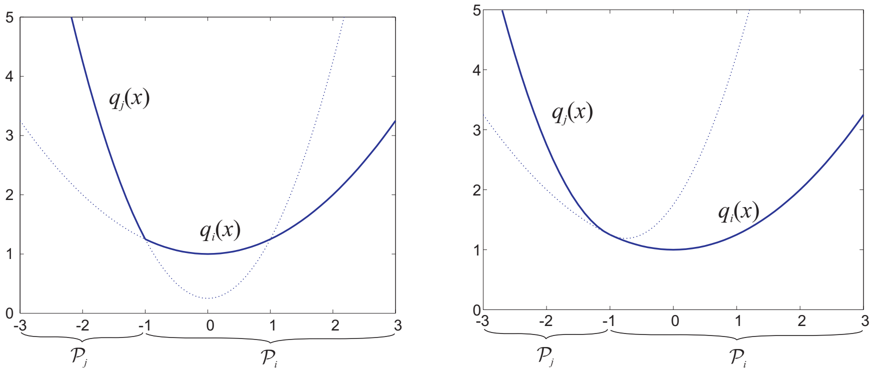
Proof: We will prove by contradiction that $\bar{b} = \gamma b$. Suppose there exists two neighboring polyhedra CR_i and CR_j such that $\bar{b} \neq \gamma b$. Without loss of generality assume that (i) $Q_i - Q_j \preceq 0$ and (ii) CR_i is in the halfspace $a'x \leq b$ defined by the common boundary.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

From (i) and from (6.70), either $\gamma < 0$ or $\gamma = 0$ if $Q_i - Q_j = 0$. Take $x_0 \in \mathcal{F}_{ij}$. For sufficiently small $\varepsilon \geq 0$, the point $x \triangleq x_0 - a\varepsilon$ belongs to CR_i.

Let $J^*(\varepsilon) \triangleq J^*(x_0 - a\varepsilon)$, $q_i(\varepsilon) \triangleq q_i(x_0 - a\varepsilon)$, and consider

$$q_i(\varepsilon) = q_j(\varepsilon) + (a'a\varepsilon)(\gamma a'a\varepsilon + (\bar{b} - \gamma b)). \tag{6.71}$$

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 ε. This implies $q'_j(\varepsilon) \leq q'_i(\varepsilon)$



(a) not differentiable (b) differentiable

Fig. 6.10 Two piecewise quadratic convex functions

where $q'_j(\varepsilon)$ and $q'_i(\varepsilon)$ 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 (6.71) $q_j(\varepsilon) < q_i(\varepsilon)$ for all $\varepsilon \in (0, \frac{-(\bar{b}-\gamma b)}{\gamma a' a})$.

Thus there exists $x \in CR_i$ with $q_j(x) < q_i(x)$. This is a contradiction since from Theorem 6.7, $A_i \subset A_j$. □

Note that in case of degeneracy the value function $J^*(x)$ in (6.63) may not be $C^{(1)}$. The following counterexample was given in [44].

Example 6.7. Consider the mp-QP (6.37) with

$$H = \begin{bmatrix} 3 & 3 & -1 \\ 3 & 11 & 23 \\ -1 & 23 & 75 \end{bmatrix}$$
$$G = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 3 & 5 \\ -1 & -1 & -1 \\ -1 & -3 & -5 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \quad W = \begin{bmatrix} 1 \\ 0 \\ -1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad S = \begin{bmatrix} 0 \\ 1 \\ 0 \\ -1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{6.72}$$

and $\mathcal{K} = \{x \in \mathbb{R} | 1 \leq x \leq 5\}$.

The problem was solved by using Algorithm 6.3.1 described next. The solution consists of 5 critical regions. The critical regions and the expression of the value function are reported in Table 6.4. The reader can verify that the value function is not continuously differentiable at $x = 3$. Indeed, at $x = 3$ the

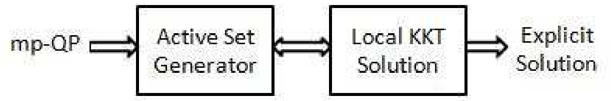


Fig. 6.11 Generic mp-QP algorithm

LICQ condition does not hold and therefore, the hypothesis of Theorem 6.9 is not fulfilled.

Region	Optimal value
$CR_{\{1,2,3,4,6\}} = \{x \ : \ 1 \leq x < 1.5\}$	$2.5x^2 - 6x + 5$
$CR_{\{1,2,3,4\}} = \{x \ : \ 1.5 \leq x \leq 2\}$	$0.5x^2 + 0.5$
$CR_{\{1,2,3,4,7\}} = \{x \ : \ 2 < x < 3\}$	$x^2 - 2x + 2.5$
$CR_{\{1,2,3,4,5,7\}} = \{x \ : \ x = 3\}$	$x^2 - 2x + 2.5$
$CR_{\{1,2,3,4,5\}} = \{x \ : \ 3 < x \leq 5\}$	$5x^2 - 24x + 32.5$

Table 6.4 Critical regions and value function corresponding to the solution of Example 6.7

6.3.5 mp-QP Algorithm

The goal of an mp-QP algorithm is to determine the partition of $\mathcal{K}^*$ into critical regions CR_i, and find the expression of the functions $J^*(\cdot)$ and $z^*(\cdot)$ for each critical region. Figure 6.11 sketches the two main components of a generic mp-QP algorithm. The active set generator computes the set of active constraints A_i. The KKT solver computes CR_{A_i} and the expression of $J^*(\cdot)$ and $z^*(\cdot)$ in CR_{A_i} as explained in Theorem 6.5. The active set generator is the critical part. In principle, one could simply generate all the possible combinations of active sets. However, in many problems only a few active constraints sets generate full-dimensional critical regions inside the region of interest $\mathcal{K}$. Therefore, the goal is to design an active set generator algorithm which computes only the active sets A_i with the associated full-dimensional critical regions covering only $\mathcal{K}^*$.

Next an implementation of a mp-QP algorithm is described. See Section 6.6 for a literature review on alternative approaches to the solution of mp-QPs.

In order to start solving the mp-QP problem, we need an initial vector x_0 inside the polyhedral set $\mathcal{K}^*$ of feasible parameters. A possible choiche for x_0 is the Chebychev center (see Section 3.4.5) of $\mathcal{K}^*$:

$$\begin{array}{ll} \max_{x,\bar{z},\epsilon} & \epsilon \\ \text{subj. to} & T_i x + \epsilon \|T_i\|_2 \leq N_i, \quad i = 1, \ldots, n_T \\ & G\bar{z} - Sx \leq W \end{array} \tag{6.73}$$

where n_T is the number of rows T_i of the matrix T defining the set $\mathcal{K}$ in (6.2). If $\epsilon \leq 0$, then the QP problem (6.37) is infeasible for all x in the interior of $\mathcal{K}$. Otherwise, we set $x = x_0$ and solve the QP problem (6.37), in order to obtain the corresponding optimal solution z_0^*. Such a solution is unique, because $H \succ 0$. The value of z_0^* defines the following optimal partition

$$\begin{array}{r} \mathrm{A}(x_0) \triangleq \{j \in J : G_j z_0^* - S_j x_0 - W_j = 0\} \\ \mathrm{NA}(x_0) \triangleq \{j \in J : G_j z_0^* - S_j x_0 - W_j < 0\} \end{array} \tag{6.74}$$

and consequently the critical region $CR_{A(x_0)}$. Once the critical region $CR_{A(x_0)}$ has been defined, the rest of the space $R^{\text{rest}} = \mathcal{K} \backslash CR_{A(x_0)}$ has to be explored and new critical regions generated. An approach for generating a polyhedral partition $\{R_1, \ldots, R_{n_{rest}}\}$ of the rest of the space R^{rest} is described in Theorem 3.1 in Section 3.4.7. Theorem 3.1 provides a way of partitioning the non-convex set $\mathcal{K} \setminus CR_0$ into polyhedral subsets R_i. For each R_i, a new vector x_i is determined by solving the LP (6.73), and, correspondingly, an optimum z_i^*, a set of active constraints A_i, and a critical region CR_i. The procedure proposed in Theorem 3.1 for partitioning the set of parameters allows one to recursively explore the parameter space. Such an iterative procedure terminates after a finite time, as the number of possible combinations of active constraints decreases with each iteration. Two following main elements need to be considered:

1. As for the mp-LP algorithm, the partitioning in Theorem 3.1 defines new polyhedral regions R_k to be explored that are not related to the critical regions which still need to be determined. This may split some of the critical regions, due to the artificial cuts induced by Theorem 3.1. Post-processing can be used to join cut critical regions [39]. As an example, in Figure 6.8 the critical region $CR_{\{3,7\}}$ is discovered twice, one part during the exploration of R_1 and the second part during the exploration of R_2. Although algorithms exist for convexity recognition and computation of the union of polyhedra, the post-processing operation is computationally expensive. Therefore, it is more efficient not to intersect the critical region obtained by (6.27) with halfspaces generated by Theorem 3.1, which is only used to drive the exploration of the parameter space. Then, no post processing is needed to join subpartitioned critical regions. On the other hand, some critical regions may appear more than once. Duplicates can be uniquely identified by the set of active constraints $A(x)$ and can be easily eliminated. To this aim, in the implementation of the algorithm we keep a list of all the critical regions which have already been generated in order to avoid duplicates. In Figure 6.8 the critical region $CR_{\{3,7\}}$ is discovered twice but stored only once.

2. If case 2 occurs in Theorem 6.5 and D is nonzero, $CR_{\mathcal{A}}$ is a lower dimensional critical region (see Remark 6.8). Therefore it is not worth to explore the actual combination G_A, S_A, W_A. On the other hand, if $D = 0$ the KKT conditions do not lead directly to (6.46)–(6.47). In this case, a full-dimensional critical region can be obtained from (6.58) by projecting the set $\mathcal{P}_{x,u^*_{A,2}}$ in (6.57), which, however, is computationally expensive. We suggest the following simpler way to handle case 2 in Theorem 6.5: collect r constraints chosen to satisfy the LICQ, and proceed with the new reduced set, therefore avoiding the computation of projections. Because of the recursive nature of the algorithm, the remaining other possible subsets of combinations of constraints leading to full-dimensional critical regions will automatically be explored later.

Based on the above discussion and results, the main steps of the mp-QP solver are outlined in the following algorithm.

Algorithm 6.3.1

```
Input: Matrices $H,\ G, W, S$ of Problem (6.37) and set $\mathcal{K}$ in (6.2)
Output: Multiparametric solution to Problem (6.37)

1    execute partition($\mathcal{K}$);
2    end.
   procedure partition($Y$)
3        let $x_0 \in Y$ and $\epsilon$ the solution to the LP (6.73);
4        if $\epsilon \leq 0$ then exit; (no full dimensional CR is in $Y$)
5        Solve the QP (6.37) for $x = x_0$ to obtain $(z_0^*, u_0^*)$;
6        Determine the set of active constraints $A$ when $z = z_0^*$, $x = x_0$,
         and build $G_A$, $W_A$, $S_A$;
7        If $r = \operatorname{rank} G_A$ is less than the number $\ell$ of rows of $G_A$, take
         a subset of $r$ linearly independent rows, and redefine
         $G_A$, $W_A$, $S_A$ accordingly;
8        Determine $u_A^*(x)$, $z^*(x)$ from (6.44) and (6.45);
9        Characterize the CR from (6.46) and (6.47);
10       Define and partition the rest of the region as in Theorem 3.1;
11       For each new sub-region $R_i$, partition($R_i$); end procedure.
```

The algorithm explores the set $\mathcal{K}$ of parameters recursively: Partition the rest of the region as in Theorem 3.1 into polyhedral sets R_i, use the same method to partition each set R_i further, and so on.

Remark 6.9. The algorithm solves the mp-QP problem by partitioning the given parameter set $\mathcal{K}$ into N_r closed polyhedral regions. Note that the algorithm generates a partition of the state space which is not strict. The algorithm could be modified to store the critical regions as defined in Section 6.1.2. (which are neither closed nor open as proven in Theorem 6.5) instead of storing their closure. This can be done by keeping track of which facet belongs to a certain critical region and which not. From a practical point of view, such procedure is not necessary since the value function and the optimizer are continuous functions of x.

6.4 Multiparametric Mixed-Integer Linear Programming

6.4.1 Formulation and Properties

Consider the mp-LP

$$\begin{aligned} J^*(x) = \min_z \ & \{J(z,x) = c'z\} \\ \text{subj. to } & Gz \leq W + Sx. \end{aligned} \tag{6.75}$$

where z is the optimization vector, $x \in \mathbb{R}^s$ is the vector of parameters, $G' = [G_1{}' \dots G_m{}']$ and $G_j \in \mathbb{R}^n$ denotes the j-th row of G, $c \in \mathbb{R}^s$, $W \in \mathbb{R}^m$, and $S \in \mathbb{R}^{m \times n}$. When we restrict some of the optimization variables to be 0 or 1, $z \triangleq \{z_c, z_d\}$, $z_c \in \mathbb{R}^{s_c}$, $z_d \in \{0,1\}^{s_d}$ and $s \triangleq s_c + s_d$, we refer to (6.75) as a (right-hand-side) *multiparametric mixed-integer linear program* (mp-MILP).

6.4.2 Geometric Algorithm for mp-MILP

Consider the mp-MILP (6.75). Given a closed and bounded polyhedral set $\mathcal{K} \subset \mathbb{R}^n$ of parameters,

$$\mathcal{K} \triangleq \{x \in \mathbb{R}^n : \ Tx \leq N\}, \tag{6.76}$$

we denote by $\mathcal{K}^* \subseteq \mathcal{K}$ the region of parameters $x \in \mathcal{K}$ such that the MILP (6.75) is feasible and the optimum $J^*(x)$ is finite. For any given $\bar{x} \in \mathcal{K}^*$, $J^*(\bar{x})$ denotes the minimum value of the objective function in problem (6.75) for $x = \bar{x}$. The function $J^* : \mathcal{K}^* \to \mathbb{R}$ will denote the function which expresses the dependence on x of the minimum value of the objective function over $\mathcal{K}^*$, J^* will be called value function. The set-valued function $Z^* : \mathcal{K}^* \to 2^{\mathbb{R}^{s_c}} \times 2^{\{0,1\}^{s_d}}$ will describe for any fixed $x \in \mathcal{K}^*$ the set of optimizers $z^*(x)$ related to $J^*(x)$.

We aim to determine the region $\mathcal{K}^* \subseteq \mathcal{K}$ of feasible parameters x and to find the expression of the value function $J^*(x)$ and the expression of an optimizer function $z^*(x) \in Z^*(x)$.

Two main approaches have been proposed for solving mp-MILP problems. In [2], the authors develop an algorithm based on branch and bound (B&B) methods. At each node of the B&B tree an mp-LP is solved. The solution at the root node represents a valid lower bound, while the solution at a node where all the integer variables have been fixed represents a valid upper bound. As in standard B&B methods, the complete enumeration of combinations of 0-1 integer variables is avoided by comparing the multiparametric solutions, and by fathoming the nodes where there is no improvement of the value function.

In [91] an alternative algorithm was proposed, which will be detailed in this section. Problem (6.75) is alternatively decomposed into an mp-LP and an MILP subproblem. When the values of the binary variable are fixed, an mp-LP is solved, and its solution provides a parametric upper bound to the value function $J^*(x)$. When the parameters in x are treated as free variables, an MILP is solved, that provides a new integer vector. The algorithm is composed of an *initialization step*, and a recursion between the solution of an *mp-LP subproblem* and an *MILP subproblem*.

6.4.2.1 Initialization

Solve the following MILP problem

$$\begin{array}{ll} \min\limits_{\{z,x\}} & c'z \\ \text{subj. to} & Gz - Sx \leq W \\ & x \in \mathcal{K} \end{array} \tag{6.77}$$

where x is treated as an independent variable. If the MILP (6.77) is infeasible then the mp-MILP (6.75) admits no solution, i.e. $\mathcal{K}^* = \emptyset$; otherwise its solution z^* provides a feasible integer variable $\bar{z}_d$.

6.4.2.2 mp-LP subproblem

At a generic step of the algorithm we have a polyhedral partition of the initial set of parameters $\mathcal{K}$. For each polyhedron of the partition we know if

1. the MILP (6.77) is infeasible for all x belonging to the polyhedron.
2. the MILP (6.77) is feasible for all x belonging to the polyhedron and we have a current upper bound on the affine value function $J^*(x)$ (in the polyhedron) and an integer variable that improves the bound at least at a point x of the polyhedron,
3. the MILP (6.77) is feasible for all x belonging to the polyhedron and we know the optimal affine value function $J^*(x)$ inside the polyhedron.

Obviously the algorithm will continue to iterate only on the polyhedra corresponding to point 2 above (if there is no such a polyhedron then the algorithm ends) and in particular, at step j we assume to have stored

1. A list of N_j polyhedral regions CR_i and for each of them an associated parametric affine upper bound $\bar{J}_i(x)$ ($\bar{J}_i(x) = +\infty$ if no integer solution has been found yet in CR_i).
2. For each CR_i a set of integer variables $Z_i = \bar{z}_{d_i}^0, \dots, \bar{z}_{d_i}^{Nb_i}$, that have already been explored in the region CR_i.
3. For each CR_i an integer feasible variable $\bar{z}_{d_i}^{Nb_i+1} \notin Z_i$ such that there exists z_c and $\hat{x} \in CR_i$ for which $Gz \leq W + S\hat{x}$ and $c'z < \bar{J}_i(\hat{x})$ where

$z = \{z_c, \bar{z}_{d_i}^{Nb_i+1}\}$. That is, $\bar{z}_{d_i}^{Nb_i+1}$ is an integer variable that improves the current bound for at least one point of the current polyhedron.

At step $j = 0$, set: $N_0 = 1$, $CR_1 = \mathcal{K}$, $Z_1 = \emptyset$, $\bar{J}_1 = +\infty$, $\bar{z}_{d_1}^1 = \bar{z}_d$.

For each CR_i we solve the following mp-LP problem

$$\begin{array}{rl} \tilde{J}_i(x) = & \min_z \; c'z \\ & \text{subj. to } Gz \le W + Sx \\ & \quad z_d = \bar{z}_{d_i}^{Nb_i+1} \\ & \quad x \in CR_i \end{array} \tag{6.78}$$

By Theorem 6.4, the solution of mp-LP (6.78) provides a partition of CR_i into polyhedral regions $R_i^k,\ k = 1, \ldots, N_{R_i}$ and and a PWA value function

$$\tilde{J}_i(x) = (\tilde{JR}_i^k(x) \triangleq c_i^k x + p_i^k) \text{ if } x \in R_i^k,\ k = 1, \ldots, N_{R_i} \tag{6.79}$$

where $\tilde{JR}_i^j(x) = +\infty$ in R_i^j if the integer variable $\bar{z}_{d_i}$ is not feasible in R_i^j and a PWA continuous control law $z^*(x)$ ($z^*(x)$ is not defined in R_i^j if $\tilde{JR}_i^j(x) = +\infty$).

The function $\tilde{J}_i(x)$ will be an upper bound of $J^*(x)$ for all $x \in CR_i$. Such a bound $\tilde{J}_i(x)$ on the value function has to be compared with the current bound $\bar{J}_i(x)$ in CR_i in order to obtain the lowest of the two parametric value functions and to update the bound.

While updating $\bar{J}_i(x)$ three cases are possible:

1. $\bar{J}_i(x) = \tilde{JR}_i^k(x)\ \forall x \in R_i^k$ if $(\tilde{JR}_i^k(x) \le \bar{J}_i(x)\ \forall\ x \in R_i^k)$
2. $\bar{J}_i(x) = \bar{J}_i(x)\ \forall x \in R_i^k$ (if $\tilde{JR}_i^k(x) \ge \bar{J}_i(x)\ \forall\ x \in R_i^k$)
3. $\bar{J}_i(x) = \begin{cases} \bar{J}_i(x) & \forall x \in (R_i^k)_1 \triangleq \{x \in R_i^k :\ \tilde{JR}_i^k(x) \ge \bar{J}_i(x)\} \\ \tilde{JR}_i^k(x) & \forall x \in (R_i^k)_2 \triangleq \{x \in R_i^k :\ \tilde{JR}_i^k(x) \le \bar{J}_i(x)\} \end{cases}$

The three cases above can be distinguished by using a simple linear program. In the third case, the region R_i^k is partitioned into two regions $(R_i^k)_1$ and $(R_i^k)_2$ that are convex polyhedra since $\tilde{JR}_i^k(x)$ and $\bar{J}_i(x)$ are affine functions of x.

After the mp-LP (6.78) has been solved for all $i = 1, \ldots, N_j$ (the subindex j denotes the that we are at step j of the recursion) and the value function has been updated, each initial region CR_i has been subdivided into at most $2N_{R_i}$ polyhedral regions R_i^k and possibly $(R_i^k)_1$ and $(R_i^k)_2$ with a corresponding updated parametric bound on the value function $\bar{J}_i(x)$. For each R_i^k, $(R_i^k)_1$ and $(R_i^k)_2$ we define the set of integer variables already explored as $Z_i = Z_i \bigcup \bar{z}_{d_i}^{Nb_i+1}$, $Nb_i = Nb_i + 1$. In the sequel the polyhedra of the new partition will be referred to as CR_i.

6.4.2.3 MILP subproblem

At step j for each critical region CR_i (note that these CR_i are the output of the previous phase) we solve the following MILP problem.

$$\begin{aligned}
\min_{\{z,x\}} \quad & c'z && (6.80)\\
\text{subj. to} \quad & Gz - Sx \leq W && (6.81)\\
& c'z \leq \bar{J}_i(z) && (6.82)\\
& z_d \neq \bar{z}^k_{d_i},\ \ k = 1, \ldots, Nb_i && (6.83)\\
& x \in CR_i && (6.84)
\end{aligned}$$

where constraints (6.83) prohibit integer solutions that have been already analyzed in CR_i from appearing again and constraint (6.82) excludes integer solutions with higher values than the current upper bound. If problem (6.84) is infeasible then the region CR_i is excluded from further recursion and the current upper bound represents the final solution. If problem (6.84) is feasible, then the discrete optimal component $z^*_{d_i}$ is stored and represents a feasible integer variable that is optimal at least in one point of CR_i.

6.4.2.4 Recursion

For all the region CR_i not excluded from the MILP's subproblem (6.80)-(6.84) the algorithm continues to iterate between the mp-LP (6.78) with $\bar{z}^{Nb_i+1}_{d_i} = z^*_{d_i}$ and the MILP (6.80)-(6.84). The algorithm terminates when all the MILPs (6.80)-(6.84) are infeasible.

Note that the algorithm generates a partition of the state space. Some parameter x could belong to the boundary of several regions. Differently from the LP and QP case, the value function may be discontinuous and therefore such a case has to be treated carefully. If a point x belong to different critical regions, the expressions of the value function associated with such regions have to be compared in order to assign to x the right optimizer. Such a procedure can be avoided by keeping track of which facet belongs to a certain critical region and which not. Moreover, if the value functions associated with the regions containing the same parameter x coincide this may imply the presence of multiple optimizers.

6.4.3 Theoretical Results

The following properties of $J^*(x)$ and $Z^*(x)$ easily follow from the algorithm described above.

Theorem 6.10. *Consider the mp-MILP (6.75). Then, the set $\mathcal{K}^*$ is the union of a finite number of (possibly open) polyhedra and the value function J^* is piecewise affine on polyhedra. If the optimizer $z^*(x)$ is unique for all $x \in \mathcal{K}^*$, then the optimizer functions $z_c^* : \mathcal{K}^* \to \mathbb{R}^{s_c}$ and $z_d^* : \mathcal{K}^* \to \{0,1\}^{s_d}$ are piecewise affine and piecewise constant, respectively, on polyhedra. Otherwise, it is always possible to define a piecewise affine optimizer function $z^*(x) \in Z^*(x)$ for all $x \in \mathcal{K}^*$.*

Note that differently from the mp-LP case, the set $\mathcal{K}^*$ can be non-convex and even disconnected.

6.5 Multiparametric Mixed-Integer Quadratic Programming

6.5.1 Formulation and Properties

Consider the mp-QP

$$\begin{array}{rl} J^*(x) = \min\limits_{z} & J(z,x) = z'H_1 z + c_1 z \\ & \text{subj. to } Gz \le W + Sx, \end{array} \tag{6.85}$$

When we restrict some of the optimization variables to be 0 or 1, $z \triangleq \{z_c, z_d\}$, where $z_c \in \mathbb{R}^{s_c}$, $z_d \in \{0,1\}^{s_d}$, we refer to (6.85) as a *multiparametric mixed-integer quadratic program* (mp-MIQP). Given a closed and bounded polyhedral set $\mathcal{K} \subset \mathbb{R}^n$ of parameters,

$$\mathcal{K} \triangleq \{x \in \mathbb{R}^n :\ Tx \le N\}, \tag{6.86}$$

we denote by $\mathcal{K}^* \subseteq \mathcal{K}$ the region of parameters $x \in \mathcal{K}$ such that the MIQP (6.85) is feasible and the optimum $J^*(x)$ is finite. For any given $\bar{x} \in \mathcal{K}^*$, $J^*(\bar{x})$ denotes the minimum value of the objective function in problem (6.85) for $x = \bar{x}$. The value function $J^* : \mathcal{K}^* \to \mathbb{R}$ denotes the function which expresses the dependence on x of the minimum value of the objective function over $\mathcal{K}^*$. The set-valued function $Z^* : \mathcal{K}^* \to 2^{\mathbb{R}^{s_c}} \times 2^{\{0,1\}^{s_d}}$ describes for any fixed $x \in \mathcal{K}^*$ the set of optimizers $z^*(x)$ related to $J^*(x)$.

We aim at determining the region $\mathcal{K}^* \subseteq \mathcal{K}$ of feasible parameters x and at finding the expression of the value function $J^*(x)$ and the expression of an optimizer function $z^*(x) \in Z^*(x)$.

We show with a simple example that the geometric approach in this chapter cannot be used for solving mp-MIQPs. Suppose z_1, z_2, x_1, $x_2 \in \mathbb{R}$ and $\delta \in \{0,1\}$, then the following mp-MIQP

$$
\begin{array}{ll}
J^*(x_1,x_2) = \min_{z_1,z_2,\delta} \; z_1^2 + z_2^2 - 25\delta + 100 & \\
\text{subj. to} & \begin{bmatrix} 1 & 0 & 10 \\ -1 & 0 & 10 \\ 0 & 1 & 10 \\ 0 & -1 & 10 \\ 1 & 0 & -10 \\ -1 & 0 & -10 \\ 0 & 1 & -10 \\ 0 & -1 & -10 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ \delta \end{bmatrix} \leq \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 10 \\ 10 \\ 10 \\ 10 \\ 0 \\ 0 \\ 0 \\ 0 \\ 10 \\ 10 \\ 10 \\ 10 \end{bmatrix}
\end{array}
\tag{6.87}
$$

can be simply solved by noting that for $\delta = 1$ $z_1 = x_1$ and $z_2 = x_2$ while for $\delta = 0$ $z_1 = z_2 = 0$. By comparing the value functions associated with $\delta = 0$ and $\delta = 1$ we obtain two critical regions

$$
\begin{array}{l}
CR_1 = \{x_1, x_2 \in \mathbb{R} : \; x_1^2 + x_2^2 \leq 75\} \\
CR_2 = \{x_1, x_2 \in \mathbb{R} : \; -10 \leq x_1 \leq 10, \; -10 \leq x_2 \leq 10, \; x_1^2 + x_2^2 > 75\}
\end{array}
\tag{6.88}
$$

with the associated parametric optimizer

$$
\begin{array}{l}
z_1^*(x_1,x_2) = \begin{cases} x_1 & \text{if } [x_1,x_2] \in CR_1 \\ 0 & \text{if } [x_1,x_2] \in CR_2 \end{cases} \\
z_2^*(x_1,x_2) = \begin{cases} x_2 & \text{if } [x_1,x_2] \in CR_1 \\ 0 & \text{if } [x_1,x_2] \in CR_2 \end{cases}
\end{array}
\tag{6.89}
$$

and the parametric value function

$$
J^*(x_1,x_2) = \begin{cases} x_1^2 + x_2^2 + 75 & \text{if } [x_1,x_2] \in CR_1 \\ 100 & \text{if } [x_1,x_2] \in CR_2 \end{cases}
\tag{6.90}
$$

The two critical regions and the value function are depicted in Figure 6.12.

This example demonstrate that, in general, the critical regions of an mp-MIQP cannot be decomposed into convex polyhedra. Therefore the method of partitioning the rest of the space presented in Theorem 3.1 cannot be applied here.

To the authors' knowledge, there does not exist an efficient method for solving general mp-MIQPs. In Chapter 15 we will present an algorithm that efficiently solves specific mp-MIQPs that stem from optimal control of discrete-time hybrid systems.

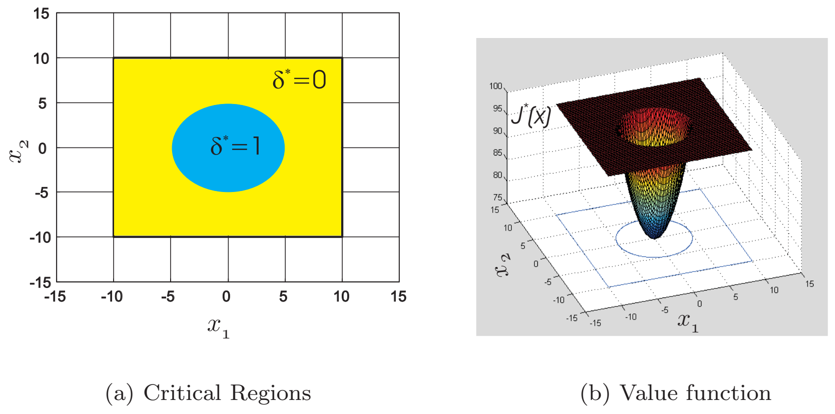


Fig. 6.12 Solution to the mp-MIQP (6.87)

6.6 Literature Review

Many of the theoretical results on parametric programming can be found in[17, 100, 43, 107].

The first method for solving *parametric linear programs* was proposed by Gass and Saaty [111], and since then extensive research has been devoted to sensitivity and (multi)-parametric linear analysis, as attested by the hundreds of references in [106] (see also [107] for recent advances in the field). One of the first methods for solving multiparametric linear programs (mp-LPs) was formulated by Gal and Nedoma [108]. The method constructs the critical regions iteratively, by visiting the graph of bases associated with the LP tableau of the original problem. Many of the results presented in this book on mp-LPs can be found in [106, p. 178-180].

Note that in [108, 106] a critical region is defined as a subset of the parameter space on which a certain basis of the linear program is optimal. The algorithm proposed in [108] for solving multiparametric linear programs generates non-overlapping critical regions by generating and exploring the graph of bases. In the graph of bases the nodes represent optimal bases of the given multiparametric problem and two nodes are connected by an edge if it is possible to pass from one basis to another by one pivot step (in this case the bases are called neighbors). In this book we use the definition (6.4) of critical regions which is not associated with the bases but with the set of active constraints and it is directly related to the definition given in [3, 191, 107].

The solution to *multiparametric quadratic programs* has been studied in detail in [17, Chapter 5]. Bemporad and coauthors in [39] presented a simple method for solving mp-QPs. The method constructs a critical region in a neighborhood of a given parameter, by using the Karush-Kuhn-Tucker conditions for optimality, and then recursively explores the parameter space outside such a region. Other algorithms for solving mp-QPs have been proposed by Seron, DeDoná and Goodwin in [236, 89] in parallel with the study of Bemporad and coauthors in [39], by Tondel, Johansen and Bemporad in [248] and by Baotic in [18]. All these algorithms are based on an iterative procedure that builds up the parametric solution by generating new polyhedral regions of the parameter space at each step. The methods differ in the way they explore the parameter space, that is, the way they identify active constraints corresponding to the critical regions neighboring to a given critical region, i.e., in the "active set generator" component.

In [236, 89] the authors construct the unconstrained critical region and then generate neighboring critical regions by enumerating all possible combinations of active constraints.

In [248] the authors explore the parameter space outside a given region CR_i by examining its set of active constraints A_i. The critical regions neighboring to CR_i are constructed by elementary operations on the active constraints set A_i that can be seen as an equivalent "pivot" for the quadratic program. For this reason the method can be considered as an extension of the method of Gal [106] to multiparametric quadratic programming.

In [18] the author uses a direct exploration of the parameter space as in [39] but he avoids the partition of the state space described in Theorem 3.1. Given a polyhedral critical region CR_i, the procedure goes through all its facets and generates the Chebychev center of each facet. For each facet $\mathcal{F}_i$ a new parameter x_ε^i is generated, by moving from the center of the facet in the direction of the normal to the facet by a small step. If such parameter x_ε^i 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 CR_i is found by solving a QP for the new parameter x_ε^i.

In [2, 91] two approaches were proposed for solving mp-MILP problems. In both methods the authors use an mp-LP algorithm and a branch and bound strategy that avoids the complete enumeration of combinations of 0-1 integer variables by comparing the available bounds on the multiparametric solutions.

Part III
Optimal Control

Chapter 7
General Formulation and Discussion

In this chapter we introduce the optimal control problem we will be studying in a very general form. We want to communicate the basic definitions and essential concepts. We will sacrifice mathematical precision for the sake of simplicity. In later chapters we will study specific versions of this problem for specific cost functions and system classes in greater detail.

7.1 Problem Formulation

We consider the nonlinear time-invariant system

$$x(t+1) = g(x(t), u(t)), \tag{7.1}$$

subject to the constraints

$$h(x(t), u(t)) \leq 0 \tag{7.2}$$

at all time instants $t \geq 0$. In (7.1)–(7.2), $x(t) \in \mathbb{R}^n$ and $u(t) \in \mathbb{R}^m$ are the state and input vector, respectively. Inequality (7.2) with $h : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^{n_c}$ expresses the n_c constraints imposed on the input and the states. These may be simple upper and lower bounds or more complicated expressions. We assume that the origin is an equilibrium point ($g(0,0) = 0$) in the interior of the feasible set, i.e., $h(0,0) < 0$.

We assumed the system to be specified in discrete time. One reason is that we are looking for solutions to engineering problems. In practice, the controller will almost always be implemented through a digital computer by sampling the variables of the system and transmitting the control action to the system at discrete time points. Another reason is that for the solution of the optimal control problems for discrete-time systems we will be able to make ready use of powerful mathematical programming software.

We want to caution the reader, however, that in many instances the discrete time model is an approximation of the continuous time model. It is generally difficult to derive "good" discrete time models from nonlinear continuous time models, and especially so when the nonlinear system has discontinuities as would be the case for switched systems. We also note that continuous time switched systems can exhibit behavioral characteristics not found in discrete-time systems, for example, an ever increasing number of switches in an ever decreasing time interval (*Zeno behavior* [118]).

We define the following *performance objective* or *cost function* from time instant 0 to time instant N

$$J_{0\to N}(x_0, U_{0\to N}) \triangleq p(x_N) + \sum_{k=0}^{N-1} q(x_k, u_k) \tag{7.3}$$

where N is the time *horizon* and x_k denotes the state vector at time k obtained by starting from the measured state $x_0 = x(0)$ and applying to the system model

$$x_{k+1} = g(x_k, u_k), \tag{7.4}$$

the input sequence $u_0, \ldots, u_{k-1}$. From this sequence we define the vector of future inputs $U_{0\to N} \triangleq [u_0', \ldots, u_{N-1}']' \in \mathbb{R}^s$, $s \triangleq mN$. The terms $q(x_k, u_k)$ and $p(x_N)$ are referred to as *stage cost* and *terminal cost*, respectively, and are assumed to be positive definite ($q \succ 0,\ p \succ 0$):

$$\begin{array}{l} p(x,u) > 0\ \forall x \neq 0,\ u \neq 0,\quad p(0,0) = 0 \\ q(x,u) > 0\ \forall x \neq 0,\ u \neq 0,\quad q(0,0) = 0 \end{array}$$

The form of the cost function (7.3) is very general. If a practical control objective can be expressed as a scalar function then this function usually takes the indicated form. Specifically, we consider the following constrained finite time optimal control (CFTOC) problem.

$$\begin{array}{rll} J^*_{0\to N}(x_0) = & \min_{U_{0\to N}} & J_{0\to N}(x_0, U_{0\to N}) \\ & \text{subj. to} & x_{k+1} = g(x_k, u_k),\ k = 0, \ldots, N-1 \\ & & h(x_k, u_k) \leq 0,\ k = 0, \ldots, N-1 \\ & & x_N \in \mathcal{X}_f \\ & & x_0 = x(0) \end{array} \tag{7.5}$$

Here $\mathcal{X}_f \subseteq \mathbb{R}^n$ is a *terminal region* that we want the system states to reach at the end of the horizon. The terminal region could be the origin, for example. We define $\mathcal{X}_{0\to N} \subseteq \mathbb{R}^n$ to be the set of initial conditions $x(0)$ for which there exists an input vector $U_{0\to N}$ so that the inputs $u_0, \ldots, u_{N-1}$ and the states $x_0, \ldots, x_N$ satisfy the model $x_{k+1} = g(x_k, u_k)$ and the constraints $h(x_k, u_k) \leq 0$ and that the state x_N lies in the terminal set $\mathcal{X}_f$.

We can determine this set of feasible initial conditions in a recursive manner. Let us denote with $\mathcal{X}_{j\to N}$ the set of states x_j at time j which can be steered into $\mathcal{X}_f$ at time N, i.e., for which the model $x_{k+1} = g(x_k, u_k)$ and the constraints $h(x_k, u_k) \leq 0$ are feasible for $k = j, \dots, N-1$ and $x_N \in \mathcal{X}_f$. This set can be defined recursively by

$$\begin{aligned} \mathcal{X}_{j\to N} &= \{x \in \mathbb{R}^n : \ \exists u \text{ such that} \, (h(x,u) \leq 0, \text{ and } g(x,u) \in \mathcal{X}_{j+1\to N} \ )\}, \\ & \quad j = 0, \dots, N-1 \qquad (7.6) \\ \mathcal{X}_{N\to N} &= \mathcal{X}_f. \qquad (7.7) \end{aligned}$$

The set $\mathcal{X}_{0\to N}$ is the final result of these iterations starting with $\mathcal{X}_f$.

The optimal cost $J^*_{0\to N}(x_0)$ is also called *value function*. In general, the problem (7.3)–(7.5) may not have a minimum, but only an infimum. We will assume that there exists a minimum. This is the case, for example, when the set of feasible input vectors $U_{0\to N}$ (defined by h and $\mathcal{X}_f$) is compact and when the functions g, p and q are continuous. Also, there might be several input vectors $U^*_{0\to N}$ which yield the minimum ($J^*_{0\to N}(x_0) = J_{0\to N}(x_0, U^*_{0\to N})$). In this case we will define one of them as the minimizer $U^*_{0\to N}$.

Note that throughout the book we will distinguish between the *current* state $x(k)$ of system (7.1) at time k and the variable x_k in the optimization problem (7.5), that is the *predicted* state of system (7.1) at time k obtained by starting from the state x_0 and applying to system (7.4) the input sequence $u_0, \dots, u_{k-1}$. Analogously, $u(k)$ is the input applied to system (7.1) at time k while u_k is the k-th optimization variable of the optimization problem (7.5). Clearly, $x(k) = x_k$ for any k if $u(k) = u_k$ for all k.

In the rest of this chapter we will be interested in the following questions related to the general optimal control problem (7.3)–(7.5).

- *Solution.* We will show that the problem can be expressed and solved either as one general nonlinear programming problem, or in a recursive manner by invoking Bellman's Principle of Optimality.
- *Infinite horizon.* We will investigate if a solution exists as $N \to \infty$, the properties of this solution, and how it can be obtained or at least approximated by using a *receding horizon* technique.

7.2 Solution via Batch Approach

If we write the equality constraints appearing in (7.5) explicitly

$$\begin{aligned} x_1 &= g(x(0), u_0) \\ x_2 &= g(x_1, u_1) \\ &\vdots \\ x_N &= g(x_{N-1}, u_{N-1}) \end{aligned} \qquad (7.8)$$

then the optimal control problem (7.3)–(7.5), rewritten below

$$\begin{array}{rl} J^*_{0\to N}(x_0) = \min_{U_{0\to N}} & p(x_N) + \sum_{k=0}^{N-1} q(x_k, u_k) \\ \text{subj. to} & x_1 = g(x_0, u_0) \\ & x_2 = g(x_1, u_1) \\ & \vdots \\ & x_N = g(x_{N-1}, u_{N-1}) \\ & h(x_k, u_k) \leq 0,\ k = 0, \ldots, N-1 \\ & x_N \in \mathcal{X}_f \\ & x_0 = x(0) \end{array} \qquad (7.9)$$

is recognized more easily as a general nonlinear programming problem with variables $u_0, \ldots, u_{N-1}$ and $x_1, \ldots, x_N$.

As an alternative we may try to eliminate the state variables and equality constraints (7.8) by successive substitution so that we are left with $u_0, \ldots, u_{N-1}$ as the only decision variables. For example, we can express x_2 as a function of $x(0), u_0$ and u_1 only, by eliminating the intermediate state x_1

$$\begin{array}{l} x_2 = g(x_1, u_1) \\ x_2 = g(g(x(0), u_0), u_1). \end{array} \qquad (7.10)$$

Except when the state equations are linear this successive substitution may become complex. Even when they are linear it may be bad from a numerical point of view.

Either with or without successive substitution the solution of the nonlinear programming problem is a sequence of present and future inputs $U^*_{0\to N} = [u_0^{*'}, \ldots, u_{N-1}^{*'}]'$ determined for the particular initial state $x(0)$.

7.3 Solution via Recursive Approach

The recursive approach, Bellman's dynamic programming technique, rests on a simple idea, the *principle of optimality*. It states that for a trajectory $x_0, x_1^*, \ldots, x_N^*$ to be optimal, the trajectory starting from any intermediate point x_j^*, i.e. $x_j^*, x_{j+1}^*, \ldots, x_N^*$, $0 \leq j \leq N-1$, must be optimal.

Consider the following example to provide an intuitive justification [46]. Suppose that the fastest route from Los Angeles to Boston passes through Chicago. Then the principle of optimality formalizes the obvious fact that the Chicago to Boston portion of the route is also the fastest route for a trip that starts from Chicago and ends in Boston.

We can utilize the principle of optimality for the optimal control problem we are investigating. We define the cost over the reduced horizon from j to N

$$J_{j\to N}(x_j, u_j, u_{j+1}, \ldots, u_{N-1}) \triangleq p(x_N) + \sum_{k=j}^{N-1} q(x_k, u_k), \tag{7.11}$$

also called the *cost-to-go*. Then the *optimal cost-to-go* $J^*_{j\to N}$ is

$$\begin{array}{lll} J^*_{j\to N}(x_j) \triangleq & \min_{u_j,u_{j+1},\ldots,u_{N-1}} & J_{j\to N}(x_j, u_j, u_{j+1}, \ldots, u_{N-1}) \\ & \text{subj. to} & x_{k+1} = g(x_k, u_k),\ k = j, \ldots, N-1 \\ & & h(x_k, u_k) \le 0,\ k = j, \ldots, N-1 \\ & & x_N \in \mathcal{X}_f \end{array} \tag{7.12}$$

Note that the optimal cost-to-go $J^*_{j\to N}(x_j)$ depends only on the initial state x_j.

The principle of optimality implies that the optimal cost-to-go $J^*_{j-1\to N}$ from time $j-1$ to the final time N can be found by minimizing the sum of the stage cost $q(x_{j-1}, u_{j-1})$ and the optimal cost-to-go $J^*_{j\to N}(x_j)$ from time j onwards.

$$\begin{array}{lll} J^*_{j-1\to N}(x_{j-1}) = & \min\limits_{u_{j-1}} & q(x_{j-1}, u_{j-1}) + J^*_{j\to N}(x_j) \\ & \text{subj. to} & x_j = g(x_{j-1}, u_{j-1}) \\ & & h(x_{j-1}, u_{j-1}) \le 0 \\ & & x_j \in \mathcal{X}_{j\to N} \end{array} \tag{7.13}$$

Here the only decision variable left for the optimization is u_{j-1}, the input at time $j-1$. All the other inputs $u^*_j, \ldots, u^*_{N-1}$ have already been selected optimally to yield the optimal cost-to-go $J^*_{j\to N}(x_j)$. We can rewrite (7.13) as

$$\begin{array}{lll} J^*_{j-1\to N}(x_{j-1}) = & \min\limits_{u_{j-1}} & q(x_{j-1}, u_{j-1}) + J^*_{j\to N}(g(x_{j-1}, u_{j-1})) \\ & \text{subj. to} & h(x_{j-1}, u_{j-1}) \le 0 \\ & & g(x_{j-1}, u_{j-1}) \in \mathcal{X}_{j\to N}, \end{array} \tag{7.14}$$

making the dependence of x_j on the initial state x_{j-1} explicit.

The optimization problem (7.14) suggests the following recursive algorithm backwards in time to determine the optimal control law. We start with the terminal cost and constraint

$$J^*_{N\to N}(x_N) = p(x_N) \tag{7.15}$$

$$\mathcal{X}_{N\to N} = \mathcal{X}_f, \tag{7.16}$$

and then proceed backwards

$$\begin{array}{llll}
J^*_{N-1\to N}(x_{N-1}) = & \min\limits_{u_{N-1}} & q(x_{N-1},u_{N-1}) + & J^*_{N\to N}(g(x_{N-1},u_{N-1})) \\
 & \text{subj. to} & h(x_{N-1},u_{N-1}) \le 0, & \\
 & & g(x_{N-1},u_{N-1}) \in \mathcal{X}_{N\to N} & \\
\vdots & & & \\
J^*_{0\to N}(x_0) = & \min\limits_{u_0} & q(x_0,u_0) + & J^*_{1\to N}(g(x_0,u_0)) \\
 & \text{subj. to} & h(x_0,u_0) \le 0, & \\
 & & g(x_0,u_0) \in \mathcal{X}_{1\to N} & \\
 & & x_0 = x(0). &
\end{array} \tag{7.17}$$

This algorithm, popularized by Bellman, is referred to as *dynamic programming*. The dynamic programming problem is appealing because it can be stated compactly and because at each step the optimization takes place over one element u_j of the optimization vector only. This optimization is rather complex, however. It is not a standard nonlinear programming problem but we have to construct the optimal cost-to-go $J^*_{j\to N}(x_j)$, a *function* defined over the subset $\mathcal{X}_{j\to N}$ of the state space.

In a few special cases we know the type of function and we can find it efficiently. For example, in the next chapter we will cover the case when the system is linear and the cost is quadratic. Then the optimal cost-to-go is also quadratic and can be constructed rather easily. Later in the book we will show that, when constraints are added to this problem, the optimal cost-to-go becomes piecewise quadratic and efficient algorithms for its construction are also available.

In general, however, we may have to resort to a "brute force" approach to construct the cost-to-go function $J^*_{j-1\to N}$ and to solve the dynamic program. Let us assume that at time $j-1$ the cost-to-go $J^*_{j\to N}$ is known and discuss how to construct an approximation of $J^*_{j-1\to N}$. With $J^*_{j\to N}$ known, for a fixed x_{j-1} the optimization problem (7.14) becomes a standard nonlinear programming problem. Thus, we can define a grid in the set $\mathcal{X}_{j-1\to N}$ of the state space and compute the optimal cost-to-go function on each grid point. We can then define an approximate value function $\tilde{J}^*_{j-1\to N}(x_{j-1})$ at intermediate points via interpolation. The complexity of constructing the cost-to-go function in this manner increases rapidly with the dimension of the state space ("curse of dimensionality").

The extra benefit of solving the optimal control problem via dynamic programming is that we do not only obtain the vector of optimal inputs $U^*_{0\to N}$ for a particular initial state $x(0)$ as with the batch approach. At each time j the optimal cost-to-go function defines implicitly a nonlinear feedback control law.

$$\begin{array}{lll}
u^*_j(x_j) = \arg & \min\limits_{u_j} & q(x_j,u_j) + \; J^*_{j+1\to N}(g(x_j,u_j)) \\
 & \text{subj. to} & h(x_j,u_j) \le 0, \\
 & & g(x_j,u_j) \in \mathcal{X}_{j+1\to N}
\end{array} \tag{7.18}$$

For a fixed x_j this nonlinear programming problem can be solved quite easily in order to find $u_j^*(x_j)$. Because the optimal cost-to-go function $J_{j\to N}^*(x_j)$ changes with time j, the nonlinear feedback control law is time-varying.

7.4 Optimal Control Problem with Infinite Horizon

We are interested in the optimal control problem (7.3)–(7.5) as the horizon N approaches infinity.

$$\begin{array}{rl} J_{0\to\infty}^*(x_0) = \min_{u_0,u_1,\ldots} & \sum_{k=0}^{\infty} q(x_k,u_k) \\ \text{subj. to} & x_{k+1} = g(x_k,u_k),\ k=0,\ldots,\infty \\ & h(x_k,u_k) \le 0,\ k=0,\ldots,\infty \\ & x_0 = x(0) \end{array} \tag{7.19}$$

We define the set of initial conditions for which this problem has a solution.

$$\mathcal{X}_{0\to\infty} = \{x(0)\in\mathbb{R}^n : \ \text{Problem (7.19) is feasible and } J_{0\to\infty}^*(x(0)) < +\infty\}. \tag{7.20}$$

For the value function $J_{0\to\infty}^*(x_0)$ to be finite it must hold that

$$\lim_{k\to\infty} q(x_k,u_k) = 0$$

and because $q(x_k,u_k) > 0$ for all $(x_k,u_k)\neq 0$

$$\lim_{k\to\infty} x_k = 0$$

and

$$\lim_{k\to\infty} u_k = 0.$$

Thus the sequence of control actions generated by the solution of the infinite horizon problem drives the system to the origin. For this solution to exists the system must be - loosely speaking - stabilizable.

Using the recursive dynamic programming approach we can seek the solution of the infinite horizon optimal control problem by increasing N until we observe convergence. If the dynamic programming algorithm converges as $N\to\infty$ then (7.14) becomes the Bellman equation

$$\begin{array}{rl} J^*(x) = \min_u & q(x,u) + J^*(g(x,u)) \\ \text{subj. to} & h(x,u) \le 0 \\ & g(x,u) \in \mathcal{X}_{0\to\infty} \end{array} \tag{7.21}$$

This procedure of simply increasing N may not be well behaved numerically and it may also be difficult to define a convergence criterion that is meaningful for the control problem. We will describe a method, called *Value Function Iteration*, in the next section.

An alternative is *receding horizon control* which can yield a time invariant controller guaranteeing convergence to the origin without requiring $N \to \infty$. We will describe this important idea later in this chapter.

7.4.1 Value Function Iteration

Once the value function $J^*(x)$ is known, the nonlinear feedback control law $u^*(x)$ is defined implicitly by (7.21)

$$\begin{array}{rl} u^*(x) = \arg\min_u & q(x,u) + J^*(g(x,u)) \\ \text{subj. to} & h(x,u) \le 0 \\ & g(x,u) \in \mathcal{X}_{0\to\infty} \end{array} \tag{7.22}$$

It is *time invariant* and guarantees convergence to the origin for all states in $\mathcal{X}_{0\to\infty}$. For a given $x \in \mathcal{X}_{0\to\infty}$, $u^*(x)$ can be found from (7.21) by solving a standard nonlinear programming problem.

In order to find the value function $J^*(x)$ we do not need to compute the sequence of cost-to-go functions implied by the dynamic program (7.14) but we can solve (7.21) directly. We can start with some initial guess $\tilde{J}_0^*(x)$ for the value function and an initial guess $\tilde{\mathcal{X}}_0$ for the region in the state space where we expect the infinite horizon problem to converge and iterate

$$\begin{array}{rl} \tilde{J}_{i+1}^*(x) = \min_u & q(x,u) + \tilde{J}_i^*(g(x,u)) \\ \text{subj. to} & h(x,u) \le 0 \\ & g(x,u) \in \tilde{\mathcal{X}}_i \end{array} \tag{7.23}$$

$$\tilde{\mathcal{X}}_{i+1} = \{x \in \mathbb{R}^n : \ \exists u \ (h(x,u) \le 0, \ \text{and } g(x,u) \in \tilde{\mathcal{X}}_i \ )\} \tag{7.24}$$

Note that here i is the iteration index and does not denote time. This iterative procedure is called *value function iteration*. It can be executed as follows. Let us assume that at iteration step i we gridded the set $\tilde{\mathcal{X}}_i$ and that $\tilde{J}_i^*(x)$ is known at each grind point from the previous iteration. We can approximate $\tilde{J}_i^*(x)$ at intermediate points via interpolation. For a fixed point $\bar{x}$ the optimization problem (7.23) is a nonlinear programming problem yielding $\tilde{J}_i^*(\bar{x})$. In this manner the approximate value function $\tilde{J}_i^*(x)$ can be constructed at all grid points and we can proceed to the next iteration step $i+1$.

7.4.2 Receding Horizon Control

Receding Horizon Control will be covered in detail in Chapter 11. Here we illustrate the main idea and discuss the fundamental properties.

Assume that at time $t = 0$ we determine the control action u_0 by solving the finite horizon optimal control problem (7.3)-(7.5). If $J^*_{0\to N}(x_0)$ converges to $J^*_{0\to\infty}(x_0)$ as $N \to \infty$ then the effect of increasing N on the value of u_0 should diminish as $N \to \infty$. Thus, intuitively, instead of making the horizon infinite we can get a similar behavior when we use a long, but finite horizon N, and repeat this optimization at each time step, in effect moving the horizon forward (*moving horizon* or *receding horizon* control).

In the *batch approach* we would solve an optimal control problem with horizon N yielding a sequence of optimal inputs $u_0^*, \ldots, u_{N-1}^*$, but we would implement only the first one of these inputs u_0^*. At the next time step we would measure the current state and then again solve the N-step problem with the current state as new initial condition x_0. If the horizon N is long enough then we expect that this approximation of the infinite horizon problem should not matter and the implemented sequence should drive the states to the origin.

In the *dynamic programming approach* we would always implement the control u_0 obtained from the optimization problem

$$
\begin{array}{rll}
J^*_{0\to N}(x_0) = & \min\limits_{u_0} & q(x_0,u_0) + \ J^*_{1\to N}(g(x_0,u_0)) \\
& \text{subj. to} & h(x_0,u_0) \le 0, \\
& & g(x_0,u_0) \in \mathcal{X}_{1\to N}, \\
& & x_0 = x(0)
\end{array}
\tag{7.25}
$$

where $J^*_{1\to N}(g(x_0,u_0))$ is the optimal cost-to-go from the state $x_1 = g(x_0,u_0)$ at time 1 to the end of the horizon N.

If the dynamic programming iterations converge as $N \to \infty$, then for a long, but finite horizon N we expect that this receding horizon approximation of the infinite horizon problem should not matter and the resulting controller will drive the system asymptotically to the origin.

In both the batch and the recursive approach, however, it is not obvious how long N must be for the receding horizon controller to inherit these desirable convergence characteristics. Indeed, for computational simplicity we would like to keep N small. We will argue next that the proposed control scheme guarantees convergence just like the infinite horizon variety if we impose a specific terminal constraint, for example, if we require the terminal region to be the origin $\mathcal{X}_f = 0$.

From the principle of optimality we know that

$$
J^*_{0\to N}(x_0) = \min_{u_0} \ q(x_0,u_0) + J^*_{1\to N}(x_1). \tag{7.26}
$$

Assume that we are at $x(0)$ at time 0 and implement the optimal u_0^* that takes us to the next state $x_1 = g(x(0), u_0^*)$. At this state at time 1 we postulate to use over the next N steps the sequence of optimal moves determined at the previous step followed by zero: $u_1^*, \ldots, u_{N-1}^*, 0$. This sequence is not optimal but the associated cost over the shifted horizon from 1 to $N+1$ can be easily determined. It consists of three parts: 1) the optimal cost $J_{0\to N}^*(x_0)$ from time 0 to N computed at time 0, minus 2) the stage cost $q(x_0, u_0)$ at time 0 plus 3) the cost at time $N+1$. But this last cost is zero because we imposed the terminal constraint $x_N = 0$ and assumed $u_N = 0$. Thus the cost over the shifted horizon for the assumed sequence of control moves is

$$J_{0\to N}^*(x_0) - q(x_0, u_0).$$

Because this postulated sequence of inputs is not optimal at time 1

$$J_{1\to N+1}^*(x_1) \leq J_{0\to N}^*(x_0) - q(x_0, u_0).$$

Because the system and the objective are time invariant $J_{1\to N+1}^*(x_1) = J_{0\to N}^*(x_1)$ so that

$$J_{0\to N}^*(x_1) \leq J_{0\to N}^*(x_0) - q(x_0, u_0).$$

As $q \succ 0$ for all $(x, u) \neq (0, 0)$, the sequence of optimal costs $J_{0\to N}^*(x_0), J_{0\to N}^*(x_1), \ldots$ is strictly decreasing for all $(x, u) \neq (0, 0)$. Because the cost $J_{0\to N}^* \geq 0$ the sequence $J_{0\to N}^*(x_0), J_{0\to N}^*(x_1), \ldots$ (and thus the sequence x_0, x_1,...) is converging. Thus we have established the following important theorem.

Theorem 7.1. *At time step j consider the cost function*

$$J_{j\to j+N}(x_j, u_j, u_{j+1}, \ldots, u_{j+N-1}) \triangleq \sum_{k=j}^{j+N} q(x_k, u_k),\ \ q \succ 0 \tag{7.27}$$

and the CFTOC problem

$$\begin{array}{lll} J_{j\to j+N}^*(x_j) \triangleq & \min_{u_j, u_{j+1}, \ldots, u_{j+N-1}} & J_{j\to j+N}(x_j, u_j, u_{j+1}, \ldots, u_{j+N-1}) \\ & \text{subj. to} & x_{k+1} = g(x_k, u_k) \\ & & h(x_k, u_k) \leq 0,\ k = j, \ldots, j+N-1 \\ & & x_N = 0 \end{array} \tag{7.28}$$

Assume that only the optimal u_j^* is implemented. At the next time step $j+1$ the CFTOC problem is solved again starting from the resulting state $x_{j+1} = g(x_j, u_j^*)$ (Receding Horizon Control). Assume that the CFTOC problem has a solution for every one of the sequence of states $x_j, x_{j+1}, \ldots$ resulting from the control policy. Then the system will converge to the origin as $j \to \infty$.

Thus we have established that a receding horizon controller with terminal constraint $x_N = 0$ has the same desirable convergence characteristics as the infinite horizon controller. At first sight the theorem appears very general and powerful. It is based on the implicit assumption, however, that at every time step the CFTOC problem has a solution. Infeasibility would occur, for example, if the underlying system is not stabilizable. It could also happen that the constraints on the inputs which restrict the control action prevent the system from reaching the terminal state in N steps. In Chapter 11 we will present special formulations of problem (7.28) such that feasibility at the initial time guarantees feasibility for all future times. Furthermore in addition to asymptotic convergence to the origin we will establish stability for the closed-loop system with the receding horizon controller.

Remark 7.1. For the sake of simplicity in the rest of the book we will use the following shorter notation

$$\begin{array}{l} J_j^*(x_j) \triangleq J_{j\to N}^*(x_j),\ j=0,\dots,N \\ J_\infty^*(x_0) \triangleq J_{0\to\infty}^*(x_0) \\ \mathcal{X}_j \triangleq \mathcal{X}_{j\to N},\ j=0,\dots,N \\ \mathcal{X}_\infty \triangleq \mathcal{X}_{0\to\infty} \\ U_0 \triangleq U_{0\to N} \end{array} \tag{7.29}$$

and use the original notation only if needed.

7.5 Lyapunov Stability

While asymptotic convergence $\lim_{k\to\infty} x_k = 0$ is a desirable property, it is generally not sufficient in practice. We would also like a system to stay in a small neighborhood of the origin when it is disturbed by a little. Formally this is expressed as Lyapunov stability.

7.5.1 General Stability Conditions

Consider the autonomous system

$$x_{k+1} = f(x_k) \tag{7.30}$$

with $f(0) = 0$.

Definition 7.1 (Lyapunov Stability). The equilibrium point $x = 0$ of system (7.30) is

- *stable* (in the sense of Lyapunov) if, for each $\varepsilon > 0$, there is $\delta > 0$ such that

$$\|x_0\| < \delta \Rightarrow \|x_k\| < \varepsilon,\ \forall k \geq 0 \tag{7.31}$$

- *unstable* if not stable
- *asymptotically stable* in $\Omega \subseteq \mathbb{R}^n$ if it is stable and

$$\lim_{k\to\infty} x_k = 0, \ \forall x_0 \in \Omega \tag{7.32}$$

- *globally asymptotically stable* if it is asymptotically stable and $\Omega = \mathbb{R}^n$
- *exponentially stable* if it is stable and there exist constants $\alpha > 0$ and $\gamma \in (0,1)$ such that

$$\|x_0\| < \delta \Rightarrow \|x_k\| \leq \alpha \|x_0\| \gamma^k, \ \forall k \geq 0 \tag{7.33}$$

The ε-δ requirement for stability (7.31) takes a challenge-answer form. To demonstrate that the origin is stable, for any value of ε that a challenger may chose (however small), we must produce a value of δ such that a trajectory starting in a δ neighborhood of the origin will never leave the ε neighborhood of the origin.

Remark 7.2. If in place of system (7.30), we consider the time-varying system $x_{k+1} = f(x_k, k)$, then δ in Definition 7.1 is a function of ε and k, i.e., $\delta = \delta(\varepsilon, k) > 0$. In this case, we introduce the concept of "uniform stability". The equilibrium point $x = 0$ is *uniformly stable* if, for each $\varepsilon > 0$, there is $\delta = \delta(\varepsilon) > 0$ (independent from k) such that

$$\|x_0\| < \delta \Rightarrow \|x_k\| < \varepsilon, \ \forall k \geq 0 \tag{7.34}$$

The following example shows that Lyapunov stability and convergence are, in general, different properties.

Example 7.1. Consider the following system with two states $x = [x(1), \ x(2)]' \in \mathbb{R}^2$:

$$\begin{aligned} x_{k+1}(1) &= x_k(1)\frac{e^{-x_k(1)/x_k(2)}}{\|x_k\|_\infty} \\ x_{k+1}(2) &= x_k(1) \end{aligned} \tag{7.35}$$

consider in a neighborhood of the origin $I_a = \{x \in \mathbb{R}^2 \ : \ x(2) = 0, \ x(1) \in (0, a), \ a > 0\}$. No matter how small we choose a, for all $x_0 \in I_a - 0$ the first component of the state at time 1, $x_1(1)$, will always be equal to the number e. In fact $x_1(2) = x_0(1)$ and thus $x_1(1) = x_0(1)\frac{e^{-x_0(1)/x_0(1)}}{x_0(1)} = e$. However it can be proven that the system converges to the origin for all $x_0 \in I_a$.

Usually to show Lyapunov stability of the origin for a particular system one constructs a so called *Lyapunov function*, i.e., a function satisfying the conditions of the following theorem.

Theorem 7.2. *Consider the equilibrium point $x = 0$ of system (7.30). Let $\Omega \subset \mathbb{R}^n$ be a closed and bounded set containing the origin. Let $V : \mathbb{R}^n \to \mathbb{R}$ be a function, continuous at the origin, such that*

$$V(0) = 0 \text{ and } V(x) > 0, \ \forall x \in \Omega \setminus \{0\} \tag{7.36a}$$

$$V(x_{k+1}) - V(x_k) < 0 \ \forall x_k \in \Omega \setminus \{0\} \tag{7.36b}$$

then $x = 0$ is asymptotically stable in Ω.

Definition 7.2. A function $V(x)$ satisfying conditions (7.36a)-(7.36b) is called a *Lyapunov Function.*

The main idea of Theorem 7.2 can be explained as follows. We aim at finding a scalar function $V(x)$ that captures qualitative characteristics of the system response, and, in particular, its stability. We can think of V as an energy function that is zero at the origin and positive elsewhere (condition (7.36a)). Condition (7.36b) of Theorem 7.2 requires that for any state $x_k \in \Omega$, $x_k \neq 0$ the energy decreases as the system evolves to x_{k+1}.

Theorem 7.2 states that if we find an energy function which satisfies the two conditions (7.36a)-(7.36b), then the system states starting from any initial state $x_0 \in \Omega$ will eventually settle to the origin.

Note that Theorem 7.2 is only sufficient. If condition (7.36b) is not satisfied for a particular choice of V nothing can be said about stability of the origin. Condition (7.36b) of Theorem 7.2 can be relaxed as follows:

$$V(x_{k+1}) - V(x_k) \leq 0, \ \forall x_k \neq 0 \tag{7.37}$$

Condition (7.37) along with condition (7.36a) are sufficient to guarantee stability of the origin as long as the set $\{x_k : V(f(x_k)) - V(x_k) = 0\}$ contains no trajectory of the system $x_{k+1} = f(x_k)$ except for $x_k = 0$ for all $k \geq 0$. This relaxation of Theorem 7.2 is the so called the Barbashin-Krasovski-LaSalle principle. It basically means that $V(x_k)$ may stay constant and non zero at one or more time instants as long as it does not do so at an equilibrium point or periodic orbit of the system.

A similar result as Theorem 7.2 can be derived for *global* asymptotic stability, i.e., $\Omega = \mathbb{R}^n$.

Theorem 7.3. *Consider the equilibrium point $x = 0$ of system (7.30). Let $V : \mathbb{R}^n \to \mathbb{R}$ be a function, continuous at the origin, such that*

$$\|x\| \to \infty \Rightarrow V(x) \to \infty \tag{7.38a}$$

$$V(0) = 0 \ \textit{and} \ V(x) > 0, \ \forall x \neq 0 \tag{7.38b}$$

$$V(x_{k+1}) - V(x_k) < 0 \ \forall x_k \neq 0 \tag{7.38c}$$

then $x = 0$ is globally asymptotically stable.

Definition 7.3. A function $V(x)$ satisfying condition (7.38a) is said to be *radially unbounded.*

Definition 7.4. A radially unbounded Lyapunov function is called a *Global Lyapunov Function.*

Note that it was not enough just to restate Theorem 7.2 with $\Omega = \mathbb{R}^n$ but we also have to require $V(x)$ to be radially unbounded to guarantee global asymptotic stability. To motivate this condition consider the candidate Lyapunov function for a system in $\mathbb{R}^2$[160]

$$V(x) = \frac{x(1)^2}{1 + x(1)^2} + x(2)^2 \tag{7.39}$$

where $x(1)$ and $x(2)$ denote the first and second components of the state vector x, respectively. $V(x)$ in (7.39) is not radially unbounded as for $x(2) = 0$

$$\lim_{x_k(1)\to\infty} V(x) = 1$$

For this Lyapunov function even if condition (7.38c) is satisfied, the state x may escape to infinity. Condition (7.38c) of Theorem 7.3 guarantees that the level sets Ω_c of $V(x)$ ($\Omega_c = \{x \in \mathbb{R}^n \ : \ V(x) \le c\}$) are closed.

The construction of suitable Lyapunov functions is a challenge except for linear systems. First of all one can quite easily show that for linear systems Lyapunov stability agrees with the notion of stability based on eigenvalue location.

Theorem 7.4. *A linear system $x_{k+1} = Ax_k$ is globally asymptotically stable in the sense of Lyapunov if and only if all its eigenvalues are inside the unit circle.*

We also note that stability is always "global" for linear systems.

7.5.2 Quadratic Lyapunov Functions for Linear Systems

A simple effective Lyapunov function for linear systems is

$$V(x) = x'Px, \ \ P \succ 0 \tag{7.40}$$

which satisfies conditions (7.38a)-(7.36a) of Theorem 7.2. In order to test condition (7.36b) we compute

$$V(x_{k+1}) - V(x_k) = x'_{k+1}Px_{k+1} - x'_kPx_k = x'_kA'PAx_k - x'_kPx_k = x'_k(A'PA - P)x_k \tag{7.41}$$

Therefore condition (7.36b) is satisfied if $P \succ 0$ can be found such that

$$A'PA - P = -Q, \ \ Q \succ 0 \tag{7.42}$$

Equation (7.42) is referred to as discrete-time Lyapunov equation. The following Theorem shows that P satisfying (7.42) exists if and only if the linear system is asymptotically stable.

Theorem 7.5. *Consider the linear system $x_{k+1} = Ax_k$. Equation (7.42) has a unique solution $P \succ 0$ for any $Q \succ 0$ if and only if A has all eigenvalues inside the unit circle.*

Thus, a quadratic form $x'Px$ is always a suitable Lyapunov function for linear systems and an appropriate P can be found by solving (7.42) for a chosen $Q \succ 0$ if the system's eigenvalues lie inside the unit circle. For nonlinear systems, determining a suitable form for $V(x)$ is generally difficult. The conditions of Theorem 7.5 can be relaxed as follows.

Theorem 7.6. *Consider the linear system $x_{k+1} = Ax_k$. Equation (7.42) has a unique solution $P \succ 0$ for any $Q = C'C \succeq 0$ if and only if A has all eigenvalues inside the unit circle and (C, A) is observable.*

It may be surprising that in order to prove stability in Theorem 7.6 we do not require that the Lyapunov function decreases at every time step, i.e. that Q is allowed to be positive *semi*definite. To understand this, let us assume that for a particular system state $\bar{x}$, V does not decrease, i.e. $\bar{x}'Q\bar{x} = (C\bar{x})'(C\bar{x}) = 0$. Then at the next time steps we have the rate of decrease $(CA\bar{x})'(CA\bar{x}), (CA^2\bar{x})'(CA^2\bar{x}), \ldots$. If the system (C, A) is observable then for all $\bar{x} \neq 0$

$$\bar{x}' \left[C \ (CA)' \ (CA^2)' \cdots (CA^{n-1})' \right] \neq 0 \tag{7.43}$$

which implies that after at most $(n-1)$ steps the rate of decrease will become nonzero. This is a special case of the the Barbashin-Krasovski-LaSalle principle.

Note that from (7.42) it follows that for stable systems and for a chosen $Q \succ 0$ one can always find $\tilde{P} \succ 0$ solving

$$A'\tilde{P}A - \tilde{P} + Q \preceq 0 \tag{7.44}$$

This *Lyapunov inequality* shows that for a stable system we can always find a $\tilde{P}$ such that $V(x) = x'\tilde{P}x$ decreases at a desires "rate" indicated by Q. We will need this result later to prove stability of receding horizon control schemes.

7.5.3 $1/\infty$ Norm Lyapunov Functions for Linear Systems

For $p = \{1, \infty\}$ the function

$$V(x) = \|Px\|_p$$

with $P \in \mathbb{R}^{l \times n}$ of full column rank satisfies the requirements (7.38a), (7.38b) of a Lyapunov function. It can be shown that a matrix P can be found such that condition (7.38c) is satisfied for the system $x_{k+1} = Ax_k$ if and only if the eigenvalues of A are inside the unit circle. The number of rows l necessary in P depends on the system. The techniques to construct P are based on the following theorem [161, 213].

Theorem 7.7. *Let $P \in \mathbb{R}^{l\times n}$ with $\operatorname{rank}(P) = n$ and $p \in \{1, \infty\}$. The function*

$$V(x) = \|Px\|_p \tag{7.45}$$

is a Lyapunov function for the discrete-time system

$$x_{k+1} = Ax_k, \qquad k \geq 0, \tag{7.46}$$

if and only if *there exists a matrix $H \in \mathbb{R}^{l\times l}$, such that*

$$PA = HP, \tag{7.47a}$$

$$\|H\|_p < 1. \tag{7.47b}$$

An effective method to find both H and P was proposed by Christophersen in [82].

To prove the stability of receding horizon control, later in this book, we will need to find a $\tilde{P}$ such that

$$-\|\tilde{P}x\|_\infty + \|\tilde{P}Ax\|_\infty + \|Qx\|_\infty \leq 0,\ \forall x \in \mathbb{R}^n. \tag{7.48}$$

Once we have constructed a P and H to fulfill the conditions of Theorem 7.7 we can easily find $\tilde{P}$ to satisfy (7.48) according to the following proposition:

Proposition 7.1. *Let P and H be matrices satisfying conditions (7.47), with P full column rank. Let $\sigma \triangleq 1 - \|H\|_\infty$, $\rho \triangleq \|QP^{\#}\|_\infty$, where $P^{\#} \triangleq (P^TP)^{-1}P^T$ is the left pseudoinverse of P. Then, the square matrix*

$$\tilde{P} = \frac{\rho}{\sigma}P \tag{7.49}$$

satisfies condition (7.48).

Proof: Since $\tilde{P}$ satisfies $\tilde{P}A = H\tilde{P}$, we obtain $-\|\tilde{P}x\|_\infty + \|\tilde{P}Ax\|_\infty + \|Qx\|_\infty = -\|\tilde{P}x\|_\infty + \|H\tilde{P}x\|_\infty + \|Qx\|_\infty \leq (\|H\|_\infty - 1)\|\tilde{P}x\|_\infty + \|Qx\|_\infty \leq (\|H\|_\infty - 1)\|\tilde{P}x\|_\infty + \|QP^{\#}\|_\infty\|Px\|_\infty = 0$. Therefore, (7.48) is satisfied. $\square$

Note that the inequality (7.48) is equivalent to the Lyapunov inequality (7.44) when $p = 1$ or $p = \infty$.

Chapter 8
Linear Quadratic Optimal Control

We consider a special case of the problem stated in the last chapter, where the system is linear and time-invariant

$$x(t+1) = Ax(t) + Bu(t) \tag{8.1}$$

Again, $x(t) \in \mathbb{R}^n$ and $u(t) \in \mathbb{R}^m$ are the state and input vectors respectively.

We define the following quadratic cost function over a finite horizon of N steps

$$J_0(x_0, U_0) \triangleq x_N' P x_N + \sum_{k=0}^{N-1} x_k' Q x_k + u_k' R u_k \tag{8.2}$$

where x_k denotes the state vector at time k obtained by starting from the state $x_0 = x(0)$ and applying to the system model

$$x_{k+1} = Ax_k + Bu_k \tag{8.3}$$

the input sequence $U_0 \triangleq [u_0', \dots, u_{N-1}']'$. Consider the finite time optimal control problem

$$\begin{array}{lll} J_0^*(x(0)) = \min_{U_0} & J_0(x(0), U_0) & \\ & \text{subj. to } x_{k+1} = Ax_k + Bu_k,\ k = 0, 1, \dots, N-1 & \\ & x_0 = x(0). & \end{array} \tag{8.4}$$

In (8.4) $U_0 = [u_0', \dots, u_{N-1}']' \in \mathbb{R}^s$, $s \triangleq mN$ is the decision vector containing all future inputs. We will assume that the state penalty is positive semi-definite $Q = Q' \succeq 0$, $P = P' \succeq 0$ and the input penalty is positive definite $R = R' \succ 0$.

As introduced in the previous chapter we will present two alternate approaches to solve problem (8.4), the batch approach and the recursive approach using dynamic programming.

8.1 Solution via Batch Approach

First we write the equality constraints in (8.4) explicitly to express all future states $x_1, x_2, \ldots$ as a function of the future inputs $u_0, u_1, \ldots$ and then we eliminate all intermediate states by successive substitution to obtain

$$\underbrace{\begin{bmatrix} x(0) \\ x_1 \\ \vdots \\ \vdots \\ x_N \end{bmatrix}}_{\mathcal{X}} = \underbrace{\begin{bmatrix} I \\ A \\ \vdots \\ \vdots \\ A^N \end{bmatrix}}_{\mathcal{S}^x} x(0) + \underbrace{\begin{bmatrix} 0 & \ldots\ldots & 0 \\ B & 0 \ \ldots & 0 \\ AB & \ddots \ \ddots & \vdots \\ \vdots & \ddots \ \ddots & \vdots \\ A^{N-1}B & \ldots\ldots & B \end{bmatrix}}_{\mathcal{S}^u} \begin{bmatrix} u_0 \\ \vdots \\ \vdots \\ u_{N-1} \end{bmatrix}. \tag{8.5}$$

Here all future states are explicit functions of the present state $x(0)$ and the future inputs $u_0, u_1, u_2, \ldots$ only. By defining the appropriate quantities we can rewrite this expression compactly as

$$\mathcal{X} = \mathcal{S}^x x(0) + \mathcal{S}^u U_0 \tag{8.6}$$

Using the same notation the objective function can be rewritten as

$$J(x(0), U_0) = \mathcal{X}'\bar{Q}\mathcal{X} + U_0{}'\bar{R}U_0 \tag{8.7}$$

where $\bar{Q} = \text{blockdiag}\{Q, \cdots, Q, P\}$, $\bar{Q} \succeq 0$, and $\bar{R} = \text{blockdiag}\{R, \cdots, R\}$, $\bar{R} \succ 0$. Substituting (8.6) into the objective function (8.7) yields

$$\begin{aligned} J_0(x(0), U_0) &= (\mathcal{S}^x x(0) + \mathcal{S}^u U_0)' \,\bar{Q}\, (\mathcal{S}^x x(0) + \mathcal{S}^u U_0) + U_0{}'\bar{R}U_0 \\ &= U_0{}' \underbrace{(\mathcal{S}^{u\prime}\bar{Q}\mathcal{S}^u + \bar{R})}_{H} U_0 + 2x'(0) \underbrace{(\mathcal{S}^{x\prime}\bar{Q}\mathcal{S}^u)}_{F} U_0 + x'(0) \underbrace{(\mathcal{S}^{x\prime}\bar{Q}\mathcal{S}^x)}_{Y} x(0) \\ &= U_0{}'HU_0 + 2x'(0)FU_0 + x'(0)Yx(0) \end{aligned} \tag{8.8}$$

Because $\bar{R} \succ 0$, also $H \succ 0$. Thus $J_0(x(0), U_0)$ is a positive definite quadratic function of U_0. Therefore, its minimum can be found by computing its gradient and setting it to zero. This yields the optimal vector of future inputs

$$\begin{aligned} U_0^*(x(0)) &= -H^{-1}F'x(0) \\ &= -\left(\mathcal{S}^{u\prime}\bar{Q}\mathcal{S}^u + \bar{R}\right)^{-1} \mathcal{S}^{u\prime}\bar{Q}\mathcal{S}^x x(0) \end{aligned} \tag{8.9}$$

With this choice of U_0 the optimal cost is

$$\begin{aligned} J_0^*(x(0)) &= -x(0)'FH^{-1}F'x(0) + x(0)'Yx(0) \\ &= x(0)' \left[\mathcal{S}^{x\prime}\bar{Q}\mathcal{S}^x - \mathcal{S}^{x\prime}\bar{Q}\mathcal{S}^u \left(\mathcal{S}^{u\prime}\bar{Q}\mathcal{S}^u + \bar{R}\right)^{-1} \mathcal{S}^{u\prime}\bar{Q}\mathcal{S}^x\right] x(0) \end{aligned} \tag{8.10}$$

Note that the optimal vector of future inputs $U_0^*(x(0))$ is a linear function (8.9) of the initial state $x(0)$ and the optimal cost $J_0^*(x(0))$ is a quadratic function (8.10) of the initial state $x(0)$.

8.2 Solution via Recursive Approach

Alternatively, we can use dynamic programming to solve the same problem in a recursive manner. We define the optimal cost $J_j^*(x_j)$ for the $N-j$ step problem starting from state x_j by

$$J_j^*(x_j) \overset{\Delta}{=} \min_{u_j,\cdots,u_{N-1}} x_N' P x_N + \sum_{k=j}^{N-1} x_k' Q x_k + u_k' R u_k$$

According to the principle of optimality the optimal one step cost-to-go can be obtained from

$$J_{N-1}^*(x_{N-1}) = \min_{u_{N-1}} x_N' P_N x_N + x_{N-1}' Q x_{N-1} + u_{N-1}' R u_{N-1} \tag{8.11}$$

$$\begin{aligned} x_N &= A x_{N-1} + B u_{N-1} \\ P_N &= P \end{aligned} \tag{8.12}$$

Here we have introduced the notation P_j to express the optimal cost-to-go $x_j{}' P_j x_j$ from time j to the end of the horizon N. Specifically if $j = N$ this is simply P. Substituting (8.12) into the objective function (8.11),

$$\begin{aligned} J_{N-1}^*(x_{N-1}) = \min\nolimits_{u_{N-1}} \{ x_{N-1}'(A' P_N A + Q) x_{N-1} \\ +2 x_{N-1}' A' P_N B u_{N-1} \\ +u_{N-1}'(B' P_N B + R) u_{N-1} \} . \end{aligned} \tag{8.13}$$

We note that the cost-to-go $J_{N-1}(x_{N-1})$ is a positive definite quadratic function of the decision variable u_{N-1}. We find the optimum by setting the gradient to zero and obtain the optimal input

$$u_{N-1}^* = \underbrace{-(B' P_N B + R)^{-1} B' P_N A}_{F_{N-1}} x_{N-1} \tag{8.14}$$

and the one-step optimal cost-to-go

$$J_{N-1}^*(x_{N-1}) = x_{N-1}' P_{N-1} x_{N-1}, \tag{8.15}$$

where we have defined

$$P_{N-1} = A' P_N A + Q - A' P_N B (B' P_N B + R)^{-1} B' P_N A \tag{8.16}$$

At the next stage, consider the two-step problem from time $N-2$ forward:

$$J^*_{N-2}(x_{N-2}) = \min_{u_{N-2}} x'_{N-1}P_{N-1}x_{N-1} + x'_{N-2}Qx_{N-2} + u'_{N-2}Ru_{N-2} \tag{8.17}$$

$$x_{N-1} = Ax_{N-2} + Bu_{N-2} \tag{8.18}$$

We recognize that (8.17), (8.18) has the same form as (8.11), (8.12). Therefore we can state the optimal solution directly.

$$u^*_{N-2} = \underbrace{-(B'P_{N-1}B+R)^{-1}B'P_{N-1}A}_{F_{N-2}}\, x_{N-2} \tag{8.19}$$

The optimal two-step cost-to-go is

$$J^*_{N-2}(x_{N-2}) = x'_{N-2}P_{N-2}x_{N-2}, \tag{8.20}$$

where we defined

$$P_{N-2} = A'P_{N-1}A + Q - A'P_{N-1}B(B'P_{N-1}B+R)^{-1}B'P_{N-1}A \tag{8.21}$$

Continuing in this manner, at some arbitrary time k the optimal control action is

$$\begin{aligned} u^*(k) &= -(B'P_{k+1}B+R)^{-1}B'P_{k+1}Ax(k), \\ &= F_k x(k), \qquad \text{for } k = 0,\dots,N-1, \end{aligned} \tag{8.22}$$

where

$$P_k = A'P_{k+1}A + Q - A'P_{k+1}B(B'P_{k+1}B+R)^{-1}B'P_{k+1}A \tag{8.23}$$

and the optimal cost-to-go starting from the measured state $x(k)$ is

$$J^*_k(x(k)) = x'(k)P_k x(k) \tag{8.24}$$

Equation (8.23) (called *Discrete Time Riccati Equation* or *Riccati Difference Equation* - RDE) is initialized with $P_N = P$ and is solved backwards, i.e., starting with P_N and solving for P_{N-1}, etc. Note from (8.22) that the optimal control action $u^*(k)$ is obtained in the form of a feedback law as a linear function of the measured state $x(k)$ at time k. The optimal cost-to-go (8.24) is found to be a quadratic function of the state at time k.

Remark 8.1. According to Section 7.4.2, the receding horizon control policy consists in solving problem (8.4) at each time step t with $x_0 = x(t)$. Consider the state-feedback solution $u^*(k)$ in (8.22) to problem (8.4). Then, the RHC policy is:

$$u^*(t) = F_0 x(t), \quad t \geq 0 \tag{8.25}$$

8.3 Comparison Of The Two Approaches

We will compare the batch and the recursive dynamic programming approach in terms of the results and the methods used to obtain the results.

Most importantly we observe that the results obtained by the two methods are fundamentally different. The batch approach yields a formula for the *sequence of inputs* as a function of the initial state.

$$U_0^* = -\left(\mathcal{S}^{u\prime}\bar{Q}\mathcal{S}^u + \bar{R}\right)^{-1}\mathcal{S}^{u\prime}\bar{Q}\mathcal{S}^x x(0) \tag{8.26}$$

The recursive dynamic programming approach yields a feedback policy, i.e., a *sequence of feedback laws* expressing at each time step the control action as a function of the state at that time.

$$u^*(k) = F_k x(k), \text{ for } k = 0, \ldots, N-1 \tag{8.27}$$

As this expression implies, we determine $u(k)$ at each time k as a function of the current state $x(k)$ rather than use a $u(k)$ precomputed at $k = 0$ as in the batch method. If the state evolves exactly according to the linear model (8.3) then the sequence of control actions $u(k)$ obtained from the two approaches is identical. In practice, the result of applying the sequence (8.26) in an open-loop fashion may be rather different from applying the time-varying feedback law (8.27) because the model (8.1) for predicting the system states may be inaccurate and the system may be subject to disturbances not included in the model. We expect the application of the feedback law to be more robust because at each time step the *observed* state $x(k)$ is used to determine the control action rather than the state x_k *predicted* at time $t = 0$.

We note that we can get the same feedback effect with the batch approach if we recalculate the optimal open-loop sequence at each time step j with the current measurement as initial condition. In this case we need to solve the following optimization problem

$$\begin{array}{ll} J_j^*(x(j)) = \min_{u_j,\cdots,u_{N-1}} & x_N' P x_N + \sum_{k=j}^{N-1} x_k' Q x_k + u_k' R u_k \\ \quad\quad\quad\quad\text{subj. to} & x_j = x(j) \end{array} \tag{8.28}$$

where we note that the horizon length is shrinking at each time step.

As seen from (8.26) the solution to (8.28) relates the sequence of inputs $u_j^*, u_{j+1}^*, \ldots$ to the state $x(j)$ through a linear expression. The first part of this expression yields again the optimal feedback law (8.27) at time j, $u^*(j) = F_j x(j)$.

Here the dynamic programming approach is clearly a more efficient way to generate the feedback policy because it only uses a simple matrix recursion (8.23). Repeated application of the batch approach, on the other hand, requires repeatedly the inversion of a potentially large matrix in (8.26). For such inversion, however, one can take advantage of the fact that only a small part of the matrix H changes at every time step.

What makes dynamic programming so effective here is that in this special case, where the system is linear and the objective is quadratic, the optimal cost-to-go, the value function $J_j^*(x(j))$ has a very simple form: it is quadratic. If we make the problem only slightly more complicated, e.g., if we add constraints on the inputs or states, the value function can still be constructed, but it is much more complex. In general, the value function can only be approximated as discussed in the previous chapter. Then a repeated application of the batch policy, where we resolve the optimization problem at each time step is an attractive alternative.

8.4 Infinite Horizon Problem

For continuous processes operating over a long time period it would be interesting to solve the following infinite horizon problem.

$$J_\infty^*(x(0)) = \min_{u_0,u_1,\dots} \sum_{k=0}^{\infty} x_k' Q x_k + u_k' R u_k \tag{8.29}$$

Since the prediction must be carried out to infinity, application of the batch method becomes impossible. On the other hand, derivation of the optimal feedback law via dynamic programming remains viable. We can initialize the RDE (8.23)

$$P_k = A' P_{k+1} A + Q - A' P_{k+1} B (B' P_{k+1} B + R)^{-1} B' P_{k+1} A \tag{8.30}$$

with the terminal cost matrix $P_0 = Q$ and solve it backwards for $k \to -\infty$. Let us assume for the moment that the iterations converge to a solution P_∞. Such P_∞ would then satisfy the *Algebraic Riccati Equation* (ARE)

$$P_\infty = A' P_\infty A + Q - A' P_\infty B (B' P_\infty B + R)^{-1} B' P_\infty A. \tag{8.31}$$

Then the optimal feedback control law is

$$u^*(k) = \underbrace{-(B' P_\infty B + R)^{-1} B' P_\infty A}_{F_\infty} x(k), \qquad k = 0, \cdots, \infty \tag{8.32}$$

and the optimal infinite horizon cost is

$$J_\infty^*(x(0)) = x(0)' P_\infty x(0). \tag{8.33}$$

Controller (8.32) is referred to as the asymptotic form of the *Linear Quadratic Regulator* (LQR) or the ∞-horizon LQR.

Convergence of the RDE has been studied extensively. A nice summary of the various results can be found in Appendix E of the book by Goodwin and Sin [120] and on page 53 of the book by Anderson and Moore [8]. Intuitively we expect that the system (8.3) must be controllable so that all states can be affected by the control and that the cost function should capture the behavior of all the states, e.g. that $Q \succ 0$. These conditions are indeed sufficient for the RDE to converge and to yield a stabilizing feedback control law. Less restrictive conditions are presented in the following theorem.

Theorem 8.1. *[173][Theorem 2.4-2] If (A, B) is a stabilizable pair and $(Q^{1/2}, A)$ is an observable pair, the controller Riccati difference equation (8.30) with $P_0 \succeq 0$ converges to a unique positive definite solution P_∞ of the ARE (8.31) and all the eigenvalues of $(A+BF_\infty)$ lie inside the unit disk.*

The first condition is clearly necessary for J^*_∞ (and P_∞) to be finite. To understand the second condition, we write the state dependent term in the objective function as $x'Qx = (x'Q^{1/2})(Q^{1/2}x)$. Thus not the state but the "output" $(Q^{1/2}x)$ is penalized in the objective. Therefore the second condition $((Q^{1/2}, A)$ observable) requires that this output captures all system modes. In this manner convergence of the output $(Q^{1/2}x)$ implies convergence of the state to zero. This is trivial if Q is nonsingular. For the singular case this is proven in the next section.

Note also that the assumption on the observability of $(Q^{1/2}, A)$ in Theorem 8.1 can be relaxed. For convergence, it is enough to require that the output $(Q^{1/2}x)$ captures all system *unstable modes* and thus $(Q^{1/2}, A)$ to be detectable.

8.5 Stability of the Infinite Horizon LQR

Consider the linear system (8.1) and the ∞-horizon LQR solution (8.31)-(8.32). We prove that the closed-loop system

$$x(t+1) = (A + BF_\infty)x(t) \tag{8.34}$$

is asymptotically stable for any F_∞ by showing that the ∞-horizon cost

$$J^*_\infty(x) = x'P_\infty x \tag{8.35}$$

is a Lyapunov function (and thus proving the last part of Theorem 8.1).

Let $Q = C'C$ and $R = D'D$ with $\det(D) \neq 0$. The proof consists of two main steps: *(i)* prove that if (C, A) is an observable pair then $\left(\begin{bmatrix} C \\ DF_\infty \end{bmatrix}, [A - BF_\infty]\right)$ is also observable, *(ii)* prove that the ARE equation (8.31) can be rewritten as

$$P_\infty = (A - BF_\infty)' P_\infty (A - BF_\infty) + \begin{bmatrix} C \\ DF_\infty \end{bmatrix}' \begin{bmatrix} C \\ DF_\infty \end{bmatrix}. \tag{8.36}$$

Once *(i)* and *(ii)* are proven, the final result is immediate. In fact, equation (8.36) is a Lyapunov equation for $(A - BF_\infty)$. Since $P_\infty \succ 0$ and $\left(\begin{bmatrix} C \\ DF_\infty \end{bmatrix}, [A - BF_\infty]\right)$ is observable we can use Theorem 7.6 and claim the asymptotic stability of the closed-loop system $A - BF_\infty$ for any F_∞.
Proof of *(i)*. Since $\det(D) \neq 0$ then $B = MD$ for some M. We have

$$\text{rank} \begin{bmatrix} zI - A \\ C \\ DF_\infty \end{bmatrix} = \text{rank} \begin{bmatrix} I & 0 & M \\ 0 & I & 0 \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} zI - A \\ C \\ DF_\infty \end{bmatrix} = \text{rank} \begin{bmatrix} zI - (A - BF_\infty) \\ C \\ DF_\infty \end{bmatrix}. \tag{8.37}$$

Since (C, A) is observable, by the Hautus condition

$$\text{rank} \begin{bmatrix} zI - A \\ C \end{bmatrix} = n \quad \text{for every } z \tag{8.38}$$

and therefore by (8.37), $\left(\begin{bmatrix} C \\ DF_\infty \end{bmatrix}, [A - BF_\infty]\right)$ is observable for any F_∞.
Proof of *(ii)*. From (8.36)

$$\begin{aligned} P_\infty &= A'P_\infty A - F_\infty' B' P_\infty A - A' P_\infty B F_\infty + F_\infty' B' P_\infty B F_\infty + Q + F_\infty R F_\infty \\ &= A'P_\infty A - F_\infty' B' P_\infty A - A' P_\infty B F_\infty + F_\infty' (B' P_\infty B + R) F_\infty + Q \end{aligned} \tag{8.39}$$

From the definition of F_∞ in (8.32), and the previous equation we have

$$\begin{aligned} P_\infty = {} & A'P_\infty A - A'P_\infty B(B'P_\infty B + R)^{-1} B'P_\infty A - A'P_\infty B(B'P_\infty B + R)^{-1} B'P_\infty A \\ & + A'P_\infty B(B'P_\infty B + R)^{-1} B'P_\infty A + Q \end{aligned} \tag{8.40}$$

which is equal to

$$P_\infty = A'P_\infty A - A'P_\infty B(B'P_\infty B + R)^{-1} B'P_\infty A + Q \tag{8.41}$$

which is the ARE equation (8.31).

Chapter 9
1/∞ Norm Optimal Control

We consider a special case of the problem stated in Chapter 7, where the system is linear and time-invariant

$$x(t+1) = Ax(t) + Bu(t) \tag{9.1}$$

Again, $x(t) \in \mathbb{R}^n$ and $u(t) \in \mathbb{R}^m$ are the state and input vector respectively.

We define the following piecewise linear cost function over a finite horizon of N steps

$$J_0(x_0, U_0) \triangleq \|Px_N\|_p + \sum_{k=0}^{N-1} \|Qx_k\|_p + \|Ru_k\|_p \tag{9.2}$$

with $p = 1$ or $p = \infty$ and where x_k denotes the state vector at time k obtained by starting from the state $x_0 = x(0)$ and applying to the system model

$$x_{k+1} = Ax_k + Bu_k \tag{9.3}$$

the input sequence $U_0 \triangleq [u_0', \ldots, u_{N-1}']'$. The weighting matrices in (9.2) could have an arbitrary number of rows. For simplicity of notation we will assume $Q \in \mathbb{R}^{n\times n}$, $R \in \mathbb{R}^{m\times m}$ and $P \in \mathbb{R}^{r\times n}$. Consider the finite time optimal control problem

$$\begin{array}{rl} J_0^*(x(0)) = \min_{U_0} & J_0(x(0), U_0) \\ \text{subj. to} & x_{k+1} = Ax_k + Bu_k,\ k = 0, 1, \ldots, N-1 \\ & x_0 = x(0). \end{array} \tag{9.4}$$

In (9.4) $U_0 = [u_0', \ldots, u_{N-1}']' \in \mathbb{R}^s$, $s \triangleq mN$ is the decision vector containing all future inputs.

We will present two alternate approaches to solve problem (9.2)-(9.4), the batch approach and the recursive approach using dynamic programming. Unlike in the 2-norm case presented in the previous chapter, there does not exist a simple closed-form solution of problem (9.2)-(9.4). In this chapter we will show how to use multiparametric linear programming to compute the solution to problem (9.2)-(9.4). We will concentrate on the use of the ∞-norm, the results can be extended easily to cost functions based on the 1-norm or mixed 1/∞ norms.

9.1 Solution via Batch Approach

First we write the equality constraints in (9.4) explicitly to express all future states $x_1, x_2, \dots$ as a function of the future inputs $u_1, u_2, \dots$ and then we eliminate all intermediate states by using

$$x_k = A^k x_0 + \textstyle\sum_{j=0}^{k-1} A^j B u_{k-1-j} \tag{9.5}$$

so that all future states are explicit functions of the present state $x(0)$ and the future inputs $u_0, u_1, u_2, \dots$ only.

The optimal control problem (9.4) with $p = \infty$ can be rewritten as a linear program by using the following standard approach (see e.g. [72]). The sum of components of any vector $\{\varepsilon_0^x, \dots, \varepsilon_N^x, \varepsilon_0^u, \dots, \varepsilon_{N-1}^u\}$ that satisfies

$$\begin{array}{l} -\mathbf{1}_n \varepsilon_k^x \le Q x_k, \; k = 0, 1, \dots, N-1 \\ -\mathbf{1}_n \varepsilon_k^x \le -Q x_k, \; k = 0, 1, \dots, N-1 \\ -\mathbf{1}_r \varepsilon_N^x \le P x_N, \\ -\mathbf{1}_r \varepsilon_N^x \le -P x_N, \\ -\mathbf{1}_m \varepsilon_k^u \le R u_k, \; k = 0, 1, \dots, N-1 \\ -\mathbf{1}_m \varepsilon_k^u \le -R u_k, \; k = 0, 1, \dots, N-1 \end{array} \tag{9.6}$$

forms an upper bound on $J_0(x(0), U_0)$, where $\mathbf{1}_k \triangleq [\underbrace{1 \; \dots \; 1}_{k}]'$, and the inequalities (9.6) hold componentwise. It is easy to prove that the vector $z_0 \triangleq \{\varepsilon_0^x, \dots, \varepsilon_N^x, \varepsilon_0^u, \dots, \varepsilon_{N-1}^u, u_0', \dots, u_{N-1}'\} \in \mathbb{R}^s$, $s \triangleq (m+1)N + N + 1$, that satisfies equations (9.6) and simultaneously minimizes $J(z_0) = \varepsilon_0^x + \dots + \varepsilon_N^x + \varepsilon_0^u + \dots + \varepsilon_{N-1}^u$ also solves the original problem (9.4), i.e., the same optimum $J_0^*(x(0))$ is achieved [267, 72]. Therefore, problem (9.4) can be reformulated as the following LP problem

$$\min_{z_0} \quad \varepsilon_0^x + \ldots + \varepsilon_N^x + \varepsilon_0^u + \ldots + \varepsilon_{N-1}^u \tag{9.7a}$$

$$\text{subj. to} \quad -\mathbf{1}_n \varepsilon_k^x \leq \pm Q \left[A^k x_0 + \sum_{j=0}^{k-1} A^j B u_{k-1-j} \right], \tag{9.7b}$$

$$-\mathbf{1}_r \varepsilon_N^x \leq \pm P \left[A^N x_0 + \sum_{j=0}^{N-1} A^j B u_{N-1-j} \right], \tag{9.7c}$$

$$-\mathbf{1}_m \varepsilon_k^u \leq \pm R u_k, \tag{9.7d}$$

$$k = 0, \ldots, N-1$$

$$x_0 = x(0) \tag{9.7e}$$

where constraints (9.7b)–(9.7d) are componentwise, and $\pm$ means that the constraint appears once with each sign, as in (9.6).

Remark 9.1. The cost function (9.2) with $p = \infty$ can be interpreted as a special case of a cost function with 1-norm over time and ∞-norm over space. For instance, the dual choice (∞-norm over time and 1-norm over space) leads to the following cost function

$$J_0(x(0), U_0) \triangleq \max_{k=0,\ldots,N} \{ \|Qx_k\|_1 + \|Ru_k\|_1 \}. \tag{9.8}$$

We remark that any combination of 1- and ∞-norms leads to a linear program. In general, ∞-norm over time could result in a poor closed-loop performance (only the largest state deviation and the largest input would be penalized over the prediction horizon), while 1-norm over space leads to an LP with a larger number of variables.

The results of this chapter hold for any combination of 1- and ∞-norms over time and space. Clearly the LP formulation will differ from the one in (9.7). For instance, the $1-$norm in space requires the introduction of nN slack variables for the terms $\|Qx_k\|_1$, $\varepsilon_{k,i} \geq \pm Q^i x_k \ k = 0, 2, \ldots, N-1, \ i = 1, 2, \ldots, n$, plus r slack variables for the terminal penalty $\|Px_N\|_1$, $\varepsilon_{N,i} \geq \pm P^i x_N \ i = 1, 2, \ldots, r$, plus mN slack variables for the input terms $\|Ru_k\|_1$, $\varepsilon_{k,i}^u \geq \pm R^i u_k \ k = 0, 1, \ldots, N-1, \ i = 1, 2, \ldots, m$. Here we have used the notation M^i to denote the i-th row of matrix M.

Problem (9.7) can be rewritten in the more compact form

$$\begin{array}{ll} \min_{z_0} & c_0' z_0 \\ \text{subj. to} & G_\varepsilon z_0 \leq W_\varepsilon + S_\varepsilon x(0) \end{array} \tag{9.9}$$

where $c_0 \in \mathbb{R}^s$ and $G_\varepsilon \in \mathbb{R}^{q \times s}$, $S_\varepsilon \in \mathbb{R}^{q \times n}$ and $W_\varepsilon \in \mathbb{R}^q$ are

$$
\begin{aligned}
c_0 &= [\overbrace{1\ \dots\ 1}^{N+1}\ \overbrace{1\ \dots\ 1}^{N}\ \overbrace{0\ \dots\ 0}^{mN}]' \\
G_\epsilon &= \left[\begin{array}{cccc|ccc|cccc}
\multicolumn{4}{c}{\overbrace{\hphantom{-\mathbf{1}_n\ 0\ \dots\ 0}}^{N+1}} & \multicolumn{3}{c}{\overbrace{\hphantom{0\ \dots\ 0}}^{N}} & \multicolumn{4}{c}{\overbrace{\hphantom{QA^{N-2}B\ QA^{N-3}B\ \dots\ 0}}^{mN}} \\
-\mathbf{1}_n & 0 & \dots & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\
-\mathbf{1}_n & 0 & \dots & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\
0 & -\mathbf{1}_n & \dots & 0 & 0 & \dots & 0 & QB & 0 & \dots & 0 \\
0 & -\mathbf{1}_n & \dots & 0 & 0 & \dots & 0 & -QB & 0 & \dots & 0 \\
\dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\
0 & \dots & -\mathbf{1}_n & 0 & 0 & \dots & 0 & QA^{N-2}B & QA^{N-3}B & \dots & 0 \\
0 & \dots & -\mathbf{1}_n & 0 & 0 & \dots & 0 & -QA^{N-2}B & -QA^{N-3}B & \dots & 0 \\
0 & \dots & 0 & -\mathbf{1}_r & 0 & \dots & 0 & PA^{N-1}B & PA^{N-2}B & \dots & PB \\
0 & \dots & 0 & -\mathbf{1}_r & 0 & \dots & 0 & -PA^{N-1}B & -PA^{N-2}B & \dots & -PB \\
0 & 0 & \dots & 0 & -\mathbf{1}_m & \dots & 0 & R & 0 & \dots & 0 \\
0 & 0 & \dots & 0 & -\mathbf{1}_m & \dots & 0 & -R & 0 & \dots & 0 \\
\dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\
0 & 0 & \dots & 0 & 0 & \dots & -\mathbf{1}_m & 0 & 0 & \dots & R \\
0 & 0 & \dots & 0 & 0 & \dots & -\mathbf{1}_m & 0 & 0 & \dots & -R
\end{array}\right] \\
W_\epsilon &= [\overbrace{0\ \dots\ 0}^{2nN+2r}\ \overbrace{0\ \dots\ 0}^{2mN}]' \\
S_\epsilon &= [\overbrace{-Q'\ Q'\ (-QA)'\ (QA)'\ (-QA^2)'\ \dots\ (QA^{N-1})'\ (-QA^{N-1})')'}^{2nN}\ \overbrace{(-PA^N)'\ (PA^N)'}^{2r}\ \overbrace{0'_m\ \dots\ 0'_m}^{2mN}]'.
\end{aligned}
\tag{9.10}
$$

Note that in (9.10) we include the zero vector W_ϵ to make the notation consistent with the one used in Section 6.2.

By treating $x(0)$ as a vector of parameters, the problem (9.9) becomes a *multiparametric linear program* (mp-LP) that can be solved as described in Section 6.2. Once the multiparametric problem (9.7) has been solved, the explicit solution $z_0^*(x(0))$ of (9.9) is available as a piecewise affine function of $x(0)$, and the optimal control law U_0^* is also available explicitly, as the optimal input U_0^* consists simply of the last part of $z_0^*(x(0))$

$$U_0^*(x(0)) = [0\ \dots 0\ I_m\ I_m\ \dots\ I_m] z_0^*(x(0)). \tag{9.11}$$

Theorem 6.4 states that there exists a continuous and PPWA solution $z_0^*(x)$ of the mp-LP problem (9.9). Clearly the same properties are inherited by the controller. The following Corollaries of Theorem 6.4 summarize the analytical properties of the optimal control law and the value function.

Corollary 9.1. *There exists a control law $U_0^* = \bar{f}_0(x(0))$, $\bar{f}_0 : \mathbb{R}^n \to \mathbb{R}^m$, obtained as a solution of the optimal control problem (9.2)-(9.4) with $p = 1$ or $p = \infty$, which is continuous and PPWA*

$$\bar{f}_0(x) = \bar{F}_0^i x \quad \text{if} \quad x \in CR_0^i,\ i = 1, \dots, N_0^r \tag{9.12}$$

where the polyhedral sets $CR_0^i \triangleq \{H_0^i x \le 0\}$, $i = 1, \dots, N_0^r$, are a partition of $\mathbb{R}^n$.

Note that in Corollary 9.1 the control law is linear (not affine) and the critical regions have a conic shape ($CR_0^i \triangleq \{H_0^i x \le 0\}$). This can be proven immediately from the results in Section 6.2 by observing that the constant term W_ϵ at the right-hand side on the mp-LP problem (9.9) is zero.

Corollary 9.2. *The value function $J^*(x)$ obtained as a solution of the optimal control problem (9.2)-(9.4) is convex and PPWA.*

Remark 9.2. Note that if the optimizer of problem (9.4) is unique for all $x(0) \in \mathbb{R}^n$, then Corollary 9.1 reads: " **The** control law $U^*(0) = \bar{f}_0(x(0))$, $\bar{f}_0 : \mathbb{R}^n \to \mathbb{R}^m$, obtained as a solution of the optimal control problem (9.2)-(9.4) with $p = 1$ or $p = \infty$, **is** continuous and PPWA,...". From the results of Section 6.2 we know that in case of multiple optima for some $x(0) \in \mathbb{R}^n$, a control law of the form (9.12) can always be computed.

9.2 Solution via Recursive Approach

Alternatively we can use dynamic programming to solve the same problem in a recursive manner. We define the optimal cost $J_j^*(x_j)$ for the $N-j$ step problem starting from state x_j by

$$J_j^*(x_j) \triangleq \min_{u_j,\cdots,u_{N-1}} \|Px_N\|_\infty + \sum_{k=j}^{N-1} \|Qx_k\|_\infty + \|Ru_k\|_\infty$$

According to the principle of optimality the optimal one step cost-to-go can be obtained from

$$J_{N-1}^*(x_{N-1}) = \min_{u_{N-1}} \|P_N x_N\|_\infty + \|Qx_{N-1}\|_\infty + \|Ru_{N-1}\|_\infty \tag{9.13}$$

$$\begin{array}{l} x_N = Ax_{N-1} + Bu_{N-1} \\ P_N = P \end{array} \tag{9.14}$$

Substituting (9.14) into the objective function (9.13), we have

$$J_{N-1}^*(x_{N-1}) = \min_{u_{N-1}} \|P_N(Ax_{N-1} + Bu_{N-1})\|_\infty + \|Qx_{N-1}\|_\infty + \|Ru_{N-1}\|_\infty \tag{9.15}$$

We find the optimum by solving the mp-LP

$$\begin{aligned} \min_{\varepsilon^x_{N-1},\varepsilon^x_N,\varepsilon^u_{N-1},u_{N-1}} \quad & \varepsilon^x_{N-1} + \varepsilon^x_N + \varepsilon^u_{N-1} && (9.16a) \\ \text{subj. to} \quad & -\mathbf{1}_n\varepsilon^x_{N-1} \le \pm Qx_{N-1} && (9.16b) \\ & -\mathbf{1}_r\varepsilon^x_N \le \pm P_N\left[Ax_{N-1} + Bu_{N-1}\right], && (9.16c) \\ & -\mathbf{1}_m\varepsilon^u_{N-1} \le \pm Ru_{N-1}, && (9.16d) \end{aligned}$$

By Theorem 6.4, J_{N-1}^* is a convex and piecewise affine function of x_{N-1}, the corresponding optimizer u_{N-1}^* is piecewise affine and continuous, and the feasible set $\mathcal{X}_{N-1}$ is $\mathbb{R}^n$. We use the equivalence of representation between convex and PPWA functions and infinity norm (see Section 4.1.5) to write the one-step optimal cost-to-go as

$$J^*_{N-1}(x_{N-1}) = \|P_{N-1}x_{N-1}\|_\infty, \tag{9.17}$$

with P_{N-1} defined appropriately. At the next stage, consider the two-step problem from time $N-2$ forward:

$$J^*_{N-2}(x_{N-2}) = \min_{u_{N-2}} \|P_{N-1}x_{N-1}\|_\infty + \|Qx_{N-2}\|_\infty + \|Ru_{N-2}\|_\infty \tag{9.18}$$

$$x_{N-1} = Ax_{N-2} + Bu_{N-2} \tag{9.19}$$

We recognize that (9.18), (9.19) has the same form as (9.13), (9.14). Therefore we can compute the optimal solution again by solving the mp-LP

$$\begin{aligned} \min_{\varepsilon^x_{N-2},\varepsilon^x_{N-1},\varepsilon^u_{N-2},u_{N-2}} \quad & \varepsilon^x_{N-2} + \varepsilon^x_{N-1} + \varepsilon^u_{N-2} && (9.20a)\\ \text{subj. to} \quad & -\mathbf{1}_n\varepsilon^x_{N-2} \le \pm Qx_{N-2} && (9.20b)\\ & -\mathbf{1}_r\varepsilon^x_{N-1} \le \pm P_{N-1}\left[Ax_{N-2} + Bu_{N-2}\right], && (9.20c)\\ & -\mathbf{1}_m\varepsilon^u_{N-2} \le \pm Ru_{N-2}, && (9.20d) \end{aligned}$$

The optimal two-step cost-to-go is

$$J^*_{N-2}(x_{N-2}) = \|P_{N-2}x_{N-2}\|_\infty, \tag{9.21}$$

Continuing in this manner, at some arbitrary time k the optimal control action is

$$u^*(k) = f_k(x(k)) \tag{9.22}$$

where $f_k(x)$ is continuous and PPWA

$$f_k(x) = F^i_k x \quad \text{if} \quad H^i_k x \le 0,\ i = 1,\dots,N^r_k \tag{9.23}$$

where the polyhedral sets $\{H^i_k x \le 0\}$, $i = 1,\dots,N^r_k$, are a partition of $\mathbb{R}^n$. The optimal cost-to-go starting from the measured state $x(k)$ is

$$J^*_k(x(k)) = \|P_k x(k)\|_\infty \tag{9.24}$$

Here we have introduced the notation P_k to express the optimal cost-to-go $J^*_k(x(k)) = \|P_k x(k)\|_\infty$ from time k to the end of the horizon N. We also remark that the rows of P_k correspond to the different affine functions constituting J^*_k and thus their number varies with the time index k. Clearly, we do not have a closed form as for the 2-norm Riccati Difference Equation (8.23) linking cost and control law at time k given their value at time $k-1$.

9.3 Comparison Of The Two Approaches

We will compare the batch and the recursive dynamic programming approach in terms of the results and the methods used to obtain the results. Most importantly we observe that the results obtained by the two methods are fundamentally different. The batch approach yields a formula for the *sequence of inputs* as a function of the initial state.

$$U_0^* = \bar{F}_0^i x(0) \quad \text{if} \quad \bar{H}_0^i x(0) \leq 0, \; i = 1, \ldots, \bar{N}_0^r \tag{9.25}$$

The recursive dynamic programming approach yields a feedback policy, i.e., a *sequence of feedback laws* expressing at each time step the control action as a function of the state at that time.

$$u^*(k) = F_k^i x(k) \quad \text{if} \quad H_k^i x(k) \leq 0, \; i = 1, \ldots, N_k^r \text{ for } k = 0, \ldots, N-1 \tag{9.26}$$

As this expression implies, we determine $u(k)$ at each time k as a function of the current state $x(k)$ rather than use a $u(k)$ precomputed at $k = 0$ as in the batch method. If the state evolves exactly according to the linear model (9.3) then the sequence of control actions $u(k)$ obtained from the two approaches is identical. In practice, the result of applying the sequence (9.25) in an open-loop fashion may be rather different from applying the time-varying feedback law (9.26) because the model (9.3) for predicting the system states may be inaccurate and the system may be subject to disturbances not included in the model. We expect the application of the feedback law to be more robust because at each time step the *observed* state $x(k)$ is used to determine the control action rather than the state x_k *predicted* at time $t = 0$.

We note that we can get the same feedback effect with the batch approach if we recalculate the optimal open-loop sequence at each time step j with the current measurement as initial condition. In this case we need to solve the following optimization problem

$$\begin{array}{lll} J_j^*(x(j)) = \min_{u_j,\cdots,u_{N-1}} & \|Px_N\|_\infty + \sum_{k=j}^{N-1} \|Qx_k\|_\infty + \|Ru_k\|_\infty \\ \quad\quad\quad\quad\;\; \text{subj. to} & x_j = x(j) \end{array} \tag{9.27}$$

where we note that the horizon length is shrinking at each time step.

As seen from (9.25) the solution to (9.27) relates the sequence of inputs $u_j^*, u_{j+1}^*, \ldots$ to the state $x(j)$ through a linear expression. The first part of this expression yields again the optimal feedback law (9.26) at time j, $u^*(j) = f_j(x(j))$.

Here the dynamic programming approach is clearly a more efficient way to generate the feedback policy because it requires the solution of a small mp-LP problem (9.7) for each time step. Repeated application of the batch approach, on the other hand, requires repeatedly the solution of a larger mp-LP for each time step.

9.4 Infinite Horizon Problem

For continuous processes operating over a long time period it would be interesting to solve the following infinite horizon problem.

$$J_\infty^*(x(0)) = \min_{u(0),u(1),\ldots} \sum_{k=0}^{\infty} \|Qx_k\|_\infty + \|Ru_k\|_\infty \tag{9.28}$$

Since the prediction must be carried out to infinity, application of the batch method becomes impossible. On the other hand, derivation of the optimal feedback law via dynamic programming remains viable. We can use the dynamic programming

$$\|P_j x_j\|_\infty = \min_{u_j} \|P_{j+1}x_{j+1}\|_\infty + \|Qx_j\|_\infty + \|Ru_j\|_\infty \tag{9.29}$$

$$x_{j+1} = Ax_j + Bu_j \tag{9.30}$$

with the terminal cost matrix $P_0 = Q$ and solve it backwards for $k \to -\infty$. Let us assume for the moment that the iterations converge to a solution P_∞ in a finite number of iterations. Then, the optimal feedback control law is time-invariant and piecewise affine

$$u^*(k) = F^i x(k) \quad \text{if} \quad H^i x \le 0, \; i = 1, \ldots, N^r \tag{9.31}$$

and the optimal infinite horizon cost is

$$J_\infty^*(x(0)) = \|P_\infty x(0)\|_\infty. \tag{9.32}$$

In general, the infinite time optimal cost $J_\infty^*(x(0))$ and the optimal feedback control law are not necessarily piecewise affine (with a finite number of regions). Convergence of the recursive scheme (9.29) has been studied in detail in [81]. If this recursive scheme converges and Q and R are of full column rank, then the resulting control law (9.31) stabilizes the system (see Section 7.4).

Example 9.1. Consider the double integrator system

$$\left\{ x(t+1) = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t) \right. \tag{9.33}$$

The aim is to compute the infinite horizon optimal controller that solves the optimization problem (9.28) with $Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ and $R = 20$.

The dynamic programming iteration (9.29) converges after 18 iterations to the following optimal solution:

$$u=\begin{cases}
\begin{bmatrix}9.44 & 29.44\end{bmatrix}x & \text{if } \begin{bmatrix}-0.10 & -1.00\\ -0.71 & -0.71\end{bmatrix}x\le\begin{bmatrix}0\\0\end{bmatrix} \quad \text{(Region \#1)}\\
\begin{bmatrix}9.00 & 25.00\end{bmatrix}x & \text{if } \begin{bmatrix}0.10 & 1.00\\ -0.11 & -0.99\end{bmatrix}x\le\begin{bmatrix}0\\0\end{bmatrix} \quad \text{(Region \#2)}\\
\begin{bmatrix}-1.00 & 19.00\end{bmatrix}x & \text{if } \begin{bmatrix}-0.45 & -0.89\\ 0.71 & 0.71\end{bmatrix}x\le\begin{bmatrix}0\\0\end{bmatrix} \quad \text{(Region \#3)}\\
\begin{bmatrix}8.00 & 16.00\end{bmatrix}x & \text{if } \begin{bmatrix}0.11 & 0.99\\ -0.12 & -0.99\end{bmatrix}x\le\begin{bmatrix}0\\0\end{bmatrix} \quad \text{(Region \#4)}\\
\begin{bmatrix}-2.00 & 17.00\end{bmatrix}x & \text{if } \begin{bmatrix}-0.32 & -0.95\\ 0.45 & 0.89\end{bmatrix}x\le\begin{bmatrix}0\\0\end{bmatrix} \quad \text{(Region \#5)}\\
\begin{bmatrix}7.00 & 8.00\end{bmatrix}x & \text{if } \begin{bmatrix}0.12 & 0.99\\ -0.14 & -0.99\end{bmatrix}x\le\begin{bmatrix}0\\0\end{bmatrix} \quad \text{(Region \#6)}\\
\begin{bmatrix}-3.00 & 14.00\end{bmatrix}x & \text{if } \begin{bmatrix}0.32 & 0.95\\ -0.24 & -0.97\end{bmatrix}x\le\begin{bmatrix}0\\0\end{bmatrix} \quad \text{(Region \#7)}\\
\begin{bmatrix}6.00 & 1.00\end{bmatrix}x & \text{if } \begin{bmatrix}0.14 & 0.99\\ -0.16 & -0.99\end{bmatrix}x\le\begin{bmatrix}0\\0\end{bmatrix} \quad \text{(Region \#8)}\\
\begin{bmatrix}-4.00 & 10.00\end{bmatrix}x & \text{if } \begin{bmatrix}0.24 & 0.97\\ -0.20 & -0.98\end{bmatrix}x\le\begin{bmatrix}0\\0\end{bmatrix} \quad \text{(Region \#9)}\\
\begin{bmatrix}5.00 & -5.00\end{bmatrix}x & \text{if } \begin{bmatrix}0.16 & 0.99\\ -0.20 & -0.98\end{bmatrix}x\le\begin{bmatrix}0\\0\end{bmatrix} \quad \text{(Region \#10)}\\
\begin{bmatrix}-5.00 & 5.00\end{bmatrix}x & \text{if } \begin{bmatrix}0.20 & 0.98\\ -0.16 & -0.99\end{bmatrix}x\le\begin{bmatrix}0\\0\end{bmatrix} \quad \text{(Region \#11)}\\
\begin{bmatrix}4.00 & -10.00\end{bmatrix}x & \text{if } \begin{bmatrix}0.20 & 0.98\\ -0.24 & -0.97\end{bmatrix}x\le\begin{bmatrix}0\\0\end{bmatrix} \quad \text{(Region \#12)}\\
\begin{bmatrix}-6.00 & -1.00\end{bmatrix}x & \text{if } \begin{bmatrix}0.16 & 0.99\\ -0.14 & -0.99\end{bmatrix}x\le\begin{bmatrix}0\\0\end{bmatrix} \quad \text{(Region \#13)}\\
\begin{bmatrix}3.00 & -14.00\end{bmatrix}x & \text{if } \begin{bmatrix}0.24 & 0.97\\ -0.32 & -0.95\end{bmatrix}x\le\begin{bmatrix}0\\0\end{bmatrix} \quad \text{(Region \#14)}\\
\begin{bmatrix}-7.00 & -8.00\end{bmatrix}x & \text{if } \begin{bmatrix}0.14 & 0.99\\ -0.12 & -0.99\end{bmatrix}x\le\begin{bmatrix}0\\0\end{bmatrix} \quad \text{(Region \#15)}\\
\begin{bmatrix}2.00 & -17.00\end{bmatrix}x & \text{if } \begin{bmatrix}0.32 & 0.95\\ -0.45 & -0.89\end{bmatrix}x\le\begin{bmatrix}0\\0\end{bmatrix} \quad \text{(Region \#16)}\\
\begin{bmatrix}-8.00 & -16.00\end{bmatrix}x & \text{if } \begin{bmatrix}0.12 & 0.99\\ -0.11 & -0.99\end{bmatrix}x\le\begin{bmatrix}0\\0\end{bmatrix} \quad \text{(Region \#17)}\\
\begin{bmatrix}1.00 & -19.00\end{bmatrix}x & \text{if } \begin{bmatrix}0.45 & 0.89\\ -0.71 & -0.71\end{bmatrix}x\le\begin{bmatrix}0\\0\end{bmatrix} \quad \text{(Region \#18)}\\
\begin{bmatrix}-9.00 & -25.00\end{bmatrix}x & \text{if } \begin{bmatrix}0.11 & 0.99\\ -0.10 & -1.00\end{bmatrix}x\le\begin{bmatrix}0\\0\end{bmatrix} \quad \text{(Region \#19)}\\
\begin{bmatrix}-9.44 & -29.44\end{bmatrix}x & \text{if } \begin{bmatrix}0.10 & 1.00\\ 0.71 & 0.71\end{bmatrix}x\le\begin{bmatrix}0\\0\end{bmatrix} \quad \text{(Region \#20)}
\end{cases}$$

with P_∞ equal to

$$P_\infty = \begin{bmatrix} 9.4444 & 29.4444 \\ 9.0000 & 25.0000 \\ -1.0000 & 19.0000 \\ 8.0000 & 16.0000 \\ -2.0000 & 17.0000 \\ 7.0000 & 8.0000 \\ -3.0000 & 14.0000 \\ 6.0000 & 1.0000 \\ -4.0000 & 10.0000 \\ 5.0000 & -5.0000 \\ -5.0000 & 5.0000 \\ 4.0000 & -10.0000 \\ -6.0000 & -1.0000 \\ 3.0000 & -14.0000 \\ -7.0000 & -8.0000 \\ 2.0000 & -17.0000 \\ -8.0000 & -16.0000 \\ 1.0000 & -19.0000 \\ -9.0000 & -25.0000 \\ -9.4444 & -29.4444 \end{bmatrix} \tag{9.34}$$

Note that P_∞ in (9.34) has 20 rows corresponding to the 20 linear terms (or pieces) of the piecewise linear value function $J^*_\infty(x) = \|P_\infty x\|_\infty$ for $x \in \mathbb{R}^2$. For instance, $J^*_\infty(x)$ in region 1 is $J^*_\infty(x) = 9.4444x_1 + 29.4444x_2$, where x_1 and x_2 denote the first and second component of the state vector x, respectively. Note that each linear term appears twice, with positive and negative sign. Therefore $J^*_\infty(x)$ can be written in minimal form as the infinity norm of a matrix $\|\tilde{P}_\infty x\|_\infty$ with $\tilde{P}_\infty$ being a matrix with ten rows. The value function $J^*_\infty(x) = \|P_\infty x\|_\infty$ is plotted in Figure 9.1.

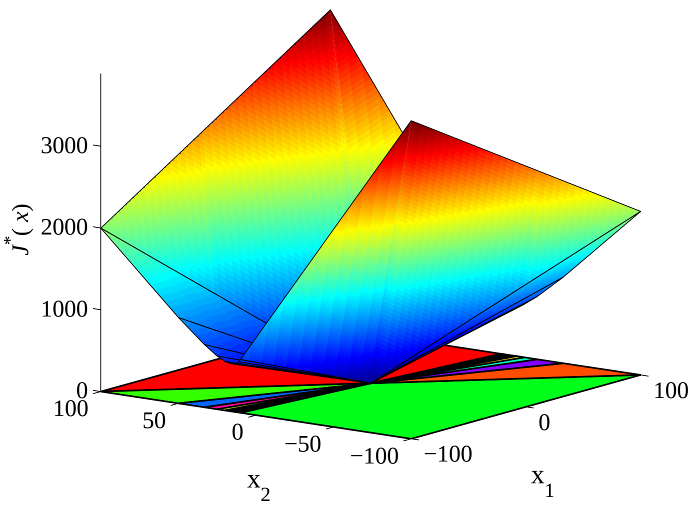


Fig. 9.1 Piecewise linear infinite time cost and corresponding polyhedral partition solution to Example 9.1

Part IV
Constrained Optimal Control of Linear Systems

Chapter 10
Constrained Optimal Control

In this chapter we study the finite time and infinite time optimal control problem for linear systems with linear constraints on inputs and state variables. We establish the structure of the optimal control law and derive algorithms for its computation. For finite time problems with linear and quadratic objective functions we show that the time varying feedback law is piecewise affine and continuous. The value function is a convex piecewise linear, resp. piecewise quadratic function of the initial state. We describe how the optimal control law can be efficiently computed by means of multiparametric linear, resp. quadratic programming. Finally we show how to compute the infinite time optimal controller for quadratic and linear objective functions and prove that, when it exists, the infinite time controller inherits all the structural properties of the finite time optimal controller.

Before formulating the finite time optimal control problem, we first introduce some fundamental concepts of set invariance theory.

10.1 Invariant Sets

In this section we deal with two types of systems, namely, autonomous systems:

$$x(t+1) = f_a(x(t)), \tag{10.1}$$

and systems subject to external inputs:

$$x(t+1) = f(x(t), u(t)). \tag{10.2}$$

We assume that the origin is an equilibrium for system (10.1) and for system (10.2) when $u = 0$. Both systems are subject to state and input constraints

$$x(t) \in \mathcal{X},\ \ u(t) \in \mathcal{U},\ \ \forall\ t \geq 0. \tag{10.3}$$

The sets $\mathcal{X}$ and $\mathcal{U}$ are polyhedra and contain the origin in their interior.

For the autonomous system (10.1) we denote the one-step reachable set for initial states x contained in the set $\mathcal{S}$ as

$$\text{Reach}(\mathcal{S}) \triangleq \{x \in \mathbb{R}^n :\ \exists\ x(0) \in \mathcal{S} \text{ s.t. } x = f_a(x(0))\}.$$

For the system (10.2) with inputs we will denote the one-step reachable set for initial states x contained in the set $\mathcal{S}$ as

$$\text{Reach}(\mathcal{S}) \triangleq \{x \in \mathbb{R}^n :\ \exists\ x(0) \in \mathcal{S},\ \exists\ u(0) \in \mathcal{U} \text{ s.t. } x = f(x(0), u(0))\}.$$

Therefore, all the states contained in $\mathcal{S}$ are mapped into the set Reach($\mathcal{S}$) under the map f_a or under the map f for some input $u \in \mathcal{U}$. “Pre” sets are the dual of one-step reachable sets. The set

$$\mathrm{Pre}(\mathcal{S}) \triangleq \{x \in \mathbb{R}^n \ : \ f_a(x) \in \mathcal{S}\} \tag{10.4}$$

defines the set of states which evolve into the target set $\mathcal{S}$ in one time step for the system (10.1). Similarly, for the system (10.2) the set of states which can be driven into the target set $\mathcal{S}$ in one time step is defined as

$$\mathrm{Pre}(\mathcal{S}) \triangleq \{x \in \mathbb{R}^n \ : \ \exists u \in \mathcal{U} \text{ s.t. } f(x,u) \in \mathcal{S}\} \tag{10.5}$$

Example 10.1. Consider the second order autonomous stable system

$$x(t+1) = Ax(t) = \begin{bmatrix} 0.5 & 0 \\ 1 & -0.5 \end{bmatrix} x(t) \tag{10.6}$$

subject to the state constraints

$$x(t) \in \mathcal{X} = \left\{ x \,\middle|\, \begin{bmatrix} -10 \\ -10 \end{bmatrix} \leq x \leq \begin{bmatrix} 10 \\ 10 \end{bmatrix} \right\},\ \forall t \geq 0 \tag{10.7}$$

The set $\mathrm{Pre}(\mathcal{X})$ can be obtained as follows: Since the set $\mathcal{X}$ is a polytope, it can be represented as a $\mathcal{H}$-polytope (Section 3.2)

$$\mathcal{X} = \{x \ : \ Hx \leq h\}, \tag{10.8}$$

where

$$H = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -1 & 0 \\ 0 & -1 \end{bmatrix} \text{ and } h = \begin{bmatrix} 10 \\ 10 \\ 10 \\ 10 \end{bmatrix}$$

By using this $\mathcal{H}$-presentation and the system equation (10.6), the set $\mathrm{Pre}(\mathcal{X})$ can be derived:

$$\begin{aligned} \mathrm{Pre}(\mathcal{X}) &= \left\{x \ : \ Hf_a(x) \leq h, \right\} &(10.9)\\ &= \{x \ : \ HAx \leq h\} &(10.10)\end{aligned}$$

The set (10.10) may contain redundant inequalities which can be removed by using Algorithm 3.1 in Section 3.4.4 to obtain its minimal representation. Note that by using the notation in Section 3.4.11, the set $\mathrm{Pre}(\mathcal{X})$ in (10.10) is simply $\mathcal{X} \circ A$.

The set $\mathrm{Pre}(\mathcal{X})$ is

$$\mathrm{Pre}(\mathcal{X}) = \left\{ x \ : \ \begin{bmatrix} 1 & 0 \\ 1 & -0.5 \\ -1 & 0 \\ -1 & -0.5 \end{bmatrix} x \leq \begin{bmatrix} 20 \\ 10 \\ 20 \\ 10 \end{bmatrix} \right\}$$

The set $\mathrm{Pre}(\mathcal{X}) \cap \mathcal{X}$, the significance of which we will discuss below, is

$$\mathrm{Pre}(\mathcal{X}) \cap \mathcal{X} = \left\{ x \ : \ \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -1 & 0 \\ 0 & -1 \\ 1 & -0.5 \\ -1 & 0.5 \end{bmatrix} x \leq \begin{bmatrix} 10 \\ 10 \\ 10 \\ 10 \\ 10 \\ 10 \end{bmatrix} \right\}$$

and it is depicted in Fig. 10.1.

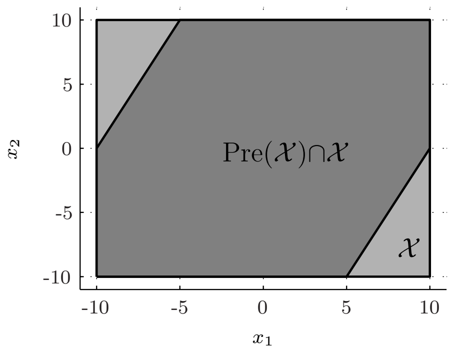


Fig. 10.1 Example 10.1: $\mathrm{Pre}(\mathcal{X}) \cap \mathcal{X}$ for system (10.6) under constraints (10.7).

The set $\mathrm{Reach}(\mathcal{X})$ is obtained by applying the map A to the set $\mathcal{X}$. Let us write $\mathcal{X}$ in $\mathcal{V}$-representation (see Section 3.1)

$$\mathcal{X} = \mathrm{conv}(V) \tag{10.11}$$

and let us map the set of vertices V through the transformation A. Because the transformation is linear, the reach set is simply the convex hull of the transformed vertices

$$\mathrm{Reach}(\mathcal{X}) = A \circ \mathcal{X} = \mathrm{conv}(AV) \tag{10.12}$$

We refer the reader to Section 3.4.11 for a detailed discussion on linear transformations of polyhedra.

The set $\mathrm{Reach}(\mathcal{X})$ in $\mathcal{H}$-representation is

$$\mathrm{Reach}(\mathcal{X}) = \left\{ x \ : \ \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 1 & -0.5 \\ -1 & 0.5 \end{bmatrix} x \leq \begin{bmatrix} 5 \\ 5 \\ 2.5 \\ 2.5 \end{bmatrix} \right\}$$

and is depicted in Fig. 10.2.

Example 10.2. Consider the second order unstable system

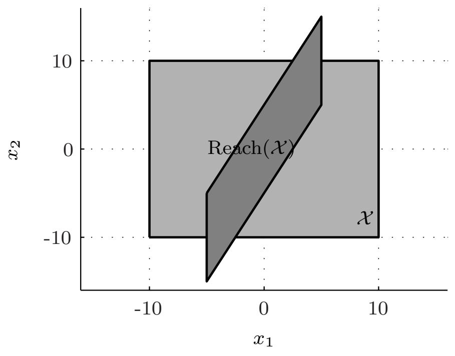


Fig. 10.2 Example 10.1: one-step reachable sets for system (10.6)

$$\left\{ x(t+1) = Ax + Bu = \begin{bmatrix} 1.5 & 0 \\ 1 & -1.5 \end{bmatrix} x(t) + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u(t) \right. \tag{10.13}$$

subject to the input and state constraints

$$u(t) \in \mathcal{U} = \{u | -5 \le u \le 5\}, \ \forall t \ge 0 \tag{10.14a}$$

$$x(t) \in \mathcal{X} = \left\{ x \mid \begin{bmatrix} -10 \\ -10 \end{bmatrix} \le x \le \begin{bmatrix} 10 \\ 10 \end{bmatrix} \right\}, \ \forall t \ge 0 \tag{10.14b}$$

For the non-autonomous system (10.13), the set Pre($\mathcal{X}$) can be computed using the $\mathcal{H}$-representation of $\mathcal{X}$ and $\mathcal{U}$,

$$\mathcal{X} = \{x \mid Hx \le h\}, \quad \mathcal{U} = \{u \mid H_u u \le h_u\}, \tag{10.15}$$

to obtain

$$\text{Pre}(\mathcal{X}) = \left\{ x \in \mathbb{R}^2 \mid \exists u \in \mathcal{U} \text{ s.t. } f(x,u) \in \mathcal{X}, \right\} \tag{10.16}$$

$$= \left\{ x \in \mathbb{R}^2 \mid \exists u \in \mathbb{R} \text{ s.t. } \begin{bmatrix} HA & HB \\ 0 & H_u \end{bmatrix} \begin{pmatrix} x \\ u \end{pmatrix} \le \begin{bmatrix} h \\ h_u \end{bmatrix} \right\}. \tag{10.17}$$

The half-spaces in (10.17) define a polytope in the state-input space, and a projection operation (see Section 3.4.6) is used to derive the half-spaces which define Pre($\mathcal{X}$) in the state space. The set Pre($\mathcal{X}$) $\cap$ $\mathcal{X}$ is depicted in Fig. 10.3 and reported below:

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -1 & 0 \\ 0 & -1 \\ 1 & -1.5 \\ -1 & 1.5 \end{bmatrix} x \le \begin{bmatrix} 10 \\ 10 \\ 10 \\ 10 \\ 10 \\ 10 \end{bmatrix}$$

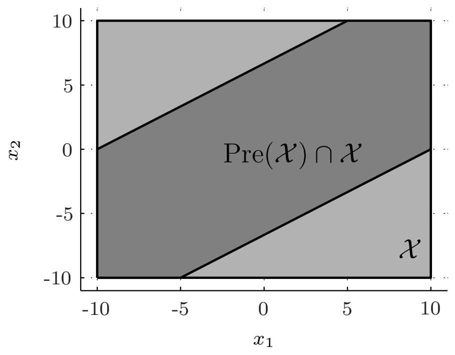


Fig. 10.3 Example 10.2: $\mathrm{Pre}(\mathcal{X}) \cap \mathcal{X}$ for system (10.13) under constraints (10.14).

Note that by using the definition of the Minkowski sum given in Section 3.4.9 and the affine operation on polyhedra in Section 3.4.11 we can compactly write the operations in (10.17) as follows:

$$\begin{aligned}
\mathrm{Pre}(\mathcal{X}) = &\{x \ : \ \exists u \in \mathcal{U} \text{ s.t. } Ax + Bu \in \mathcal{X}\} \\
&\{x \ : \ y = Ax + Bu, \ y \in \mathcal{X}, \ u \in \mathcal{U}\} \\
&\{x \ : \ Ax = y + (-Bu), \ y \in \mathcal{X}, \ u \in \mathcal{U}\} \\
&\{x \ : \ Ax \in \mathcal{C}, \ \mathcal{C} = \mathcal{X} \oplus (-B) \circ \mathcal{U}\} \\
&\{x \ : \ x \in \mathcal{C} \circ A, \ \mathcal{C} = \mathcal{X} \oplus (-B) \circ \mathcal{U}\} \\
&\{x \ : \ x \in (\mathcal{X} \oplus (-B) \circ \mathcal{U}) \circ A\}
\end{aligned} \tag{10.18}$$

The set $\mathrm{Reach}(\mathcal{X}) = \{Ax + Bu \in \mathbb{R}^2 \ : \ x \in \mathcal{X}, \ u \in \mathcal{U}\}$ is obtained by applying the map A to the set $\mathcal{X}$ and then considering the effect of the input $u \in \mathcal{U}$. As shown before,

$$A \circ \mathcal{X} = \mathrm{conv}(AV) \tag{10.19}$$

and therefore

$$\mathrm{Reach}(\mathcal{X}) = \{y + Bu \ : \ y \in A \circ \mathcal{X}, \ u \in \mathcal{U}\}$$

We can use the definition of the Minkowski sum given in Section 3.4.9 and rewrite the set $\mathrm{Reach}(\mathcal{X})$ as

$$\mathrm{Reach}(\mathcal{X}) = (A \circ \mathcal{X}) \oplus (B \circ \mathcal{U})$$

We can compute the Minkowski sum via projection or vertex enumeration as explained in Section 3.4.9 and obtain the set $\mathrm{Reach}(\mathcal{X})$ in $\mathcal{H}$-representation

$$\text{Reach}(\mathcal{X}) = \left\{ x \;:\; \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 1 & -1.5 \\ 1 & -1.5 \\ -1 & 1.5 \end{bmatrix} x \le \begin{bmatrix} 20 \\ 20 \\ 25 \\ 25 \\ 27.5 \\ 27.5 \end{bmatrix} \right\}$$

which is depicted in Fig. 10.4.

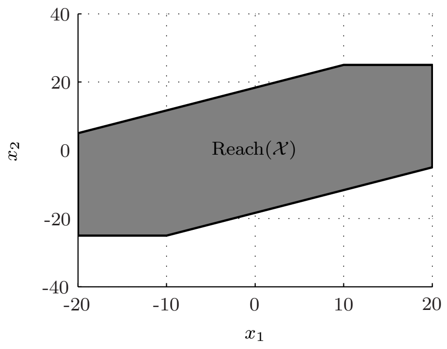


Fig. 10.4 Example 10.2: one-step reachable sets for system (10.13) under constraints (10.14).

Remark 10.1. In summary, the sets Pre($\mathcal{X}$) and Reach($\mathcal{X}$) are the results of linear operations on the polyhedra $\mathcal{X}$ and $\mathcal{U}$ and therefore are polyhedra. By using the definition of the Minkowski sum given in Section 3.4.9 and of affine operation on polyhedra in Section 3.4.11 we can compactly summarize the Pre and Reach operations on linear systems as follows:

	$x(t+1) = Ax(t)$	$x(k+1) = Ax(t) + Bu(t)$
Pre($\mathcal{X}$)	$\mathcal{X} \circ A$	$(\mathcal{X} \oplus (-B \circ \mathcal{U})) \circ A$
Reach($\mathcal{X}$)	$A \circ \mathcal{X}$	$(A \circ \mathcal{X}) \oplus (B \circ \mathcal{U})$

Table 10.1 Pre and Reach operations for linear systems subject to polyhedral input and state constraints $x(t) \in \mathcal{X},\ u(t) \in \mathcal{U}$

Two different types of sets are considered in this chapter: *invariant sets* and *control invariant sets*. We will first discuss invariant sets. Invariant sets are computed for autonomous systems. These types of sets are useful to answer questions such as: “For a *given* feedback controller $u = g(x)$, find the set of initial states whose trajectory will never violate the system constraints”. The following definitions, derived from [158, 52, 47, 45, 163, 128, 129, 130], introduce the different types of invariant sets.

Definition 10.1 (Positive Invariant Set). A set $\mathcal{O} \subseteq \mathcal{X}$ is said to be a positive invariant set for the autonomous system (10.1) subject to the constraints in (10.3), if

$$x(0) \in \mathcal{O} \quad \Rightarrow \quad x(t) \in \mathcal{O}, \quad \forall t \in \mathbb{N}_+$$

Definition 10.2 (Maximal Positive Invariant Set $\mathcal{O}_\infty$). The set $\mathcal{O}_\infty \subseteq \mathcal{X}$ is the maximal invariant set of the autonomous system (10.1) subject to the constraints in (10.3) if $0 \in \mathcal{O}_\infty$, $\mathcal{O}_\infty$ is invariant and $\mathcal{O}_\infty$ contains all the invariant sets that contain the origin.

Remark 10.2. The condition that $\mathcal{O}_\infty$ must contain the origin is added because system (10.1) may have multiple equilibrium points and thus multiple invariant sets which are disconnected. Furthermore, if these sets are used as a target sets in control problems, they should contain only one equilibrium point in order to get predictable closed-loop behavior.

Remark 10.3. The maximal invariant sets defined here are often referred to as 'maximal admissible sets' or 'maximal output admissible sets' in the literature (e.g. [115]), depending on whether the system state or output is constrained.

Theorem 10.1 (Geometric condition for invariance [90]). *A set $\mathcal{O} \subseteq \mathcal{X}$ is a positive invariant set for the autonomous system (10.1) subject to the constraints in (10.3), if and only if*

$$\mathcal{O} \subseteq \text{Pre}(\mathcal{O}) \tag{10.20}$$

Proof: We prove both the necessary and sufficient parts by contradiction. ($\Leftarrow$:) If $\mathcal{O} \nsubseteq \text{Pre}(\mathcal{O})$ then $\exists \bar{x} \in \mathcal{O}$ such that $\bar{x} \notin \text{Pre}(\mathcal{O})$. From the definition of $\text{Pre}(\mathcal{O})$, $A\bar{x} \notin \mathcal{O}$ and thus $\mathcal{O}$ is not positive invariant. ($\Rightarrow$:) If $\mathcal{O}$ is not a positive invariant set then $\exists \bar{x} \in \mathcal{O}$ such that $A\bar{x} \notin \mathcal{O}$. This implies that $\bar{x} \in \mathcal{O}$ and $\bar{x} \notin \text{Pre}(\mathcal{O})$ and thus $\mathcal{O} \nsubseteq \text{Pre}(\mathcal{O})$ □

It is immediate to prove that condition (10.20) of Theorem 10.1 is equivalent to the following condition

$$\text{Pre}(\mathcal{O}) \cap \mathcal{O} = \mathcal{O} \tag{10.21}$$

Based on condition (10.21), the following algorithm provides a procedure for computing the maximal positive invariant subset $\mathcal{O}_\infty$ for system (10.1),(10.3) [10, 45, 158, 115].

Algorithm 10.1 (Computation of $\mathcal{O}_\infty$)

```
INPUT $f_a$ , $\mathcal{X}$
OUTPUT $\mathcal{O}_\infty$
1. LET $\Omega_0 \leftarrow \mathcal{X}$
2. LET $\Omega_{k+1} \leftarrow \text{Pre}(\Omega_k) \cap \Omega_k$
3. IF $\Omega_{k+1} = \Omega_k$ THEN
       $\mathcal{O}_\infty \leftarrow \Omega_{k+1}$
   ELSE GOTO 2
4. END
```

Algorithm 10.1 generates the set sequence $\{\Omega_k\}$ satisfying $\Omega_{k+1} \subseteq \Omega_k, \forall k \in \mathbb{N}$ and, if it terminates, it terminates when $\Omega_{k+1} = \Omega_k$ so that Ω_k is the maximal positive invariant set $\mathcal{O}_\infty$ for the system (10.1)-(10.3). In general, Algorithm 10.1 may never terminate. Condition for finite time termination of Algorithm 10.1 can be found in [114].

Example 10.3. Consider the second order stable system in Example 10.1. The maximal positive invariant set of system (10.6) subject to constraints (10.7) is depicted in Fig. 10.5 and reported below:

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -1 & 0 \\ 0 & -1 \\ 1 & -0.5 \\ -1 & 0.5 \end{bmatrix} x \leq \begin{bmatrix} 10 \\ 10 \\ 10 \\ 10 \\ 10 \\ 10 \end{bmatrix}$$

Note from the previous discussion of the example and from Figure 10.1 that here the maximal positive invariant set $\mathcal{O}_\infty$ is obtained after a single step of Algorithm 10.1, i.e.

$$\mathcal{O}_\infty = \Omega_1 = \text{Pre}(\mathcal{X}) \cap \mathcal{X} .$$

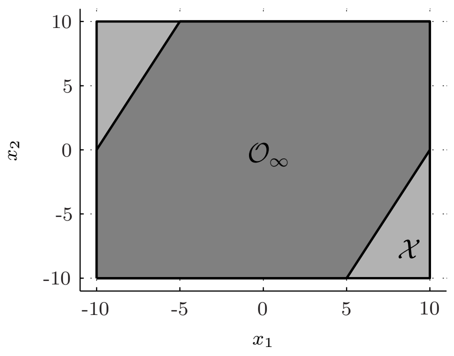


Fig. 10.5 Maximal Positive Invariant Set of system (10.6) under constraints (10.7).

Control invariant sets are defined for systems subject to external inputs. These types of sets are useful to answer questions such as: “Find the set of initial states for which *there exists* a controller such that the system constraints are never violated”. The following definitions, adopted from [158, 52, 47, 45, 163], introduce the different types of control invariant sets.

Definition 10.3 (Control Invariant Set). A set $\mathcal{C} \subseteq \mathcal{X}$ is said to be a control invariant set for the system in (10.2) subject to the constraints in

(10.3), if

$$x(t) \in \mathcal{C} \quad \Rightarrow \quad \exists u(t) \in \mathcal{U} \text{ such that } f(x(t), u(t)) \in \mathcal{C}, \quad \forall t \in \mathbb{N}_+$$

Definition 10.4 (Maximal Control Invariant Set $\mathcal{C}_\infty$). The set $\mathcal{C}_\infty \subseteq \mathcal{X}$ is said to be the maximal control invariant set for the system in (10.2) subject to the constraints in (10.3), if it is control invariant and contains all control invariant sets contained in $\mathcal{X}$.

Remark 10.4. The geometric conditions for invariance (10.20), (10.21) hold for control invariant sets.

The following algorithm provides a procedure for computing the maximal control invariant set $\mathcal{C}_\infty$ for system (10.2),(10.3) [10, 45, 158, 115].

Algorithm 10.2 (Computation of $\mathcal{C}_\infty$)

```
INPUT $f$, $\mathcal{X}$ and $\mathcal{U}$
OUTPUT $\mathcal{C}_\infty$
1. LET $\Omega_0 \leftarrow \mathcal{X}$
2. LET $\Omega_{k+1} \leftarrow \text{Pre}(\Omega_k) \cap \Omega_k$
3. IF $\Omega_{k+1} = \Omega_k$ THEN
       $\mathcal{C}_\infty \leftarrow \Omega_{k+1}$
   ELSE GOTO 2
4. END
```

Algorithm 10.2 generates the set sequence $\{\Omega_k\}$ satisfying $\Omega_{k+1} \subseteq \Omega_k, \forall k \in \mathbb{N}$ and $\mathcal{O}_\infty = \bigcap_{k \geq 0} \Omega_k$. If $\Omega_k = 0$ for some integer k then the simple conclusion is that $\mathcal{O}_\infty = 0$. Algorithm 10.2 terminates if $\Omega_{k+1} = \Omega_k$ so that Ω_k is the maximal control invariant set $\mathcal{C}_\infty$ for the system (10.2)-(10.3). In general, Algorithm 10.2 may never terminate [10, 45, 158, 151]. The same holds true for non-autonomous systems.

Example 10.4. Consider the second order unstable system in example 10.2. Algorithm 10.2 is used to compute the maximal control invariant set of system (10.13) subject to constraints (10.14). Algorithm 10.2 terminates after 45 iterations and the maximal control invariant set $\mathcal{C}_\infty$ is:

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -1 & 0 \\ 0 & -1 \\ -0.25 & 0.375 \\ 0.25 & -0.375 \end{bmatrix} x \leq \begin{bmatrix} 4 \\ 10 \\ 10 \\ 4 \\ 1 \\ 1 \end{bmatrix}$$

The results of the iterations and $\mathcal{C}_\infty$ are depicted in Fig. 10.6.

Definition 10.5 (Finitely determined set). Consider Algorithm 10.1 (Algorithm 10.2). The set $\mathcal{O}_\infty$ ($\mathcal{C}_\infty$) is finitely determined if and only if $\exists\ i \in \mathbb{N}$ such that $\Omega_{i+1} = \Omega_i$. The smallest element $i \in \mathbb{N}$ such that $\Omega_{i+1} = \Omega_i$ is called the determinedness index.

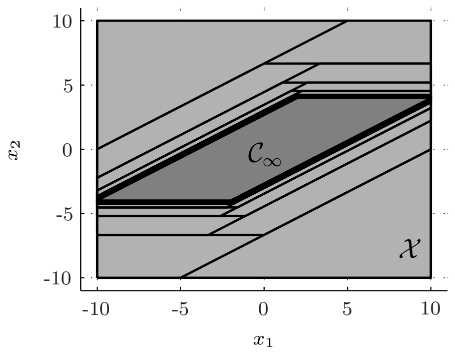


Fig. 10.6 Maximal Control Invariant Set of system (10.13) subject to constraints (10.14).

Remark 10.5. From Remark 10.1, for linear system with linear constraints the sets $\mathcal{O}_\infty$ and $\mathcal{C}_\infty$ are polyhedra if they are finitely determined.

For all states contained in the maximal control invariant set $\mathcal{C}_\infty$ there exists a control law, such that the system constraints are never violated. This does not imply that there exists a control law which can drive the state into a user-specified target set. This issue is addressed in the following by introducing the concept of stabilizable sets.

Definition 10.6 (N-Step Controllable Set $\mathcal{K}_N(\mathcal{O})$). For a given target set $\mathcal{O} \subseteq \mathcal{X}$, the N-step controllable set $\mathcal{K}_N(\mathcal{O})$ of the system (10.2) subject to the constraints (10.3) is defined recursively as:

$$\mathcal{K}_j(\mathcal{O}) \triangleq \text{Pre}(\mathcal{K}_{j-1}(\mathcal{O})), \quad \mathcal{K}_0(\mathcal{O}) = \mathcal{O}, \qquad j \in \{1, \ldots, N\}$$

From Definition 10.6, all states x_0 belonging to the N-Step Controllable Set $\mathcal{K}_N(\mathcal{O})$ can be driven, by a suitable control sequence, to the target set $\mathcal{O}$ in N steps, while satisfying input and state constraints.

Definition 10.7 (Maximal Controllable Set $\mathcal{K}_\infty(\mathcal{O})$). For a given target set $\mathcal{O} \subseteq \mathcal{X}$, the maximal controllable set $\mathcal{K}_\infty(\mathcal{O})$ for system (10.2) subject to the constraints in (10.3) is the union of all N-step controllable sets $\mathcal{K}_N(\mathcal{O})$ contained in $\mathcal{X}$ ($N \in \mathbb{N}$).

We will often deal with controllable sets $\mathcal{K}_N(\mathcal{O})$ where the target $\mathcal{O}$ is a control invariant set. They are special sets, since in addition to guaranteeing that from $\mathcal{K}_N(\mathcal{O})$ we reach $\mathcal{O}$ in N steps, one can ensure that once it has reached $\mathcal{O}$, the system can stay there at all future time instants.

Definition 10.8 (N-step (Maximal) Stabilizable Set). For a given control invariant set $\mathcal{O} \subseteq \mathcal{X}$, the N-step (maximal) stabilizable set of the system (10.2) subject to the constraints (10.3) is the N-step (maximal) controllable set $\mathcal{K}_N(\mathcal{O})$ ($\mathcal{K}_\infty(\mathcal{O})$).

The set $\mathcal{K}_\infty(\mathcal{O})$ contains all states which can be steered into the control invariant set $\mathcal{O}$ and hence $\mathcal{K}_\infty(\mathcal{O}) \subseteq \mathcal{C}_\infty$. For *linear systems* the set $\mathcal{K}_\infty(\mathcal{O}) \subseteq \mathcal{C}_\infty$ can be computed as follows (cf. [52, 48]):

Algorithm 10.3 (Maximal Stabilizable Set $\mathcal{K}_\infty(\mathcal{O})$)

1. *$\mathcal{K}_0 \leftarrow \mathcal{O}$, where $\mathcal{O}$ is a control invariant set*
2. *$\mathcal{K}_{c+1} \leftarrow \mathrm{Pre}(\mathcal{K}_c)$*
3. *If $\mathcal{K}_{c+1} = \mathcal{K}_c$, then $\mathcal{K}_\infty(\mathcal{O}) \leftarrow \mathcal{K}_c$, return; Else, set $c = c+1$ and goto 2.*

Since $\mathcal{O}$ is control invariant, it holds $\forall c \in \mathbb{N}$ that $\mathcal{K}_c(\mathcal{O})$ is control invariant and $\mathcal{K}_c \subseteq \mathcal{K}_{c+1}$. Note that Algorithm 10.3 is not guaranteed to terminate in finite time.

Remark 10.6. In general, the maximal stabilizable set $\mathcal{K}_\infty(\mathcal{O})$ is not equal to the maximal control invariant set $\mathcal{C}_\infty$, even for linear systems. $\mathcal{K}_\infty(\mathcal{O}) \subseteq \mathcal{C}_\infty$ for all control invariant sets $\mathcal{O}$. The set $\mathcal{C}_\infty \setminus \mathcal{K}_\infty(\mathcal{O})$ includes all initial states from which it is not possible to steer the system to the stabilizable region $\mathcal{K}_\infty(\mathcal{O})$ and hence $\mathcal{O}$.

Example 10.5. Consider the simple constrained one-dimensional system

$$x(t+1) = 2x(t) + u(t), \tag{10.22a}$$
$$|x(t)| \leq 1, \text{ and } |u(t)| \leq 1 \tag{10.22b}$$

and the state-feedback control law

$$u(t) = \begin{cases} 1 & \text{if} \quad x(t) \in \left[-1, \ -\frac{1}{2}\right], \\ -2x(t) & \text{if} \quad x(t) \in \left[-\frac{1}{2}, \ \frac{1}{2}\right], \\ -1 & \text{if} \quad x(t) \in \left[\frac{1}{2}, \ 1\right] \end{cases} \tag{10.23}$$

The closed-loop system has three equilibria at -1, 0, and 1 and system (10.22) is always feasible for all initial states in $[-1, \ 1]$ and therefore $\mathcal{C}_\infty = [-1, \ 1]$. It is, however, asymptotically stable only for the open set $(-1, \ 1)$. In fact, $u(t)$ and any other feasible control law cannot stabilize the system from $x = 1$ and from $x = -1$ and therefore when $\mathcal{O} = 0$ then $\mathcal{K}_\infty(\mathcal{O}) = (-1, \ 1) \subset \mathcal{C}_\infty$. We note in this example that the maximal stabilizable set is open. One can easily argue that, in general, if the maximal stabilizable set is closed then it is equal to the maximal control invariant set.

N-step reachable sets are defined analogously to N-step controllable sets.

Definition 10.9 (N-Step Reachable Set $\mathcal{R}_N(\mathcal{X}_0)$). For a given initial set $\mathcal{X}_0 \subseteq \mathcal{X}$, the N-step reachable set $\mathcal{R}_N(\mathcal{X}_0)$ of the system (10.1) or (10.2) subject to the constraints (10.3) is defined as:

$$\mathcal{R}_{i+1}(\mathcal{X}_0) \triangleq \mathrm{Reach}(\mathcal{R}_i(\mathcal{X}_0)), \quad \mathcal{R}_0(\mathcal{X}_0) = \mathcal{X}_0, \qquad i = 0, \ldots, N-1$$

From Definition 10.9, all states x_0 belonging to $\mathcal{X}_0$ will evolve to the N-step reachable set $\mathcal{R}_N(\mathcal{X}_0)$ in N steps.

10.2 Constrained Optimal Control Problem Formulation

Consider the linear time-invariant system

$$x(t+1) = Ax(t) + Bu(t) \tag{10.24}$$

where $x(t) \in \mathbb{R}^n$, $u(t) \in \mathbb{R}^m$ are the state and input vectors, respectively, subject to the constraints

$$x(t) \in \mathcal{X},\ u(t) \in \mathcal{U},\ \forall t \geq 0 \tag{10.25}$$

The sets $\mathcal{X} \subseteq \mathbb{R}^n$ and $\mathcal{U} \subseteq \mathbb{R}^m$ are polyhedra.

Remark 10.7. The results of this chapter also hold for more general forms of linear constraints such as mixed input and state constraints

$$[x(t)', u(t)'] \in \mathcal{P}_{x,u} \tag{10.26}$$

where $\mathcal{P}_{x,u}$ is a polyhedron in $\mathbb{R}^{n+m}$ or mixed input and state constraints over a finite time

$$[x(0)', \ldots, x(N-1)', u(0)', \ldots, u(N-1)'] \in \mathcal{P}_{x,u,N} \tag{10.27}$$

where $\mathcal{P}_{x,u,N}$ is a polyhedron in $\mathbb{R}^{N(n+m)}$. Note that constraints of the type (10.27) can arise, for example, from constraints on the input rate $\Delta u(t) \triangleq u(t) - u(t-1)$. In this chapter, for the sake of simplicity, we will use the less general form (10.25).

Define the cost function

$$J_0(x(0), U_0) \triangleq p(x_N) + \sum_{k=0}^{N-1} q(x_k, u_k) \tag{10.28}$$

where x_k denotes the state vector at time k obtained by starting from the state $x_0 = x(0)$ and applying to the system model

$$x_{k+1} = Ax_k + Bu_k \tag{10.29}$$

the input sequence $U_0 \triangleq [u_0', \ldots, u_{N-1}']'$.

If the 1-norm or ∞-norm is used in the cost function (10.28), then we set $p(x_N) = \|Px_N\|_p$ and $q(x_k, u_k) = \|Qx_k\|_p + \|Ru_k\|_p$ with $p = 1$ or $p = \infty$ and P, Q, R full column rank matrices. Cost (10.28) is rewritten as

$$J_0(x(0),U_0) \triangleq \|Px_N\|_p + \sum_{k=0}^{N-1} \|Qx_k\|_p + \|Ru_k\|_p \tag{10.30}$$

If the squared euclidian norm is used in the cost function (10.28), then we set $p(x_N) = x_N'Px_N$ and $q(x_k,u_k) = x_k'Qx_k + u_k'Ru_k$ with $P \succeq 0$, $Q \succeq 0$ and $R \succ 0$. Cost (10.28) is rewritten as

$$J_0(x(0),U_0) \triangleq x_N'Px_N + \sum_{k=0}^{N-1} x_k'Qx_k + u_k'Ru_k \tag{10.31}$$

Consider the constrained finite time optimal control problem (CFTOC)

$$\begin{array}{rl} J_0^*(x(0)) = \min_{U_0} & J_0(x(0),U_0) \\ \text{subj. to} & x_{k+1} = Ax_k + Bu_k,\ k=0,\ldots,N-1 \\ & x_k \in \mathcal{X},\ u_k \in \mathcal{U},\ k=0,\ldots,N-1 \\ & x_N \in \mathcal{X}_f \\ & x_0 = x(0) \end{array} \tag{10.32}$$

where N is the time horizon and $\mathcal{X}_f \subseteq \mathbb{R}^n$ is a terminal polyhedral region. In (10.28)–(10.32) $U_0 = [u_0',\ldots,u_{N-1}']' \in \mathbb{R}^s$, $s \triangleq mN$ is the optimization vector. We denote with $\mathcal{X}_0 \subseteq \mathcal{X}$ the set of initial states $x(0)$ for which the optimal control problem (10.28)–(10.32) is feasible, i.e.,

$$\begin{array}{l} \mathcal{X}_0 = \{x_0 \in \mathbb{R}^n :\ \exists(u_0,\ldots,u_{N-1}) \text{ such that } x_k \in \mathcal{X},\ u_k \in \mathcal{U},\ k=0,\ldots,N-1,\ x_N \in \mathcal{X}_f \\ \qquad \text{where } x_{k+1} = Ax_k + Bu_k,\ k=0,\ldots,N-1\}, \end{array} \tag{10.33}$$

Note that we distinguish between the *current* state $x(k)$ of system (10.24) at time k and the variable x_k in the optimization problem (10.32), that is the *predicted* state of system (10.24) at time k obtained by starting from the state $x_0 = x(0)$ and applying to system (10.29) the input sequence $u_0,\ldots,u_{k-1}$. Analogously, $u(k)$ is the input applied to system (10.24) at time k while u_k is the k-th optimization variable of the optimization problem (10.32).

If we use cost (10.31) with the squared euclidian norm and set

$$\{(x,u) \in \mathbb{R}^{n+m}\ :\ x \in \mathcal{X},\ u \in \mathcal{U}\} = \mathbb{R}^{n+m},\ \mathcal{X}_f = \mathbb{R}^n, \tag{10.34}$$

problem (10.32) becomes the standard unconstrained finite time optimal control problem (Chapter 8) whose solution (under standard assumptions on A, B, P, Q and R) can be expressed through the time varying state feedback control law (8.27)

$$u^*(k) = F_k x(k) \quad k=0,\ldots,N-1 \tag{10.35}$$

The optimal cost is given by

$$J_0^*(x(0)) = x(0)'P_0x(0). \tag{10.36}$$

If we let $N \to \infty$ as discussed in Section 8.4, then Problem (10.31)–(10.32)–(10.34) becomes the standard infinite horizon linear quadratic regulator (LQR) problem whose solution (under standard assumptions on A, B, P, Q and R) can be expressed as the state feedback control law (cf. (8.32))

$$u^*(k) = F_\infty x(k), \quad k = 0, 1, \ldots \tag{10.37}$$

In the following chapters we will show that the solution to Problem (10.32) can again be expressed in feedback form where now $u^*(k)$ is a continuous piecewise affine function on polyhedra of the state $x(k)$, i.e., $u^*(k) = f_k(x(k))$ where

$$f_k(x) = F_k^j x + g_k^j \quad \text{if} \quad H_k^j x \le K_k^j, \; j = 1, \ldots, N_k^r. \tag{10.38}$$

Matrices H_k^j and K_k^j in equation (10.38) describe the j-th polyhedron $CR_k^j = \{x \in \mathbb{R}^n | H_k^j x \le K_k^j\}$ inside which the feedback optimal control law $u^*(k)$ at time k has the affine form $F_k^j x + g_k^j$. The set of polyhedra CR_k^j, $j = 1, \ldots, N_k^r$ is a *polyhedral partition* of the set of feasible states $\mathcal{X}_k$ of Problem (10.32) at time k. The sets $\mathcal{X}_k$ are discussed in detail in the next section. Since the functions $f_k(x(k))$ are continuous, the use of polyhedral partitions rather than strict polyhedral partitions (Definition 3.5) will not cause any problem, indeed it will simplify the exposition.

In the rest of this chapter we will characterize the structure of the value function and describe how the optimal control law can be efficiently computed by means of multiparametric linear and quadratic programming. We will distinguish the cases 1- or ∞-norm and squared 2-norm.

10.3 Feasible Solutions

We denote with $\mathcal{X}_i$ the set of states x_i at time i for which (10.28)–(10.32) is feasible, for $i = 0, \ldots, N$. The sets $\mathcal{X}_i$ for $i = 0, \ldots, N$ play an important role in the the solution of (10.32). They are independent of the cost function (as long as it guarantees the existence of a minimum) and of the algorithm used to compute the solution to problem (10.32). There are two ways to rigourously define and compute the sets $\mathcal{X}_i$: the *batch approach* and the *recursive approach*. In the batch approach

$$\begin{aligned}\mathcal{X}_i = \{x_i \in \mathcal{X} :\; & \exists (u_i, \ldots, u_{N-1}) \text{ such that } x_k \in \mathcal{X},\; u_k \in \mathcal{U},\; k = i, \ldots, N-1,\; x_N \in \mathcal{X}_f \\ & \text{where } x_{k+1} = Ax_k + Bu_k,\; k = i, \ldots, N-1\},\end{aligned} \tag{10.39}$$

The definition of $\mathcal{X}_i$ in (10.39) requires that for any initial state $x_i \in \mathcal{X}_i$ there exists a feasible sequence of inputs $U_i \triangleq [u_i', \ldots, u_{N-1}']$ which keeps the state evolution in the feasible set $\mathcal{X}$ at future time instants $k = i+1, \ldots, N-1$

and forces x_N into $\mathcal{X}_f$ at time N. Clearly $\mathcal{X}_N = \mathcal{X}_f$. Next we show how to compute $\mathcal{X}_i$ for $i = 0, \dots, N-1$. Let the state and input constraint sets $\mathcal{X}$, $\mathcal{X}_f$ and $\mathcal{U}$ be the $\mathcal{H}$-polyhedra $A_x x \leq b_x$, $A_f x \leq b_f$, $A_u u \leq b_u$, respectively. Define the polyhedron $\mathcal{P}_i$ for $i = 0, \dots, N-1$ as follows

$$\mathcal{P}_i = \{(U_i, x_i) \in \mathbb{R}^{m(N-i)+n} : \ G_i U_i - E_i x_i \leq W_i\} \tag{10.40}$$

where G_i, E_i and W_i are defined as follows

$$G_i = \begin{bmatrix} A_u & 0 & \dots 0 \\ 0 & A_u & \dots 0 \\ \vdots & \vdots & \vdots \ \ \vdots \\ 0 & 0 & \dots A_u \\ 0 & 0 & \dots 0 \\ A_x B & 0 & \dots 0 \\ A_x A B & A_x B & \dots 0 \\ \vdots & \vdots & \vdots \ \ \vdots \\ A_f A^{N-i-1} B & A_x A^{N-i-2} B & \dots A_x B \end{bmatrix}, \ E_i = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ -A_x \\ -A_x A \\ -A_x A^2 \\ \vdots \\ -A_f A^{N-i} \end{bmatrix}, \ W_i = \begin{bmatrix} b_u \\ b_u \\ \vdots \\ b_u \\ b_x \\ b_x \\ b_x \\ \vdots \\ b_f \end{bmatrix} \tag{10.41}$$

The set $\mathcal{X}_i$ is a polyhedron as it is the projection of the polyhedron $\mathcal{P}_i$ in (10.40)-(10.41) on the x_i space.

In the *recursive approach*,

$$\begin{aligned} \mathcal{X}_i = \{x \in \mathcal{X} : \ &\exists u \in \mathcal{U} \text{ such that } Ax + Bu \in \mathcal{X}_{i+1} \ \}, \\ &i = 0, \dots, N-1 \\ \mathcal{X}_N = \mathcal{X}_f&. \end{aligned} \tag{10.42}$$

The definition of $\mathcal{X}_i$ in (10.42) is recursive and requires that for any feasible initial state $x_i \in \mathcal{X}_i$ there exists a feasible input u_i which keeps the next state $Ax_i + Bu_i$ in the feasible set $\mathcal{X}_{i+1}$. It can be compactly written as

$$\mathcal{X}_i = \text{Pre}(\mathcal{X}_{i+1}) \cap \mathcal{X} \tag{10.43}$$

Initializing $\mathcal{X}_N$ to $\mathcal{X}_f$ and solving (10.42) backward in time yields *the same* sets $\mathcal{X}_i$ as the batch approach. This recursive method, however, leads to an alternative approach for computing the sets $\mathcal{X}_i$. Let $\mathcal{X}_i$ be the $\mathcal{H}$-polyhedra $A_{\mathcal{X}_i} x \leq b_{\mathcal{X}_i}$. Then the set $\mathcal{X}_{i-1}$ is the projection of the following polyhedron

$$\begin{bmatrix} A_u \\ 0 \\ A_{\mathcal{X}_i} B \end{bmatrix} u_i + \begin{bmatrix} 0 \\ A_x \\ A_{\mathcal{X}_i} A \end{bmatrix} x_i \leq \begin{bmatrix} b_u \\ b_x \\ b_{\mathcal{X}_i} \end{bmatrix} \tag{10.44}$$

on the x_i space.

Consider problem (10.32). The set $\mathcal{X}_0$ is the set of all initial states x_0 for which (10.32) is feasible. The sets $\mathcal{X}_i$ with $i = 1, \dots, N-1$ are somehow hidden. A given $\bar{U}_0 = [\bar{u}_0, \ldots, \bar{u}_{N-1}]$ is feasible for problem (10.32) if and only if at all time instants i, the state x_i obtained by applying $\bar{u}_0, \ldots, \bar{u}_{i-1}$ to the system model $x_{k+1} = Ax_k + Bu_k$ with initial state $x_0 \in \mathcal{X}_0$ belongs to $\mathcal{X}_i$. Also, $\mathcal{X}_i$ is the set of feasible initial states for problem

$$\begin{array}{rl} J_i^*(x(0)) = \min_{U_i} & p(x_N) + \sum_{k=i}^{N-1} q(x_k, u_k) \\ \text{subj. to} & x_{k+1} = Ax_k + Bu_k,\ k = i, \ldots, N-1 \\ & x_k \in \mathcal{X},\ u_k \in \mathcal{U},\ k = i, \ldots, N-1 \\ & x_N \in \mathcal{X}_f \end{array} \tag{10.45}$$

Next, we provide more insights into the set $\mathcal{X}_i$ by using the invariant set theory of Section 10.1. We consider two cases: (1) $\mathcal{X}_f = \mathcal{X}$ which corresponds to effectively "remove" the terminal constraint set and (2) $\mathcal{X}_f$ chosen to be a control invariant set.

Theorem 10.2. *[158, Theorem 5.3]. Let the terminal constraint set $\mathcal{X}_f$ be equal to $\mathcal{X}$. Then,*

1. *The feasible set $\mathcal{X}_i$, $i = 0, \ldots, N-1$ is equal to the $(N-i)$-step controllable set:*
$$\mathcal{X}_i = \mathcal{K}_{N-i}(\mathcal{X})$$
2. *The feasible set $\mathcal{X}_i$, $i = 0, \ldots, N-1$ contains the maximal control invariant set:*
$$\mathcal{C}_\infty \subseteq \mathcal{X}_i$$
3. *The feasible set $\mathcal{X}_i$ is control invariant if and only if the maximal control invariant set is finitely determined and $N - i$ is equal to or greater than its determinedness index $\bar{N}$, i.e.*
$$\mathcal{X}_i \subseteq Pre(\mathcal{X}_i) \Leftrightarrow \mathcal{C}_\infty = \mathcal{K}_{N-i}(\mathcal{X}) \quad \text{for all } i \leq N - \bar{N}$$
4. *$\mathcal{X}_i \subseteq \mathcal{X}_j$ if $i < j$ for $i = 0, \ldots, N-1$. The size of the feasible set $\mathcal{X}_i$ stops decreasing (with decreasing i) if and only if the maximal control invariant set is finitely determined and $N-i$ is larger than its determinedness index, i.e.*
$$\mathcal{X}_i \subset \mathcal{X}_j \ \text{if } N - \bar{N} < i < j < N$$
Furthermore,
$$\mathcal{X}_i = \mathcal{C}_\infty \ \text{if } i \leq N - \bar{N}$$

Theorem 10.3. *[158, Theorem 5.4]. Let the terminal constraint set $\mathcal{X}_f$ be a control invariant subset of $\mathcal{X}$. Then,*

1. *The feasible set $\mathcal{X}_i$, $i = 0, \ldots, N-1$ is equal to the $(N-i)$-step stabilizable set:*
$$\mathcal{X}_i = \mathcal{K}_{N-i}(\mathcal{X}_f)$$
2. *The feasible set $\mathcal{X}_i$, $i = 0, \ldots, N-1$ is control invariant and contained within the maximal control invariant set:*

$$\mathcal{X}_i \subseteq \mathcal{C}_\infty$$

3. *$\mathcal{X}_i \supseteq \mathcal{X}_j$ if $i < j$, $i = 0, \ldots, N-1$. The size of the feasible $\mathcal{X}_i$ set stops increasing (with decreasing i) if and only if the maximal stabilizable set is finitely determined and $N - i$ is larger than its determinedness index, i.e.*
$$\mathcal{X}_i \supset \mathcal{X}_j \text{ if } N - \bar{N} < i < j < N$$
Furthermore,
$$\mathcal{X}_i = \mathcal{K}_\infty(\mathcal{X}_f) \text{ if } i \le N - \bar{N}$$

Remark 10.8. Theorems 10.2 and 10.3 help us understand how the feasible sets $\mathcal{X}_i$ propagate backward in time as a function of the terminal set $\mathcal{X}_f$. In particular, when $\mathcal{X}_f = \mathcal{X}$ the set $\mathcal{X}_i$ shrinks as i becomes smaller and stops shrinking when it becomes the maximal control invariant set. Also, depending on i, either it is not a control invariant set or it is the maximal control invariant set. We have the opposite if a control invariant set is chosen as terminal constraint $\mathcal{X}_f$. The set $\mathcal{X}_i$ grows as i becomes smaller and stops growing when it becomes the maximal stabilizable set. Both cases are shown in the Example 10.6 below.

Remark 10.9. In this section we investigated the behavior of $\mathcal{X}_i$ as i varies for a fixed horizon N. Equivalently, we could study the behavior of $\mathcal{X}_0$ as the horizon N varies. Specifically, the sets $\mathcal{X}_{0\to N_1}$ and $\mathcal{X}_{0\to N_2}$ with $N_2 > N_1$ are equal to the sets $\mathcal{X}_{N_2-N_1\to N}$ and $\mathcal{X}_{0\to N}$, respectively, with $N = N_2$.

Example 10.6. Consider the double integrator

$$\begin{cases} x(t+1) = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t) \\ y(t) = \begin{bmatrix} 1 & 0 \end{bmatrix} x(t) \end{cases} \tag{10.46}$$

subject to the input constraints

$$-1 \le u(k) \le 1 \text{ for all } k \ge 0 \tag{10.47}$$

and the state constraints

$$\begin{bmatrix} -5 \\ -5 \end{bmatrix} \le x(k) \le \begin{bmatrix} 5 \\ 5 \end{bmatrix} \text{ for all } k \ge 0 \tag{10.48}$$

We compute the feasible sets $\mathcal{X}_i$ and plot them in Figure 10.7 in two cases.

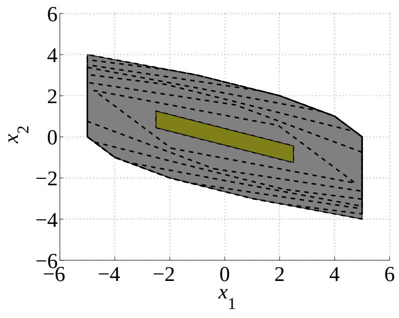
Case 1. $\mathcal{X}_f$ is the control invariant set

$$\begin{bmatrix} -0.32132 & -0.94697 \\ 0.32132 & 0.94697 \\ 1 & 0 \\ -1 & 0 \end{bmatrix} x \le \begin{bmatrix} 0.3806 \\ 0.3806 \\ 2.5 \\ 2.5 \end{bmatrix} \tag{10.49}$$

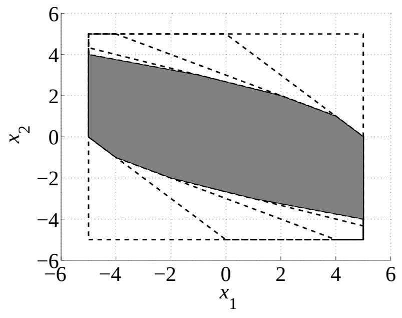
After six iterations the sets $\mathcal{X}_i$ converge to the following $\mathcal{K}_\infty(\mathcal{X}_f)$

$$\begin{bmatrix} -0.44721 & -0.89443 \\ -0.24254 & -0.97014 \\ -0.31623 & -0.94868 \\ 0.24254 & 0.97014 \\ 0.31623 & 0.94868 \\ 0.44721 & 0.89443 \\ 1 & 0 \\ -1 & 0 \\ 0.70711 & 0.70711 \\ -0.70711 & -0.70711 \end{bmatrix} x \leq \begin{bmatrix} 2.6833 \\ 2.6679 \\ 2.5298 \\ 2.6679 \\ 2.5298 \\ 2.6833 \\ 5 \\ 5 \\ 3.5355 \\ 3.5355 \end{bmatrix} \qquad (10.50)$$

Note that in this case $\mathcal{C}_\infty = \mathcal{K}_\infty(\mathcal{X}_f)$ and the determinedness index is six.
Case 2. $\mathcal{X}_f = \mathcal{X}$. After six iterations the sets $\mathcal{X}_i$ converge to $\mathcal{K}_\infty(\mathcal{X}_f)$ in (10.50).



(a) Case 1: $\mathcal{X}_f$ = control invariant set in (10.49)



(b) Case 2: $\mathcal{X}_f = \mathcal{X}$

Fig. 10.7 Example 10.6: Propagation of the feasible sets $\mathcal{X}_i$

10.4 State Feedback Solution, 2-Norm Case

Consider Problem (10.32) with $J_0(\cdot)$ defined by (10.31). In this chapter we always assume that $Q = Q' \succeq 0$, $R = R' \succ 0$, $P = P' \succeq 0$.

$$
\begin{aligned}
J_0^*(x(0)) = \min_{U_0} \quad & J_0(x(0),U_0) = x_N'Px_N + \sum_{k=0}^{N-1} x_k'Qx_k + u_k'Ru_k \\
\text{subj. to } & x_{k+1} = Ax_k + Bu_k,\ k = 0,\ldots,N-1 \\
& x_k \in \mathcal{X},\ u_k \in \mathcal{U},\ k = 0,\ldots,N-1 \\
& x_N \in \mathcal{X}_f \\
& x_0 = x(0)
\end{aligned}
\tag{10.51}
$$

10.4.1 Solution via Batch Approach

As shown in Section 8.1, Problem (10.51) can be rewritten as

$$
\begin{aligned}
J_0^*(x(0)) = \min_{U_0} \quad & J_0(x(0),U_0) = U_0'HU_0 + 2x'(0)FU_0 + x'(0)Yx(0) \\
= \min_{U_0} \quad & J_0(x(0),U_0) = [U_0'\ x'(0)] \left[\begin{smallmatrix} H & F' \\ F & Y \end{smallmatrix}\right] [U_0{}'\ x(0)']' \\
\text{subj. to } & G_0U_0 \le W_0 + E_0x(0)
\end{aligned}
\tag{10.52}
$$

with G_0, W_0 and E_0 defined in (10.41) for $i = 0$ and H,F, Y defined in (8.8). As $J_0(x(0),U_0) \ge 0$ by definition it follows that $\left[\begin{smallmatrix} H & F' \\ F & Y \end{smallmatrix}\right] \succeq 0$. Note that the optimizer U_0^* is independent of the term involving Y in (10.52).

We view $x(0)$ as a vector of parameters and our goal is to solve (10.52) for all values of $x(0) \in \mathcal{X}_0$ and to make this dependence *explicit*. The computation of the set $\mathcal{X}_0$ of initial states for which problem (10.52) is feasible was discussed in Section 10.3.

Before proceeding further, it is convenient to define

$$
z \triangleq U_0 + H^{-1}F'x(0) \tag{10.53}
$$

$z \in \mathbb{R}^s$, remove $x(0)'Yx(0)$ and to transform (10.52) to obtain the equivalent problem

$$
\begin{aligned}
\hat{J}^*(x(0)) = \min_{z} \quad & z'Hz \\
\text{subj. to } & G_0z \le W_0 + S_0x(0),
\end{aligned}
\tag{10.54}
$$

where $S_0 \triangleq E_0 + G_0H^{-1}F'$, and $\hat{J}^*(x(0)) = J_0^*(x(0)) - x(0)'(Y - FH^{-1}F')x(0)$. In the transformed problem the parameter vector $x(0)$ appears only on the rhs of the constraints.

Problem (10.54) is a multiparametric quadratic program that can be solved by using the algorithm described in Section 6.3.1. Once the multiparametric problem (10.54) has been solved, the solution $U_0^* = U_0^*(x(0))$ of CFTOC (10.51) and therefore $u^*(0) = u^*(x(0))$ is available explicitly as a function of the initial state $x(0)$ for all $x(0) \in \mathcal{X}_0$.

Theorem 6.6 states that the solution $z^*(x(0))$ of the mp-QP problem (10.54) is a continuous and piecewise affine function on polyhedra of $x(0)$. Clearly the same properties are inherited by the controller. The following corollaries of Theorem 6.6 establish the analytical properties of the optimal control law and of the value function.

Corollary 10.1. *The control law $u^*(0) = f_0(x(0))$, $f_0 : \mathbb{R}^n \to \mathbb{R}^m$, obtained as a solution of the CFTOC (10.51) is continuous and piecewise affine on polyhedra*

$$f_0(x) = F_0^j x + g_0^j \quad \text{if} \quad x \in CR_0^j, \quad j = 1, \dots, N_0^r \tag{10.55}$$

where the polyhedral sets $CR_0^j = \{x \in \mathbb{R}^n | H_0^j x \leq K_0^j\}$, $j = 1, \dots, N_0^r$ are a partition of the feasible polyhedron $\mathcal{X}_0$.

Proof: From (10.53) $U_0^*(x(0)) = z^*(x(0)) - H^{-1}F'x(0)$. From Theorem 6.6 we know that $z^*(x(0))$, solution of (10.54), is PPWA and continuous. As $U_0^*(x(0))$ is a linear combination of a linear function and a PPWA function, it is PPWA. As $U_0^*(x(0))$ is a linear combination of two continuous functions it is continuous. In particular, these properties hold for the first component $u^*(0)$ of U_0^*. $\square$

Remark 10.10. Note that, as discussed in Remark 6.9, the critical regions defined in (6.4) are in general sets that are neither closed nor open. In Corollary 10.1 the polyhedron CR_0^i describes the closure of a critical region. The function $f_0(x)$ is continuous and therefore it is simpler to use a polyhedral partition rather than a strict polyhedral partition.

Corollary 10.2. *The value function $J_0^*(x(0))$ obtained as solution of the CFTOC (10.51) is convex and piecewise quadratic on polyhedra. Moreover, if the mp-QP problem (10.54) is not degenerate, then the value function $J_0^*(x(0))$ is $C^{(1)}$.*

Proof: By Theorem 6.6 $\hat{J}^*(x(0))$ is a a convex function of $x(0)$. As $\left[\begin{smallmatrix} H & F' \\ F & Y \end{smallmatrix}\right] \succeq 0$, its Schur complement $Y - FH^{-1}F' \succeq 0$, and therefore $J_0^*(x(0)) = \hat{J}^*(x(0)) + x(0)'(Y - FH^{-1}F')x(0)$ is a convex function, because it is the sum of convex functions. If the mp-QP problem (10.54) is not degenerate, then Theorem 6.9 implies that $\hat{J}^*(x(0))$ is a $C^{(1)}$ function of $x(0)$ and therefore $J_0^*(x(0))$ is a $C^{(1)}$ function of $x(0)$. The results of Corollary 10.1 imply that $J_0^*(x(0))$ is piecewise quadratic. $\square$

Remark 10.11. The relation between the design parameters of the optimal control problem (10.51) and the degeneracy of the mp-QP problem (10.54) is complex, in general.

The solution of the multiparametric problem (10.54) provides the state feedback solution $u^*(k) = f_k(x(k))$ of CFTOC (10.51) for $k = 0$ and it also provides the open-loop optimal control laws $u^*(k)$ as function of the initial state, i.e., $u^*(k) = u^*(k, x(0))$. The state feedback PPWA optimal controllers

$u^*(k) = f_k(x(k))$ with $f_k : \mathcal{X}_k \mapsto \mathcal{U}$ for $k = 1, \ldots, N$ are computed in the following way. Consider the same CFTOC (10.51) over the shortened time-horizon $[i, N]$

$$\begin{array}{ll} \min_{U_i} & x_N' P x_N + \sum_{k=i}^{N-1} x_k' Q x_k + u_k' R u_k \\ \text{subj. to} & x_{k+1} = Ax_k + Bu_k,\ k = i, \ldots, N-1 \\ & x_k \in \mathcal{X},\ u_k \in \mathcal{U},\ k = i, \ldots, N-1 \\ & x_N \in \mathcal{X}_f \\ & x_i = x(i) \end{array} \tag{10.56}$$

where $U_i \triangleq [u_i', \ldots, u_{N-1}']$. As defined in (10.39) and discussed in Section 10.3, $\mathcal{X}_i \subseteq \mathbb{R}^n$ is the set of initial states $x(i)$ for which the optimal control problem (10.56) is feasible. We denote by U_i^* the optimizer of the optimal control problem (10.56).

Problem (10.56) can be translated into the mp-QP

$$\begin{array}{ll} \min & U_i{}' H_i U_i + 2x'(i) F_i U_i + x'(i) Y_i x(i) \\ \text{subj. to} & G_i U_i \le W_i + E_i x(i). \end{array} \tag{10.57}$$

where $H_i = H_i' \succ 0$, F_i, Y_i are appropriately defined for each i and G_i, W_i, E_i are defined in (10.41). The first component of the multiparametric solution of (10.57) has the form

$$u_i^*(x(i)) = f_i(x(i)),\ \forall x(i) \in \mathcal{X}_i, \tag{10.58}$$

where the control law $f_i : \mathbb{R}^n \to \mathbb{R}^m$, is continuous and PPWA

$$f_i(x) = F_i^j x + g_i^j \quad \text{if} \quad x \in CR_i^j,\ j = 1, \ldots, N_i^r \tag{10.59}$$

and where the polyhedral sets $CR_i^j = \{x \in \mathbb{R}^n | H_i^j x \le K_i^j\}$, $j = 1, \ldots, N_i^r$ are a partition of the feasible polyhedron $\mathcal{X}_i$. Therefore the feedback solution $u^*(k) = f_k(x(k)),\ k = 0, \ldots, N-1$ of the CFTOC (10.51) is obtained by solving N mp-QP problems of decreasing size. The following corollary summarizes the final result.

Corollary 10.3. *The state-feedback control law $u^*(k) = f_k(x(k))$, $f_k : \mathcal{X}_k \subseteq \mathbb{R}^n \to \mathcal{U} \subseteq \mathbb{R}^m$, obtained as a solution of the CFTOC (10.51) and $k = 0, \ldots, N-1$ is time-varying, continuous and piecewise affine on polyhedra*

$$f_k(x) = F_k^j x + g_k^j \quad \textit{if} \quad x \in CR_k^j, \quad j = 1, \ldots, N_k^r \tag{10.60}$$

where the polyhedral sets $CR_k^j = \{x \in \mathbb{R}^n \ : \ H_k^j x \le K_k^j\}$, $j = 1, \ldots, N_k^r$ are a partition of the feasible polyhedron $\mathcal{X}_k$.

10.4.2 Solution via Recursive Approach

Consider the dynamic programming formulation of the CFTOC (10.51)

$$\begin{array}{rll} J_j^*(x_j) \triangleq \min\limits_{u_j} & x_j'Qx_j + u_j'Ru_j + J_{j+1}^*(Ax_j + Bu_j) & \\ \text{subj. to} & x_j \in \mathcal{X},\ u_j \in \mathcal{U}, & (10.61) \\ & Ax_j + Bu_j \in \mathcal{X}_{j+1} & \end{array}$$

for $j = 0, \ldots, N-1$, with boundary conditions

$$J_N^*(x_N) = x_N' P x_N \quad (10.62)$$

$$\mathcal{X}_N = \mathcal{X}_f, \quad (10.63)$$

where $\mathcal{X}_j$ denotes the set of states x for which the CFTOC (10.51) is feasible at time j (as defined in (10.39)). Note that according to Corollary 10.2, $J_{j+1}^*(Ax_j + Bu_j)$ is piecewise quadratic for $j < N-1$. Therefore (10.61) is not simply an mp-QP and, contrary to the unconstrained case (Section 8.1), the computational advantage of the iterative over the batch approach is not obvious. Nevertheless an algorithm was developed and can be found in Section 15.6.

10.4.3 Infinite Horizon Problem

Assume $Q \succ 0$, $R \succ 0$ and that the constraint sets $\mathcal{X}$ and $\mathcal{U}$ contain the origin in their interior[1]. Consider the following infinite-horizon linear quadratic regulation problem with constraints (CLQR)

$$\begin{array}{rll} J_\infty^*(x(0)) = \min_{u_0,u_1,\ldots} & \sum\limits_{k=0}^{\infty} x_k'Qx_k + u_k'Ru_k & \\ \text{subj. to} & x_{k+1} = Ax_k + Bu_k,\ k = 0, \ldots, \infty & (10.64) \\ & x_k \in \mathcal{X},\ u_k \in \mathcal{U},\ k = 0, \ldots, \infty & \\ & x_0 = x(0) & \end{array}$$

and the set

$$\mathcal{X}_\infty = \{x(0) \in \mathbb{R}^n |\ \text{Problem (10.64) is feasible and } J_\infty^*(x(0)) < +\infty\} \quad (10.65)$$

Because $Q \succ 0$, $R \succ 0$ *any* optimizer u_k^* of problem (10.64) *must* converge to the origin ($u_k^* \to 0$) and so must the state trajectory resulting from the application of u_k^* ($x_k^* \to 0$). Thus the origin $x = 0, u = 0$ must lie in the

[1] As in the unconstrained case, the assumption $Q \succ 0$ can be relaxed by requiring that $(Q^{1/2}, A)$ is an observable pair (Section 8.4)

interior of the constraint set ($\mathcal{X}$,$\mathcal{U}$) (if the origin were not contained in the constraint set then $J^*_\infty(x(0))$ would be infinite). For this reason, the set $\mathcal{X}_\infty$ in (10.65) is the maximal stabilizable set $\mathcal{K}_\infty(\mathcal{O})$ of system (10.24) subject to the constraints (10.25) with $\mathcal{O}$ being the origin $\mathbf{0}$ (Definition 10.8).

If the initial state $x_0 = x(0)$ is sufficiently close to the origin, then the constraints will never become active and the solution of Problem (10.64) will yield the *unconstrained* LQR (8.32). More formally we can define a corresponding invariant set around the origin.

Definition 10.10 (Maximal LQR Invariant Set $\mathcal{O}_\infty^{\mathbf{LQR}}$). Consider the system $x(k+1) = Ax(k)+Bu(k)$. $\mathcal{O}_\infty^{\rm LQR} \subseteq \mathbb{R}^n$ denotes the maximal positively invariant set for the autonomous constrained linear system:

$$x(k+1) = (A + BF_\infty)x(k),\ \ x(k) \in \mathcal{X},\ \ u(k) \in \mathcal{U},\ \ \forall\ k \geq 0$$

where $u(k) = F_\infty x(k)$ is the unconstrained LQR control law (8.32) obtained from the solution of the ARE (8.31).

Therefore, from the previous discussion, there is some finite time $\bar{N}(x_0)$, depending on the initial state x_0, at which the state enters $\mathcal{O}_\infty^{\rm LQR}$ and after which the system evolves in an unconstrained manner ($x_k^* \in \mathcal{X}$, $u_k^* \in \mathcal{U}$, $\forall k > \bar{N}$). This consideration allows us to split Problem (10.64) into two parts by using the dynamic programming principle, one up to time $k = \bar{N}$ where the constraints may be active and one for longer times $k > \bar{N}$ where there are no constraints.

$$\begin{array}{rll} J^*_\infty(x(0)) = & \min_{u_0,u_1,\ldots} & \displaystyle\sum_{k=0}^{\bar{N}-1} x_k'Qx_k + u_k'Ru_k + J^*_{\bar{N}\to\infty}(x_{\bar{N}}) \\ & \text{subj. to} & x_k \in \mathcal{X},\ \ u_k \in \mathcal{U},\ \ k = 0,\ldots,\bar{N}-1 \\ & & x_{k+1} = Ax_k + Bu_k,\ \ k \geq 0 \\ & & x_0 = x(0). \end{array} \tag{10.66}$$

where

$$\begin{array}{rll} J^*_{\bar{N}\to\infty}(x_{\bar{N}}) = & \min_{u_{\bar{N}},u_{\bar{N}+1},\ldots} & \displaystyle\sum_{k=\bar{N}}^{\infty} x_k'Qx_k + u_k'Ru_k \\ & \text{subj. to} & x_{k+1} = Ax_k + Bu_k,\ \ k \geq \bar{N} \\ & = x_{\bar{N}}'P_\infty x_{\bar{N}} & \end{array} \tag{10.67}$$

This key insight due to Sznaier and Damborg [246] is formulated precisely in the following.

Theorem 10.4 (Equality of Finite and Infinite Optimal Control, [233]). *For any given initial state $x(0)$, the solution to (10.66, 10.67) is equal to the infinite-time solution of (10.64), if the terminal state $x_{\bar{N}}$ of (10.66) lies* *in the positive invariant set $\mathcal{O}_\infty^{LQR}$ and no terminal set constraint is applied in (10.66), i.e. the state 'voluntarily' enters the set $\mathcal{O}_\infty^{LQR}$ after $\bar{N}$ steps.*

Theorem 10.4 suggests to obtain the infinite horizon constrained linear quadratic regulator CLQR by solving the finite horizon problem for a horizon of $\bar{N}$ with a terminal weight of $P = P_\infty$ and *no* terminal constraint. The critical question of how to determine $\bar{N}(x_0)$ or at least an upper bound was studied by several researchers. Chmielewski and Manousiouthakis [79] presented an approach that provides a conservative estimate N_{est} of the finite horizon $\bar{N}(x_0)$ for all x_0 belonging to a compact set of initial conditions $\mathcal{S} \subseteq \mathcal{X}_\infty = \mathcal{K}_\infty(\mathbf{0})$ ($N_{\text{est}} \geq \bar{N}_{\mathcal{S}}(x_0),\ \forall x_0 \in \mathcal{S}$). They solve a single, finite dimensional, convex program to obtain N_{est}. Their estimate can be used to compute the PWA solution of (10.66) for a particular set $\mathcal{S}$. Alternatively, the quadratic program with horizon N_{est} can be solved to determine $u_0^*,\ u_1^*, \ldots, u_{N_{\text{est}}}^*$ for a particular $x(0) \in \mathcal{S}$. For a given initial state $x(0)$, rather then a set $\mathcal{S}$, Scokaert and Rawlings [233] presented an algorithm that attempts to identify $\bar{N}(x(0))$ iteratively. In summary we can state the following Theorem.

Theorem 10.5 (Explicit solution of CLQR). *Assume that (A, B) is a stabilizable pair and $(Q^{1/2}, A)$ is an observable pair, $R \succ 0$. The state-feedback solution to the CLQR problem (10.64) in a compact set of the initial conditions $\mathcal{S} \subseteq \mathcal{X}_\infty = \mathcal{K}_\infty(\mathbf{0})$ is time-invariant, continuous and piecewise affine on polyhedra*

$$u^*(k) = f_\infty(x(k)), \quad f_\infty(x) = F^j x + g^j \quad \text{if} \quad x \in CR_\infty^j, \quad j = 1, \ldots, N_\infty^r \tag{10.68}$$

where the polyhedral sets $CR_\infty^j = \{x \in \mathbb{R}^n \ :\ H^j x \leq K^j\}$, $j = 1, \ldots, N_\infty^r$ are a finite partition of the feasible compact polyhedron $\mathcal{S} \subseteq \mathcal{X}_\infty$.

As discussed previously, the complexity of the solution manifested by the number of polyhedral regions depends on the chosen horizon. As the various discussed techniques yield an N_{est} that may be too large by orders of magnitude this is not a viable proposition. An efficient algorithm for computing the PPWA solution to the CLQR problem is presented next.

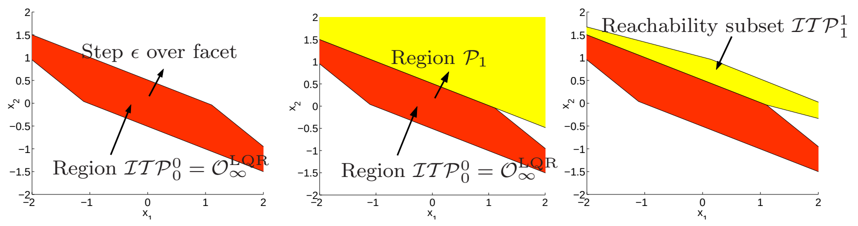
10.4.4 CLQR Algorithm

In this section we will sketch an efficient algorithm to compute the PWA solution to the CLQR problem in (10.64) for a given set $\mathcal{S}$ of initial conditions. Details are available in [123, 124]. As a side product, the algorithm also computes $\bar{N}_{\mathcal{S}}$, the shortest horizon $\bar{N}$ for which the problem (10.66), (10.67) is equivalent to the infinite horizon problem (10.64).

The idea is as follows. For the CFTOC problem (10.51) with a horizon N with no terminal constraint ($\mathcal{X}_f = \mathbb{R}^n$) and terminal cost $P = P_\infty$, where P_∞

is the solution to the ARE (8.31), we solve an mp-QP and obtain the PWA control law. From Theorem 10.4 we can conclude that for all states which enter the invariant set $\mathcal{O}_\infty^{\text{LQR}}$ introduced in Definition 10.10 with the computed control law in N steps, the infinite-horizon problem has been solved. For these states, which we can identify via a reachability analysis, the computed feedback law is infinite-horizon optimal.

In more detail, we start the procedure by computing the Maximal LQR Invariant Set $\mathcal{O}_\infty^{\text{LQR}}$ introduced in Definition 10.10, the polyhedron $\mathcal{P}_0 \triangleq \mathcal{O}_\infty^{\text{LQR}} = \{x \in \mathbb{R}^n | H_0 x \leq K_0\}$. Figure 10.8(a) depicts $\mathcal{O}_\infty^{\text{LQR}}$. Then, the



(a) Compute positive invariant region $\mathcal{O}_\infty^{\text{LQR}}$ and step over facet with step-size ϵ.

(b) Solve QP for new point with horizon $N = 1$ to create the first constrained region $\mathcal{P}_1$.

(c) Compute reachability subset of $\mathcal{P}_1$ to obtain $\mathcal{ITP}_1^1$.

Fig. 10.8 CLQR Algorithm: Region Exploration

algorithm finds a point $\bar{x}$ by stepping over a facet of $\mathcal{O}_\infty^{\text{LQR}}$ with a small step ϵ, as described in [18]. If (10.51) is feasible for horizon $N = 1$ (terminal set constraint $\mathcal{X}_f = \mathbb{R}^n$, terminal cost $P = P_\infty$ and $x(0) = \bar{x}$), the active constraints will define the neighboring polyhedron $\mathcal{P}_1 = \{x \in \mathbb{R}^n | H_1 x \leq K_1\}$ ($\bar{x} \in \mathcal{P}_1$, see Figure 10.8(b)) [39]. By Theorem 10.4, the finite time optimal solution computed above equals the infinite-time optimal solution if $x_1 \in \mathcal{O}_\infty^{\text{LQR}}$. Therefore we extract from $\mathcal{P}_1$ the set of points that will enter $\mathcal{O}_\infty^{\text{LQR}}$ in $N = 1$ time-steps, provided that the optimal control law associated with $\mathcal{P}_1$ (i.e., $U_1^* = F_1 x(0) + G_1$) is applied. The Infinite-Time Polyhedron ($\mathcal{ITP}_1^1$) is therefore defined by the intersection of the following two polyhedra:

$$x_1 \in \mathcal{O}_\infty^{\text{LQR}},\ x_1 = Ax_0 + BU_1^*, \tag{10.69a}$$
$$x_0 \in \mathcal{P}_1 \tag{10.69b}$$

Equation (10.69a) is the reachability constraint and (10.69b) defines the set of states for which the computed feedback law is feasible and optimal over $N = 1$ steps (see [39] for details). The intersection is the set of points for which the control law is infinite-time optimal.

A general step r of the algorithm involves stepping over a facet to a new point $\bar{x}$ and determining the polyhedron $\mathcal{P}_r$ and the associated control law ($U_N^* = F_r x(0) + G_r$) from (10.51) with horizon N. Then we extract from $\mathcal{P}_r$ the set of points that will enter $\mathcal{O}_\infty^{\rm LQR}$ in N time-steps, provided that the optimal control law associated with $\mathcal{P}_r$ is applied. The Infinite-Time Polyhedron ($\mathcal{ITP}_r^N$) is therefore defined by the intersection of the following two polyhedra:

$$x_N \in \mathcal{O}_\infty^{\rm LQR} \tag{10.70a}$$

$$x_0 \in \mathcal{P}_r \tag{10.70b}$$

This intersection is the set of points for which the control law is infinite-time optimal. Note that, as for x_1 in the one-step case, x_N in (10.70a) can be described as a linear function of x_0 by substituting the feedback sequence $U_N^* = F_r x_0 + G_r$ into the LTI system dynamics (10.24).

We continue exploring the facets increasing N when necessary. The algorithm terminates when we have covered $\mathcal{S}$ or when we can no longer find a new feasible polyhedron $\mathcal{P}_r$. The following theorem shows that the algorithm also provides the horizon $\bar{N}_\mathcal{S}$ for compact sets. Exact knowledge of $\bar{N}_\mathcal{S}$ can serve to improve the performance of a wide array of algorithms presented in the literature.

Theorem 10.6 (Exact Computation of $\bar{N}_\mathcal{S}$, [123, 124]). *If we explore any given compact set $\mathcal{S}$ with the proposed algorithm, the largest resulting horizon is equal to $\bar{N}_\mathcal{S}$, i.e.,*

$$\bar{N}_\mathcal{S} = \max_{\mathcal{ITP}_r^N \quad r=0,\ldots,R} N$$

For certain classes of problems the proposed algorithm is more efficient than standard multi-parametric solvers, even if finite horizon optimal controllers are sought. The initial polyhedral representation $\mathcal{P}_r$ contains redundant constraints which need to be removed in order to obtain a minimal representation of the controller region. The intersection with the reachability constraint, as proposed here, can simplify this constraint removal.

10.4.5 Examples

Example 10.7. Consider the double integrator (10.46). We want to compute the state feedback optimal controller that solves problem (10.51) with $N = 6$, $Q = \left[\begin{smallmatrix}1 & 0\\ 0 & 1\end{smallmatrix}\right]$, $R = 0.1$, P is equal to the solution of the Riccati equation (8.31), $\mathcal{X}_f = \mathbb{R}^2$. The input constraints are

$$-1 \le u(k) \le 1, \;\; k = 0, \ldots, 5 \tag{10.71}$$

and the state constraints

$$\begin{bmatrix}-10\\-10\end{bmatrix} \leq x(k) \leq \begin{bmatrix}10\\10\end{bmatrix}, \; k = 0, \ldots, 5 \tag{10.72}$$

This task is addressed as shown in Section (10.4.1). The feedback optimal solution $u^*(0), \ldots, u^*(5)$ is computed by solving six mp-QP problems and the corresponding polyhedral partitions of the state-space are depicted in Fig. 10.9. Only the last two optimal control moves are reported below:

$$u^*(5) = \begin{cases} \begin{bmatrix} -0.58 & -1.55 \end{bmatrix} x(5) & \text{if } \begin{bmatrix} -0.35 & -0.94 \\ 0.35 & 0.94 \\ 1.00 & 0.00 \\ -1.00 & 0.00 \end{bmatrix} x(5) \leq \begin{bmatrix} 0.61 \\ 0.61 \\ 10.00 \\ 10.00 \end{bmatrix} \quad \text{(Region \#1)} \\ 1.00 & \text{if } \begin{bmatrix} 0.35 & 0.94 \\ 0.00 & -1.00 \\ -0.71 & -0.71 \\ 1.00 & 0.00 \\ -1.00 & 0.00 \end{bmatrix} x(5) \leq \begin{bmatrix} -0.61 \\ 10.00 \\ 7.07 \\ 10.00 \\ 10.00 \end{bmatrix} \quad \text{(Region \#2)} \\ -1.00 & \text{if } \begin{bmatrix} -0.35 & -0.94 \\ 1.00 & 0.00 \\ 0.00 & 1.00 \\ -1.00 & 0.00 \\ 0.71 & 0.71 \end{bmatrix} x(5) \leq \begin{bmatrix} -0.61 \\ 10.00 \\ 10.00 \\ 10.00 \\ 7.07 \end{bmatrix} \quad \text{(Region \#3)} \end{cases}$$

$$u^*(4) = \begin{cases}
\begin{bmatrix} -0.58 & -1.55 \end{bmatrix} x(4) & \text{if } \begin{bmatrix} -0.35 & -0.94 \\ 0.35 & 0.94 \\ -0.77 & -0.64 \\ 0.77 & 0.64 \end{bmatrix} x(4) \leq \begin{bmatrix} 0.61 \\ 0.61 \\ 2.43 \\ 2.43 \end{bmatrix} \quad \text{(Region \#1)} \\
1.00 & \text{if } \begin{bmatrix} 0.29 & 0.96 \\ 0.00 & -1.00 \\ -0.71 & -0.71 \\ -0.45 & -0.89 \\ 1.00 & 0.00 \\ -1.00 & 0.00 \end{bmatrix} x(4) \leq \begin{bmatrix} -0.98 \\ 10.00 \\ 7.07 \\ 4.92 \\ 10.00 \\ 10.00 \end{bmatrix} \quad \text{(Region \#2)} \\
1.00 & \text{if } \begin{bmatrix} 0.29 & 0.96 \\ 0.35 & 0.94 \\ 0.00 & -1.00 \\ -0.71 & -0.71 \\ -0.45 & -0.89 \\ 1.00 & 0.00 \\ -1.00 & 0.00 \end{bmatrix} x(4) \leq \begin{bmatrix} -0.37 \\ -0.61 \\ 10.00 \\ 7.07 \\ 4.92 \\ 10.00 \\ 10.00 \end{bmatrix} \quad \text{(Region \#3)} \\
-1.00 & \text{if } \begin{bmatrix} -0.29 & -0.96 \\ 1.00 & 0.00 \\ 0.00 & 1.00 \\ -1.00 & 0.00 \\ 0.71 & 0.71 \\ 0.45 & 0.89 \end{bmatrix} x(4) \leq \begin{bmatrix} -0.98 \\ 10.00 \\ 10.00 \\ 10.00 \\ 7.07 \\ 4.92 \end{bmatrix} \quad \text{(Region \#4)} \\
-1.00 & \text{if } \begin{bmatrix} -0.29 & -0.96 \\ -0.35 & -0.94 \\ 1.00 & 0.00 \\ 0.00 & 1.00 \\ -1.00 & 0.00 \\ 0.71 & 0.71 \\ 0.45 & 0.89 \end{bmatrix} x(4) \leq \begin{bmatrix} -0.37 \\ -0.61 \\ 10.00 \\ 10.00 \\ 10.00 \\ 7.07 \\ 4.92 \end{bmatrix} \quad \text{(Region \#5)} \\
\begin{bmatrix} -0.44 & -1.43 \end{bmatrix} x(4) - 0.46 & \text{if } \begin{bmatrix} -0.29 & -0.96 \\ 0.29 & 0.96 \\ -0.77 & -0.64 \\ 1.00 & 0.00 \end{bmatrix} x(4) \leq \begin{bmatrix} 0.98 \\ 0.37 \\ -2.43 \\ 10.00 \end{bmatrix} \quad \text{(Region \#6)} \\
\begin{bmatrix} -0.44 & -1.43 \end{bmatrix} x(4) + 0.46 & \text{if } \begin{bmatrix} -0.29 & -0.96 \\ 0.29 & 0.96 \\ 0.77 & 0.64 \\ -1.00 & 0.00 \end{bmatrix} x(4) \leq \begin{bmatrix} 0.37 \\ 0.98 \\ -2.43 \\ 10.00 \end{bmatrix} \quad \text{(Region \#7)}
\end{cases}$$

Note that by increasing the horizon N, the control law changes only far away from the origin, the larger N the more in the periphery. This must be expected from the results of Section 10.4.3. The control law does not change anymore with increasing N in the set where the CFTOC law becomes equal to the constrained infinite-horizon linear quadratic regulator (CLQR) problem. This set gets larger as N increases [79, 233].

Example 10.8. The infinite-time CLQR (10.64) was determined for Example 10.7 by using the approach presented in Section 10.4.3. The resulting $\bar{N}_{\mathcal{S}}$ is 12. The state-space is divided into 117 polyhedral regions and is depicted in Fig. 10.10(a). In Fig. 10.10(b) the same control law is represented where polyhedra with the same affine control law were merged.

10.5 State Feedback Solution, 1-Norm and ∞-Norm Case

We are considering problem (10.32) with $J_0(\cdot)$ defined by (10.30) with $p = 1$ or $p = \infty$. In the following section we will concentrate on the ∞-norm, the results can be extended easily to cost functions with 1-norm or mixed $1/\infty$ norms.

$$
\begin{array}{rl}
J_0^*(x(0)) = \min_{U_0} & J_0(x(0), U_0) = \|Px_N\|_p + \sum_{k=0}^{N-1} \|Qx_k\|_p + \|Ru_k\|_p \\
\text{subj. to} & x_{k+1} = Ax_k + Bu_k,\ k = 0, \ldots, N-1 \\
& x_k \in \mathcal{X},\ u_k \in \mathcal{U},\ k = 0, \ldots, N-1 \\
& x_N \in \mathcal{X}_f \\
& x_0 = x(0)
\end{array}
\tag{10.73}
$$

10.5.1 Batch Approach

The optimal control problem (10.73) with $p = \infty$ can be rewritten as a linear program by using the standard approach (see e.g. [72]) presented in Section 9.1. Therefore, problem (10.73) can be reformulated as the following LP problem

$$\begin{aligned}
\min_{z_0} \quad & \varepsilon_0^x + \ldots + \varepsilon_N^x + \varepsilon_0^u + \ldots + \varepsilon_{N-1}^u & (10.74a)\\
\text{subj. to} \quad & -\mathbf{1}_n \varepsilon_k^x \leq \pm Q \left[A^k x_0 + \sum_{j=0}^{k-1} A^j B u_{k-1-j} \right], & (10.74b)\\
& -\mathbf{1}_r \varepsilon_N^x \leq \pm P \left[A^N x_0 + \sum_{j=0}^{N-1} A^j B u_{N-1-j} \right], & (10.74c)\\
& -\mathbf{1}_m \varepsilon_k^u \leq \pm R u_k, & (10.74d)\\
& A^k x_0 + \sum_{j=0}^{k-1} A^j B u_{k-1-j} \in \mathcal{X},\ u_k \in \mathcal{U}, & (10.74e)\\
& A^N x_0 + \sum_{j=0}^{N-1} A^j B u_{N-1-j} \in \mathcal{X}_f, & (10.74f)\\
& k = 0, \ldots, N-1 & \\
& x_0 = x(0) & (10.74g)
\end{aligned}$$

where constraints (10.74b)–(10.74f) are componentwise, and $\pm$ means that the constraint appears once with each sign. Problem (10.74) can be rewritten in the more compact form

$$\begin{aligned}
\min_{z_0} \quad & c_0' z_0 \\
\text{subj. to} \quad & \bar{G}_0 z_0 \leq \bar{W}_0 + \bar{S}_0 x(0)
\end{aligned} \qquad (10.75)$$

where $z_0 \triangleq \{\varepsilon_0^x, \ldots, \varepsilon_N^x, \varepsilon_0^u, \ldots, \varepsilon_{N-1}^u, u_0', \ldots, u_{N-1}'\} \in \mathbb{R}^s$, $s \triangleq (m+1)N + N + 1$ and

$$\bar{G}_0 = \begin{bmatrix} G_\varepsilon & 0 \\ 0 & G_0 \end{bmatrix},\ \bar{S}_0 = \begin{bmatrix} S_\varepsilon \\ S_0 \end{bmatrix},\ \bar{W}_0 = \begin{bmatrix} W_\varepsilon \\ W_0 \end{bmatrix} \qquad (10.76)$$

Vector c_0 and the submatrices G_ϵ, W_ϵ, S_ϵ associated with the constraints (10.74b)-(10.74d) are defined in (9.10). The matrices G_0, W_0 and E_0 are defined in (10.41) for $i = 0$.
As in the 2-norm case, by treating $x(0)$ as a vector of parameters, Problem (10.75) becomes a *multiparametric linear program* (mp-LP) that can be solved as described in Section 6.2. Once the multiparametric problem (10.74) has been solved, the explicit solution $z_0^*(x(0))$ of (10.75) is available as a piecewise affine function of $x(0)$, and the optimal control law $u^*(0)$ is also available explicitly, as the optimal input $u^*(0)$ consists simply of m components of $z_0^*(x(0))$

$$u^*(0) = [0 \ \ldots 0 \ I_m \ 0 \ \ldots \ 0] z_0^*(x(0)). \qquad (10.77)$$

Theorem 6.4 states that there always exists a continuous PPWA solution $z_0^*(x)$ of the mp-LP problem (10.75). Clearly the same properties are inherited by the controller. The following Corollaries of Theorem 6.4 summarize the analytical properties of the optimal control law and of the value function.

Corollary 10.4. *There exists a control law $u^*(0) = f_0(x(0))$, $f_0 : \mathbb{R}^n \to \mathbb{R}^m$, obtained as a solution of the CFTOC (10.73) with $p = 1$ or $p = \infty$, which is continuous and PPWA*

$$f_0(x) = F_0^j x + g_0^j \quad \text{if} \quad x \in CR_0^j, \ j = 1, \dots, N_0^r \tag{10.78}$$

where the polyhedral sets $CR_0^j \triangleq \{H_0^j x \le k_0^j\}$, $j = 1, \dots, N_0^r$, are a partition of the feasible set $\mathcal{X}_0$.

Corollary 10.5. *The value function $J^*(x)$ obtained as a solution of the CFTOC (10.73) is convex and PPWA.*

Remark 10.12. Note that if the optimizer of problem (10.73) is unique for all $x(0) \in \mathcal{X}_0$, then Corollary 10.4 reads: “ **The** control law $u^*(0) = f_0(x(0))$, $f_0 : \mathbb{R}^n \to \mathbb{R}^m$, obtained as a solution of the CFTOC (10.73) with $p = 1$ or $p = \infty$, **is** continuous and PPWA,...”. From the results of Section 6.2 we know that in case of multiple optimizers for some $x(0) \in \mathcal{X}_0$, a control law of the form (10.78) can always be computed.

The multiparametric solution of (10.75) provides the open-loop optimal sequence $u^*(0), \dots, u^*(N-1)$ as an affine function of the initial state $x(0)$. The state feedback PPWA optimal controllers $u^*(k) = f_k(x(k))$ with $f_k : \mathcal{X}_k \mapsto \mathcal{U}$ for $k = 1, \dots, N$ are computed in the following way. Consider the same CFTOC (10.73) over the shortened time horizon $[i, N]$

$$\begin{array}{ll} \min_{U_i} & \|Px_N\|_p + \sum_{k=i}^{N-1} \|Qx_k\|_p + \|Ru_k\|_p \\ \text{subj. to} & x_{k+1} = Ax_k + Bu_k, \ k = i, \dots, N-1 \\ & x_k \in \mathcal{X}, \ u_k \in \mathcal{U}, \ k = i, \dots, N-1 \\ & x_N \in \mathcal{X}_f \\ & x_i = x(i) \end{array} \tag{10.79}$$

where $U_i \triangleq [u_i', \dots, u_{N-1}']$ and $p = 1$ or $p = \infty$. As defined in (10.39) and discussed in Section 10.3, $\mathcal{X}_i \subseteq \mathbb{R}^n$ is the set of initial states $x(i)$ for which the optimal control problem (10.79) is feasible. We denote by U_i^* one of the optimizers of the optimal control problem (10.79).

Problem (10.79) can be translated into the mp-LP

$$\begin{array}{ll} \min\limits_{z_i} & c_i' z_i \\ \text{subj. to} & \bar{G}_i z_i \le \bar{W}_i + \bar{S}_i x(i) \end{array} \tag{10.80}$$

where $z_i \triangleq \{\varepsilon_i^x, \ldots, \varepsilon_N^x, \varepsilon_i^u, \ldots, \varepsilon_{N-1}^u, u_i, \ldots, u_{N-1}\}$ and c_i, $\bar{G}_i$, $\bar{S}_i$, $\bar{W}_i$, are appropriately defined for each i. The component u_i^* of the multiparametric solution of (10.80) has the form

$$u_i^*(x(i)) = f_i(x(i)),\ \forall x(i) \in \mathcal{X}_i, \tag{10.81}$$

where the control law $f_i : \mathbb{R}^n \to \mathbb{R}^m$, is continuous and PPWA

$$f_i(x) = F_i^j x + g_i^j \quad \text{if} \quad x \in CR_i^j,\ j = 1, \ldots, N_i^r \tag{10.82}$$

and where the polyhedral sets $CR_i^j = \{x \in \mathbb{R}^n \ :\ H_i^j x \leq K_i^j\}$, $j = 1, \ldots, N_i^r$ are a partition of the feasible polyhedron $\mathcal{X}_i$. Therefore the feedback solution $u^*(k) = f_k(x(k)),\ k = 0, \ldots, N-1$ of the CFTOC (10.73) with $p = 1$ or $p = \infty$ is obtained by solving N mp-LP problems of decreasing size. The following corollary summarizes the final result.

Corollary 10.6. *There exists a state-feedback control law $u^*(k) = f_k(x(k))$, $f_k : \mathcal{X}_k \subseteq \mathbb{R}^n \to \mathcal{U} \subseteq \mathbb{R}^m$, solution of the CFTOC (10.73) for $p = 1$ or $p = \infty$ and $k = 0, \ldots, N-1$ which is time-varying, continuous and piecewise affine on polyhedra*

$$f_k(x) = F_k^j x + g_k^j \quad \textit{if} \quad x \in CR_k^j, \quad j = 1, \ldots, N_k^r \tag{10.83}$$

where the polyhedral sets $CR_k^j = \{x \in \mathbb{R}^n \ :\ H_k^j x \leq K_k^j\}$, $j = 1, \ldots, N_k^r$ are a partition of the feasible polyhedron $\mathcal{X}_k$.

10.5.2 Recursive Approach

Consider the dynamic programming formulation of (10.73) with $J_0(\cdot)$ defined by (10.30) with $p = 1$ or $p = \infty$

$$\begin{array}{rll} J_j^*(x_j) \triangleq & \min_{u_j} & \|Qx_j\|_p + \|Ru_j\|_p + J_{j+1}^*(Ax_j + Bu_j) \\ & \text{subj. to} & x_j \in \mathcal{X},\ u_j \in \mathcal{U}, \\ & & Ax_j + Bu_j \in \mathcal{X}_{j+1} \end{array} \tag{10.84}$$

for $j = 0, \ldots, N-1$, with boundary conditions

$$J_N^*(x_N) = \|Px_N\|_p \tag{10.85}$$

$$\mathcal{X}_N = \mathcal{X}_f, \tag{10.86}$$

Unlike for the 2-norm case the dynamic program (10.84)-(10.86) can be solved nicely and efficiently as explained in the next theorem.

Theorem 10.7. *The state feedback piecewise affine solution (10.83) of the CFTOC (10.73) for $p = 1$ or $p = \infty$ is obtained by solving the equations (10.84)-(10.86) via N mp-LPs.*

Proof: Consider the first step $j = N-1$ of dynamic programming (10.84)-(10.86)

$$\begin{array}{rll} J^*_{N-1}(x_{N-1}) \triangleq & \min_{u_{N-1}} & \|Qx_{N-1}\|_p + \|Ru_{N-1}\|_p + J^*_N(Ax_{N-1} + Bu_{N-1}) \\ & \text{subj. to} & x_{N-1} \in \mathcal{X},\ u_{N-1} \in \mathcal{U}, \\ & & Ax_{N-1} + Bu_{N-1} \in \mathcal{X}_f \end{array} \tag{10.87}$$

$J^*_{N-1}(x_{N-1})$, $u^*_{N-1}(x_{N-1})$ and $\mathcal{X}_{N-1}$ are computable via the mp-LP:

$$\begin{array}{rll} J^*_{N-1}(x_{N-1}) \triangleq & \min_{\mu, u_{N-1}} & \mu \\ & \text{subj. to} & \mu \geq \|Qx_{N-1}\|_p + \|Ru_{N-1}\|_p + \|P(Ax_{N-1} + Bu_{N-1})\|_p \\ & & x_{N-1} \in \mathcal{X},\ u_{N-1} \in \mathcal{U}, \\ & & Ax_{N-1} + Bu_{N-1} \in \mathcal{X}_f \end{array} \tag{10.88}$$

By Theorem 6.4, J^*_{N-1} is a convex and piecewise affine function of x_{N-1}, the corresponding optimizer u^*_{N-1} is piecewise affine and continuous, and the feasible set $\mathcal{X}_{N-1}$ is a polyhedron. Without any loss of generality we assume J^*_{N-1} to be described as follows: $J^*_{N-1}(x_{N-1}) = \max_{i=1,\ldots,n_{N-1}}\{c_i x_{N-1} + d_i\}$ (see Section 4.1.5 for convex PPWA functions representation) where n_{N-1} is the number of affine components comprising the value function J^*_{N-1}. At step $j = N-2$ of dynamic programming (10.84)-(10.86) we have

$$\begin{array}{rll} J^*_{N-2}(x_{N-2}) \triangleq & \min_{u_{N-2}} & \|Qx_{N-2}\|_p + \|Ru_{N-2}\|_p + J^*_{N-1}(Ax_{N-2} + Bu_{N-2}) \\ & \text{subj. to} & x_{N-2} \in \mathcal{X},\ u_{N-2} \in \mathcal{U}, \\ & & Ax_{N-2} + Bu_{N-2} \in \mathcal{X}_{N-1} \end{array} \tag{10.89}$$

Since $J^*_{N-1}(x)$ is a convex and piecewise affine function of x, the problem (10.89) can be recast as the following mp-LP (see Section 4.1.5 for details)

$$\begin{array}{rll} J^*_{N-2}(x_{N-2}) \triangleq & \min_{\mu, u_{N-2}} & \mu \\ & \text{subj. to} & \mu \geq \|Qx_{N-2}\|_p + \|Ru_{N-2}\|_p + c_i(Ax_{N-2} + Bu_{N-2}) + d_i, \\ & & i = 1, \ldots, n_{N-1}, \\ & & x_{N-2} \in \mathcal{X},\ u_{N-2} \in \mathcal{U}, \\ & & Ax_{N-2} + Bu_{N-2} \in \mathcal{X}_{N-1} \end{array} \tag{10.90}$$

$J^*_{N-2}(x_{N-2})$, $u^*_{N-2}(x_{N-2})$ and $\mathcal{X}_{N-2}$ are computed by solving the mp-LP (10.90). By Theorem 6.4, J^*_{N-2} is a convex and piecewise affine function of x_{N-2}, the corresponding optimizer u^*_{N-2} is piecewise affine and continuous, and the feasible set $\mathcal{X}_{N-2}$ is a convex polyhedron.

The convexity and piecewise linearity of J^*_j and the polyhedra representation of $\mathcal{X}_j$ still hold for $j = N-3, \ldots, 0$ and the procedure can be iterated backwards in time, proving the theorem. $\square$

Consider the state feedback piecewise affine solution (10.83) of the CFTOC (10.73) for $p = 1$ or $p = \infty$ and assume we are interested only in the optimal controller at time 0. In this case, by using duality arguments we can solve the equations (10.84)-(10.86) by using vertex enumerations and one mp-LP. This is proven in the next theorem.

Theorem 10.8. *The state feedback piecewise affine solution (10.83) at time $k = 0$ of the CFTOC (10.73) for $p = 1$ or $p = \infty$ is obtained by solving the equations (10.84)-(10.86) via one mp-LP.*

Proof: Consider the first step $j = N-1$ of dynamic programming (10.84)-(10.86)

$$\begin{array}{rl} J^*_{N-1}(x_{N-1}) \triangleq \min_{u_{N-1}} & \|Qx_{N-1}\|_p + \|Ru_{N-1}\|_p + J^*_N(Ax_{N-1}+Bu_{N-1}) \\ \text{subj. to} & x_{N-1} \in \mathcal{X},\ u_{N-1} \in \mathcal{U}, \\ & Ax_{N-1}+Bu_{N-1} \in \mathcal{X}_f \end{array} \tag{10.91}$$

and the corresponding mp-LP:

$$\begin{array}{rl} J^*_{N-1}(x_{N-1}) \triangleq \min_{\mu,u_{N-1}} & \mu \\ \text{subj. to} & \mu \geq \|Qx_{N-1}\|_p + \|Ru_{N-1}\|_p + \|P(Ax_{N-1}+Bu_{N-1})\|_p \\ & x_{N-1} \in \mathcal{X},\ u_{N-1} \in \mathcal{U}, \\ & Ax_{N-1}+Bu_{N-1} \in \mathcal{X}_f \end{array} \tag{10.92}$$

By Theorem 6.4, J^*_{N-1} is a convex and piecewise affine function of x_{N-1}, and the feasible set $\mathcal{X}_{N-1}$ is a polyhedron. J^*_{N-1} and $\mathcal{X}_{N-1}$ are computed without explicitely solving the mp-LP (10.92). Rewrite problem (10.92) in the more compact form

$$\begin{array}{rl} \min\limits_{z_{N-1}} & c'_{N-1}z_{N-1} \\ \text{subj. to} & \bar{G}_{N-1}z_{N-1} \leq \bar{W}_{N-1} + \bar{S}_{N-1}x_{N-1} \end{array} \tag{10.93}$$

where $z_{N-1} = [\mu,\ u_{N-1}]$. Consider the Linear Program dual of (10.93)

$$\begin{array}{rl} \max_v & -(\bar{W}_{N-1} + \bar{S}_{N-1}x_{N-1})'v \\ \text{subj. to} & \bar{G}'_{N-1}v = -c_{N-1} \\ & v \geq 0 \end{array} \tag{10.94}$$

Consider the dual feasibility polyheron $\mathcal{P}_d = \{v \geq 0\ :\ \bar{G}'_{N-1}v = -c_{N-1}\}$. Let $\{V_1, \ldots, V_k\}$ be the vertices of $\mathcal{P}_d$ and $\{y_1, \ldots, y_e\}$ be the rays of $\mathcal{P}_d$. Since in we have zero duality gap we have that

$$J^*_{N-1}(x_{N-1}) = \max_{i=1,\ldots,k}\{-(\bar{W}_{N-1} + \bar{S}_{N-1}x_{N-1})'V_i\}$$

i.e.,

$$J^*_{N-1}(x_{N-1}) = \max_{i=1,\ldots,k}\{-(V_i'\bar{S}_{N-1})x_{N-1} - \bar{W}'_{N-1}V_i\}$$

Recall that if the dual of a mp-LP is unbounded, then the primal is infeasible (Theorem 6.2). For this reason the feasible set $\mathcal{X}_{N-1}$ is obtained by requiring that the cost of (10.94) does not decrease in the direction of the rays:

$$\mathcal{X}_{N-1} = \{x_{N-1} \ : \ -(\bar{W}_{N-1} + \bar{S}_{N-1}x_{N-1})'y_i \le 0, \ \forall \ i = 1, \dots, e\}$$

At this point the proof follows the one of Theorem 10.8 by iterating the procedure backwards in time. One mp-LP will be required at step 0 in order to compute $u_0^*(x(0))$. $\square$

10.5.3 Example

Example 10.9. Consider the double integrator system (10.46). We want to compute the state feedback optimal controller that solves (10.73) with $p = \infty$, $N = 4$, $P = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$, $Q = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$, $R = 0.8$, subject to the input constraints

$$\mathcal{U} = \{u \in \mathbb{R} \ : \ -1 \le u \le 1\} \tag{10.95}$$

and the state constraints

$$\mathcal{X} = \{x \in \mathbb{R}^2 \ : \ \begin{bmatrix} -10 \\ -10 \end{bmatrix} \le x \le \begin{bmatrix} 10 \\ 10 \end{bmatrix}\} \tag{10.96}$$

and $\mathcal{X}_f = \mathcal{X}$. The optimal feedback solution $u^*(0), \dots, u^*(3)$ was computed by solving four mp-LP problems and the corresponding polyhedral partitions of the state-space are depicted in Fig. 10.11, where polyhedra with the same control law were merged. Only the last optimal control move is reported below:

$$
u^*(5) = \begin{cases}
0 & \text{if } \begin{bmatrix} 0.45 & 0.89 \\ 1.00 & 0.00 \\ -0.71 & -0.71 \\ -1.00 & -0.00 \end{bmatrix} x(5) \le \begin{bmatrix} 0.00 \\ 0.00 \\ 7.07 \\ 10.00 \end{bmatrix} & \text{(Region \#1)} \\
0 & \text{if } \begin{bmatrix} -0.45 & -0.89 \\ -1.00 & 0.00 \\ 0.71 & 0.71 \\ 1.00 & -0.00 \end{bmatrix} x(5) \le \begin{bmatrix} 0.00 \\ 0.00 \\ 7.07 \\ 10.00 \end{bmatrix} & \text{(Region \#2)} \\
\begin{bmatrix} -1.00 & -2.00 \end{bmatrix} x(5) & \text{if } \begin{bmatrix} -0.45 & -0.89 \\ 0.45 & 0.89 \\ 0.71 & 0.71 \\ -1.00 & -0.00 \end{bmatrix} x(5) \le \begin{bmatrix} 0.00 \\ 0.45 \\ 0.00 \\ 10.00 \end{bmatrix} & \text{(Region \#3)} \\
\begin{bmatrix} -1.00 & -2.00 \end{bmatrix} x(5) & \text{if } \begin{bmatrix} -0.45 & -0.89 \\ 0.45 & 0.89 \\ -0.71 & -0.71 \\ 1.00 & -0.00 \end{bmatrix} x(5) \le \begin{bmatrix} 0.45 \\ 0.00 \\ 0.00 \\ 10.00 \end{bmatrix} & \text{(Region \#4)} \\
\begin{bmatrix} 1.00 & 0.00 \end{bmatrix} x(5) & \text{if } \begin{bmatrix} -0.71 & -0.71 \\ -1.00 & 0.00 \\ 1.00 & 0.00 \\ -0.00 & 1.00 \end{bmatrix} x(5) \le \begin{bmatrix} 0.00 \\ 1.00 \\ 0.00 \\ 10.00 \end{bmatrix} & \text{(Region \#5)} \\
\begin{bmatrix} 1.00 & 0.00 \end{bmatrix} x(5) & \text{if } \begin{bmatrix} 0.71 & 0.71 \\ -1.00 & 0.00 \\ 1.00 & 0.00 \\ -0.00 & -1.00 \end{bmatrix} x(5) \le \begin{bmatrix} 0.00 \\ 0.00 \\ 1.00 \\ 10.00 \end{bmatrix} & \text{(Region \#6)} \\
1.00 & \text{if } \begin{bmatrix} 0.45 & 0.89 \\ -1.00 & 0.00 \\ -0.00 & -1.00 \\ 1.00 & -0.00 \end{bmatrix} x(5) \le \begin{bmatrix} -0.45 \\ -1.00 \\ 10.00 \\ 10.00 \end{bmatrix} & \text{(Region \#7)} \\
-1.00 & \text{if } \begin{bmatrix} -0.45 & -0.89 \\ 1.00 & 0.00 \\ -1.00 & -0.00 \\ -0.00 & 1.00 \end{bmatrix} x(5) \le \begin{bmatrix} -0.45 \\ -1.00 \\ 10.00 \\ 10.00 \end{bmatrix} & \text{(Region \#8)}
\end{cases}
\tag{10.97}
$$

Note that the controller (10.97) is piecewise linear around the origin. In fact, the origin belongs to multiple regions (1 to 6). Note that the number N_i^r of regions is not always increasing with decreasing i ($N_5^r = 8$, $N_4^r = 12$, $N_3^r = 12$, $N_2^r = 26$, $N_1^r = 28$, $N_0^r = 26$). This is due to the merging procedure, before merging we have $N_5^r = 12$, $N_4^r = 22$, $N_3^r = 40$, $N_2^r = 272$, $N_1^r = 108$, $N_0^r = 152$.

10.5.4 Infinite-Time Solution

Assume that Q and R have full column rank and that the constraint sets $\mathcal{X}$ and $\mathcal{U}$ contain the origin in their interior. Consider the following infinite-horizon problem with constraints

$$\begin{array}{rll} J_\infty^*(x(0)) = \min_{u_0,u_1,\ldots} & \sum_{k=0}^{\infty} \|Qx_k\|_p + \|Ru_k\|_p & \\ \text{subj. to} & x_{k+1} = Ax_k + Bu_k,\ k = 0,\ldots,\infty & \\ & x_k \in \mathcal{X},\ u_k \in \mathcal{U},\ k = 0,\ldots,\infty & \\ & x_0 = x(0) & \end{array} \tag{10.98}$$

and the set

$$\mathcal{X}_\infty = \{x(0) \in \mathbb{R}^n | \text{ Problem (10.98) is feasible and } J_\infty^*(x(0)) < +\infty\}. \tag{10.99}$$

Because Q and R have full column rank, *any* optimizer u_k^* of problem (10.64) *must* converge to the origin ($u_k^* \to 0$) and so must the state trajectory resulting from the application of u_k^* ($x_k^* \to 0$). Thus the origin $x = 0, u = 0$ must lie in the interior of the constraint set ($\mathcal{X}$,$\mathcal{U}$). (If the origin were not contained in the constraint set then $J_\infty^*(x(0))$ would be infinite.) Furthermore, if the initial state $x_0 = x(0)$ is sufficiently close to the origin, then the state and input constraints will never become active and the solution of Problem (10.64) will yield the *unconstrained* optimal controller (9.31).

The discussion for the solution of the infinite horizon constrained linear quadratic regulator (Section 10.4.3) by means of the batch approach can be repeated here with one precaution. Since the unconstrained optimal controller (if it exists) is PPWA the computation of the Maximal Invariant Set for the autonomous constrained piecewise linear system is more involved and requires algorithms which will be presented later in Chapter 15.

Differently from the 2-norm case, here the use of dynamic programming for computing the infinite horizon solution is a viable alternative to the batch approach. Convergence conditions of the dynamic programming strategy and stability guarantees for the resulting possibly discontinuous closed-loop system are given in [81]. A computationally efficient algorithm to obtain the infinite time optimal solution, based on a dynamic programming exploration strategy with a multi-parametric linear programming solver and basic polyhedral manipulations, is also presented in [81].

10.6 State-Feedback Solution, Minimum-Time Control

In this section we consider the solution of minimum-time optimal control problems

$$\begin{array}{rll} J_0^*(x(0)) = \min\limits_{U_0,N} & N & \\ \text{subj. to} & x_{k+1} = Ax_k + Bu_k,\ k = 0,\ldots,N-1 & \\ & x_k \in \mathcal{X},\ u_k \in \mathcal{U},\ k = 0,\ldots,N-1 & \\ & x_N \in \mathcal{X}_f & \\ & x_0 = x(0) & \end{array} \tag{10.100}$$

where $\mathcal{X}_f \subset \mathbb{R}^n$ is a terminal target set to be reached in minimum time.

We can find the controller that brings the states into $\mathcal{X}_f$ in one time step by solving the following multiparametric program

$$\begin{array}{ll}\min\limits_{u_0} & c(x_0, u_0) \\ \text{subj. to} & x_1 = Ax_0 + Bu_0 \\ & x_0 \in \mathcal{X},\ u_0 \in \mathcal{U} \\ & x_1 \in \mathcal{X}_f \end{array} \tag{10.101}$$

where $c(x_0, u_0)$ is any convex quadratic function. Let us assume that the solution of the multiparametric program generates R^1 regions $\{\mathcal{P}_r^1\}_{r=1}^{R^1}$ with the affine control law $u_0 = F_r^1 x + G_r^1$ in each region r. By construction we have

$$\mathcal{X}_0 = \mathcal{K}_1(\mathcal{X}_f)$$

Continuing setting up simple multiparametric programs bring the states into $\mathcal{X}_f$ in $2, 3, \ldots$ steps, we have for step j

$$\begin{array}{ll}\min\limits_{u_0} & c(x_0, u_0) \\ \text{subj. to} & x_1 = Ax_0 + Bu_0 \\ & x_0 \in \mathcal{X},\ u_0 \in \mathcal{U} \\ & x_1 \in \mathcal{K}_{j-1}(\mathcal{X}_f) \end{array} \tag{10.102}$$

which yields R^j regions $\{\mathcal{P}_r^j\}_{r=1}^{R^j}$ with the affine control law $u_0 = F_r^j x + G_r^j$ in each region r. By construction we have

$$\mathcal{X}_0 = \mathcal{K}_j(\mathcal{X}_f)$$

Thus to obtain $\mathcal{K}_1(\mathcal{X}_f), \ldots, \mathcal{K}_N(\mathcal{X}_f)$ we need to solve N multiparametric programs with a prediction horizon of 1. Since the overall complexity of a multiparametric program is exponential in N, this scheme can be exploited to yield controllers of lower complexity than the optimal control scheme introduced in the previous sections.

Since N multiparametric programs have been solved, the controller regions overlap in general. In order to achieve minimum time behavior, the feedback law associated with the region computed for the smallest number of steps c, is selected for any given state x.

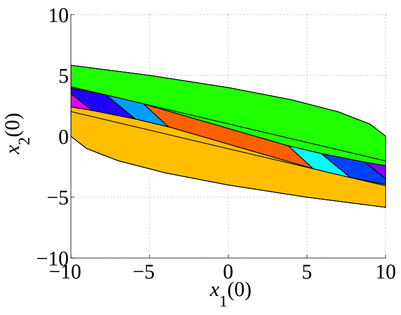
Algorithm 10.4 (Minimum-Time Controller: On-Line Application)

```
1. Obtain state measurement $x$.
2. Find controller partition $c_{min} = \min_{c \in \{0,\ldots,N\}}\ c,\quad s.t.\ x \in \mathcal{K}_c(\mathcal{X}_f)$.
3. Find controller region $r$, such that $x \in \mathcal{P}_r^{c_{min}}$ and compute $u_0 = F_r^{c_{min}} x + G_r^{c_{min}}$.
4. Apply input $u_0$ to system and go to Step 1.
```

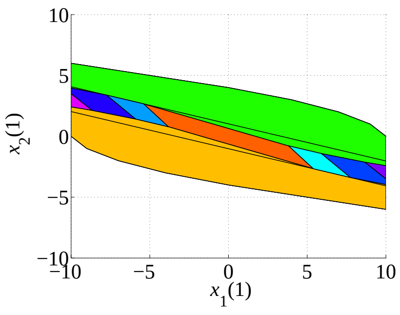
Note that the region identification for this type of controller partition is much more efficient than simply checking all the regions. Steps 2 and 3 in Algorithm 10.4 correspond to two levels of a search tree, where the search is first performed over the feasible sets $\mathcal{K}_c(\mathcal{X}_f)$ and then over the controller partition $\{\mathcal{P}_r^c\}_{r=1}^{R^c}$. Furthermore, one may discard all regions $\mathcal{P}_r^i$ which are completely covered by previously computed controllers (i.e. $\mathcal{P}_r^i \subseteq \bigcup_{j\in\{1,\dots,i-1\}} \mathcal{K}_j(\mathcal{X}_f)$) since they are not time optimal.

Example 10.10. Consider again the double integrator from Example 10.6. The Minimum-Time Controller is computed which steers the system to the Maximal LQR Invariant Set $\mathcal{O}_\infty^{\mathrm{LQR}}$ in the minimum number of time steps N. The Algorithm terminated after 11 iterations, covering the Maximal Controllable Set $\mathcal{K}_\infty(\mathcal{O}_\infty^{\mathrm{LQR}})$. The resulting controller is defined over 33 regions. The regions are depicted in Fig. 10.12(a). The Maximal LQR Invariant Set $\mathcal{O}_\infty^{\mathrm{LQR}}$ is the gray region. The color of each facet corresponds to the controller partition and thus to the number of steps needed to reach $\mathcal{O}_\infty^{\mathrm{LQR}}$.

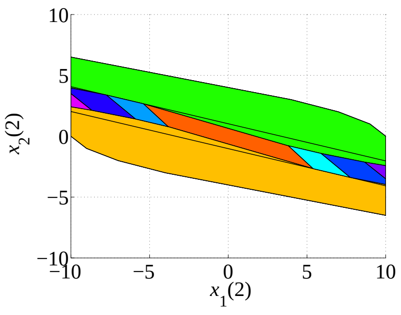
The control law on this partition is depicted in Fig. 10.12(b). Note that, in general, the minimum-time control law is not continuous as can be seen in Fig. 10.12(b).



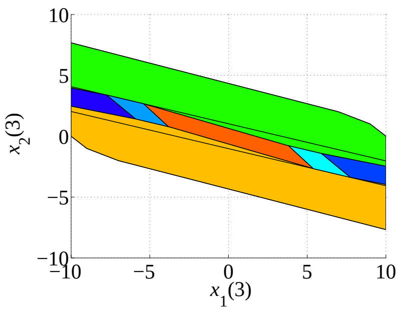
(a) Partition of the state space for the affine control law $u^*(0)$ ($N_0^r = 13$)



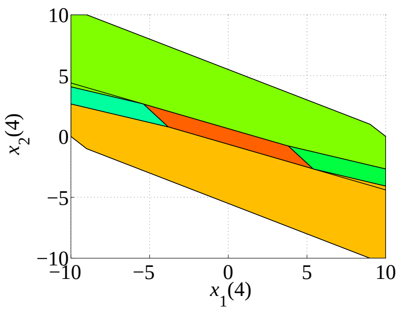
(b) Partition of the state space for the affine control law $u^*(1)$ ($N_1^r = 13$)



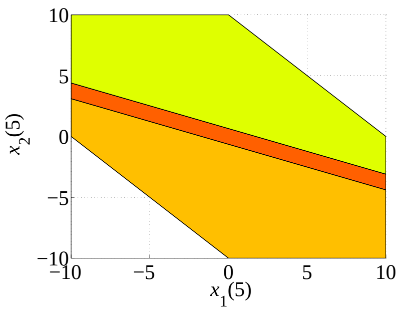
(c) Partition of the state space for the affine control law $u^*(2)$ ($N_2^r = 13$)



(d) Partition of the state space for the affine control law $u^*(3)$ ($N_3^r = 11$)

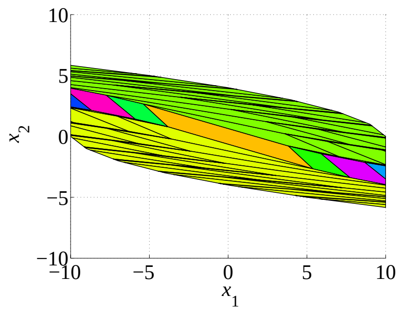


(e) Partition of the state space for the affine control law $u^*(4)$ ($N_4^r = 7$)

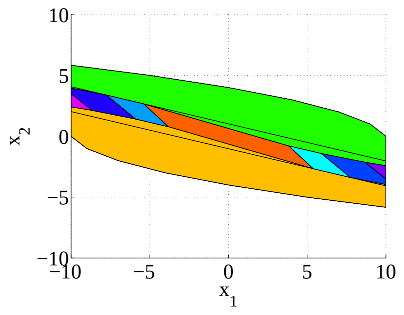


(f) Partition of the state space for the affine control law $u^*(5)$ ($N_5^r = 3$)

Fig. 10.9 Example 10.7: Partition of the state space for optimal control law. Polyhedra with the same control law were merged.

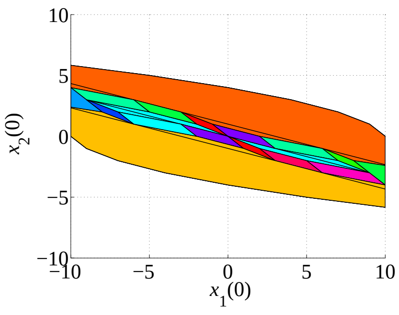


(a) Partition before merging ($N^r_\infty = 117$)

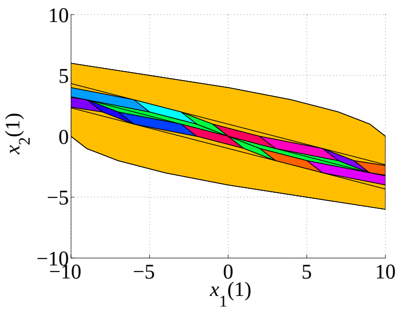


(b) Partition after merging ($N^r_\infty = 13$)

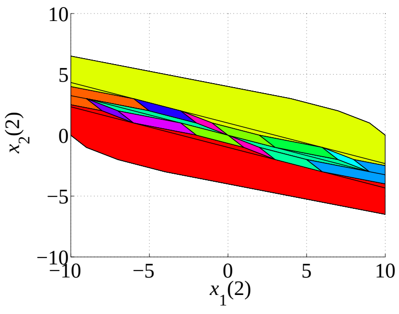
Fig. 10.10 Example 10.8: Partition of the state space for infinite time optimal control law



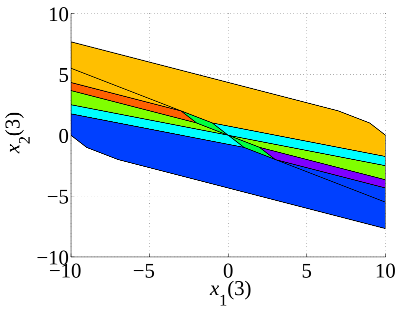
(a) Partition of the state space for the affine control law $u^*(0)(N_0^r = 26)$



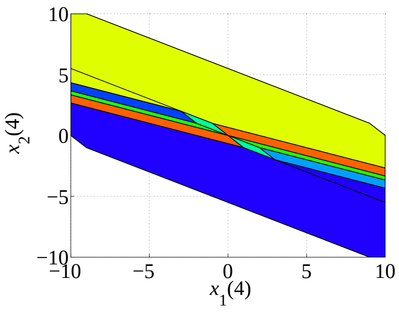
(b) Partition of the state space for the affine control law $u^*(1)$ $(N_1^r = 28)$



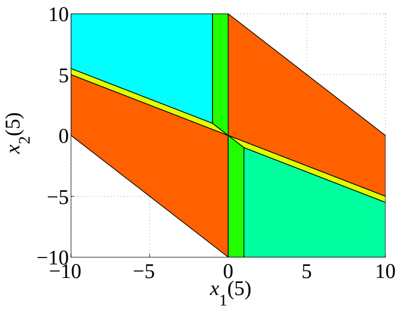
(c) Partition of the state space for the affine control law $u^*(2)$ $(N_2^r = 26)$



(d) Partition of the state space for the affine control law $u^*(3)$ $(N_3^r = 12)$

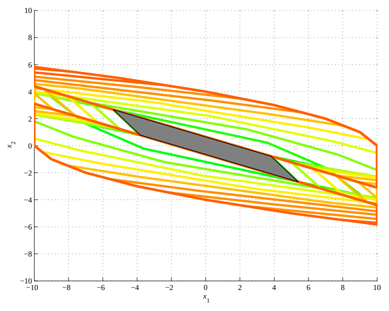


(e) Partition of the state space for the affine control law $u^*(4)$ $(N_4^r = 12)$

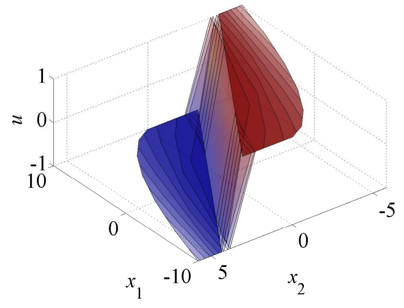


(f) Partition of the state space for the affine control law $u^*(5)$ $(N_5^r = 8)$

Fig. 10.11 Example 10.9: Partition of the state space for optimal control law



(a) Minimum-Time State Space Partition



(b) Minimum-Time Control Law

Fig. 10.12 Example 10.10: Minimum-Time Partition of the State Space and Control Law

Chapter 11
Receding Horizon Control

Model predictive control is the only advanced control technology that has made a substantial impact on industrial control problems: its success is largely due to its almost unique ability to handle, simply and effectively, hard constraints on control and states.

(D.Mayne, 2001 [186])

11.1 Introduction

In the previous chapter we discussed the solution of constrained finite time and infinite time optimal control problems for linear systems. An infinite horizon "sub-optimal" controller can be designed by repeatedly solving finite time optimal control problems in a receding horizon fashion as described next.

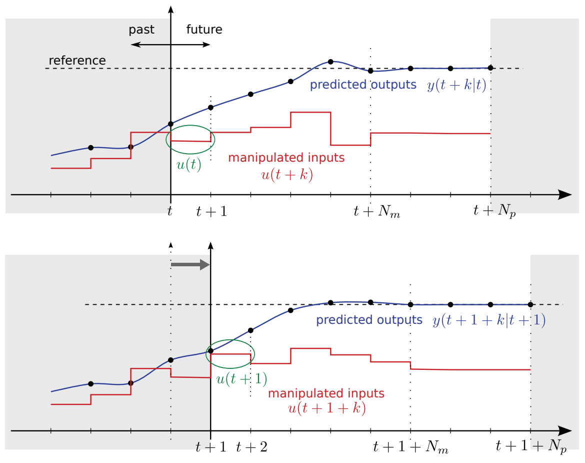


Fig. 11.1 Receding Horizon Idea

At each sampling time, starting at the current state, an open-loop optimal control problem is solved over a finite horizon (top diagram in Figure 11.1). The computed optimal manipulated input signal is applied to the process only during the following sampling interval $[t, t+1]$. At the next time step $t+1$ a new optimal control problem based on new measurements of the state is solved over a shifted horizon (bottom diagram in Figure 11.1). The resulting controller is referred to as Receding Horizon Controller (RHC).

A receding horizon controller where the finite time optimal control law is computed by solving an optimization problem on-line is usually referred to as *Model Predictive Control* (MPC).

Since RHC requires to solve at each sampling time an open-loop constrained finite time optimal control problem as a function of the current state, the results of the previous chapters lead to a different approach for RHC implementation. Precomputing off-line the explicit piecewise affine feedback policy (that provides the optimal control for all states) 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.

This technique is attractive for a wide range of practical problems where the computational complexity of on-line optimization is prohibitive. It also provides insight into the structure underlying optimization-based controllers, describing the behaviour of the RHC controller in different regions of the state space. Moreover, for applications where safety is crucial, the correctness of a piecewise affine control law is easier to verify than of a mathematical program solver.

In this chapter we review the basics of RHC. We discuss the stability and the feasibility of RHC and we provide guidelines for choosing the terminal weight so that closed-loop stability is achieved. Then, in Sections 11.4 and 11.5 the piecewise affine feedback control structure of QP-based and LP-based RHC is obtained as a simple corollary of the results of the previous chapters.

11.2 RHC Implementation

Consider the problem of regulating to the origin the discrete-time linear time-invariant system

$$\begin{cases} x(t+1) = Ax(t) + Bu(t) \\ y(t) = Cx(t) \end{cases} \tag{11.1}$$

where $x(t) \in \mathbb{R}^n$, $u(t) \in \mathbb{R}^m$, and $y(t) \in \mathbb{R}^p$ are the state, input, and output vectors, respectively, subject to the constraints

$$x(t) \in \mathcal{X},\ u(t) \in \mathcal{U},\ \forall t \geq 0 \tag{11.2}$$

where the sets $\mathcal{X} \subseteq \mathbb{R}^n$ and $\mathcal{U} \subseteq \mathbb{R}^m$ are polyhedra. Receding Horizon Control (RHC) approaches such a constrained regulation problem in the following way. Assume that a full measurement or estimate of the state $x(t)$ is available at the current time t. Then the finite time optimal control problem

$$\begin{array}{rll} J_t^*(x(t)) = \min_{U_{t\to t+N|t}} & J_t(x(t), U_{t\to t+N|t}) \triangleq p(x_{t+N|t}) + \displaystyle\sum_{k=0}^{N-1} q(x_{t+k|t}, u_{t+k|t}) \\ \text{subj. to} & x_{t+k+1|t} = Ax_{t+k|t} + Bu_{t+k|t},\ k = 0, \ldots, N-1 \\ & x_{t+k|t} \in \mathcal{X},\ u_{t+k|t} \in \mathcal{U},\ k = 0, \ldots, N-1 \\ & x_{t+N|t} \in \mathcal{X}_f \\ & x_{t|t} = x(t) \end{array} \tag{11.3}$$

is solved at time t, where $U_{t\to t+N|t} = \{u_{t|t}, \ldots, u_{t+N-1|t}\}$ and where $x_{t+k|t}$ denotes the state vector at time $t+k$ predicted at time t obtained by starting from the current state $x_{t|t} = x(t)$ and applying to the system model

$$x_{t+k+1|t} = Ax_{t+k|t} + Bu_{t+k|t} \tag{11.4}$$

the input sequence $u_{t|t}, \ldots, u_{t+N-1|t}$. Often the symbol $x_{t+k|t}$ is read as "the state x at time $t+k$ predicted at time t". Similarly, $u_{t+k|t}$ is read as "the input u at time $t+k$ computed at time t". For instance, $x_{3|1}$ represents the predicted state at time 3 when the prediction is done at time $t = 1$ starting from the current state $x(1)$. It is different, in general, from $x_{3|2}$ which is the predicted state at time 3 when the prediction is done at time $t = 2$ starting from the current state $x(2)$.

Let $U^*_{t\to t+N|t} = \{u^*_{t|t}, \ldots, u^*_{t+N-1|t}\}$ be the optimal solution of (11.3) at time t and $J_t^*(x(t))$ the corresponding value function. Then, the first element of $U^*_{t\to t+N|t}$ is applied to system (11.1)

$$u(t) = u^*_{t|t}(x(t)). \tag{11.5}$$

The optimization problem (11.3) is repeated at time $t+1$, based on the new state $x_{t+1|t+1} = x(t+1)$, yielding a *moving* or *receding horizon* control strategy.

Let $f_t : \mathbb{R}^n \to \mathbb{R}^m$ denote the *receding horizon* control law that associates the optimal input $u^*_{t|t}$ to the current state $x(t)$, $f_t(x(t)) = u^*_{t|t}(x(t))$. Then, the closed-loop system obtained by controlling (11.1) with the RHC (11.3)-(11.5) is

$$x(k+1) = Ax(k) + Bf_k(x(k)) \triangleq f_{cl}(x(k), k),\ k \geq 0 \tag{11.6}$$

Note that the notation used in this chapter is slightly different from the one used in Chapter 10. Because of the receding horizon strategy, there is the need to distinguish between the input $u^*(t+k)$ applied to the plant at time $t+k$, and optimizer $u^*_{t+k|t}$ of the problem (11.3) at time $t+k$ obtained by solving (11.3) at time t with $x_{t|t} = x(t)$.

Consider problem (11.3). As the system, the constraints and the cost function are time-invariant, the solution to problem (11.3) is a time-invariant function of the initial state $x(t)$. Therefore, in order to simplify the notation, we can set $t = 0$ in (11.3) and remove the term "$|0$" since it is now redundant and rewrite (11.3) as

$$\begin{array}{rll} J_0^*(x(t)) = \min_{U_0} & J_0(x(t),U_0) \triangleq p(x_N) + \displaystyle\sum_{k=0}^{N-1} q(x_k,u_k) \\ & \text{subj. to } x_{k+1} = Ax_k + Bu_k,\ k=0,\ldots,N-1 \\ & x_k \in \mathcal{X},\ u_k \in \mathcal{U},\ k=0,\ldots,N-1 \\ & x_N \in \mathcal{X}_f \\ & x_0 = x(t) \end{array} \tag{11.7}$$

where $U_0 = \{u_0,\ldots,u_{N-1}\}$ and the notation in Remark 7.1 applies. Similarly to previous chapters, we will focus on two classes of cost functions. If the 1-norm or ∞-norm is used in (11.7), then we set $p(x_N) = \|Px_N\|_p$ and $q(x_k,u_k) = \|Qx_k\|_p + \|Ru_k\|_p$ with $p=1$ or $p=\infty$ and P, Q, R full column rank matrices. The cost function is rewritten as

$$J_0(x(0),U_0) \triangleq \|Px_N\|_p + \sum_{k=0}^{N-1} \|Qx_k\|_p + \|Ru_k\|_p \tag{11.8}$$

If the squared euclidian norm is used in (11.7), then we set $p(x_N) = x_N'Px_N$ and $q(x_k,u_k) = x_k'Qx_k + u_k'Ru_k$ with $P \succeq 0$, $Q \succeq 0$ and $R \succ 0$. The cost function is rewritten as

$$J_0(x(0),U_0) \triangleq x_N'Px_N + \sum_{k=0}^{N-1} x_k'Qx_k + u_k'Ru_k \tag{11.9}$$

The control law (11.5)

$$u(t) = f_0(x(t)) = u_0^*(x(t)). \tag{11.10}$$

and closed-loop system (11.6)

$$x(k+1) = Ax(k) + Bf_0(x(k)) = f_{cl}(x(k)),\ k \geq 0 \tag{11.11}$$

are time-invariant as well.

Note that the notation in (11.7) does not allow us to distinguish at which time step a certain state prediction or optimizer is computed and is valid for time-invariant problems only. Nevertheless, we will prefer the RHC notation in (11.7) to the one in (11.3) in order to simplify the exposition.

Compare problem (11.7) and the CFTOC (10.32). The *only* difference is that problem (11.7) is solved for $x_0 = x(t)$, $t \geq 0$ rather than for $x_0 = x(0)$. For this reason we can make use of all the results of the previous chapter. In particular, $\mathcal{X}_0$ denotes the set of feasible states $x(t)$ for problem (11.7) as defined and studied in Section 10.3. Recall from Section 10.3 that $\mathcal{X}_0$ is a polyhedron.

From the above explanations it is clear that a fixed prediction horizon is shifted or *receded* over time, hence its name, receding horizon control. The procedure of this *on-line* optimal control technique is summarized in the following algorithm.

Algorithm 11.1 (On-line receding horizon control)

```
1. MEASURE the state $x(t)$ at time instant $t$
2. OBTAIN $U_0^*(x(t))$ by solving the optimization problem (11.7)
3. IF $U_0^*(x(t)) = \emptyset$ THEN 'problem infeasible' STOP
4. APPLY the first element $u_0^*$ of $U_0^*$ to the system
5. WAIT for the new sampling time $t+1$, GOTO (1.)
```

Example 11.1. Consider the double integrator system (10.46) rewritten below:

$$\begin{cases} x(t+1) = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t) \\ \quad y(t) = \begin{bmatrix} 1 & 0 \end{bmatrix} x(t) \end{cases} \tag{11.12}$$

The aim is to compute the receding horizon controller that solves the optimization problem (11.7) with $p(x_N) = x_N' P x_N$, $q(x_k, u_k) = x_k' Q x_k + u_k' R u_k$ $N = 3, P = Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, R = 10$, $\mathcal{X}_f = \mathbb{R}^2$ subject to the input constraints

$$-0.5 \leq u(k) \leq 0.5, \; k = 0, \ldots, 3 \tag{11.13}$$

and the state constraints

$$\begin{bmatrix} -5 \\ -5 \end{bmatrix} \leq x(k) \leq \begin{bmatrix} 5 \\ 5 \end{bmatrix}, \; k = 0, \ldots, 3. \tag{11.14}$$

The QP problem associated with the RHC has the form (10.52) with

$$H = \begin{bmatrix} 13.50 & -10.00 & -0.50 \\ -10.00 & 22.00 & -10.00 \\ -0.50 & -10.00 & 31.50 \end{bmatrix}, \quad F = \begin{bmatrix} -10.50 & 10.00 & -0.50 \\ -20.50 & 10.00 & 9.50 \end{bmatrix}, \quad Y = \begin{bmatrix} 14.50 & 23.50 \\ 23.50 & 54.50 \end{bmatrix} \tag{11.15}$$

and

$$G_0 = \begin{bmatrix} 0.50 & -1.00 & 0.50 \\ -0.50 & 1.00 & -0.50 \\ -0.50 & 0.00 & 0.50 \\ -0.50 & 0.00 & -0.50 \\ 0.50 & 0.00 & -0.50 \\ 0.50 & 0.00 & 0.50 \\ -1.00 & 0.00 & 0.00 \\ 0.00 & -1.00 & 0.00 \\ 1.00 & 0.00 & 0.00 \\ 0.00 & 1.00 & 0.00 \\ 0.00 & 0.00 & -1.00 \\ 0.00 & 0.00 & 1.00 \\ 0.00 & 0.00 & 0.00 \\ -0.50 & 0.00 & 0.50 \\ 0.00 & 0.00 & 0.00 \\ 0.50 & 0.00 & -0.50 \\ -0.50 & 0.00 & 0.50 \\ 0.50 & 0.00 & -0.50 \\ 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 \end{bmatrix}, \; E_0 = \begin{bmatrix} 0.50 & 0.50 \\ -0.50 & -0.50 \\ 0.50 & 0.50 \\ -0.50 & -0.50 \\ -0.50 & -0.50 \\ 0.50 & 0.50 \\ 0.00 & 0.00 \\ 0.00 & 0.00 \\ 0.00 & 0.00 \\ 0.00 & 0.00 \\ 0.00 & 0.00 \\ 0.00 & 0.00 \\ 1.00 & 1.00 \\ -0.50 & -0.50 \\ -1.00 & -1.00 \\ 0.50 & 0.50 \\ -0.50 & -1.50 \\ 0.50 & 1.50 \\ 1.00 & 0.00 \\ 0.00 & 1.00 \\ -1.00 & 0.00 \\ 0.00 & -1.00 \end{bmatrix}, \; W_0 = \begin{bmatrix} 0.50 \\ 0.50 \\ 5.00 \\ 5.00 \\ 5.00 \\ 5.00 \\ 5.00 \\ 5.00 \\ 5.00 \\ 5.00 \\ 0.50 \\ 0.50 \\ 5.00 \\ 5.00 \\ 5.00 \\ 5.00 \\ 0.50 \\ 0.50 \\ 5.00 \\ 5.00 \\ 5.00 \\ 5.00 \end{bmatrix} \tag{11.16}$$

The RHC (11.7)-(11.10) algorithm becomes

1. MEASURE the state $x(t)$ at time instance t
2. COMPUTE $\tilde{F} = 2x'(t)F$ and $\tilde{W}_0 = W_0 + E_0 x(t)$
3. OBTAIN $U_0^*(x(t))$ by solving the optimization problem $[U_0^*, \text{Flag}] = \text{QP}(H, \tilde{F}, G_0, \tilde{W}_0)$
4. IF Flag='infeasible' THEN STOP
5. APPLY the first element u_0^* of U_0^* to the system
6. WAIT for the new sampling time $t+1$, GOTO (1.)

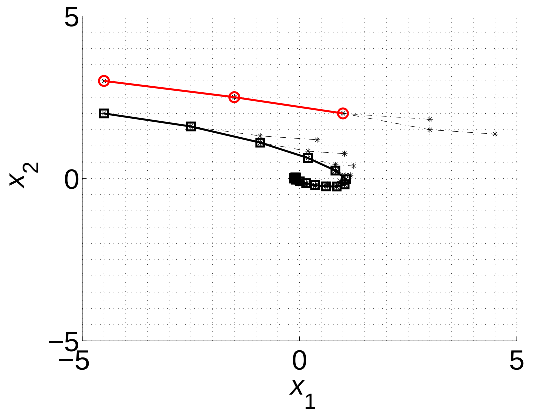
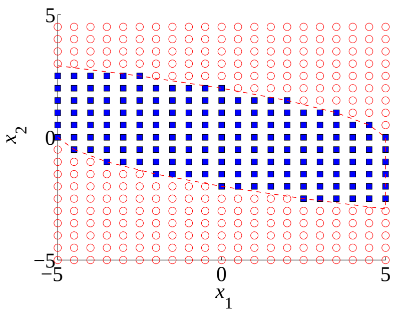


Fig. 11.2 Example 11.1. Closed-loop trajectories for the initial state x(0)=[-4.5,2] (*boxes*) and x(0)=[-4.5,3] (*circles*).

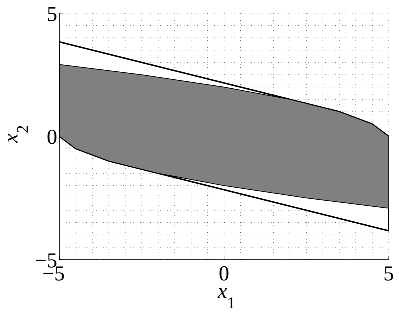
Fig. 11.2 shows two closed-loop trajectories starting at state $x(0) = [-4.5, 2]$ and $x(0) = [-4.5, 3]$. The trajectory starting from $x(0) = [-4.5, 2]$ converges to the origin and satisfies input and state constraints. The trajectory starting from $x(0) = [-4.5, 3]$ stops at $x(2) = [1, 2]$ because of infeasibility. At each time step, the open-loop predictions are depicted with dashed lines. This shows that the closed-loop trajectories are different from the open-loop predicted trajectories because of the receding horizon nature of the controller.

In Fig. 11.3(a) the feasible state space was gridded and each point of the grid was marked with a square if the RHC law (11.7)-(11.10) generates feasible closed-loop trajectories and with a circle if it does not. The set of all initial conditions generating feasible closed-loop trajectories is the maximal positive invariant set $\mathcal{O}_\infty$ of the autonomous system (11.11). We remark that this set is different from the set $\mathcal{X}_0$ of feasible initial conditions for the QP problem (10.52) with matrices (11.15)-(11.16). Both sets $\mathcal{O}_\infty$ and $\mathcal{X}_0$ are depicted in figure 11.3(b). The computation of f_0 is discussed later in this chapter. Because of the nonlinear nature of f_0, the computation of $\mathcal{O}_\infty$ for the system (11.11) is not an easy task. Therefore we will show how to choose a terminal invariant set $\mathcal{X}_f$ such that $\mathcal{O}_\infty = \mathcal{X}_0$ is guaranteed automatically.

Note that a feasible closed-loop trajectory does not necessarily converge to the origin. Feasibility, convergence and stability of RHC are discussed in detail in the next sections. Before that we want to illustrate these issues through another example.



(a) *Boxes* (*Circles*) are initial points leading (not leading) to feasible closed-loop trajectories



(b) Maximal positive invariant set $\mathcal{O}_\infty$ (grey) and set of initial feasible states $\mathcal{X}_0$ (white and gray)

Fig. 11.3 Double integrator Example (11.1)

Example 11.2. Consider the unstable system

$$x(t+1) = \begin{bmatrix} 2 & 1 \\ 0 & 0.5 \end{bmatrix} x(t) + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u(t) \qquad (11.17)$$

with the input constraints

$$-1 \le u(k) \le 1, \ k = 0, \ldots, N-1 \tag{11.18}$$

and the state constraints

$$\begin{bmatrix} -10 \\ -10 \end{bmatrix} \le x(k) \le \begin{bmatrix} 10 \\ 10 \end{bmatrix}, \ k = 0, \ldots, N-1. \tag{11.19}$$

In the following, we study the receding horizon control problem (11.7) with $p(x_N) = x_N' P x_N$, $q(x_k, u_k) = x_k' Q x_k + u_k' R u_k$ for different horizons N and weights R. We set $Q = I$ and omit both the terminal set constraint and the terminal weight, i.e. $\mathcal{X}_f = \mathbb{R}^2$, $P = 0$.

Fig. 11.4 shows closed-loop trajectories for receding horizon control loops that were obtained with the following parameter settings

Setting 1:	$N = 2, R = 10$
Setting 2:	$N = 3, R = 2$
Setting 3:	$N = 4, R = 1$

For *Setting 1* (Fig. 11.4(a)) there is evidently no initial state that can be steered to the origin. Indeed, it turns out, that *all* non-zero initial states $x(0) \in \mathbb{R}^2$ diverge from the origin and eventually become infeasible. Different from that, *Setting 2* leads to a receding horizon controller, that manages to get some of the initial states converge to the origin, as seen in Fig. 11.4(b). Finally, Fig. 11.4(c) shows that *Setting 3* can expand the set of those initial states that can be brought to the origin.

Note the behavior of particular initial states:

1. Closed-loop trajectories starting at state $x(0) = [-4, 7]$ behave differently depending on the chosen setting. Both *Setting 1* and *Setting 2* cannot bring this state to the origin, but the controller with *Setting 3* succeeds.
2. There are initial states, e.g. $x(0) = [-4, 8.5]$, that always lead to infeasible trajectories independent of the chosen settings. It turns out, that *no* setting can be found that brings those states to the origin.

These results illustrate that the choice of parameters for receding horizon control influences the behavior of the resulting closed-loop trajectories in a complex manner. A better understanding of the effect of parameter changes can be gained from an inspection of maximal positive invariant sets $\mathcal{O}_\infty$ for the different settings, and the maximal control invariant set $\mathcal{C}_\infty$ as depicted in Fig. 11.5.
The maximal positive invariant set stemming from *Setting 1* only contains the origin ($\mathcal{O}_\infty = \{0\}$) which explains why all non-zero initial states diverge from the origin. For *Setting 2* the maximal positive invariant set has grown considerably, but does not contain the initial state $x(0) = [-4, 7]$, thus leading to infeasibility eventually. *Setting 3* leads to a maximal positive invariant set that contains this state and thus keeps the closed-loop trajectory inside this set for all future time steps.

From Fig. 11.5 we also see that a trajectory starting at $x(0) = [-4, 8.5]$ cannot be kept inside any bounded set by *any* setting (indeed, by any controller) since it is outside the maximal control invariant set $\mathcal{C}_\infty$.

11.3 RHC Main Issues

If we solve the receding horizon problem for the special case of an infinite horizon (setting $N = \infty$ in (11.7) as we did for LQR in Section 8.4 and CLQR in Section 10.4.3) then it is almost immediate that the closed-loop system with this controller has some nice properties. Most importantly, the differences between the open-loop predicted and the actual closed-loop trajectories observed in Example 11.1 disappear. As a consequence, if the optimization problem is feasible, then the closed-loop trajectories will be feasible for all times. If the optimization problem has a finite solution, then in closed loop the states and inputs will converge to the origin asymptotically.

In RHC, when we solve the optimization problem over a finite horizon repeatedly at each time step, we hope that the controller resulting from this "short-sighted" strategy will lead to a closed-loop behavior that mimics that of the infinite horizon controller. The examples in the last section indicated that at least two problems may occur. First of all, the controller may lead us into a situation where after a few steps the finite horizon optimal control problem that we need to solve at each time step is infeasible, i.e., that there does not exist a sequence of control inputs for which the constraints are obeyed. Second, even if the feasibility problem does not occur, the generated control inputs may not lead to trajectories that converge to the origin, i.e., that the closed-loop system is asymptotically stable.

In general, stability and feasibility are not ensured by the RHC law (11.7)-(11.10). In principle, we could analyze the RHC law for feasibility, stability and convergence but this is difficult as the examples in the last section illustrated. Therefore, conditions will be derived on how the terminal weight P and the terminal constraint set $\mathcal{X}_f$ should be chosen such that closed-loop stability and feasibility are ensured.

11.3.1 Feasibility of RHC

The examples in the last section illustrate that feasibility at the initial time $x(0) \in \mathcal{X}_0$ does not necessarily imply feasibility for all future times. It is desirable to design a RHC such that feasibility for all future times is guaranteed, a property we refer to as *persistent feasibility*.

We would like to gain some insight when persistent feasibility occurs and how it is affected by the formulation of the control problem and the choice of the controller parameters. Let us first recall the various sets introduced in Sections 10.1-10.3 and how they are influenced.

$\mathcal{C}_\infty$: The maximal control invariant set $\mathcal{C}_\infty$ is only affected by the sets $\mathcal{X}$ and $\mathcal{U}$, the constraints on states and inputs. It is the largest set over which we can expect *any* controller to work.

$\mathcal{X}_0$: A control input U_0 can only be found, i.e. the control problem is feasible, if $x(0) \in \mathcal{X}_0$. The set $\mathcal{X}_0$ depends on $\mathcal{X}$ and $\mathcal{U}$, on the controller horizon N and on the controller terminal set $\mathcal{X}_f$. It does not depend on the objective function and it has generally no relation with $\mathcal{C}_\infty$ (it can be larger, smaller, etc.).

$\mathcal{O}_\infty$:The maximal positive invariant set for the closed-loop system depends on the controller and as such on all parameters affecting the controller, i.e., $\mathcal{X}$, $\mathcal{U}$, N, $\mathcal{X}_f$ and the objective function with its parameters P, Q and R. Clearly $\mathcal{O}_\infty \subseteq \mathcal{X}_0$ because if it were not there would be points in $\mathcal{O}_\infty$ for which the control problem is not feasible. Because of invariance, the closed-loop is persistently feasible for all states $x(0) \in \mathcal{O}_\infty$.

We can now state necessary and sufficient conditions guaranteeing persistent feasibility by means of invariant set theory.

Lemma 11.2. *Let $\mathcal{O}_\infty$ be the maximal positive invariant set for the closed-loop system $x(k+1) = f_{cl}(x(k))$ in (11.11) with constraints (11.2). The RHC problem is persistently feasible if and only if $\mathcal{X}_0 = \mathcal{O}_\infty$*

Proof: For the RHC problem to be persistently feasible $\mathcal{X}_0$ must be positive invariant for the closed-loop system. We argued above that $\mathcal{O}_\infty \subseteq \mathcal{X}_0$. As the positive invariant set $\mathcal{X}_0$ cannot be larger than the maximal positive invariant set $\mathcal{O}_\infty$, it follows that $\mathcal{X}_0 = \mathcal{O}_\infty$. □

As $\mathcal{X}_0$ does not depend on the controller parameters P, Q and R but $\mathcal{O}_\infty$ does, the requirement $\mathcal{X}_0 = \mathcal{O}_\infty$ for persistent feasibility shows that, in general, only some P, Q and R are allowed. The parameters P, Q and R affect the performance. The complex effect they have on persistent feasibility makes their choice extremely difficult for the design engineer. In the following we will remedy this undesirable situation. We will make use of the following important *sufficient* condition for persistent feasibility.

Lemma 11.3. *Consider the RHC law (11.7)-(11.10) with $N \geq 1$. If $\mathcal{X}_1$ is a control invariant set for system (11.1)-(11.2) then the RHC is persistently feasible. Also, $\mathcal{O}_\infty$ is independent of P, Q and R.*

Proof: If $\mathcal{X}_1$ is control invariant then, by definition, $\mathcal{X}_1 \subseteq \text{Pre}(\mathcal{X}_1)$. Also recall that $\text{Pre}(\mathcal{X}_1) = \mathcal{X}_0$ from the properties of the feasible sets in equation (10.43) (note that $\text{Pre}(\mathcal{X}_1) \cap \mathcal{X} = \text{Pre}(\mathcal{X}_1)$ from control invariance). Pick some $x \in \mathcal{X}_0$ and some feasible control u for that x and define $x^+ = Ax + Bu \in \mathcal{X}_1$. Then $x^+ \in \mathcal{X}_1 \subseteq \text{Pre}(\mathcal{X}_1) = \mathcal{X}_0$. As u was arbitrary (as long as it is feasible) $x^+ \in \mathcal{X}_0$ for all feasible u. As $\mathcal{X}_0$ is positive invariant,

$\mathcal{X}_0 = \mathcal{O}_\infty$ from Lemma 11.2. As $\mathcal{X}_0$ is positive invariant for all feasible u, $\mathcal{O}_\infty$ does not depend on P, Q and R. □

Note that in the proof of Lemma 11.3, persistent feasibility does not depended on the input u as long as it is feasible. For this reason, sometimes in the literature this property is referred to "persistently feasible for all feasible u".

We can use Lemma 11.3 in the following manner. For $N = 1$, $\mathcal{X}_1 = \mathcal{X}_f$. If we choose the terminal set to be control invariant then $\mathcal{X}_0 = \mathcal{O}_\infty$ and RHC will be persistently feasible independent of chosen control objectives and parameters. Thus the designer can choose the parameters to affect performance without affecting persistent feasibility. A control horizon of $N = 1$ is often too restrictive, but we can easily extend Lemma 11.3.

Theorem 11.1. *Consider the RHC law (11.7)-(11.10) with $N \geq 1$. If $\mathcal{X}_f$ is a control invariant set for system (11.1)-(11.2) then the RHC is persistently feasible.*

Proof: If $\mathcal{X}_f$ is control invariant, then $\mathcal{X}_{N-1}$, $\mathcal{X}_{N-2}, \ldots, \mathcal{X}_1$ are control invariant and Lemma 11.3 establishes persistent feasibility for all feasible u. □

Corollary 11.1. *Consider the RHC law (11.7)-(11.10) with $N \geq 1$. If there exists $i \in [1, N]$ such that $\mathcal{X}_i$ is a control invariant set for system (11.1)-(11.2), then the RHC is persistently feasible for all cost functions.*

Proof: Follows directly from the proof of Theorem 11.3. □

Recall that Theorem 10.3 together with Remark 10.9 define the properties of the set $\mathcal{X}_0$ as N varies. Therefore, Theorem 11.3 and Corollary 11.1 provide also guidelines on the choice of the horizon N for guaranteeing persistent feasibility for all feasible u. For instance, if the RHC problem (11.7) for $N = \bar{N}$ yields a control invariant set $\mathcal{X}_0$, then from Theorem 10.3 the RHC law (11.7)-(11.10) with $N = \bar{N} + 1$ is persistently feasible for all feasible u. Moreover, from Corollary 11.1 the RHC law (11.7)-(11.10) with $N \geq \bar{N} + 1$ is persistently feasible for all feasible u.

Corollary 11.2. *Consider the RHC problem (11.7)-(11.10). If N is greater than the determinedness index $\bar{N}$ of $\mathcal{K}_\infty(\mathcal{X}_f)$ for system (11.1)-(11.2), then the RHC is persistently feasible.*

Proof: The feasible set $\mathcal{X}_i$ for $i = 1, \ldots, N-1$ is equal to the $(N-i)$-step controllable set $\mathcal{X}_i = \mathcal{K}_{N-i}(\mathcal{X}_f)$. If the maximal controllable set is finitely determined then $\mathcal{X}_i = \mathcal{K}_\infty(\mathcal{X}_f)$ for $i \leq N - \bar{N}$. Note that $\mathcal{K}_\infty(\mathcal{X}_f)$ is control invariant. Then persistent feasibility for all feasible u follows from Corollary 11.1. □

Persistent feasibility does not guarantee that the closed-loop trajectories converge towards the desired equilibrium point. From Theorem 11.3 it is clear that one can only guarantee that $x(k) \in \mathcal{X}_1$ for all $k > 0$ if $x(0) \in \mathcal{X}_0$.

One of the most popular approaches to guarantee persistent feasibility and stability of the RHC law (11.7)-(11.10) makes use of a control invariant terminal set $\mathcal{X}_f$ and a terminal cost P which drives the closed-loop optimal trajectories towards $\mathcal{X}_f$. A detailed discussion follows in the next section.

11.3.2 Stability of RHC

In this section we will derive the main stability result for RHC. Our objective is to find a Lyapunov function for the closed-loop system. We will show next that if terminal cost and constraint are appropriately chosen, then the value function $J_0^*(\cdot)$ is a Lyapunov function.

Theorem 11.2. *Consider system (11.1)-(11.2), the RHC law (11.7)-(11.10), the cost function (11.9) or (11.8) and the closed-loop system (11.11). Assume that*

(A0) $Q = Q' \succ 0$, $R = R' \succ 0$, $P \succ 0$, *if cost (11.9) is used, or* Q, R, P *full column rank matrices if cost (11.8) is used.*
(A1) The sets $\mathcal{X}$, $\mathcal{X}_f$ *and* $\mathcal{U}$ *contain the origin in their interior and are closed.*
(A2) $\mathcal{X}_f$ *is control invariant,* $\mathcal{X}_f \subseteq \mathcal{X}$.
(A3) $\min_{v\in\mathcal{U},\ Ax+Bv\in\mathcal{X}_f} (-p(x) + q(x,v) + p(Ax+Bv)) \leq 0,\ \forall x \in \mathcal{X}_f$.

Then,
(i) the state of the closed-loop system (11.11) converges to the origin, i.e., $\lim_{k\to\infty} x(k) = 0$,
(ii) the origin of the closed-loop system (11.11) is asymptotically stable with domain of attraction $\mathcal{X}_0$.

Proof: Based on Lemma 11.2, 11.3 and Theorem 11.3, from hypothesis (A2) we conclude that $\mathcal{X}_0 = \mathcal{O}_\infty$ is a positive invariant set for the closed-loop system (11.11) for any choice of $N \geq 1$, Q, R and P. Thus persistent feasibility for any feasible input is guaranteed.

Next we prove convergence and stability. We establish that the function $J_0^*(\cdot)$ in (11.7) is a Lyapunov function for the closed-loop system. Because J_0, system and constraints are time-invariant we can study the properties of J_0^* between step $k = 0$ and step $k + 1 = 1$.

Consider problem (11.7) at time $t = 0$. Let $x(0) \in \mathcal{X}_0$ and let $U_0^* = \{u_0^*, \ldots, u_{N-1}^*\}$ be the optimizer of problem (11.7) and $\mathbf{x}_0 = \{x(0), x_1, \ldots, x_N\}$ be the corresponding optimal state trajectory. After the implementation of u_0^* we obtain $x(1) = x_1 = Ax(0) + Bu_0^*$. Consider now problem (11.7) for $t = 1$. We will construct an upper bound on $J_0^*(x(1))$. Consider the sequence $\tilde{U}_1 = \{u_1^*, \ldots, u_{N-1}^*, v\}$ and the corresponding state trajectory resulting from the initial state $x(1)$, $\tilde{\mathbf{x}}_1 = \{x_1, \ldots, x_N, Ax_N + Bv\}$. Because $x_N \in \mathcal{X}_f$ and (A2) there exists a feasible v such that $x_{N+1} = Ax_N + Bv \in \mathcal{X}_f$ and with this

v the sequence $\tilde{U}_1 = \{u_1^*, \dots, u_{N-1}^*, v\}$ is feasible. Because $\tilde{U}_1$ is not optimal $J_0(x(1), \tilde{U}_1)$ is an upper bound on $J_0^*(x(1))$.

Since the trajectories generated by U_0^* and $\tilde{U}_1$ overlap, except for the first and last sampling intervals, it is immediate to show that

$$\begin{aligned} J_0^*(x(1)) \leq J_0(x(1), \tilde{U}_1) = J_0^*(x(0)) - q(x_0, u_0^*) - p(x_N) + \\ (q(x_N, v) + p(Ax_N + Bv)) \end{aligned} \tag{11.20}$$

Let $x = x_0 = x(0)$ and $u = u_0^*$. Under assumption (A3) equation (11.20) becomes

$$J_0^*(Ax + Bu) - J_0^*(x) \leq -q(x, u), \ \forall x \in \mathcal{X}_0. \tag{11.21}$$

Equation (11.21) and the hypothesis (A0) on the matrices R and Q ensure that $J_0^*(x)$ strictly decreases along the state trajectories of the closed-loop system (11.11) for any $x \in \mathcal{X}_0$, $x \neq 0$. In addition to the fact that $J_0^*(x)$ decreases, $J_0^*(x)$ is lower-bounded by zero and since the state trajectories generated by the closed-loop system (11.11) starting from any $x(0) \in \mathcal{X}_0$ lie in $\mathcal{X}_0$ for all $k \geq 0$, equation (11.21) is sufficient to ensure that the state of the closed-loop system converges to zero as $k \rightarrow 0$ if the initial state lies in $\mathcal{X}_0$. We have proven (i).

In order to prove stability via Theorem 7.2 we have to establish that $J_0^*(x)$ is a Lyapunov function. Positivity holds by definition, decrease follows from (11.21). For continuity at the origin we will show that $J_0^*(x) \leq p(x)$, $\forall x \in \mathcal{X}_f$ and as $p(x)$ is continuous at the origin $J_0^*(x)$ must be continuous as well. From assumption (A2), $\mathcal{X}_f$ is control invariant and thus for any $x \in \mathcal{X}_f$ there exists a feasible input sequence $\{u_0, \dots, u_{N-1}\}$ for problem (11.7) starting from the initial state $x_0 = x$ whose corresponding state trajectory is $\{x_0, x_1, \dots, x_N\}$ stays in $\mathcal{X}_f$, i.e., $x_i \in \mathcal{X}_f \ \forall\, i = 0, \dots, N$. Among all the aforementioned input sequences $\{u_0, \dots, u_{N-1}\}$ we focus on the one where u_i satisfies assumption (A3) for all $i = 0, \dots, N-1$. Such a sequence provides an upper bound on the function J_0^*:

$$J_0^*(x_0) \leq \left(\sum_{i=0}^{N-1} q(x_i, u_i) \right) + p(x_N), \ x_i \in \mathcal{X}_f, \ i = 0, \dots, N \tag{11.22}$$

which can be rewritten as

$$\begin{aligned} J_0^*(x_0) \leq & \left(\textstyle\sum_{i=0}^{N-1} q(x_i, u_i) \right) + p(x_N), \\ & = p(x_0) + \left(\textstyle\sum_{i=0}^{N-1} q(x_i, u_i) + p(x_{i+1}) - p(x_i) \right) \ x_i \in \mathcal{X}_f, \ i = 0, \dots, N \end{aligned} \tag{11.23}$$

which from assumption (A3) yields

$$J_0^*(x) \leq p(x), \ \forall x \in \mathcal{X}_f. \tag{11.24}$$

In conclusion, there exist a finite time in which any $x \in \mathcal{X}_0$ is steered to a level set of $J_0^*(x)$ contained in $\mathcal{X}_f$ after which convergence to and stability of the origin follows. □

Remark 11.1. The assumption on the positive definiteness of Q in Theorem 11.2 can be relaxed as in standard optimal control by allowing $Q \succeq 0$ with $(Q^{\frac{1}{2}}, A)$ observable.

Remark 11.2. The procedure outlined in Theorem 11.2 is, in general, conservative because it requires the introduction of an artificial terminal set $\mathcal{X}_f$ to guarantee persistent feasibility and a terminal cost to guarantee stability. Requiring $x_N \in \mathcal{X}_f$ usually decreases the size of the region of attraction $\mathcal{X}_0 = \mathcal{O}_\infty$. Also the performance may be negatively affected.

Remark 11.3. A function $p(x)$ satisfying assumption (A3) of Theorem 11.2 is often called control Lyapunov function .

The hypothesis (A2) of Theorem 11.2 is required for guaranteeing persistent feasibility as discussed in Section 11.3.1. In some part of the literature the constraint $\mathcal{X}_f$ is not used. However, in this literature the terminal region constraint $\mathcal{X}_f$ is implicit. In fact, it is typically required that the horizon N is sufficiently large to ensure feasibility of the RHC (11.7)–(11.10) at all time instants t. Technically this means that N has to be greater than the determinedness index $\bar{N}$ of system (11.1)-(11.2) which by Corollary 11.2 guarantees persistent feasibility for all inputs. We refer the reader to Section 11.3.1 for more details on feasibility.

Next we will show a few simple choices for P and $\mathcal{X}_f$ satisfying the hypothesis (A2) and (A3) of Theorem 11.3.

Stability, 2-Norm case

Consider system (11.1)-(11.2), the RHC law (11.7)-(11.10), the cost function(11.9) and the closed-loop system (11.11). A simple choice $\mathcal{X}_f$ is obtained by choosing $\mathcal{X}_f$ as the maximal positive invariant set (see Section 10.1) for the closed-loop system $x(k+1) = (A + BF_\infty)x(k)$ where F_∞ is the associated unconstrained infinite-time optimal controller (8.32). With this choice the assumption (A3) in Theorem 11.2 becomes

$$x'(A'(P - PB(B'PB + R)^{-1}BP)A + Q - P)x \leq 0, \ \forall x \in \mathcal{X}_f \qquad (11.25)$$

which is satisfied as an equality if P is chosen as the solution P_∞ of the algebraic Riccati equation (8.31) for system (11.1) (see proof in Section 8.5).

If system (11.1) is asymptotically stable, then $\mathcal{X}_f$ can be alternatively chosen as the positive invariant set of the autonomous system $x(k+1) =$

$Ax(k)$ subject to the state constraints $x \in \mathcal{X}$. Therefore in $\mathcal{X}_f$ the input $\mathbf{0}$ is feasible and the assumption (A3) in Theorem 11.2 becomes

$$-x'Px + x'A'PAx + x'Qx \leq 0, \forall x \in \mathcal{X}_f \tag{11.26}$$

which is satisfied if P solves $x'(-P + A'PA + Q)x = 0$, i.e. the standard Lyapunov equation (7.42). In this case stability implies exponential stability. The argument is simple. As the system is closed-loop stable it enters the terminal region in finite time. If $\mathcal{X}_f$ is chose as suggested, the closed-loop system is unconstrained after entering $\mathcal{X}_f$. For an unconstrained linear system the convergence to the origin is exponential.

Stability, 1, ∞-Norm case

Consider system (11.1)-(11.2), the RHC law (11.7)-(11.10), the cost function(11.8) and the closed-loop system (11.11). Let $p = 1$ or $p = \infty$. If system (11.1) is asymptotically stable, then $\mathcal{X}_f$ can be chosen as the positively invariant set of the autonomous system $x(k+1) = Ax(k)$ subject to the state constraints $x \in \mathcal{X}$. Therefore in $\mathcal{X}_f$ the input $\mathbf{0}$ is feasible and the assumption (A3) in Theorem 11.2 becomes

$$-\|Px\|_p + \|PAx\|_p + \|Qx\|_p \leq 0, \forall x \in \mathcal{X}_f \tag{11.27}$$

which is the corresponding Lyapunov inequality for the 1, ∞-norm case (7.48) whose solution has been discussed in Section 7.5.3.

In general, if the unconstrained optimal controller (9.31) exists it is PPWA. In this case the computation of the maximal invariant set $\mathcal{X}_f$ for the closed-loop PWA system

$$x(k+1) = (A + F^i)x(k) \quad \text{if} \quad H^i x \leq 0, \ i = 1, \dots, N^r \tag{11.28}$$

is more involved. However if such $\mathcal{X}_f$ can be computed it can be used as terminal constraint in Theorem 11.2. With this choice the assumption (A3) in Theorem 11.2 is satisfied by the infinite-time unconstrained optimal cost P_∞ in (9.32).

11.4 State Feedback Solution of RHC, 2-Norm Case

The state feedback receding horizon controller (11.10) with cost (11.9) for system (11.1) is

$$u(t) = f_0^*(x(t)) \tag{11.29}$$

where $f_0^*(x_0) : \mathbb{R}^n \to \mathbb{R}^m$ is the piecewise affine solution to the CFTOC (11.7) and is obtained as explained in Section 10.4.

We remark that the implicit form (11.7) and the explicit form (11.29) describe the same function, and therefore the stability, feasibility, and performance properties mentioned in the previous sections are automatically inherited by the piecewise affine control law (11.29). Clearly, the explicit form (11.29) has the advantage of being easier to implement, and provides insight into the type of action of controller action in different regions CR_i of the state space.

Example 11.3. Consider the double integrator system (11.12) subject to the input constraints

$$-1 \le u(k) \le 1 \tag{11.30}$$

and the state constraints

$$-10 \le x(k) \le 10 \tag{11.31}$$

We want to regulate the system to the origin by using the RHC problem (11.7)–(11.10) with cost (11.9), $Q = \left[\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right]$, $R = 0.01$, and $P = P_\infty$ where P_∞ solves the algebraic Riccati equation (8.31). We consider three cases:

Case 1. $N = 2$, $\mathcal{X}_f = 0$,
Case 2. $N = 2$, $\mathcal{X}_f$ is the positively invariant set of the closed-loop system $x(k+1) = (A + BF_\infty)$ where F_∞ is the infinite-time unconstrained optimal controller (8.32).
Case 3. No terminal state constraints: $\mathcal{X}_f = \mathbb{R}^n$ and $N = 6 =$ determinedness index+1.

From the results presented in this chapter, all three cases guarantee persistent feasibility for all cost functions and asymptotic stability of the origin with region of attraction $\mathcal{X}_0$ (with $\mathcal{X}_0$ different for each case) . Next we will detail the matrices of the quadratic program for the on-line solution as well as the explicit solution for the three cases.

Case 1: $\mathcal{X}_f = 0$. The mp-QP problem associated with the RHC has the form (10.52) with

$$H = \left[\begin{smallmatrix} 19.08 & 8.55 \\ 8.55 & 5.31 \end{smallmatrix}\right], \; F = \left[\begin{smallmatrix} -10.57 & -5.29 \\ -10.59 & -5.29 \end{smallmatrix}\right], \; Y = \left[\begin{smallmatrix} 10.31 & 9.33 \\ 9.33 & 10.37 \end{smallmatrix}\right] \tag{11.32}$$

and

$$G_0 = \begin{bmatrix} 0.00 & -1.00 \\ 0.00 & 1.00 \\ 0.00 & 0.00 \\ -1.00 & 0.00 \\ 0.00 & 0.00 \\ 1.00 & 0.00 \\ -1.00 & 0.00 \\ -1.00 & -1.00 \\ 1.00 & 0.00 \\ 1.00 & 1.00 \\ 0.00 & 0.00 \\ -1.00 & 0.00 \\ 0.00 & 0.00 \\ 1.00 & 0.00 \\ -1.00 & 0.00 \\ 1.00 & 0.00 \\ 0.00 & 0.00 \\ 0.00 & 0.00 \\ 0.00 & 0.00 \\ 0.00 & 0.00 \\ 0.00 & 0.00 \\ 0.00 & 0.00 \\ 0.00 & 0.00 \\ 0.00 & 0.00 \\ 1.00 & 0.00 \\ 1.00 & 1.00 \\ -1.00 & 0.00 \\ -1.00 & -1.00 \end{bmatrix}, \; E_0 = \begin{bmatrix} 0.00 & 0.00 \\ 0.00 & 0.00 \\ 1.00 & 1.00 \\ -1.00 & -1.00 \\ -1.00 & -1.00 \\ 1.00 & 1.00 \\ 0.00 & 0.00 \\ -1.00 & -1.00 \\ 0.00 & 0.00 \\ 1.00 & 1.00 \\ 1.00 & 1.00 \\ -1.00 & -1.00 \\ -1.00 & -1.00 \\ 1.00 & 1.00 \\ -1.00 & -2.00 \\ 1.00 & 2.00 \\ 1.00 & 0.00 \\ 0.00 & 1.00 \\ -1.00 & 0.00 \\ 0.00 & -1.00 \\ 1.00 & 0.00 \\ 0.00 & 1.00 \\ -1.00 & 0.00 \\ 0.00 & -1.00 \\ 0.00 & 0.00 \\ 1.00 & 1.00 \\ 0.00 & 0.00 \\ -1.00 & -1.00 \end{bmatrix}, \; W_0 = \begin{bmatrix} 1.00 \\ 1.00 \\ 10.00 \\ 10.00 \\ 10.00 \\ 10.00 \\ 10.00 \\ 10.00 \\ 10.00 \\ 10.00 \\ 5.00 \\ 5.00 \\ 5.00 \\ 5.00 \\ 1.00 \\ 1.00 \\ 10.00 \\ 10.00 \\ 10.00 \\ 10.00 \\ 5.00 \\ 5.00 \\ 5.00 \\ 5.00 \\ 0.00 \\ 0.00 \\ 0.00 \\ 0.00 \end{bmatrix} \qquad (11.33)$$

The corresponding polyhedral partition of the state-space is depicted in Fig. 11.6(a). The RHC law is:

$$u = \begin{cases} \begin{bmatrix} -0.61 & -1.61 \end{bmatrix} x \text{ if } \begin{bmatrix} 0.70 & 0.71 \\ -0.70 & -0.71 \\ -0.70 & -0.71 \\ 0.70 & 0.71 \end{bmatrix} x \leq \begin{bmatrix} 0.00 \\ 0.00 \\ 0.00 \\ 0.00 \end{bmatrix} & \text{(Region \#1)} \\ \begin{bmatrix} -1.00 & -2.00 \end{bmatrix} x \text{ if } \begin{bmatrix} -0.71 & -0.71 \\ -0.70 & -0.71 \\ -0.45 & -0.89 \\ 0.45 & 0.89 \\ 0.71 & 0.71 \\ -0.70 & -0.71 \end{bmatrix} x \leq \begin{bmatrix} 0.00 \\ -0.00 \\ 0.45 \\ 0.45 \\ 0.71 \\ -0.00 \end{bmatrix} & \text{(Region \#2)} \\ \begin{bmatrix} -1.00 & -2.00 \end{bmatrix} x \text{ if } \begin{bmatrix} 0.45 & 0.89 \\ -0.70 & -0.71 \\ 0.71 & 0.71 \end{bmatrix} x \leq \begin{bmatrix} 0.45 \\ -0.00 \\ -0.00 \end{bmatrix} & \text{(Region \#3)} \\ \begin{bmatrix} -0.72 & -1.72 \end{bmatrix} x \text{ if } \begin{bmatrix} 0.39 & 0.92 \\ 0.70 & 0.71 \\ -0.70 & -0.71 \\ 0.70 & 0.71 \end{bmatrix} x \leq \begin{bmatrix} 0.54 \\ 0.00 \\ 0.00 \\ -0.00 \end{bmatrix} & \text{(Region \#4)} \\ \begin{bmatrix} -1.00 & -2.00 \end{bmatrix} x \text{ if } \begin{bmatrix} 0.45 & 0.89 \\ -0.71 & -0.71 \\ 0.70 & 0.71 \\ -0.45 & -0.89 \\ 0.71 & 0.71 \\ 0.70 & 0.71 \end{bmatrix} x \leq \begin{bmatrix} 0.45 \\ 0.71 \\ -0.00 \\ 0.45 \\ 0.00 \\ -0.00 \end{bmatrix} & \text{(Region \#5)} \\ \begin{bmatrix} -1.00 & -2.00 \end{bmatrix} x \text{ if } \begin{bmatrix} -0.45 & -0.89 \\ -0.71 & -0.71 \\ 0.70 & 0.71 \end{bmatrix} x \leq \begin{bmatrix} 0.45 \\ -0.00 \\ -0.00 \end{bmatrix} & \text{(Region \#6)} \\ \begin{bmatrix} -0.72 & -1.72 \end{bmatrix} x \text{ if } \begin{bmatrix} -0.39 & -0.92 \\ 0.70 & 0.71 \\ -0.70 & -0.71 \\ -0.70 & -0.71 \end{bmatrix} x \leq \begin{bmatrix} 0.54 \\ 0.00 \\ 0.00 \\ -0.00 \end{bmatrix} & \text{(Region \#7)} \end{cases}$$

The union of the regions depicted in Fig. 11.6(a) is $\mathcal{X}_0$. From Theorem 11.2, $\mathcal{X}_0$ is also the domain of attraction of the RHC law.

Case 2: $\mathcal{X}_f$ positively invariant set. The set $\mathcal{X}_f$ is

$$\mathcal{X}_f = \{x \in \mathbb{R}^2 : \begin{bmatrix} -0.35617 & -0.93442 \\ 0.35617 & 0.93442 \\ 0.71286 & 0.70131 \\ -0.71286 & -0.70131 \end{bmatrix} x \leq \begin{bmatrix} 0.58043 \\ 0.58043 \\ 1.9049 \\ 1.9049 \end{bmatrix}\} \tag{11.34}$$

The mp-QP problem associated with the RHC has the form (10.52) with

$$H = \begin{bmatrix} 19.08 & 8.55 \\ 8.55 & 5.31 \end{bmatrix},\ F = \begin{bmatrix} -10.57 & -5.29 \\ -10.59 & -5.29 \end{bmatrix},\ Y = \begin{bmatrix} 10.31 & 9.33 \\ 9.33 & 10.37 \end{bmatrix} \tag{11.35}$$

$$G_0 = \begin{bmatrix} 0.00 & -1.00 \\ 0.00 & 1.00 \\ 0.00 & 0.00 \\ -1.00 & 0.00 \\ 0.00 & 0.00 \\ 1.00 & 0.00 \\ -1.00 & 0.00 \\ -1.00 & -1.00 \\ 1.00 & 0.00 \\ 1.00 & 1.00 \\ 0.00 & 0.00 \\ -1.00 & 0.00 \\ 0.00 & 0.00 \\ 1.00 & 0.00 \\ -1.00 & 0.00 \\ 1.00 & 0.00 \\ 0.00 & 0.00 \\ 0.00 & 0.00 \\ 0.00 & 0.00 \\ 0.00 & 0.00 \\ 0.00 & 0.00 \\ 0.00 & 0.00 \\ 0.00 & 0.00 \\ 0.00 & 0.00 \\ -1.29 & -0.93 \\ 1.29 & 0.93 \\ 1.41 & 0.70 \\ -1.41 & -0.70 \end{bmatrix}, E_0 = \begin{bmatrix} 0.00 & 0.00 \\ 0.00 & 0.00 \\ 1.00 & 1.00 \\ -1.00 & -1.00 \\ -1.00 & -1.00 \\ 1.00 & 1.00 \\ 0.00 & 0.00 \\ -1.00 & -1.00 \\ 0.00 & 0.00 \\ 1.00 & 1.00 \\ 1.00 & 1.00 \\ -1.00 & -1.00 \\ -1.00 & -1.00 \\ 1.00 & 1.00 \\ -1.00 & -2.00 \\ 1.00 & 2.00 \\ 1.00 & 0.00 \\ 0.00 & 1.00 \\ -1.00 & 0.00 \\ 0.00 & -1.00 \\ 1.00 & 0.00 \\ 0.00 & 1.00 \\ -1.00 & 0.00 \\ 0.00 & -1.00 \\ -0.93 & -0.93 \\ 0.93 & 0.93 \\ 0.70 & 0.70 \\ -0.70 & -0.70 \end{bmatrix},\ W_0 = \begin{bmatrix} 1.00 \\ 1.00 \\ 10.00 \\ 10.00 \\ 10.00 \\ 10.00 \\ 10.00 \\ 10.00 \\ 10.00 \\ 10.00 \\ 5.00 \\ 5.00 \\ 5.00 \\ 5.00 \\ 1.00 \\ 1.00 \\ 10.00 \\ 10.00 \\ 10.00 \\ 10.00 \\ 5.00 \\ 5.00 \\ 5.00 \\ 5.00 \\ 0.58 \\ 0.58 \\ 1.90 \\ 1.90 \end{bmatrix} \tag{11.36}$$

The corresponding polyhedral partition of the state-space is depicted in Fig. 11.6(b). The RHC law is:

$$u = \begin{cases}
\begin{bmatrix} -0.61 & -1.61 \end{bmatrix} x & \text{if } \begin{bmatrix} 1.00 & 0.00 \\ -1.00 & 0.00 \\ -0.36 & -0.93 \\ 0.36 & 0.93 \\ 0.71 & 0.70 \\ -0.71 & -0.70 \end{bmatrix} x \leq \begin{bmatrix} 5.00 \\ 5.00 \\ 0.58 \\ 0.58 \\ 1.90 \\ 1.90 \end{bmatrix} \quad \text{(Region \#1)} \\
1.00 & \text{if } \begin{bmatrix} 1.00 & 0.00 \\ -1.00 & 0.00 \\ -0.71 & -0.71 \\ -0.21 & -0.98 \\ 0.21 & 0.98 \\ -0.32 & -0.95 \\ 0.27 & 0.96 \end{bmatrix} x \leq \begin{bmatrix} 5.00 \\ 5.00 \\ 3.54 \\ 1.67 \\ -0.98 \\ 1.79 \\ -1.13 \end{bmatrix} \quad \text{(Region \#2)} \\
1.00 & \text{if } \begin{bmatrix} 0.29 & 0.96 \\ 1.00 & 0.00 \\ -1.00 & 0.00 \\ -0.27 & -0.96 \\ -0.45 & -0.89 \\ 0.36 & 0.93 \end{bmatrix} x \leq \begin{bmatrix} -0.36 \\ 5.00 \\ 5.00 \\ 1.13 \\ 1.65 \\ -0.58 \end{bmatrix} \quad \text{(Region \#3)} \\
1.00 & \text{if } \begin{bmatrix} -1.00 & 0.00 \\ -0.21 & -0.98 \\ 0.45 & 0.89 \end{bmatrix} x \leq \begin{bmatrix} 5.00 \\ 0.98 \\ -1.65 \end{bmatrix} \quad \text{(Region \#4)} \\
-1.00 & \text{if } \begin{bmatrix} -0.29 & -0.96 \\ 1.00 & 0.00 \\ -1.00 & 0.00 \\ 0.27 & 0.96 \\ 0.45 & 0.89 \\ -0.36 & -0.93 \end{bmatrix} x \leq \begin{bmatrix} -0.36 \\ 5.00 \\ 5.00 \\ 1.13 \\ 1.65 \\ -0.58 \end{bmatrix} \quad \text{(Region \#5)} \\
-1.00 & \text{if } \begin{bmatrix} 1.00 & 0.00 \\ 0.21 & 0.98 \\ -0.45 & -0.89 \end{bmatrix} x \leq \begin{bmatrix} 5.00 \\ 0.98 \\ -1.65 \end{bmatrix} \quad \text{(Region \#6)} \\
-1.00 & \text{if } \begin{bmatrix} 1.00 & 0.00 \\ -1.00 & 0.00 \\ 0.71 & 0.71 \\ -0.21 & -0.98 \\ 0.21 & 0.98 \\ 0.32 & 0.95 \\ -0.27 & -0.96 \end{bmatrix} x \leq \begin{bmatrix} 5.00 \\ 5.00 \\ 3.54 \\ -0.98 \\ 1.67 \\ 1.79 \\ -1.13 \end{bmatrix} \quad \text{(Region \#7)} \\
\begin{bmatrix} -0.45 & -1.44 \end{bmatrix} x - 0.45 & \text{if } \begin{bmatrix} 1.00 & 0.00 \\ 0.29 & 0.96 \\ -0.71 & -0.70 \end{bmatrix} x \leq \begin{bmatrix} 5.00 \\ 0.36 \\ -1.90 \end{bmatrix} \quad \text{(Region \#8)} \\
\begin{bmatrix} -0.45 & -1.44 \end{bmatrix} x + 0.45 & \text{if } \begin{bmatrix} -1.00 & 0.00 \\ -0.29 & -0.96 \\ 0.71 & 0.70 \end{bmatrix} x \leq \begin{bmatrix} 5.00 \\ 0.36 \\ -1.90 \end{bmatrix} \quad \text{(Region \#9)}
\end{cases}$$

The union of the regions depicted in Fig. 11.6(b) is $\mathcal{X}_0$. Note that from Theorem 11.2 the set $\mathcal{X}_0$ is also the domain of attraction of the RHC law.

Case 3: $\mathcal{X}_f = \mathbb{R}^n$, $N = 6$. The QP problem associated with the RHC has the form (10.52) with

$$H = \begin{bmatrix} 5.30 & -0.89 & 0.01 & -1.10 & -0.50 & 0.21 \\ -0.89 & 2.48 & -0.01 & 1.56 & 0.00 & -1.11 \\ 0.01 & -0.01 & 5.31 & -0.01 & 0.00 & 3.25 \\ -1.10 & 1.56 & -0.01 & 2.48 & 0.01 & -0.89 \\ -0.50 & 0.00 & 0.00 & 0.01 & 1.53 & -0.00 \\ 0.21 & -1.11 & 3.25 & -0.89 & -0.00 & 7.03 \end{bmatrix}, \; F = \begin{bmatrix} -0.51 & -0.00 & 0.00 & -0.01 & -0.50 & 0.00 \\ -0.52 & -0.00 & 0.00 & -0.01 & -0.49 & 0.00 \end{bmatrix}, \; Y = \begin{bmatrix} 4.51 & 3.52 \\ 3.52 & 4.55 \end{bmatrix} \tag{11.37}$$

$$G_0 = \begin{bmatrix}
-0.33 & 0.67 & -1.00 & 0.33 & 0.00 & 0.33 \\
0.33 & -0.67 & 1.00 & -0.33 & 0.00 & -0.33 \\
-0.33 & 0.67 & 0.00 & 0.33 & 0.00 & -0.67 \\
0.33 & -0.67 & 0.00 & -0.33 & 0.00 & -0.33 \\
0.33 & -0.67 & 0.00 & -0.33 & 0.00 & 0.67 \\
-0.33 & 0.67 & 0.00 & 0.33 & 0.00 & 0.33 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & -1.00 \\
0.00 & 0.00 & -1.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 \\
0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 \\
-0.33 & 0.67 & 0.00 & 0.33 & 0.00 & -0.67 \\
0.33 & -0.67 & 0.00 & -0.33 & 0.00 & -0.33 \\
0.33 & -0.67 & 0.00 & -0.33 & 0.00 & 0.67 \\
-0.33 & 0.67 & 0.00 & 0.33 & 0.00 & 0.33 \\
0.00 & -1.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
-0.67 & 0.33 & 0.00 & 0.67 & 0.00 & -0.33 \\
0.33 & 0.33 & 0.00 & -0.33 & 0.00 & -0.33 \\
0.67 & -0.33 & 0.00 & -0.67 & 0.00 & 0.33 \\
-0.33 & -0.33 & 0.00 & 0.33 & 0.00 & 0.33 \\
-0.67 & 0.33 & 0.00 & 0.67 & 0.00 & -0.33 \\
0.33 & 0.33 & 0.00 & -0.33 & 0.00 & -0.33 \\
0.67 & -0.33 & 0.00 & -0.67 & 0.00 & 0.33 \\
-0.33 & -0.33 & 0.00 & 0.33 & 0.00 & 0.33 \\
0.00 & 0.00 & 0.00 & -1.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 \\
-1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.33 & 0.33 & 0.00 & 0.67 & 0.00 & -0.33 \\
1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
-0.33 & -0.33 & 0.00 & -0.67 & 0.00 & 0.33 \\
-1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.33 & 0.33 & 0.00 & 0.67 & 0.00 & -0.33 \\
1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
-0.33 & -0.33 & 0.00 & -0.67 & 0.00 & 0.33 \\
0.83 & 0.33 & 0.00 & 0.67 & 0.50 & -0.33 \\
-0.83 & -0.33 & 0.00 & -0.67 & -0.50 & 0.33 \\
-0.50 & 0.00 & 0.00 & 0.00 & 0.50 & 0.00 \\
-0.50 & 0.00 & 0.00 & 0.00 & -0.50 & 0.00 \\
0.50 & 0.00 & 0.00 & 0.00 & -0.50 & 0.00 \\
0.50 & 0.00 & 0.00 & 0.00 & 0.50 & 0.00 \\
-0.50 & 0.00 & 0.00 & 0.00 & 0.50 & 0.00 \\
-0.50 & 0.00 & 0.00 & 0.00 & -0.50 & 0.00 \\
0.50 & 0.00 & 0.00 & 0.00 & -0.50 & 0.00 \\
0.50 & 0.00 & 0.00 & 0.00 & 0.50 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & -1.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
-0.50 & 0.00 & 0.00 & 0.00 & 0.50 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.50 & 0.00 & 0.00 & 0.00 & -0.50 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
-0.50 & 0.00 & 0.00 & 0.00 & 0.50 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.50 & 0.00 & 0.00 & 0.00 & -0.50 & 0.00 \\
-0.50 & 0.00 & 0.00 & 0.00 & 0.50 & 0.00 \\
0.50 & 0.00 & 0.00 & 0.00 & -0.50 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & -0.93 & 0.00 & 0.00 & -0.36 \\
0.00 & 0.00 & 0.93 & 0.00 & 0.00 & 0.36 \\
0.00 & 0.00 & 0.70 & 0.00 & 0.00 & 0.71 \\
0.00 & 0.00 & -0.70 & 0.00 & 0.00 & -0.71
\end{bmatrix}, \; E_0 = \begin{bmatrix}
0.00 & 0.00 \\
0.00 & 0.00 \\
0.00 & 0.00 \\
0.00 & 0.00 \\
0.00 & 0.00 \\
0.00 & 0.00 \\
0.00 & 0.00 \\
0.00 & 0.00 \\
0.00 & 0.00 \\
0.00 & 0.00 \\
0.00 & 0.00 \\
0.00 & 0.00 \\
0.00 & 0.00 \\
0.00 & 0.00 \\
0.00 & 0.00 \\
0.00 & 0.00 \\
0.00 & 0.00 \\
0.00 & 0.00 \\
0.00 & 0.00 \\
0.00 & 0.00 \\
0.00 & 0.00 \\
0.00 & 0.00 \\
0.00 & 0.00 \\
0.00 & 0.00 \\
0.00 & 0.00 \\
0.00 & 0.00 \\
0.00 & 0.00 \\
0.00 & 0.00 \\
0.00 & 0.00 \\
0.00 & 0.00 \\
0.00 & 0.00 \\
0.00 & 0.00 \\
0.00 & 0.00 \\
0.00 & 0.00 \\
0.50 & 0.50 \\
-0.50 & -0.50 \\
0.50 & 0.50 \\
-0.50 & -0.50 \\
-0.50 & -0.50 \\
0.50 & 0.50 \\
0.50 & 0.50 \\
-0.50 & -0.50 \\
-0.50 & -0.50 \\
0.50 & 0.50 \\
0.00 & 0.00 \\
0.00 & 0.00 \\
1.00 & 1.00 \\
-0.50 & -0.50 \\
-1.00 & -1.00 \\
0.50 & 0.50 \\
1.00 & 1.00 \\
-0.50 & -0.50 \\
-1.00 & -1.00 \\
0.50 & 0.50 \\
-0.50 & -1.50 \\
0.50 & 1.50 \\
1.00 & 0.00 \\
0.00 & 1.00 \\
-1.00 & 0.00 \\
0.00 & -1.00 \\
1.00 & 0.00 \\
0.00 & 1.00 \\
-1.00 & 0.00 \\
0.00 & -1.00 \\
0.00 & 0.00 \\
0.00 & 0.00 \\
0.00 & 0.00 \\
0.00 & 0.00
\end{bmatrix}, \; W_0 = \begin{bmatrix}
1.00 \\
1.00 \\
10.00 \\
10.00 \\
10.00 \\
10.00 \\
10.00 \\
10.00 \\
10.00 \\
10.00 \\
5.00 \\
5.00 \\
5.00 \\
5.00 \\
1.00 \\
1.00 \\
10.00 \\
10.00 \\
10.00 \\
10.00 \\
5.00 \\
5.00 \\
5.00 \\
5.00 \\
1.00 \\
1.00 \\
10.00 \\
10.00 \\
10.00 \\
10.00 \\
5.00 \\
5.00 \\
5.00 \\
5.00 \\
1.00 \\
1.00 \\
10.00 \\
10.00 \\
10.00 \\
10.00 \\
5.00 \\
5.00 \\
5.00 \\
5.00 \\
1.00 \\
1.00 \\
10.00 \\
10.00 \\
10.00 \\
10.00 \\
5.00 \\
5.00 \\
5.00 \\
5.00 \\
1.00 \\
1.00 \\
10.00 \\
10.00 \\
10.00 \\
10.00 \\
5.00 \\
5.00 \\
5.00 \\
5.00 \\
0.58 \\
0.58 \\
1.90 \\
1.90
\end{bmatrix} \tag{11.38}$$

The corresponding polyhedral partition of the state-space is depicted in Fig. 11.6(c). The RHC law is:

$$u = \begin{cases}
\begin{bmatrix} -0.61 & -1.61 \end{bmatrix} x & \text{if } \begin{bmatrix} 1.00 & 0.00 \\ -1.00 & 0.00 \\ -0.36 & -0.93 \\ 0.36 & 0.93 \\ 0.71 & 0.70 \\ -0.71 & -0.70 \end{bmatrix} x \leq \begin{bmatrix} 5.00 \\ 5.00 \\ 0.58 \\ 0.58 \\ 1.90 \\ 1.90 \end{bmatrix} \quad \text{(Region \#1)} \\
1.00 & \text{if } \begin{bmatrix} 1.00 & 0.00 \\ -0.71 & -0.71 \\ -0.21 & -0.98 \\ -0.32 & -0.95 \\ 0.27 & 0.96 \\ -1.00 & 0.00 \\ 0.23 & 0.97 \end{bmatrix} x \leq \begin{bmatrix} 5.00 \\ 3.54 \\ 1.67 \\ 1.79 \\ -1.13 \\ 5.00 \\ -1.03 \end{bmatrix} \quad \text{(Region \#2)} \\
1.00 & \text{if } \begin{bmatrix} -1.00 & 0.00 \\ -0.23 & -0.97 \\ 0.45 & 0.89 \end{bmatrix} x \leq \begin{bmatrix} 5.00 \\ 1.03 \\ -1.65 \end{bmatrix} \quad \text{(Region \#3)} \\
1.00 & \text{if } \begin{bmatrix} 1.00 & 0.00 \\ -0.71 & -0.71 \\ -0.45 & -0.89 \\ -0.18 & -0.98 \\ 0.18 & 0.98 \\ -0.24 & -0.97 \\ 0.21 & 0.98 \end{bmatrix} x \leq \begin{bmatrix} 5.00 \\ 3.54 \\ 2.68 \\ 2.19 \\ -1.62 \\ 2.10 \\ -1.67 \end{bmatrix} \quad \text{(Region \#4)} \\
1.00 & \text{if } \begin{bmatrix} 0.29 & 0.96 \\ 1.00 & 0.00 \\ -1.00 & 0.00 \\ -0.27 & -0.96 \\ -0.45 & -0.89 \\ 0.36 & 0.93 \end{bmatrix} x \leq \begin{bmatrix} -0.36 \\ 5.00 \\ 5.00 \\ 1.13 \\ 1.65 \\ -0.58 \end{bmatrix} \quad \text{(Region \#5)} \\
1.00 & \text{if } \begin{bmatrix} 0.32 & 0.95 \\ -0.71 & -0.71 \\ -0.18 & -0.98 \end{bmatrix} x \leq \begin{bmatrix} -1.79 \\ 3.54 \\ 1.62 \end{bmatrix} \quad \text{(Region \#6)} \\
-1.00 & \text{if } \begin{bmatrix} -1.00 & 0.00 \\ 0.71 & 0.71 \\ 0.21 & 0.98 \\ 0.32 & 0.95 \\ -0.27 & -0.96 \\ 1.00 & 0.00 \\ -0.23 & -0.97 \end{bmatrix} x \leq \begin{bmatrix} 5.00 \\ 3.54 \\ 1.67 \\ 1.79 \\ -1.13 \\ 5.00 \\ -1.03 \end{bmatrix} \quad \text{(Region \#7)} \\
-1.00 & \text{if } \begin{bmatrix} 1.00 & 0.00 \\ 0.23 & 0.97 \\ -0.45 & -0.89 \end{bmatrix} x \leq \begin{bmatrix} 5.00 \\ 1.03 \\ -1.65 \end{bmatrix} \quad \text{(Region \#8)} \\
-1.00 & \text{if } \begin{bmatrix} -1.00 & 0.00 \\ 0.71 & 0.71 \\ 0.45 & 0.89 \\ -0.18 & -0.98 \\ 0.18 & 0.98 \\ 0.24 & 0.97 \\ -0.21 & -0.98 \end{bmatrix} x \leq \begin{bmatrix} 5.00 \\ 3.54 \\ 2.68 \\ -1.62 \\ 2.19 \\ 2.10 \\ -1.67 \end{bmatrix} \quad \text{(Region \#9)} \\
-1.00 & \text{if } \begin{bmatrix} -0.29 & -0.96 \\ 1.00 & 0.00 \\ -1.00 & 0.00 \\ 0.27 & 0.96 \\ 0.45 & 0.89 \\ -0.36 & -0.93 \end{bmatrix} x \leq \begin{bmatrix} -0.36 \\ 5.00 \\ 5.00 \\ 1.13 \\ 1.65 \\ -0.58 \end{bmatrix} \quad \text{(Region \#10)} \\
-1.00 & \text{if } \begin{bmatrix} -0.32 & -0.95 \\ 0.71 & 0.71 \\ 0.18 & 0.98 \end{bmatrix} x \leq \begin{bmatrix} -1.79 \\ 3.54 \\ 1.62 \end{bmatrix} \quad \text{(Region \#11)} \\
\begin{bmatrix} -0.45 & -1.44 \end{bmatrix} x - 0.45 & \text{if } \begin{bmatrix} 1.00 & 0.00 \\ 0.29 & 0.96 \\ -0.71 & -0.70 \end{bmatrix} x \leq \begin{bmatrix} 5.00 \\ 0.36 \\ -1.90 \end{bmatrix} \quad \text{(Region \#12)} \\
\begin{bmatrix} -0.45 & -1.44 \end{bmatrix} x + 0.45 & \text{if } \begin{bmatrix} -1.00 & 0.00 \\ -0.29 & -0.96 \\ 0.71 & 0.70 \end{bmatrix} x \leq \begin{bmatrix} 5.00 \\ 0.36 \\ -1.90 \end{bmatrix} \quad \text{(Region \#13)}
\end{cases}$$

Comparing the feasibility regions $\mathcal{X}_0$ in Figure 11.6 we notice that in Case 2 we obtain a larger region than in Case 1 and that in Case 3 we obtain a feasibility region larger than Case 1 and Case 2. This can be easily explained from the theory presented in this and the previous chapter. In particular we have seen that if a control invariant set is chosen as terminal constraint $\mathcal{X}_f$, the size of the feasibility region increases with the number of control moves (increase from Case 2 to Case 3) (Remark 10.8). Also, the size of the feasibility region increases with the size of the target set (increase from Case 1 to Case 2).

11.5 State Feedback Solution of RHC, $1, \infty$-Norm Case

The state feedback receding horizon controller (11.7)–(11.10) with cost (11.8) for system (11.1) is

$$u(t) = f_0^*(x(t)) \tag{11.39}$$

where $f_0^*(x_0) : \mathbb{R}^n \to \mathbb{R}^m$ is the piecewise affine solution to the CFTOC (11.7) and is computed as explained in Section 10.5. As in the 2-norm case the explicit form (11.39) has the advantage of being easier to implement, and provides insight into the type of control action in different regions CR_i of the state space.

Example 11.4. Consider the double integrator system (11.12) subject to the input constraints

$$-1 \leq u(k) \leq 1 \tag{11.40}$$

and the state constraints

$$-5 \leq x(k) \leq 5 \tag{11.41}$$

We want to regulate the system to the origin by using the RHC controller (11.7)–(11.10) with cost (11.9), $p = \infty$, $Q = \left[\begin{smallmatrix}1 & 0\\ 0 & 1\end{smallmatrix}\right]$, $R = 20$. We consider two cases:
Case 1. $\mathcal{X}_f = \mathbb{R}^n$, $N = 6$ (determinedness index+1) and $P = Q$
Case 2. $\mathcal{X}_f = \mathbb{R}^n$, $N = 6$ and $P = \|P_\infty\|_\infty$ given in (9.34) measuring the infinite-time unconstrained optimal cost in (9.32).

From Corollary 11.2 in both cases persistent feasibility is guaranteed for all cost functions and $\mathcal{X}_0 = \mathcal{C}_\infty$. However, in Case 1 the terminal cost P does not satisfy (11.27) which is assumption (A3) in Theorem 11.2 and therefore the convergence to and the stability of the origin cannot be guaranteed. In order to satisfy Assumption (A3) in Theorem 11.2, in Case 2 we select the terminal cost to be equal to the infinite-time unconstrained optimal cost computed in Example 9.1.

Next we will detail the explicit solutions for the two cases.

Case 1. The LP problem associated with the RHC has the form (10.75) with $\bar{G}_0 \in \mathbb{R}^{124\times 18}$, $\bar{S}_0 \in \mathbb{R}^{124\times 2}$ and $c' = [\mathbf{0}_6\ \mathbf{1}_{12}]$. The corresponding polyhedral partition of the state-space is depicted in Fig. 11.7(a). The RHC law is:

$$
u = \begin{cases}
 & \text{if } \begin{bmatrix} 0.16 & 0.99 \\ -0.16 & -0.99 \\ -1.00 & 0.00 \\ 1.00 & 0.00 \end{bmatrix} x \le \begin{bmatrix} 0.82 \\ 0.82 \\ 5.00 \\ 5.00 \end{bmatrix} & \text{(Region \#1)} \\
\begin{bmatrix} -0.29 & -1.71 \end{bmatrix} x + 1.43 & \text{if } \begin{bmatrix} 1.00 & 0.00 \\ -0.16 & -0.99 \\ -1.00 & 0.00 \\ 0.16 & 0.99 \end{bmatrix} x \le \begin{bmatrix} 5.00 \\ -0.82 \\ 5.00 \\ 1.40 \end{bmatrix} & \text{(Region \#2)} \\
-1.00 & \text{if } \begin{bmatrix} -0.16 & -0.99 \\ 1.00 & 0.00 \\ 0.71 & 0.71 \\ -1.00 & 0.00 \\ 0.20 & 0.98 \\ 0.16 & 0.99 \\ 0.24 & 0.97 \\ 0.45 & 0.89 \\ 0.32 & 0.95 \end{bmatrix} x \le \begin{bmatrix} -1.40 \\ 5.00 \\ 4.24 \\ 5.00 \\ 2.94 \\ 3.04 \\ 2.91 \\ 3.35 \\ 3.00 \end{bmatrix} & \text{(Region \#3)} \\
\begin{bmatrix} -0.29 & -1.71 \end{bmatrix} x - 1.43 & \text{if } \begin{bmatrix} -1.00 & 0.00 \\ 0.16 & 0.99 \\ 1.00 & 0.00 \\ -0.16 & -0.99 \end{bmatrix} x \le \begin{bmatrix} 5.00 \\ -0.82 \\ 5.00 \\ 1.40 \end{bmatrix} & \text{(Region \#4)} \\
1.00 & \text{if } \begin{bmatrix} -0.32 & -0.95 \\ -0.24 & -0.97 \\ -0.20 & -0.98 \\ -0.16 & -0.99 \\ -1.00 & 0.00 \\ 0.16 & 0.99 \\ -0.71 & -0.71 \\ -0.45 & -0.89 \\ 1.00 & 0.00 \end{bmatrix} x \le \begin{bmatrix} 3.00 \\ 2.91 \\ 2.94 \\ 3.04 \\ 5.00 \\ -1.40 \\ 4.24 \\ 3.35 \\ 5.00 \end{bmatrix} & \text{(Region \#5)}
\end{cases}
$$

The union of the regions depicted in Fig. 11.7(a) is $\mathcal{X}_0$ and is shown in white in Fig. 11.7(c). Since N is equal to the determinedness index plus one, $\mathcal{X}_0$ is a positive invariant set for the closed-loop system and thus persistent feasibility is guaranteed for all $x(0) \in \mathcal{X}_0$. However, it can be noticed from Fig. 11.7(c) that convergence to the origin is not guaranteed. Starting from the initial conditions [-4,2], [-2, 2], [0, 0.5], [4,-2], [-1,-1] and [2,-0.5], the closed-loop system converges to either $[-5, 0]$ or $[5, 0]$.

Case 2. The LP problem associated with the RHC has the form (10.75) with $\bar{G}_0 \in \mathbb{R}^{174\times 18}$, $\bar{S}_0 \in \mathbb{R}^{174\times 2}$ and $c' = [\mathbf{0}_6\ \mathbf{1}_{12}]$. The RHC law is defined over 21 regions and the corresponding polyhedral partition of the state-space is depicted in Fig. 11.7(b).

The union of the regions depicted in Fig. 11.7(b) is $\mathcal{X}_0$ and is shown in white in Fig. 11.7(d). Since N is equal to the determinedness index plus one, $\mathcal{X}_0$ is a positive invariant set for the closed-loop system and thus persistent feasibility is guaranteed for all $x(0) \in \mathcal{X}_0$. Convergence to the origin is also guaranteed by the choice of P as shown by the closed-loop trajectories in Fig. 11.7(d) starting from the same initial conditions as in Case 1.

Idle Control and Multiple Optimizers*

There are two potential problems with the RHC control law based on linear programming: idle control and multiple solutions. The first corresponds to the situation when the optimal move $u(t)$ is always zero, the second to the degeneracy of the LP problem. The explicit mp-LP approach allows us to easily recognize both situations.

By analyzing the explicit solution of the RHC law, one can locate immediately the critical regions where the matrices F_0^i, g_0^i in (11.39) are zero, i.e., idle control. A different tuning of the controller is required when such polyhedral regions appear and the overall performance is not satisfactory. The second issue is the presence of multiple solutions, that might arise from the degeneracy of the dual problem (6.14). Multiple optimizers are undesirable, as they might lead to a fast switching between the different optimal control moves when the optimization program (10.75) is solved on-line, unless interior-point methods are used. The mp-LP solvers [106, 56] can detect critical regions of degeneracy and partition them into sub-regions where a unique optimizer is defined. Example 11.5 illustrates a RHC law where multiple optimizers and idle control occur.

Example 11.5. Consider the double integrator of Example 11.4, with $N = 1$, $Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, $R = 1$, $P = Q$ subject to the input constraints

$$\mathcal{U} = \{u : \ -1 \leq u \leq 1\} \tag{11.42}$$

and the state constraints

$$\mathcal{X} = \{x : \ \begin{bmatrix} -10 \\ -10 \end{bmatrix} \leq x \leq \begin{bmatrix} 10 \\ 10 \end{bmatrix}\} \tag{11.43}$$

The associated mp-LP problem is

$$\begin{array}{ll} \min\limits_{\varepsilon_1, \varepsilon_2, u_0} & \varepsilon_1 + \varepsilon_2 \\ \\ \text{subj. to} & \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & -1 \\ 0 & -1 & 0 \\ 0 & -1 & -1 \\ 0 & -1 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ u_0 \end{bmatrix} \leq \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 10 \\ 10 \\ 10 \\ 10 \\ 1 \\ 1 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 1 \\ 0 & 1 \\ -1 & -1 \\ 0 & -1 \\ 0 & -1 \\ -1 & -1 \\ 0 & 1 \\ 1 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} x(t) \end{array} \tag{11.44}$$

The solution of (11.44) gives rise to idle control and multiple optimizers.

In fact, the corresponding polyhedral partition of the state-space is depicted in Fig. 11.8. The RHC law is

$$u = \begin{cases} \text{degenerate if} \begin{bmatrix} -1.00 & -2.00 \\ 1.00 & 0.00 \\ 1.00 & 1.00 \\ -1.00 & -1.00 \\ 0.00 & 1.00 \end{bmatrix} x \leq \begin{bmatrix} 0.00 \\ 0.00 \\ 10.00 \\ 10.00 \\ 11.00 \end{bmatrix} & \text{(Region \#1)} \\ 0 \qquad \text{if} \begin{bmatrix} 1.00 & 0.00 \\ 1.00 & 2.00 \\ -1.00 & -1.00 \end{bmatrix} x \leq \begin{bmatrix} 0.00 \\ 0.00 \\ 10.00 \end{bmatrix} & \text{(Region \#2)} \\ \text{degenerate if} \begin{bmatrix} -1.00 & 0.00 \\ 1.00 & 2.00 \\ 1.00 & 1.00 \\ -1.00 & -1.00 \\ 0.00 & -1.00 \end{bmatrix} x \leq \begin{bmatrix} 0.00 \\ 0.00 \\ 10.00 \\ 10.00 \\ 11.00 \end{bmatrix} & \text{(Region \#3)} \\ 0 \qquad \text{if} \begin{bmatrix} -1.00 & -2.00 \\ -1.00 & 0.00 \\ 1.00 & 1.00 \end{bmatrix} x \leq \begin{bmatrix} 0.00 \\ 0.00 \\ 10.00 \end{bmatrix} & \text{(Region \#4)} \end{cases}$$

Note the presence of idle control in regions #2, #4 and multiple optimizers in regions #1, #3. Two sub-partitions of the degenerate regions #1, #3 are possible. Region #1 can be partitioned as

$$u_{1A} = \begin{cases} 0 \qquad \text{if} \begin{bmatrix} -1.00 & 0.00 \\ 1.00 & 2.00 \\ 0.00 & -1.00 \end{bmatrix} x \leq \begin{bmatrix} 0.00 \\ 0.00 \\ 10.00 \end{bmatrix} & \text{(Region \#1a)} \\ \begin{bmatrix} 0.00 & -1.00 \end{bmatrix} x + 10.00 \ \text{if} \begin{bmatrix} 0.00 & -2.00 \\ 1.00 & 1.00 \\ -1.00 & -1.00 \\ 0.00 & 1.00 \end{bmatrix} x \leq \begin{bmatrix} -20.00 \\ 10.00 \\ 10.00 \\ 11.00 \end{bmatrix} & \text{(Region \#1b)} \end{cases}$$

or

$$u_{1B} = \begin{cases} -1.00 \qquad \text{if} \begin{bmatrix} -1.00 & -2.00 \\ 1.00 & 0.00 \\ 0.00 & 1.00 \end{bmatrix} x \leq \begin{bmatrix} -1.00 \\ -1.00 \\ 11.00 \end{bmatrix} & \text{(Region \#1a)} \\ 0 \qquad \text{if} \begin{bmatrix} 1.00 & 2.00 \\ -1.00 & -2.00 \\ 1.00 & 0.00 \\ 0.00 & 1.00 \end{bmatrix} x \leq \begin{bmatrix} 1.00 \\ 0.00 \\ 0.00 \\ 10.00 \end{bmatrix} & \text{(Region \#1b)} \\ \begin{bmatrix} 0.00 & -1.00 \end{bmatrix} x + 10.00 \ \text{if} \begin{bmatrix} 1.00 & 2.00 \\ 0.00 & -2.00 \\ -1.00 & -1.00 \end{bmatrix} x \leq \begin{bmatrix} 1.00 \\ -20.00 \\ 10.00 \end{bmatrix} & \text{(Region \#1c)} \\ 0 \qquad \text{if} \begin{bmatrix} -1.00 & 0.00 \\ -1.00 & -2.00 \\ 1.00 & 0.00 \\ 0.00 & 1.00 \end{bmatrix} x \leq \begin{bmatrix} 1.00 \\ -1.00 \\ 0.00 \\ 10.00 \end{bmatrix} & \text{(Region \#1d)} \\ \begin{bmatrix} 0.00 & -1.00 \end{bmatrix} x + 10.00 \ \text{if} \begin{bmatrix} -1.00 & 0.00 \\ 0.00 & -2.00 \\ 1.00 & 1.00 \end{bmatrix} x \leq \begin{bmatrix} 1.00 \\ -20.00 \\ 10.00 \end{bmatrix} & \text{(Region \#1e)} \end{cases}$$

Region #3 can be partitioned symmetrically as:

$$u_{3A} = \begin{cases} 0 & \text{if } \begin{bmatrix} -1.00 & -2.00 \\ 1.00 & 0.00 \\ 0.00 & 1.00 \end{bmatrix} x \leq \begin{bmatrix} 0.00 \\ 0.00 \\ 10.00 \end{bmatrix} & \text{(Region \#3a)} \\ \begin{bmatrix} 0.00 & -1.00 \end{bmatrix} x - 10.00 & \text{if } \begin{bmatrix} 0.00 & 2.00 \\ 1.00 & 1.00 \\ -1.00 & -1.00 \\ 0.00 & -1.00 \end{bmatrix} x \leq \begin{bmatrix} -20.00 \\ 10.00 \\ 10.00 \\ 11.00 \end{bmatrix} & \text{(Region \#3b)} \end{cases}$$

or

$$u_{3B} = \begin{cases} 1.00 & \text{if } \begin{bmatrix} -1.00 & 0.00 \\ 1.00 & 2.00 \\ 0.00 & -1.00 \end{bmatrix} x \leq \begin{bmatrix} -1.00 \\ -1.00 \\ 11.00 \end{bmatrix} & \text{(Region \#3a)} \\ 0 & \text{if } \begin{bmatrix} 1.00 & 0.00 \\ -1.00 & 0.00 \\ 1.00 & 2.00 \\ 0.00 & -1.00 \end{bmatrix} x \leq \begin{bmatrix} 1.00 \\ 0.00 \\ 0.00 \\ 10.00 \end{bmatrix} & \text{(Region \#3b)} \\ \begin{bmatrix} 0.00 & -1.00 \end{bmatrix} x - 10.00 & \text{if } \begin{bmatrix} 1.00 & 0.00 \\ 0.00 & 2.00 \\ -1.00 & -1.00 \end{bmatrix} x \leq \begin{bmatrix} 1.00 \\ -20.00 \\ 10.00 \end{bmatrix} & \text{(Region \#3c)} \\ 0 & \text{if } \begin{bmatrix} -1.00 & -2.00 \\ -1.00 & 0.00 \\ 1.00 & 2.00 \\ 0.00 & -1.00 \end{bmatrix} x \leq \begin{bmatrix} 1.00 \\ -1.00 \\ 0.00 \\ 10.00 \end{bmatrix} & \text{(Region \#3d)} \\ \begin{bmatrix} 0.00 & -1.00 \end{bmatrix} x - 10.00 & \text{if } \begin{bmatrix} -1.00 & -2.00 \\ 0.00 & 2.00 \\ 1.00 & 1.00 \end{bmatrix} x \leq \begin{bmatrix} 1.00 \\ -20.00 \\ 10.00 \end{bmatrix} & \text{(Region \#3e)} \end{cases}$$

The two possible partitions with unique optimizer for problem (11.44) are depicted in Figure 11.9. Note that controllers u_{1A} and u_{3A} are continuous in Regions 1 and 3, respectively, while controllers u_{1B} and u_{3B} are discontinuous in Regions 1 and 3, respectively.

11.6 RHC Extensions

In order to reduce the size of the optimization problem at the price of possibly reduced performance, the basic RHC formulation (11.7) is often modified as follows

$$\begin{array}{ll} \min\limits_{U_0} & p(x_{N_y}) + \sum\limits_{k=0}^{N_y-1} q(x_k, u_k) \\ \text{subj. to} & x_{k+1} = Ax_k + Bu_{k,t},\ \ k = 0, \ldots, N \\ & x_k \in \mathcal{X}, k = 0, \ldots, N_c \\ & u_k \in \mathcal{U},\ \ k = 0, \ldots, N_u \\ & u_k = Kx_k,\ \ N_u \leq k < N_y \end{array} \tag{11.45}$$

where K is some feedback gain, N_y, N_u, N_c are the prediction, input, and state constraint horizons, respectively, with $N_u \leq N_y$ and $N_c \leq N_y$. This formulation reduces the number of constraints and as a consequence makes the long horizon prediction used in the optimization less accurate as it is not forced to obey all the constraints. As this approximation affects only the states far in the future, it is hoped hat it will not affect significantly the present control action. Note, however, that all the theoretical properties derived for RHC do not hold for this approximate formulation as the constraints sets $\mathcal{X}$ and $\mathcal{U}$ vary with time. For Theorem 11.2 to hold, for example, it is essential that $\mathcal{X}$ and $\mathcal{U}$ are constant.

Formulation (11.45) can be extended naturally to situations where the control task is more demanding. As long as the control task can be expressed as an mp-QP or mp-LP, a piecewise affine controller results which can be easily implemented. As an example the bounds $y_{\min}$, $y_{\max}$, $\delta u_{\min}$, $\delta u_{\max}$, $u_{\min}$, $u_{\max}$ may change depending on the operating conditions, or in the case of a stuck actuator the constraints become $\delta u_{\min} = \delta u_{\max} = 0$. This possibility can again be built into the control law. The bounds can be treated as parameters in the QP and added to the vector x. The control law will have the form $u(t) = F(x(t), y_{\min}, y_{\max}, \delta u_{\min}, \delta u_{\max}, u_{\min}, u_{\max})$.

Example 11.6.

11.7 Offset-Free Reference Tracking

This section describes how the RHC problem has to be formulated to track constant references without offset under model mismatch. We distinguish between the number p of measured outputs, the number r of outputs which one desires to track (called "tracked outputs"), and the number n_d of disturbances. First we summarize the conditions that need to be satisfied to obtain offset-free RHC by using the arguments of the internal model principle. Then we provide a simple proof of zero steady-state offset when $r \leq p = n_d$. Extensive treatment of reference tracking for RHC can be found in in [13, 201, 203, 204, 181]. Consider the discrete-time time-invariant system

$$\begin{cases} x_m(t+1) = f(x_m(t), u(t)) \\ y_m(t) = g(x_m(t)) \\ z(t) = H y_m(t) \end{cases} \tag{11.46}$$

In (11.46), $x_m(t) \in \mathbb{R}^n$, $u(t) \in \mathbb{R}^m$ and $y_m(t) \in \mathbb{R}^p$ are the state, input, measured output vector, respectively. The controlled variables $z(t) \in \mathbb{R}^r$ are a linear combination of the measured variables. Without any loss of generality we assume H to have full row rank.

The objective is to design an RHC based on the linear system model (11.1) of (11.46) in order to have $z(t)$ track $r(t)$, where $r(t) \in \mathbb{R}^p$ is the reference signal, which we assume to converge to a constant, i.e. $r(t) \to r_\infty$ as $t \to \infty$. We require zero steady-state tracking error, i.e., $(z(t) - r(t)) \to 0$ for $t \to \infty$.

The Observer Design

The plant model (11.1) is augmented with a disturbance model in order to capture the mismatch between (11.46) and (11.1) in steady state. Several disturbance models have been presented in the literature [13, 190, 176, 204, 203, 268]. Here we follow [204] and use the form:

$$\begin{cases} x(t+1) = Ax(t) + Bu(t) + B_d d(t) \\ d(t+1) = d(t) \\ y(t) = Cx(t) + C_d d(t) \end{cases} \tag{11.47}$$

with $d(t) \in \mathbb{R}^{n_d}$. With abuse of notation we have used the same symbols for state and outputs of system (11.1) and system (11.47). Later we will focus on specific versions of the model (11.47).

The observer estimates both states and disturbances based on this augmented model. Conditions for the observability of (11.47) are given in the following theorem.

Theorem 11.3. *[199, 200, 204, 13] The augmented system (11.47) is observable if and only if (C, A) is observable and*

$$\begin{bmatrix} A - I & B_d \\ C & C_d \end{bmatrix} \tag{11.48}$$

has full column rank.

Proof: From the Hautus observability condition system (11.47) is observable iff

$$\begin{bmatrix} A' - \lambda I & 0 & C' \\ B_d' & I - \lambda I & C_d' \end{bmatrix} \text{ has full row rank } \forall \lambda \tag{11.49}$$

Again from the Hautus condition, the first set of rows is linearly independent iff (C, A) is observable. The second set of rows is linearly independent from the first n rows except possibly for $\lambda = 1$. Thus, for the augmented system the Hautus condition needs to be checked for $\lambda = 1$ only, where it becomes (11.48). □

Remark 11.4. Note that for condition (11.48) to be satisfied the number of disturbances in d needs to be smaller or equal to the number of available measurements in y, $n_d \leqslant p$. Condition (11.48) can be nicely interpreted. It requires that the model of the disturbance effect on the output $d \to y$ must not have a zero at $(1, 0)$. Alternatively we can look at the steady state of system (11.47)

$$\begin{bmatrix} A - I & B_d \\ C & C_d \end{bmatrix} \begin{bmatrix} x_\infty \\ d_\infty \end{bmatrix} = \begin{bmatrix} 0 \\ y_\infty \end{bmatrix} \tag{11.50}$$

where we have denoted the steady state values with a subscript ∞ and have omitted the forcing term u for simplicity. We note that from the observability condition (11.48) for system (11.47) equation (11.50) is required to have a unique solution, which means, that we must be able to deduce a unique value for the disturbance d_∞ from a measurement of y_∞ in steady state.

The following corollary follows directly from Theorem 11.3.

Corollary 11.3. *The augmented system (11.47) with $n_d = p$ and $C_d = I$ is observable if and only if (C, A) is observable and*

$$det \begin{bmatrix} A - I & B_d \\ C & I \end{bmatrix} = det(A - I - B_d C) \neq 0. \tag{11.51}$$

Remark 11.5. We note here how the observability requirement restricts the choice of the disturbance model. If the plant has no integrators, then $\det(A - I) \neq 0$ and we can choose $B_d = 0$. If the plant has integrators then B_d has to be chosen specifically to make $\det(A - I - B_d C) \neq 0$.

The state and disturbance estimator is designed based on the augmented model as follows:

$$\begin{bmatrix} \hat{x}(t+1) \\ \hat{d}(t+1) \end{bmatrix} = \begin{bmatrix} A & B_d \\ 0 & I \end{bmatrix} \begin{bmatrix} \hat{x}(t) \\ \hat{d}(t) \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u(t) + \begin{bmatrix} L_x \\ L_d \end{bmatrix} (-y_m(t) + C\hat{x}(t) + C_d \hat{d}(t)) \tag{11.52}$$

where L_x and L_d are chosen so that the estimator is stable. We remark that the results below are independent of the choice of the method for computing L_x and L_d. We then have the following property.

Lemma 11.4. *Suppose the observer (11.52) is stable. Then, rank$(L_d) = n_d$.*

Proof: From (11.52) it follows

$$\begin{bmatrix} \hat{x}(t+1) \\ \hat{d}(t+1) \end{bmatrix} = \begin{bmatrix} A + L_x C & B_d + L_x C_d \\ L_d C & I + L_d C_d \end{bmatrix} \begin{bmatrix} \hat{x}(t) \\ \hat{d}(t) \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u(t) - \begin{bmatrix} L_x \\ L_d \end{bmatrix} y_m(t) \tag{11.53}$$

By stability, the observer has no poles at $(1, 0)$ and therefore

$$\det \left(\begin{bmatrix} A - I + L_x C & B_d + L_x C_d \\ L_d C & L_d C_d \end{bmatrix} \right) \neq 0 \tag{11.54}$$

For (11.54) to hold, the last n_d rows of the matrix have to be of full row rank. A necessary condition is that L_d has full row rank. □

In the rest of this section, we will focus on the case $n_d = p$.

Lemma 11.5. *Suppose the observer (11.52) is stable. Choose $n_d = p$. The steady state of the observer (11.52) satisfies:*

$$\begin{bmatrix} A - I & B \\ C & 0 \end{bmatrix} \begin{bmatrix} \hat{x}_\infty \\ u_\infty \end{bmatrix} = \begin{bmatrix} -B_d \hat{d}_\infty \\ y_{m,\infty} - C_d \hat{d}_\infty \end{bmatrix} \tag{11.55}$$

where $y_{m,\infty}$ and u_∞ are the steady state measured output and input of the system (11.46), $\hat{x}_\infty$ and $\hat{d}_\infty$ are state and disturbance estimates from the observer (11.52) at steady state, respectively.

Proof: From (11.52) we note that the disturbance estimate $\hat{d}$ converges only if $L_d(-y_{m,\infty} + C\hat{x}_\infty + C_d\hat{d}_\infty) = 0$. As L_d is square by assumption and nonsingular by Lemma 11.4 this implies that at steady state, the observer estimates (11.52) satisfy

$$-y_{m,\infty} + C\hat{x}_\infty + C_d\hat{d}_\infty = 0 \tag{11.56}$$

Equation (11.55) follows directly from (11.56) and (11.52). □

The MPC design

Denote by $z_\infty = Hy_{m,\infty}$ and r_∞ the tracked measured outputs and their references at steady state, respectively. For offset-free tracking at steady state we want $z_\infty = r_\infty$. The observer condition (11.55) suggests that at steady state the MPC should satisfy

$$\begin{bmatrix} A-I & B \\ HC & 0 \end{bmatrix} \begin{bmatrix} x_\infty \\ u_\infty \end{bmatrix} = \begin{bmatrix} -B_d\hat{d}_\infty \\ r_\infty - HC_d\hat{d}_\infty \end{bmatrix} \tag{11.57}$$

where x_∞ is the MPC state at steady state. For x_∞ and u_∞ to exist for any $\hat{d}_\infty$ and r_∞ the matrix $\begin{bmatrix} A-I & B \\ HC & 0 \end{bmatrix}$ must be of full row rank which implies $m \geq r$.

The MPC is designed as follows

$$\begin{array}{ll} \min_{U_0} & (x_N - \bar{x}_t)'P(x_N - \bar{x}_t) + \sum_{k=0}^{N-1} (x_k - \bar{x}_t)'Q(x_k - \bar{x}_t) + (u_k - \bar{u}_t)'R(u_k - \bar{u}_t) \\ \text{subj. to} & x_{k+1} = Ax_k + Bu_k + B_d d_k,\ k = 0,\ldots,N \\ & x_k \in \mathcal{X},\ u_k \in \mathcal{U},\ k = 0,\ldots,N-1 \\ & x_N \in \mathcal{X}_f \\ & d_{k+1} = d_k,\ k = 0,\ldots,N \\ & x_0 = \hat{x}(t) \\ & d_0 = \hat{d}(t), \end{array} \tag{11.58}$$

with the targets $\bar{u}_t$ and $\bar{x}_t$ given by

$$\begin{bmatrix} A-I & B \\ HC & 0 \end{bmatrix} \begin{bmatrix} \bar{x}_t \\ \bar{u}_t \end{bmatrix} = \begin{bmatrix} -B_d\hat{d}(t) \\ r(t) - HC_d\hat{d}(t) \end{bmatrix} \tag{11.59}$$

and where $\|x\|_M^2 \triangleq x'Mx$, $Q \succeq 0$, $R \succ 0$, and $P \succ 0$.

Let $U^*(t) = \{u_0^*, \dots, u_{N-1}^*\}$ be the optimal solution of (11.58)-(11.59) at time t. Then, the first sample of $U^*(t)$ is applied to system (11.46)

$$u(t) = u_0^*. \tag{11.60}$$

Denote by $c_0(\hat{x}(t), \hat{d}(t), r(t)) = u_0^*(\hat{x}(t), \hat{d}(t), r(t))$ the control law when the estimated state and disturbance are $\hat{x}(t)$ and $\hat{d}(t)$, respectively. Then the closed-loop system obtained by controlling (11.46) with the MPC (11.58)-(11.59)-(11.60) and the observer (11.52) is:

$$\begin{aligned}
&x(t+1) = f(x(t), c_0(\hat{x}(t), \hat{d}(t), r(t)))\\
&\hat{x}(t+1) = (A + L_xC)\hat{x}(t) + (B_d + L_xC_d)\hat{d}(t) + Bc_0(\hat{x}(t), \hat{d}(t), r(t)) - L_xy_m(t)\\
&\hat{d}(t+1) = L_dC\hat{x}(t) + (I + L_dC_d)\hat{d}(t) - L_dy_m(t)
\end{aligned} \tag{11.61}$$

Often in practice, one desires to track all measured outputs with zero offset. Choosing $n_d = p = r$ is thus a natural choice. Such zero-offset property continues to hold if only a subset of the measured outputs are to be tracked, i.e., $n_d = p > r$. Next we provide a very simple proof for offset-free control when $n_d = p$.

Theorem 11.4. *Consider the case $n_d = p$. Assume that for $r(t) \to r_\infty$ as $t \to \infty$, the MPC problem (11.58)-(11.59) is feasible for all $t \in \mathbb{N}_+$, unconstrained for $t \geq j$ with $j \in \mathbb{N}_+$ and the closed-loop system (11.61) converges to $\hat{x}_\infty$, $\hat{d}_\infty$, $y_{m,\infty}$, i.e., $\hat{x}(t) \to \hat{x}_\infty$, $\hat{d}(t) \to \hat{d}_\infty$, $y_m(t) \to y_{m,\infty}$ as $t \to \infty$. Then $z(t) = Hy_m(t) \to r_\infty$ as $t \to \infty$.*

Proof: Consider the MPC problem (11.58)-(11.59). At steady state $u(t) \to u_\infty = c_0(\hat{x}_\infty, \hat{d}_\infty, r_\infty)$, $\bar{x}_t \to \bar{x}_\infty$ and $\bar{u}_t \to \bar{u}_\infty$. Note that the steady state controller input u_∞ (computed and implemented) might be different from the steady state target input $\bar{u}_\infty$.

The asymptotic values $\hat{x}_\infty$, $\bar{x}_\infty$, u_∞ and $\bar{u}_\infty$ satisfy the observer conditions (11.55)

$$\begin{bmatrix} A - I & B \\ C & 0 \end{bmatrix} \begin{bmatrix} \hat{x}_\infty \\ u_\infty \end{bmatrix} = \begin{bmatrix} -B_d\hat{d}_\infty \\ y_{m,\infty} - C_d\hat{d}_\infty \end{bmatrix} \tag{11.62}$$

and the controller requirement (11.59)

$$\begin{bmatrix} A - I & B \\ HC & 0 \end{bmatrix} \begin{bmatrix} \bar{x}_\infty \\ \bar{u}_\infty \end{bmatrix} = \begin{bmatrix} -B_d\hat{d}_\infty \\ r_\infty - HC_d\hat{d}_\infty \end{bmatrix} \tag{11.63}$$

Define $\delta x = \hat{x}_\infty - \bar{x}_\infty$, $\delta u = u_\infty - \bar{u}_\infty$ and the offset $\varepsilon = z_\infty - r_\infty$. Notice that the steady state target values $\bar{x}_\infty$ and $\bar{u}_\infty$ are both functions of r_∞ and $\hat{d}_\infty$ as given by (11.63). Left multiplying the second row of (11.62) by H and subtracting (11.63) from the result, we obtain

$$\begin{aligned}
&(A - I)\delta x + B\delta u = 0\\
&HC\delta x = \varepsilon.
\end{aligned} \tag{11.64}$$

Next we prove that δx =0 and thus $\varepsilon = 0$.

Consider the MPC problem (11.58)-(11.59) and the following change of variables $\delta x_k = x_k - \bar{x}_t$, $\delta u_k = u_k - \bar{u}_t$. Notice that $Hy_k - r(t) = HCx_k + HC_d d_k - r(t) = HC\delta x_k + HC\bar{x}_t + HC_d d_k - r(t) = HC\delta x_k$ from condition (11.59) with $\hat{d}(t) = d_k$. Similarly, one can show that $\delta x_{k+1} = A\delta x_k + B\delta u_k$. Then, the MPC problem (11.58) becomes:

$$\begin{array}{ll} \min_{\delta u_0,\ldots,\delta u_{N-1}} & \delta x_N' P \delta x_N + \sum_{k=0}^{N-1} \delta x_k' Q \delta x_k + \delta u_k' R \delta u_k \\ \text{subj. to} & \delta x_{k+1} = A\delta x_k + B\delta u_k, \quad 0 \le k \le N \\ & x_k \in \mathcal{X},\ u_k \in \mathcal{U},\ k = 0,\ldots,N-1 \\ & x_N \in \mathcal{X}_f \\ & \delta x_0 = \delta x(t), \\ & \delta x(t) = \hat{x}(t) - \bar{x}_t. \end{array} \tag{11.65}$$

Denote by K_{MPC} the unconstrained MPC controller (11.65), i.e., $\delta u_0^* = K_{MPC}\delta x(t)$. At steady state $\delta u_0^* \to u_\infty - \bar{u}_\infty = \delta u$ and $\delta x(t) \to \hat{x}_\infty - \bar{x}_\infty = \delta x$. Therefore, at steady state, $\delta u = K_{MPC}\delta x$. From (11.64)

$$(A - I + BK_{MPC})\delta x = 0. \tag{11.66}$$

By assumption the unconstrained system with the MPC controller converges. Thus K_{MPC} is a stabilizing control law, which implies that $(A - I + BK_{MPC})$ is nonsingular and hence $\delta x = 0$. □

Remark 11.6. Theorem 11.4 was proven in [204] by using a different approach.

Remark 11.7. Theorem 11.4 can be extended to prove local Lyapunov stability of the closed-loop system (11.61) under standard regularity assumptions on the state update function f in (11.61) [185].

Remark 11.8. The proof of Theorem 11.4 assumes only that the models used for the control design (11.1) and the observer design (11.47) are identical in steady state in the sense that they give rise to the same relation $z = z(u, d, r)$. It does not make any assumptions about the behavior of the real plant (11.46), i.e. the model-plant mismatch, with the exception that the closed-loop system (11.61) must converge to a fixed point. The models used in the controller and the observer could even be different as long as they satisfy the same steady state relation.

Remark 11.9. If condition (11.59) does not specify $\bar{x}_t$ and $\bar{u}_t$ uniquely, it is customary to determine $\bar{x}_t$ and $\bar{u}_t$ through an optimization problem, for example, minimizing the magnitude of $\bar{u}_t$ subject to the constraint (11.59) [204].

Remark 11.10. Note that in order to achieve no offset we augmented the model of the plant with as many disturbances (and integrators) as we have measurements ($n_d = p$) (cf. equation (11.53)). Our design procedure requires the addition of p integrators even if we wish to control only a subset of $r < p$ measured variables. This is actually not necessary as we suspect from basic system theory. The design procedure for the case $n_d = r < p$ is, however, more involved.

If the 2-norm in the objective function of (11.58) is replaced with a 1 or ∞ norm ($\|P(x_N - \bar{x}_t)\|_p + \sum_{k=0}^{N-1} \|Q(x_k - \bar{x}_t)\|_p + \|R(u_k - \bar{u}_t)\|_p$, where $p = 1$ or $p = \infty$), then our results continue to hold. In particular, Theorem 11.4 continues to hold. In fact, the unconstrained MPC controlled K_{MPC} in (11.65) is piecewise linear around the origin [40]. In particular, around the origin, $\delta u^*(t) \triangleq \delta u_0^* = K_{MPC}(\delta x(t))$ is a continuous piecewise linear function of the state variation δx:

$$K_{MPC}(\delta x) = F^j \delta x \quad \text{if} \quad H^j \delta x \leq K^j,\ j = 1, \ldots, N^r, \tag{11.67}$$

where H^j and K^j in equation (11.67) are the matrices describing the j-th polyhedron $CR^j = \{\delta x \in \mathbb{R}^n | H^j \delta x \leq K^j\}$ inside which the feedback optimal control law $\delta u^*(t)$ has the linear form $F^j \delta x(k)$. The polyhedra CR^j, $j = 1, \ldots, N^r$ are a partition of the set of feasible states of problem (11.58) and they all contain the origin. For Theorem 11.4 to hold, it is sufficient to require that all the linear feedback laws $F^j \delta x(k)$ for $i = j, \ldots, N^r$ are stabilizing.

Explicit Controller

Examining (11.58), (11.59) we note that the control law depends on $\hat{x}(t)$, $\hat{d}(t)$ and $r(t)$. Thus in order to achieve offset free tracking of r outputs out of p measurements we had to add the $p + r$ "parameters" $\hat{d}(t)$ and $r(t)$ to the usual parameters $\hat{x}(t)$.

There are more involved RHC design techniques to obtain offset-free control for models with $n_d < p$ and in particular, with minimum order disturbance models $n_d = r$. The total size of the parameter vector can thus be reduced to $n + 2r$. This is significant only if a small subset of the plant outputs are to be controlled. A greater reduction of parameters can be achieved by the following method. By Corollary 11.3, we are allowed to choose $B_d = 0$ in the disturbance model if the plant has no integrators. Recall the target conditions 11.59 with $B_d = 0$

$$\begin{bmatrix} A - I & B \\ HC & 0 \end{bmatrix} \begin{bmatrix} \bar{x}_t \\ \bar{u}_t \end{bmatrix} = \begin{bmatrix} 0 \\ r(t) - HC_d \hat{d}(t) \end{bmatrix}. \tag{11.68}$$

Clearly, any solution to (11.68) can be parameterized by $r(t) - HC_d \hat{d}(t)$. The explicit control law is written as $u(t) = c_0(\hat{x}(t), r(t) - HC_d \hat{d}(t))$, with only $n + r$ parameters. Since the observer is unconstrained, complexity is much less of an issue. Hence, a full disturbance model with $n_d = p$ can be chosen to yield offset-free control.

Remark 11.11. The choice of $B_d = 0$ might be limiting in practice. In [257], the authors have shown that for a wide range of systems, if $B_d = 0$ and the observer is designed through a Kalman filter, then the closed-loop system might suffer a dramatic performance deterioration.

Delta Input (δu) Formulation.

In the δu formulation, the MPC scheme uses the following linear time-invariant system model of (11.46):

$$\left\{ \begin{aligned} x(t+1) &= Ax(t) + Bu(t) \\ u(t) &= u(t-1) + \delta u(t) \\ y(t) &= Cx(t) \end{aligned} \right. \tag{11.69}$$

System (11.69) is controllable if (A, B) is controllable. The δu formulation often arises naturally in practice when the actuator is subject to uncertainty, e.g. the exact gain is unknown or is subject to drift. In these cases, it can be advantageous to consider changes in the control value as input to the plant. The absolute control value is estimated by the observer, which is expressed as follows

$$\begin{bmatrix} \hat{x}(t+1) \\ \hat{u}(t+1) \end{bmatrix} = \begin{bmatrix} A & B \\ 0 & I \end{bmatrix} \begin{bmatrix} \hat{x}(t) \\ \hat{u}(t) \end{bmatrix} + \begin{bmatrix} B \\ I \end{bmatrix} \delta u(t) + \begin{bmatrix} L_x \\ L_u \end{bmatrix} (-y_m(t) + C\hat{x}(t)) \tag{11.70}$$

The MPC problem is readily modified

$$\begin{array}{ll} \min_{\delta u_0,\ldots,\delta u_{N-1}} & \|y_k - r_k\|_Q^2 + \|\delta u_k\|_R^2 \\ \text{subj. to} & x_{k+1} = Ax_k + Bu_k,\ k \geq 0 \\ & y_k = Cx_k\ k \geq 0 \\ & x_k \in \mathcal{X},\ u_k \in \mathcal{U},\ k = 0, \ldots, N-1 \\ & x_N \in \mathcal{X}_f \\ & u_k = u_{k-1} + \delta u_k,\ k \geq 0 \\ & u_{-1} = \hat{u}(t) \\ & x_0 = \hat{x}(t) \end{array} \tag{11.71}$$

The control input applied to the system is

$$u(t) = \delta u_0^* + u(t-1). \tag{11.72}$$

The input estimate $\hat{u}(t)$ is not necessarily equal to the actual input $u(t)$. This scheme inherently achieves offset-free control, there is no need to add a disturbance model. To see this, we first note that $\delta u_0^* = 0$ in steady-state. Hence our analysis applies as the δu formulation is equivalent to a disturbance model in steady-state. This is due to the fact that any plant/model mismatch is lumped into $\hat{u}(t)$. Indeed this approach is equivalent to an input disturbance model ($B_d = B$, $C_d = 0$). If in (11.71) the measured $u(t)$ were substituted for its estimate, i.e. $u_{-1} = u(t-1)$, then the algorithm would show offset.

In this formulation the computation of a target input $\bar{u}_t$ and state $\bar{x}_t$ is not required. A disadvantage of the formulation is that it is not applicable when there is an excess of manipulated variables u compared to measured variables y, since detectability of the augmented system (11.69) is then lost.

Minimum-Time Controller

In minimum-time control, the cost function minimizes the predicted number of steps necessary to reach a target region, usually the invariant set associated to the unconstrained LQR controller [155]. This scheme can reduce the on-line computation time significantly, especially for explicit controllers (Section 10.6). While minimum-time MPC is computed and implemented differently from standard MPC controllers, there is no difference between the two control schemes at steady-state. In particular, one can choose the target region to be the unconstrained region of (11.58). When the state and disturbance estimates and reference are within this region, the control law is switched to (11.58). The analysis and methods presented in this section therefore apply directly.

11.8 Literature Review

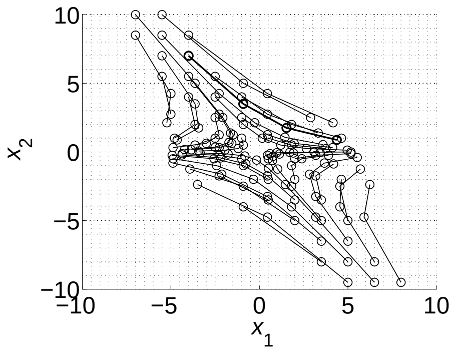
Although the basic idea of receding horizon control was already indicated by the theoretical work of Propoi [215] in 1963 it did not gain much attention until the mid-1970s, when Richalet and coauthors [223, 224] proposed the MPC technique (they called it "Model Predictive Heuristic Control (MPHC)"). Shortly thereafter, Cutler and Ramaker [83] introduced the predictive control algorithm called Dynamic Matrix Control (DMC) which has been hugely successful in the petro-chemical industry. A vast variety of different names and methodologies followed, such as Quadratic Dynamic Matrix Control (QDMC), Adaptive Predictive Control (APC), Generalized Predictive Control (GPC), Sequential Open Loop Optimization (SOLO), and others.

While the mentioned algorithms are seemingly different, they all share the same structural features: a model of the plant, the receding horizon idea, and an optimization procedure to obtain the control action by optimizing the system's predicted evolution.

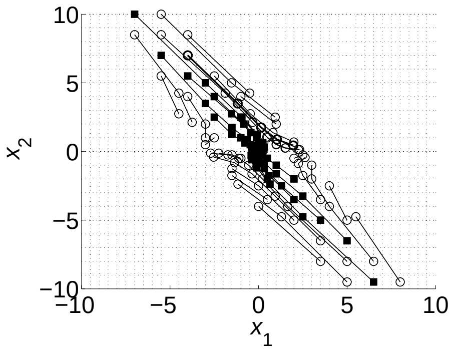
For complex constrained multivariable control problems, model predictive control has become the accepted standard in the process industries [216]. If in the finite time optimal control problem solved by MPC at each time step, system model and constraints are linear and the performance index is expressed as weighted sum of 2-norm, 1-norm or ∞-norm vectors, then the resulting optimization problem can be cast as a quadratic program (QP) or linear program (LP), respectively, for which a rich variety of efficient active-set and interior-point solvers are available.

Some of the first industrial MPC algorithms like IDCOM [224] and DMC [83] were developed for constrained MPC with quadratic performance indices. However, in those algorithms input and output constraints were treated in an indirect ad-hoc fashion. Only later, algorithms like QDMC [109] overcame this limitation by employing quadratic programming to solve constrained MPC problems with quadratic performance indices. Later an extensive theoretical effort was devoted to analyze such schemes, provide conditions for guaranteeing feasibility and closed-loop stability, and highlight the relations between MPC and the linear quadratic regulator (see the recent survey [188]).

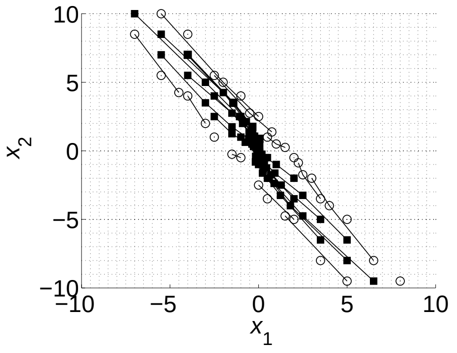
An extensive treatment of conditions for feasibility and stability of MPC can also be found in the surveys [188, 187]. Theorem 11.2 in this book presents the main result on feasibility and stability of MPC ant it has been extracted from [188, 187]. The main idea of Theorem 11.2 dates back to Keerthi and Gilbert [157], the first to propose specific choices of a terminal cost P and a terminal constraint $\mathcal{X}_f$, namely $\mathcal{X}_f = 0$ and $P = 0$. Under this assumption (and a few other mild ones) Keerthi and Gilbert prove the stability for general nonlinear performance functions and nonlinear models. The work of Keerthi and Gilbert has been followed by a number of stability result for RHC including the one in [157, 30, 1, 144, 78]. If properly analyzed all these results have the main components required in Theorem 11.2.



(a) *Setting 1*: $N = 2, R = 10$



(b) *Setting 2*: $N = 3, R = 2$



(c) *Setting 3*: $N = 4, R = 1$

Fig. 11.4 Example 11.2. Closed-loop trajectories for different settings of horizon N and weight R. *Boxes* (*Circles*) are initial points leading (not leading) to feasible closed-loop trajectories.

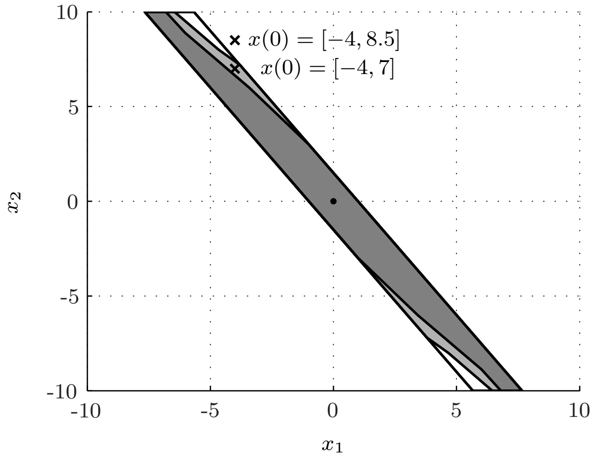
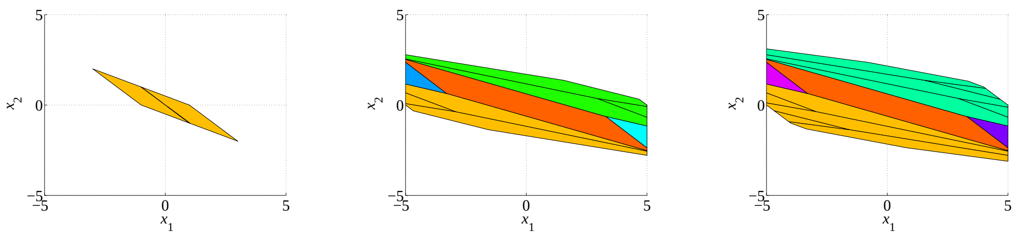


Fig. 11.5 Example 11.2. Maximal positive invariant sets $\mathcal{O}_\infty$ for different parameter settings: *Setting 1* (origin), *Setting 2* (dark-gray) and *Setting 3* (gray and dark-gray). Also depicted is the maximal control invariant set $\mathcal{C}_\infty$ (white and gray and dark-gray).

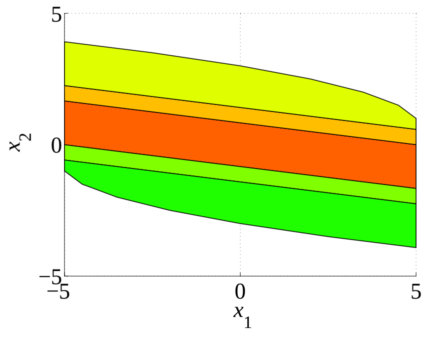


(a) Polyhedral partition of $\mathcal{X}_0$, Case 1. $N_0^r = 7$.

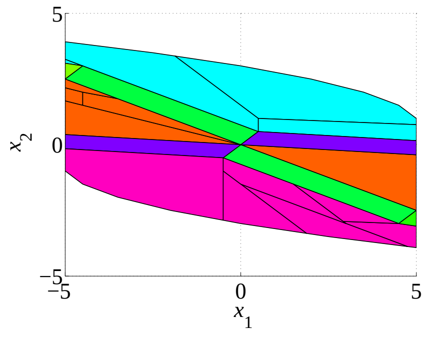
(b) Polyhedral partition of $\mathcal{X}_0$, Case 2. $N_0^r = 9$.

(c) Polyhedral partition of $\mathcal{X}_0$, Case 3. $N_0^r = 13$.

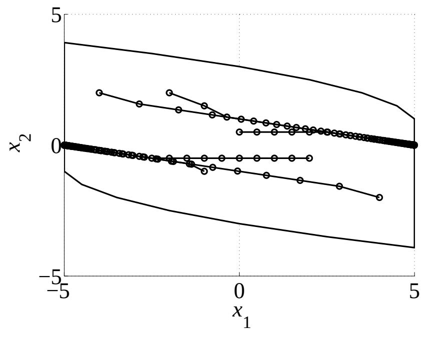
Fig. 11.6 Double integrator Example (11.3)



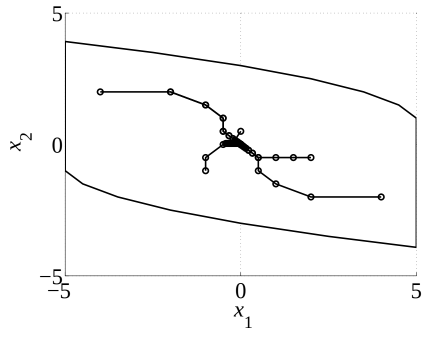
(a) Polyhedral partition of $\mathcal{X}_0$, Case 1



(b) Polyhedral partition of $\mathcal{X}_0$, Case 2



(c) Closed-loop Trajectories, Case 1



(d) Closed-loop Trajectories, Case 2

Fig. 11.7 Double integrator Example (11.4)

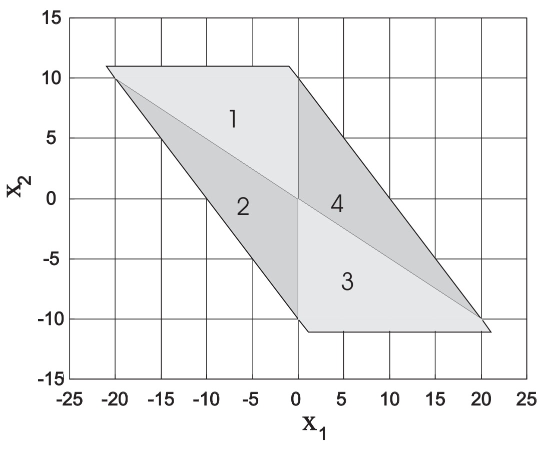
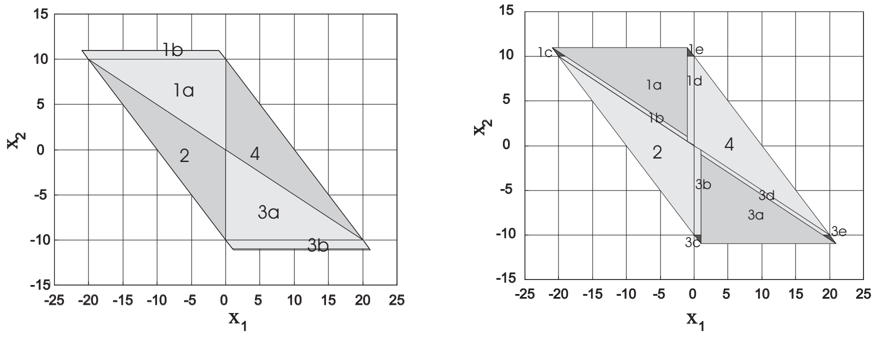


Fig. 11.8 Polyhedral partition of the state-space corresponding to the PPWA solution to problem (11.44)



(a) A possible solution of Example 11.5 obtained by choosing, for the degenerate regions 1 and 3, the control laws u_{1A} and u_{3A}

(b) A possible solution of Example 11.5 obtained by choosing, for the degenerate regions 1 and 3, the control laws u_{1B} and u_{3B}

Fig. 11.9 ∞-Norm control of the double integrator in Example 11.5: example of degeneracy

Chapter 12
Constrained Robust Optimal Control

12.1 Introduction

A control system is robust when the performance specifications are met for a specified range of model variations and a class of disturbance signals (uncertainty range). A typical robust strategy involves solving a min-max problem to optimize worst-case performance while enforcing input and state constraints for all possible disturbances. This chapter shows how to formulate and compute the state feedback solutions to min-max robust constrained optimal control problems for systems affected by additive bounded input disturbances and/or polyhedral parametric uncertainty.

Before formulating the robust finite time optimal control problem, we first introduce some fundamental concepts of robust set invariance theory.

12.2 Robust Invariant Sets

In this section we deal with two types of systems, namely, autonomous systems

$$x(k+1) = f_a(x(k), w(k)), \tag{12.1}$$

and systems subject to external controllable inputs

$$x(k+1) = f(x(k), u(k), w(k)). \tag{12.2}$$

Both systems are subject to the disturbance $w(k)$ and to the constraints

$$x(k) \in \mathcal{X},\ \ u(k) \in \mathcal{U},\ \ w(k) \in \mathcal{W}\ \forall\ k \geq 0. \tag{12.3}$$

The sets $\mathcal{X}$ and $\mathcal{U}$ and $\mathcal{W}$ are polytopes and contain the origin in their interior.

For the autonomous system (12.1) we will denote the one-step robust reachable set for initial states x contained in the set $\mathcal{S}$ as

$$\text{Reach}(\mathcal{S}, \mathcal{W}) \triangleq \{x \in \mathbb{R}^n\ :\ \exists\, x(0) \in \mathcal{S},\ \exists\, w \in \mathcal{W} \text{ such that } x = f_a(x(0), w)\}.$$

For the system (12.2) with inputs we will denote the one-step robust reachable set for initial states x contained in the set $\mathcal{S}$ as

$$\text{Reach}(\mathcal{S}, \mathcal{W}) \triangleq \{x \in \mathbb{R}^n\ :\ \exists\, x(0) \in \mathcal{S},\ \exists\, u \in \mathcal{U},\ \exists\, w \in \mathcal{W}, \text{ such that } x = f(x(0), u, w)\}.$$

Thus, all the states contained in $\mathcal{S}$ are mapped into the set $\text{Reach}(\mathcal{S}, \mathcal{W})$ under the map f_a for all disturbances $w \in \mathcal{W}$, and under the map f for all inputs $u \in \mathcal{U}$ and for all disturbances $w \in \mathcal{W}$.

"Pre" sets are the dual of one-step robust reachable sets.

$$\text{Pre}(\mathcal{S}, \mathcal{W}) \triangleq \{x \in \mathbb{R}^n\ :\ f_a(x, w) \in \mathcal{S},\ \forall w \in \mathcal{W}\} \tag{12.4}$$

defines the set of system (12.1) states which evolve into the target set $\mathcal{S}$ in one time step for all possible disturbance $w \in \mathcal{W}$.

Similarly, for the system (12.2) the set of states which can be robustly driven into the target set $\mathcal{S}$ in one time step is defined as

$$\text{Pre}(\mathcal{S},\mathcal{W}) \triangleq \{x \in \mathbb{R}^n \ : \ \exists u \in \mathcal{U} \text{ s.t. } f(x,u,w) \subseteq \mathcal{S}, \ \forall w \in \mathcal{W}\}. \tag{12.5}$$

Computation of Robust Pre and Reach for Linear Systems without Inputs

Example 12.1. Consider the second order autonomous system

$$x(t+1) = Ax(t) + w(t) = \begin{bmatrix} 0.5 & 0 \\ 1 & -0.5 \end{bmatrix} x(t) + w(t) \tag{12.6}$$

subject to the state constraints

$$x(t) \in \mathcal{X} = \left\{ x \ : \ \begin{bmatrix} -10 \\ -10 \end{bmatrix} \leq x \leq \begin{bmatrix} 10 \\ 10 \end{bmatrix} \right\}, \ \forall t \geq 0 \tag{12.7}$$

and where the additive disturbance belongs to the set

$$w(t) \in \mathcal{W} = \left\{ w \ : \ \begin{bmatrix} -1 \\ -1 \end{bmatrix} \leq w \leq \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right\}, \ \forall t \geq 0. \tag{12.8}$$

The set $\text{Pre}(\mathcal{X},\mathcal{W})$ can be obtained as described next. Since the set $\mathcal{X}$ is a polytope, it can be represented as an $\mathcal{H}$-polytope (Section 3.2)

$$\mathcal{X} = \{x : \ Hx \leq h\}, \tag{12.9}$$

where

$$H = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -1 & 0 \\ 0 & -1 \end{bmatrix} \text{ and } h = \begin{bmatrix} 10 \\ 10 \\ 10 \\ 10 \end{bmatrix}.$$

By using this $\mathcal{H}$-presentation and the system equation (12.6), the set $\text{Pre}(\mathcal{X},\mathcal{W})$ can be rewritten as

$$\begin{aligned} \text{Pre}(\mathcal{X},\mathcal{W}) &= \left\{ x \ : \ Hf_a(x,w) \leq h, \ \forall w \in \mathcal{W} \right\} & \text{(12.10a)} \\ &= \{x \ : \ HAx \leq h - Hw, \ \forall w \in \mathcal{W}\}. & \text{(12.10b)} \end{aligned}$$

The set (12.10) can be represented as a the following polyhedron

$$\text{Pre}(\mathcal{X},\mathcal{W}) = \{x \in \mathbb{R}^n \ : \ HAx \leq \tilde{h}\} \tag{12.11}$$

with

$$\tilde{h}_i = \min_{w \in \mathcal{W}} (h_i - H_i w). \tag{12.12}$$

In general, a linear program is required to solve problems (12.12). In this example H_i and $\mathcal{W}$ have simple expressions and we get $\tilde{h} = \begin{bmatrix} 9 \\ 9 \\ 9 \\ 9 \end{bmatrix}$. The set (12.11) might contain redundant inequalities which can be removed by using Algorithm 3.1 in Section 3.4.4 to obtain its minimal representation.

The set $\text{Pre}(\mathcal{X}, \mathcal{W})$ is

$$\text{Pre}(\mathcal{X}, \mathcal{W}) = \left\{ x \ : \ \begin{bmatrix} 1 & 0 \\ 1 & -0.5 \\ -1 & 0 \\ -1 & -0.5 \end{bmatrix} x \leq \begin{bmatrix} 18 \\ 9 \\ 18 \\ 9 \end{bmatrix} \right\}.$$

The set $\text{Pre}(\mathcal{X}, \mathcal{W}) \cap \mathcal{X}$, the significance of which we will discuss below, is

$$\text{Pre}(\mathcal{X}, \mathcal{W}) \cap \mathcal{X} = \left\{ x \ : \ \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -1 & 0 \\ 0 & -1 \\ 1 & -0.5 \\ -1 & 0.5 \end{bmatrix} x \leq \begin{bmatrix} 10 \\ 10 \\ 10 \\ 10 \\ 9 \\ 9 \end{bmatrix} \right\}$$

and is depicted in Fig. 12.1.

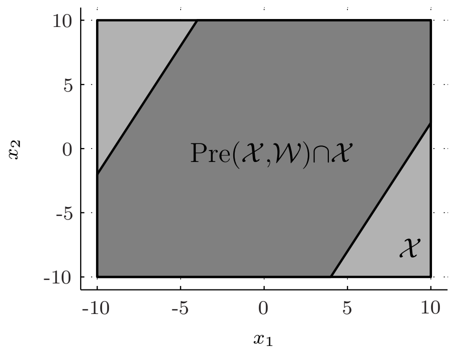


Fig. 12.1 Example 12.1: $\text{Pre}(\mathcal{X}, \mathcal{W}) \cap \mathcal{X}$ for system (12.6) under constraints (12.7)-(12.8).

Note that by using the definition of Pontryagin difference given in Section 3.4.8 and affine operations on polyhedra in Section 3.4.11 we can compactly summarize the operations in (12.10) and write the set Pre in (12.4) as

$$\begin{aligned}\text{Pre}(\mathcal{X},\mathcal{W}) &= \{x \in \mathbb{R}^n \ : \ Ax + w \in \mathcal{S}, \ \forall w \in \mathcal{W}\} = \{x \in \mathbb{R}^n \ : \ Ax \in \mathcal{S} \ominus \mathcal{W}\} = \\ &= (\mathcal{X} \ominus \mathcal{W}) \circ A.\end{aligned}$$

The set

$$\text{Reach}(\mathcal{X},\mathcal{W}) = \{y \ : \ \exists x \in \mathcal{X}, \ \exists w \in \mathcal{W} \text{ such that } y = Ax + w\} \tag{12.13}$$

is obtained by applying the map A to the set $\mathcal{X}$ and then considering the effect of the disturbance $w \in \mathcal{W}$. Let us write $\mathcal{X}$ in $\mathcal{V}$-representation (see section 3.1)

$$\mathcal{X} = \text{conv}(V), \tag{12.14}$$

and let us map the set of vertices V through the transformation A. Because the transformation is linear, the composition of the map A with the set $\mathcal{X}$, denoted as $A \circ \mathcal{X}$, is simply the convex hull of the transformed vertices

$$A \circ \mathcal{X} = \text{conv}(AV). \tag{12.15}$$

We refer the reader to Section 3.4.11 for a detailed discussion on linear transformations of polyhedra. Rewrite (12.13) as

$$\text{Reach}(\mathcal{X},\mathcal{W}) = \{y \in \mathbb{R}^n \ : \ \exists \ z \in A \circ \mathcal{X}, \ \exists w \in \mathcal{W} \text{ such that } y = z + w\}.$$

We can use the definition of Minkowski sum given in Section 3.4.9 and rewrite the Reach set as

$$\text{Reach}(\mathcal{X},\mathcal{W}) = (A \circ \mathcal{X}) \oplus \mathcal{W}.$$

We can compute the Minkowski sum via projection or vertex enumeration as explained in Section 3.4.9. The set Reach$(\mathcal{X},\mathcal{W})$ in $\mathcal{H}$-representation is

$$\text{Reach}(\mathcal{X},\mathcal{W}) = \left\{ x \ : \ \begin{bmatrix} 1 & -0.5 \\ 0 & -1 \\ -1 & 0 \\ -1 & 0.5 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} x \le \begin{bmatrix} 4 \\ 16 \\ 6 \\ 4 \\ 16 \\ 6 \end{bmatrix} \right\},$$

and is depicted in Fig. 12.2.

Computation of Robust Pre and Reach Sets for Linear Systems with Inputs

Example 12.2. Consider the second order unstable system

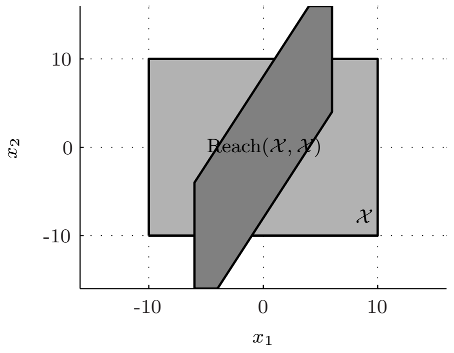


Fig. 12.2 Example 12.1: one-step reachable set for system (12.6) under constraints (12.8).

$$\begin{cases} x(t+1) = Ax + Bu = \begin{bmatrix} 1.5 & 0 \\ 1 & -1.5 \end{bmatrix} x(t) + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u(t) + w(t) \end{cases} \tag{12.16}$$

subject to the input and state constraints

$$u(t) \in \mathcal{U} = \{u \ : \ -5 \le u \le 5\}, \ \forall t \ge 0 \tag{12.17a}$$

$$x(t) \in \mathcal{X} = \left\{x \ : \ \begin{bmatrix} -10 \\ -10 \end{bmatrix} \le x \le \begin{bmatrix} 10 \\ 10 \end{bmatrix}\right\}, \ \forall t \ge 0 \tag{12.17b}$$

where

$$w(t) \in \mathcal{W} = \{w \ : \ -1 \le w \le 1\}, \ \forall t \ge 0 \tag{12.18}$$

For the non-autonomous system (12.16), the set Pre($\mathcal{X}, \mathcal{W}$) can be computed using the $\mathcal{H}$-presentation of $\mathcal{X}$ and $\mathcal{U}$,

$$\mathcal{X} = \{x \ : \ Hx \le h\}, \quad \mathcal{U} = \{u \ : \ H_u u \le h_u\}, \tag{12.19}$$

to obtain

$$\text{Pre}(\mathcal{X}, \mathcal{W}) = \{x \in \mathbb{R}^2 \ : \ \exists u \in \mathcal{U} \text{ s.t. } Ax + Bu + w \in \mathcal{X}, \ \forall \ w \in \mathcal{W}\} \tag{12.20a}$$

$$= \left\{x \in \mathbb{R}^2 \ : \ \exists u \in \mathbb{R} \text{ s.t. } \begin{bmatrix} HA & HB \\ 0 & H_u \end{bmatrix} \begin{pmatrix} x \\ u \end{pmatrix} \le \begin{bmatrix} h - Hw \\ h_u \end{bmatrix}, \ \forall \ w \in \mathcal{W}\right\}. \tag{12.20b}$$

As in example (12.1), the set Pre($\mathcal{X}, \mathcal{W}$) can be compactly written as

$$\text{Pre}(\mathcal{X}, \mathcal{W}) = \left\{x \in \mathbb{R}^2 \ : \ \exists u \in \mathbb{R} \text{ s.t. } \begin{bmatrix} HA & HB \\ 0 & H_u \end{bmatrix} \begin{pmatrix} x \\ u \end{pmatrix} \le \begin{bmatrix} \tilde{h} \\ h_u \end{bmatrix}\right\} \tag{12.21}$$

where

$$\tilde{h}_i = \min_{w \in \mathcal{W}} (h_i - H_i w). \tag{12.22}$$

In general, a linear program is required to solve problems (12.22). In this example H_i and $\mathcal{W}$ have simple expressions and we get $\tilde{h} = \begin{bmatrix} 9 \\ 9 \\ 9 \\ 9 \end{bmatrix}$.

The halfspaces in (12.21) define a polytope in the state-input space, and a projection operation (see Section 3.4.6) is used to derive the halfspaces which define $\text{Pre}(\mathcal{X})$ in the state space. The set $\text{Pre}(\mathcal{X}) \cap \mathcal{X}$ is depicted in Fig. 12.3 and reported below:

$$\begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 1 & -1.5 \\ -1 & 1.5 \\ 0 & 1 \\ 0 & -1 \end{bmatrix} x \leq \begin{bmatrix} 9.3 \\ 9.3 \\ 9 \\ 9 \\ 10 \\ 10 \end{bmatrix}$$

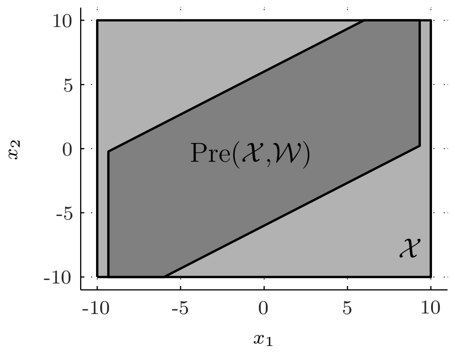


Fig. 12.3 Example 12.2: $\text{Pre}(\mathcal{X},\mathcal{W}) \cap \mathcal{X}$ for system (12.16) under constraints (12.17)-(12.18).

Note that by using the definition of a Minkowski sum given in Section 3.4.9 and the affine operation on polyhedra in Section 3.4.11 we can compactly write the operations in (12.20) as follows:

$$\begin{aligned} \text{Pre}(\mathcal{X},\mathcal{W}) &= \{x \ : \ \exists u \in \mathcal{U} \text{ s.t. } Ax + Bu + w \in \mathcal{X}, \ \forall \ w \in \mathcal{W}\} \\ &= \{x \ : \ \exists y \in \mathcal{X}, \ \exists u \in \mathcal{U} \text{ s.t. } y = Ax + Bu + w, \ \forall \ w \in \mathcal{W}\} \\ &= \{x \ : \ \exists y \in \mathcal{X}, \ \exists u \in \mathcal{U} \text{ s.t. } Ax = y + (-Bu) - w, \ \forall \ w \in \mathcal{W}\} \\ &= \{x \ : \ Ax \in \mathcal{C} \text{ and } \mathcal{C} = \mathcal{X} \oplus (-\text{B}) \circ \mathcal{U} \ominus \mathcal{W}\} \\ &= \{x \ : \ x \in \mathcal{C} \circ A, \ \mathcal{C} = \mathcal{X} \oplus (-B) \circ \mathcal{U} \ominus \mathcal{W}\} \\ &= \{x \ : \ x \in ((\mathcal{X} \ominus \mathcal{W}) \oplus (-B \circ \mathcal{U})) \circ A\} \,. \end{aligned} \tag{12.23}$$

Note that in (12.23) we have used the fact that if a set $\mathcal{S}$ is described as $\mathcal{S} = \{v \ : \ \exists z \in \mathcal{Z}, \text{ s.t. } v = z - w, \ \forall \ w \in \mathcal{W}\}$, then $\mathcal{S} = \{v \ : \ \exists z \in \mathcal{Z}, \text{ s.t. } z =$

$v+w,\ \forall\ w\in\mathcal{W}\}$ or $\mathcal{S}=\{v\ :\ v+w\in\mathcal{Z},\ \forall\ w\in\mathcal{W}\}=\mathcal{Z}\ominus\mathcal{W}$.

The set $\text{Reach}(\mathcal{X},\mathcal{W})=\{y:\ \exists x\in\mathcal{X},\ \exists u\in\mathcal{U},\ \exists w\in\mathcal{W}\text{ s.t. } y=Ax+Bu+w\}$ is obtained by applying the map A to the set $\mathcal{X}$ and then considering the effect of the input $u\in\mathcal{U}$ and of the disturbance $w\in\mathcal{W}$. We can use the definition of Minkowski sum given in Section 3.4.9 and rewrite $\text{Reach}(\mathcal{X},\mathcal{W})$ as

$$\text{Reach}(\mathcal{X},\mathcal{W})=(A\circ\mathcal{X})\oplus(B\circ\mathcal{U})\oplus\mathcal{W}.$$

Remark 12.1. In summary, for linear systems with additive disturbances the sets $\text{Pre}(\mathcal{X},\mathcal{W})$ and $\text{Reach}(\mathcal{X},\mathcal{W})$ are the results of linear operations on the polytopes $\mathcal{X}$, $\mathcal{U}$ and $\mathcal{W}$ and therefore are polytopes. By using the definition of Minkowski sum given in Section 3.4.9, Pontryagin difference given in Section 3.4.8 and affine operation on polyhedra in Section 3.4.11 we can compactly summarize the operations in Table 12.1.

	$x(t+1)=Ax(t)+w(t)$	$x(k+1)=Ax(t)+Bu(t)+w(t)$
$\text{Pre}(\mathcal{X},\mathcal{W})$	$(\mathcal{X}\ominus\mathcal{W})\circ A$	$(\mathcal{X}\ominus\mathcal{W}\oplus -B\circ\mathcal{U})\circ A$
$\text{Reach}(\mathcal{X}),\mathcal{W}$	$(A\circ\mathcal{X})\oplus\mathcal{W}$	$(A\circ\mathcal{X})\oplus(B\circ\mathcal{U})\oplus\mathcal{W}$

Table 12.1 Pre and Reach operations for uncertain linear systems subject to polyhedral input and state constraints $x(t)\in\mathcal{X}$, $u(t)\in\mathcal{U}$ with additive polyhedral disturbances $w(t)\in\mathcal{W}$

Remark 12.2. Note that the summary in remark (12.1) applies also to the class of systems $x(k+1)=Ax(t)+Bu(t)+E\tilde{d}(t)$ where $\tilde{d}\in\tilde{\mathcal{W}}$. This can be transformed into $x(k+1)=Ax(t)+Bu(t)+w(t)$ where $w\in\mathcal{W}\triangleq E\circ\tilde{\mathcal{W}}$.

Computation of Robust Pre and Reach for Linear Systems with Parametric Uncertainty

The following proposition will help us computing Pre and Reach sets for linear systems with parametric uncertainty.

Proposition 12.1. *Let $g:\mathbb{R}^{n_z}\times\mathbb{R}^{n}\times\mathbb{R}^{n_w}\rightarrow\mathbb{R}^{n_g}$ be a function of (z,x,w) convex in w for each (z,x). Assume that the variable w belongs to the polytope $\mathcal{W}$ with vertices $\{\bar{w}_i\}_{i=1}^{n_\mathcal{W}}$. Then, the constraint*

$$g(z,x,w)\ \leq 0\ \forall w\in\mathcal{W} \tag{12.24}$$

is satisfied if and only if

$$g(z, x, \bar{w}_i) \leq 0, \;\; i = 1, \ldots, n_{\mathcal{W}}. \tag{12.25}$$

Proof: Easily follows by the fact that the maximum of a convex function over a compact convex set is attained at an extreme point of the set. □

The following proposition shows how to reduce the number of constraints in (12.25) for a specific class of constraint functions.

Proposition 12.2. *Assume $g(z, x, w) = g^1(z, x) + g^2(w)$. Then, the constraint (12.24) can be replaced by $g(z, x) \leq -\bar{g}$, where $\bar{g} \triangleq \left[\bar{g}_1, \ldots, \bar{g}_{n_g}\right]'$ is a vector whose i-th component is*

$$\bar{g}_i = \max_{w \in \mathcal{W}} \; g_i^2(w), \tag{12.26}$$

and $g_i^2(w)$ denotes the i-th component of $g^2(w)$.

Example 12.3. Consider the second order autonomous system

$$x(t+1) = A(w^p(t))x(t) + w^a(t) = \begin{bmatrix} 0.5 + w^p(t) & 0 \\ 1 & -0.5 \end{bmatrix} x(t) + w^a(t) \tag{12.27}$$

subject to the state constraints

$$\begin{aligned} x(t) \in \mathcal{X} &= \left\{ x \;:\; \begin{bmatrix} -10 \\ -10 \end{bmatrix} \leq x \leq \begin{bmatrix} 10 \\ 10 \end{bmatrix} \right\}, \;\; \forall t \geq 0 \\ w^a(t) \in \mathcal{W}^a &= \left\{ w^a \;:\; \begin{bmatrix} -1 \\ -1 \end{bmatrix} \leq w^a \leq \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right\}, \;\; \forall t \geq 0 \\ w^p(t) \in \mathcal{W}^p &= \{ w^p \;:\; 0 \leq w^p \leq 0.5 \}, \;\; \forall t \geq 0. \end{aligned} \tag{12.28}$$

Let $w = [w^a; \; w^p]$ and $\mathcal{W} = \mathcal{W}^a \times \mathcal{W}^p$. The set $\text{Pre}(\mathcal{X}, \mathcal{W})$ can be obtained as follows. The set $\mathcal{X}$ is a polytope and it can be represented as an $\mathcal{H}$-polytope (Section 3.2)

$$\mathcal{X} = \{x : \; Hx \leq h\}, \tag{12.29}$$

where

$$H = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -1 & 0 \\ 0 & -1 \end{bmatrix} \text{ and } h = \begin{bmatrix} 10 \\ 10 \\ 10 \\ 10 \end{bmatrix}.$$

By using this $\mathcal{H}$-presentation and the system equation (12.27), the set $\text{Pre}(\mathcal{X}, \mathcal{W})$ can be rewritten as

$$\text{Pre}(\mathcal{X}, \mathcal{W}) = \left\{ x \;:\; Hf_a(x, w) \leq h, \; \forall w \in \mathcal{W} \right\} \tag{12.30}$$

$$= \{ x \;:\; HA(w^p)x \leq h - Hw^a, \; \forall w^a \in \mathcal{W}^a, \; w^p \in \mathcal{W}^p \}. \tag{12.31}$$

By using Propositions 12.1 and 12.2, the set (12.31) can be rewritten as a the polytope

$$x \in \operatorname{Pre}(\mathcal{X},\mathcal{W}) = \{x \in \mathbb{R}^n \ : \ \begin{bmatrix} HA(0) \\ HA(0.5) \end{bmatrix} x \le \begin{bmatrix} \tilde{h} \\ \tilde{h} \end{bmatrix}\} \tag{12.32}$$

with

$$\tilde{h}_i = \min_{w^a \in \mathcal{W}^a} (h_i - H_i w^a). \tag{12.33}$$

The set $\operatorname{Pre}(\mathcal{X},\mathcal{W}) \cap \mathcal{X}$ is depicted in Fig. 12.4 and reported below:

$$\begin{bmatrix} 1 & 0 \\ 0.89 & -0.44 \\ 1 & 0 \\ -0.89 & 0.44 \\ 0 & 1 \\ 0 & -1 \end{bmatrix} x \le \begin{bmatrix} 9 \\ 8.049 \\ 9 \\ 8.049 \\ 10 \\ 10 \end{bmatrix}$$

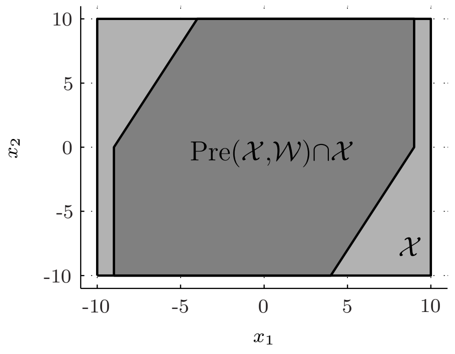


Fig. 12.4 Example 12.3: $\operatorname{Pre}(\mathcal{X},\mathcal{W}) \cap \mathcal{X}$ for system (12.27) under constraints (12.28).

Robust Invariant and Robust Control Invariant Sets

Two different types of sets are considered in this chapter: *robust invariant sets* and *robust control invariant sets*. We will first discuss robust invariant sets. Invariant sets are computed for autonomous systems. These types of sets are useful to answer questions such as: "For a *given* feedback controller $u = g(x)$, find the set of states whose trajectory will never violate the system constraints for all possible disturbances". The following definitions introduce the different types of robust invariant sets.

Definition 12.1 (Robust Positive Invariant Set). A set $\mathcal{O} \subseteq \mathcal{X}$ is said to be a robust positive invariant set for the autonomous system (12.1) subject to the constraints in (12.3), if

$$x(0) \in \mathcal{O} \quad \Rightarrow \quad x(t) \in \mathcal{O}, \quad \forall w(t) \in \mathcal{W},\ t \in \mathbb{N}_+$$

Definition 12.2 (Maximal Robust Positive Invariant Set $\mathcal{O}_\infty$). The set $\mathcal{O}_\infty \subseteq \mathcal{X}$ is the maximal robust invariant set of the autonomous system (12.1) subject to the constraints in (12.3) if $\mathcal{O}_\infty$ is a robust invariant set and $\mathcal{O}_\infty$ contains all the robust positive invariant sets contained in $\mathcal{X}$ that contain the origin.

Theorem 12.1 (Geometric condition for invariance). *A set $\mathcal{O} \subseteq \mathcal{X}$ is a robust positive invariant set for the autonomous system (12.1) subject to the constraints in (12.3), if and only if*

$$\mathcal{O} \subseteq \operatorname{Pre}(\mathcal{O}, \mathcal{W}) \tag{12.34}$$

The proof of Theorem 12.1 follows the same lines of the proof of Theorem 10.1. □

It is immediate to prove that condition (12.34) of Theorem 12.1 is equivalent to the following condition

$$\operatorname{Pre}(\mathcal{O}, \mathcal{W}) \cap \mathcal{O} = \mathcal{O} \tag{12.35}$$

Based on condition (12.35), the following algorithm provides a procedure for computing the maximal robust positive invariant subset $\mathcal{O}_\infty$ for system (12.1)-(12.3) (for reference to proofs and literature see Chapter 10.3).

Algorithm 12.1 (Computation of $\mathcal{O}_\infty$)

```
INPUT $f_a$, $\mathcal{X}$, $\mathcal{W}$
OUTPUT $\mathcal{O}_\infty$
1. LET $\Omega_0 = \mathcal{X}$
2. LET $\Omega_{k+1} = \operatorname{Pre}(\Omega_k, \mathcal{W}) \cap \Omega_k$
3. IF $\Omega_{k+1} = \Omega_k$ THEN
       $\mathcal{O}_\infty \leftarrow \Omega_{k+1}$
   ELSE GOTO 2
4. END
```

Algorithm 12.1 generates the set sequence $\{\Omega_k\}$ satisfying $\Omega_{k+1} \subseteq \Omega_k, \forall k \in \mathbb{N}$ and it terminates if $\Omega_{k+1} = \Omega_k$ so that Ω_k is the maximal robust positive invariant set $\mathcal{O}_\infty$ for system (12.1)-(12.3).

Example 12.4. Consider the second order stable system in Example 12.1

$$x(t+1) = Ax(t) + w(t) = \begin{bmatrix} 0.5 & 0 \\ 1 & -0.5 \end{bmatrix} x(t) + w(t) \tag{12.36}$$

subject to the state constraints

$$\begin{aligned} x(t) \in \mathcal{X} &= \left\{ x \ : \ \begin{bmatrix} -10 \\ -10 \end{bmatrix} \leq x \leq \begin{bmatrix} 10 \\ 10 \end{bmatrix} \right\}, \ \forall t \geq 0 \\ w(t) \in \mathcal{W} &= \left\{ w \ : \ \begin{bmatrix} -1 \\ -1 \end{bmatrix} \leq w \leq \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right\}, \ \forall t \geq 0 \end{aligned} \tag{12.37}$$

The maximal robust positive invariant set of system (12.36) subject to constraints (12.37) is depicted in Fig. 12.5 and reported below:

$$\begin{bmatrix} 0.89 & -0.44 \\ -0.89 & 0.44 \\ -1 & 0 \\ 0 & -1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} x \leq \begin{bmatrix} 8.04 \\ 8.04 \\ 10 \\ 10 \\ 10 \\ 10 \end{bmatrix}$$

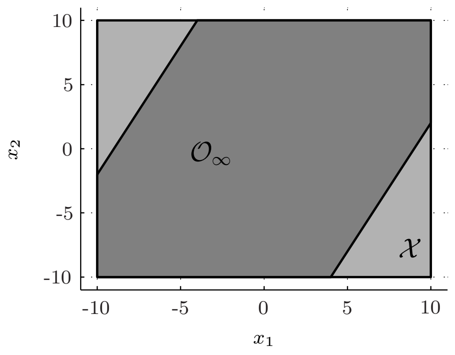


Fig. 12.5 Maximal Robust Positive Invariant Set of system (12.36) subject to constraints (12.37).

Robust control invariant sets are defined for systems subject to controllable inputs. These types of sets are useful to answer questions such as: "Find the set of states for which *there exists* a controller such that the system constraints are never violated for all possible disturbances". The following definitions introduce the different types of robust control invariant sets.

Definition 12.3 (Robust Control Invariant Set). A set $\mathcal{C} \subseteq \mathcal{X}$ is said to be a robust control invariant set for the system in (12.2) subject to the constraints in (12.3), if

$$x(t) \in \mathcal{C} \quad \Rightarrow \quad \exists u(t) \in \mathcal{U} \text{ such that } f(x(t), u(t), w(t)) \in \mathcal{C}, \ \forall\, w(t) \in \mathcal{W}, \ \forall\, t \in \mathbb{N}_+$$

Definition 12.4 (Maximal Robust Control Invariant Set $\mathcal{C}_\infty$). The set $\mathcal{C}_\infty \subseteq \mathcal{X}$ is said to be the maximal robust control invariant set for the system in (12.2) subject to the constraints in (12.3), if it is robust control invariant and contains all robust control invariant sets contained in $\mathcal{X}$.

Remark 12.3. The geometric conditions for invariance (12.34)-(12.35) hold for control invariant sets.

The following algorithm provides a procedure for computing the maximal control robust invariant set $\mathcal{C}_\infty$ for system (12.2)-(12.3).

Algorithm 12.2 (Computation of $\mathcal{C}_\infty$)

```
INPUT $A$ and $B$, $\mathcal{X}$, $\mathcal{U}$ and $\mathcal{W}$
OUTPUT $\mathcal{C}_\infty$
1. LET $\Omega_0 = \mathcal{X}$
2. LET $\Omega_{k+1} = \text{Pre}(\Omega_k, \mathcal{W}) \cap \Omega_k$
3. IF $\Omega_{k+1} = \Omega_k$ THEN
       $\mathcal{C}_\infty \leftarrow \Omega_{k+1}$
   ELSE GOTO 2
4. END
```

Algorithm 12.2 generates the set sequence $\{\Omega_k\}$ satisfying $\Omega_{k+1} \subseteq \Omega_k, \forall k \in \mathbb{N}$ and $\mathcal{O}_\infty = \bigcap_{k\geq 0} \Omega_k$. If $\Omega_k = \emptyset$ for some integer k then the simple conclusion is that $\mathcal{O}_\infty = \emptyset$. Algorithm 12.2 terminates if $\Omega_{k+1} = \Omega_k$ so that Ω_k is the maximal robust control invariant set $\mathcal{C}_\infty$ for the system (12.2)-(12.3). The same holds true for non-autonomous systems.

Example 12.5. Consider the second order unstable system in example 12.2. The maximal robust control invariant set of system (12.16) subject to constraints (12.17) is depicted in Fig. 12.6 and reported below:

$$\begin{bmatrix} 0 & 1 \\ 0 & -1 \\ 0.55 & -0.83 \\ -0.55 & 0.83 \end{bmatrix} x \leq \begin{bmatrix} 3.72 \\ 3.72 \\ 2.0 \\ 2.0 \end{bmatrix}$$

Definition 12.5 (Finitely determined set). Consider Algorithm 12.1. The set $\mathcal{C}_\infty$ ($\mathcal{O}_\infty$) is finitely determined if and only if $\exists\ i \in \mathbb{N}$ such that $\Omega_{i+1} = \Omega_i$. The smallest element $i \in \mathbb{N}$ such that $\Omega_{i+1} = \Omega_i$ is called the determinedness index.

For all states contained in the maximal robust control invariant set $\mathcal{C}_\infty$ there exists a control law, such that the system constraints are never violated for all feasible disturbances. This does not imply that there exists a control law which can drive the state into a user-specified target set. This issue is addressed in the following by introducing the concept of robust controllable and stabilizable sets.

Definition 12.6 (N-Step Robust Controllable Set $\mathcal{K}_N(\mathcal{O}, \mathcal{W})$). For a given target set $\mathcal{O} \subseteq \mathcal{X}$, the N-step robust controllable set $\mathcal{K}_N(\mathcal{O}, \mathcal{W})$ of the system (12.2) subject to the constraints (12.3) is defined as:

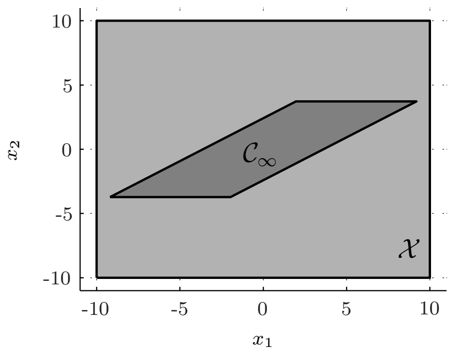


Fig. 12.6 Maximal Robust Control Invariant Set of system (12.16) subject to constraints (12.17).

$$\mathcal{K}_j(\mathcal{O},\mathcal{W}) \triangleq \text{Pre}(\mathcal{K}_{j-1}(\mathcal{O}),\mathcal{W}), \quad \mathcal{K}_0(\mathcal{O},\mathcal{W}) = \mathcal{O}, \qquad j \in \{1,\dots,N\}.$$

From Definition 12.6, all states x_0 belonging to the N-Step Robust Controllable Set $\mathcal{K}_N(\mathcal{O},\mathcal{W})$ can be robustly driven, through a time-varying control law, to the target set $\mathcal{O}$ in N steps, while satisfying input and state constraints for all possible disturbances.

Definition 12.7 (Maximal Robust Controllable Set $\mathcal{K}_\infty(\mathcal{O},\mathcal{W})$). For a given target set $\mathcal{O} \subseteq \mathcal{X}$, the maximal robust controllable set $\mathcal{K}_\infty(\mathcal{O},\mathcal{W})$ for the system (12.2) subject to the constraints in (12.3) is the union of all N-step robust controllable sets contained in $\mathcal{X}$ ($N \in \mathbb{N}$).

Robust controllable sets $\mathcal{K}_N(\mathcal{O},\mathcal{W})$ where the target $\mathcal{O}$ is a robust control invariant set are special sets, since in addition to guaranteeing that from $\mathcal{K}_N(\mathcal{O},\mathcal{W})$ we robustly reach $\mathcal{O}$ in N steps, one can ensure that once reached $\mathcal{O}$, the system can stay there at all future time instants and for all possible disturbance realizations.

Definition 12.8 (N-step (Maximal) Robust Stabilizable Set). For a given robust control invariant set $\mathcal{O} \subseteq \mathcal{X}$, the N-step (maximal) robust stabilizable set of the system (12.2) subject to the constraints (12.3) is the N-step (maximal) robust controllable set $\mathcal{K}_N(\mathcal{O},\mathcal{W})$ ($\mathcal{K}_\infty(\mathcal{O},\mathcal{W})$).

The set $\mathcal{K}_\infty(\mathcal{O},\mathcal{W})$ contains all states which can be robustly steered into the robust control invariant set $\mathcal{O}$ and hence $\mathcal{K}_\infty(\mathcal{O},\mathcal{W}) \subseteq \mathcal{C}_\infty$. For *linear systems* the set $\mathcal{K}_\infty(\mathcal{O},\mathcal{W}) \subseteq \mathcal{C}_\infty$ can be computed as follows:

Algorithm 12.3 (The Maximal Robust Stabilizable Set $\mathcal{K}_\infty(\mathcal{O},\mathcal{W})$)

```
1. $\mathcal{K}_0 = \mathcal{O}$, where $\mathcal{O}$ is a control invariant set
2. $\mathcal{K}_{c+1} = \text{Pre}(\mathcal{K}_c,\mathcal{W}) \cap \mathcal{K}_c$
3. If $\mathcal{K}_{c+1} = \mathcal{K}_c$, then $\mathcal{K}_\infty(\mathcal{O},\mathcal{W}) = \mathcal{K}_c$, return; Else, set $c = c+1$ and goto 2.
```

Since $\mathcal{O}$ is robust control invariant, it holds $\forall c \in \mathbb{N}$ that $\mathcal{K}_c$ is robust control invariant and $\mathcal{K}_c \subseteq \mathcal{K}_{c+1}$. Note that Algorithm 12.3 is not guaranteed to terminate in finite time.

N-step robust reachable sets are defined analogously to N-step robust controllable set.

Definition 12.9 (N-Step Robust Reachable Set $\mathcal{R}_N(\mathcal{X}_0)$). For a given initial set $\mathcal{X}_0 \subseteq \mathcal{X}$, the N-step robust reachable set $\mathcal{R}_N(\mathcal{X}_0, \mathcal{W})$ of the system (12.1) or (12.2) subject to the constraints (12.3) is defined as:

$$\mathcal{R}_{i+1}(\mathcal{X}_0, \mathcal{W}) \triangleq \text{Reach}(\mathcal{R}_i(\mathcal{X}_0), \mathcal{W}), \quad \mathcal{R}_0(\mathcal{X}_0, \mathcal{W}) = \mathcal{X}_0, \quad i = 0, \ldots, N-1.$$

From Definition 12.9, all states x_0 belonging to $\mathcal{X}_0$ will evolve to the N-step robust reachable set $\mathcal{R}_N(\mathcal{X}_0, \mathcal{W})$ in N steps.

12.3 Problem Formulation

Consider the following discrete-time linear uncertain system

$$x(t+1) = A(w^p(t))x(t) + B(w^p(t))u(t) + Ew^a(t) \tag{12.38}$$

where $x(t) \in \mathbb{R}^n$ and $u(t) \in \mathbb{R}^m$ are the state and input vectors, respectively, subject to the constraints

$$x(t) \in \mathcal{X}, \; u(t) \in \mathcal{U}, \; \forall t \geq 0. \tag{12.39}$$

The sets $\mathcal{X} \subseteq \mathbb{R}^n$ and $\mathcal{U} \subseteq \mathbb{R}^m$ are polytopes. Vectors $w^a(t) \in \mathbb{R}^{n_a}$ and $w^p(t) \in \mathbb{R}^{n_p}$ are unknown additive disturbances and parametric uncertainties, respectively. The disturbance vector is $w(t) = [w^a(t); \; w^p(t)] \in \mathbb{R}^{n_w}$ with $n_w = n_a + n_p$. We assume that only bounds on $w^a(t)$ and $w^p(t)$ are known, namely that $w \in \mathcal{W}$. In particular $\mathcal{W} = \mathcal{W}^a \times \mathcal{W}^p$ with $w^a(t) \in \mathcal{W}^a$, where $\mathcal{W}^a \subset \mathbb{R}^{n_a}$ is a given polytope represented in terms if its vertices $\mathcal{W}^a = \text{conv}(\{w^{a,1}, \ldots, w^{a,n_{\mathcal{W}^a}}\})$, and $w^p(t) \in \mathcal{W}^p$, where $\mathcal{W}^p = \text{conv}(\{w^{p,1}, \ldots, w^{p,n_{\mathcal{W}^p}}\})$ is a polytope in $\mathbb{R}^{n_p}$. We also assume that $A(\cdot)$, $B(\cdot)$ are affine functions of w^p

$$A(w^p) = A^0 + \sum_{i=1}^{n_p} A^i w_c^{p,i}, \; B(w^p) = B^0 + \sum_{i=1}^{n_p} B^i w_c^{p,i} \tag{12.40}$$

where $A^i \in \mathbb{R}^{n \times n}$ and $B^i \in \mathbb{R}^{n \times m}$ are given matrices for $i = 0 \ldots, n_p$ and $w_c^{p,i}$ is the i-th component of the vector w^p, i.e., $w^p = [w_c^{p,1}, \ldots, w_c^{p,n_p}]$.

Open-Loop Predictions

Define the worst case cost function as

$$J_0(x(0),U_0) \triangleq \max_{w_0,\ldots,w_{N-1}} \left[p(x_N) + \sum_{k=0}^{N-1} q(x_k,u_k)\right]$$
$$\text{subj. to } \begin{cases} x_{k+1} = A(w_k^p)x_k + B(w_k^p)u_k + Ew_k^a \\ w_k^a \in \mathcal{W}^a, w_k^p \in \mathcal{W}^p, \\ k = 0,\ldots,N-1 \end{cases} \tag{12.41}$$

where N is the time horizon and $U_0 \triangleq [u_0',\ldots,u_{N-1}']' \in \mathbb{R}^s$, $s \triangleq mN$ the vector of the input sequence. If the 1-norm or ∞-norm is used in the cost function of problem (12.41), then we set

$$p(x_N) = \|Px_N\|_p,\ q(x_k,u_k) = \|Qx_k\|_p + \|Ru_k\|_p \tag{12.42}$$

with $p = 1$ or $p = \infty$. If the squared euclidian norm is used in the cost function of problem (12.41), then we set

$$p(x_N) = x_N'Px_N,\ q(x_k,u_k) = x_k'Qx_k + u_k'Ru_k. \tag{12.43}$$

Note that in (12.41) x_k denotes the state vector at time k obtained by starting from the state $x_0 = x(0)$ and applying to the system model

$$x_{k+1} = A(w_k^p)x_k + B(w_k^p)u_k + Ew_k^a$$

the input sequence $u_0,\ldots,u_{k-1}$ and the disturbance sequences $\mathbf{w}^a \triangleq \{w_0^a,\ldots,w_{N-1}^a\}$, $\mathbf{w}^p \triangleq \{w_0^p,\ldots,w_{N-1}^p\}$.

Consider the robust optimal control problem

$$J_0^*(x_0) \triangleq \min_{U_0} J_0(x_0,U_0) \tag{12.44}$$

$$\text{subj. to } \left\{\begin{array}{l} x_k \in \mathcal{X},\ u_k \in \mathcal{U} \\ x_{k+1} = A(w_k^p)x_k + B(w_k^p)u_k + Ew_k^a \\ x_N \in \mathcal{X}_f \\ k = 0,\ldots,N-1 \end{array}\right\} \begin{array}{l} \forall w_k^a \in \mathcal{W}^a, w_k^p \in \mathcal{W}^p \\ \forall k = 0,\ldots,N-1 \end{array} \tag{12.45}$$

where $\mathcal{X}_f \subseteq \mathbb{R}^n$ is a terminal polyhedral region. We denote by $U_0^* = \{u_0^*,\ldots,u_{N-1}^*\}$ the optimal solution to (12.44)–(12.45).

We denote with $\mathcal{X}_i^{OL} \subseteq \mathcal{X}$ the set of states x_i for which the robust optimal control problem (12.41)-(12.45) is feasible, i.e.,

$$\begin{aligned} \mathcal{X}_i^{OL} = \{x_i \in \mathcal{X} :\ & \exists (u_i,\ldots,u_{N-1}) \text{ such that } x_k \in \mathcal{X},\ u_k \in \mathcal{U},\ k = i,\ldots,N-1,\ x_N \in \mathcal{X}_f \\ & \forall\ w_k^a \in \mathcal{W}^a, w_k^p \in \mathcal{W}^p\ k = i,\ldots,N-1,\ \text{where } x_{k+1} = A(w_k^p)x_k + B(w_k^p)u_k + Ew_k^a\}. \end{aligned} \tag{12.46}$$

Remark 12.4. Note that we distinguish between the *current* state $x(k)$ of system (12.38) at time k and the variable x_k in the optimization problem (12.45), that is the *predicted* state of system (12.38) at time k obtained by starting from the state $x_0 = x(0)$ and applying to system $x_{k+1} = A(w_k^p)x_k + B(w_k^p)u_k + Ew_k^a$ the input sequence $u_0, \ldots, u_{k-1}$ and the disturbance sequences $w_0^p, \ldots, w_{k-1}^p$, $w_0^a, \ldots, w_{k-1}^a$. Analogously, $u(k)$ is the input applied to system (12.38) at time k while u_k is the k-th optimization variable of the optimization problem (12.45).

Problem (12.41) looks for the worst value $J(x_0, U)$ of the performance index and the corresponding worst sequences $\mathbf{w}^{p*}$, $\mathbf{w}^{a*}$ as a function of x_0 and U_0. Problem (12.44)–(12.45) minimizes such a worst performance subject to the constraint that the input sequence must be feasible *for all* possible disturbance realizations. In other words, the worst-case performance is minimized under constraint fulfillment against all possible realizations of $\mathbf{w}^a$, $\mathbf{w}^p$. Note that worst sequences $\mathbf{w}^{a*}$, $\mathbf{w}^{p*}$for the performance are not necessarily worst sequences in terms of constraints satisfaction.

The min-max formulation (12.41)–(12.45) is based on an *open-loop* prediction and thus referred to as Constrained Robust Optimal Control with open-loop predictions (CROC-OL). The optimal control problem (12.41)–(12.45) can be viewed as a deterministic zero-sum dynamic game between two players: the controller U and the disturbance W [22, pag. 266-272]. The player U plays first. Given the initial state $x(0)$, U chooses his action over the whole horizon $\{u_0, \ldots, u_{N-1}\}$, reveals his plan to the opponent W, who decides on his actions next $\{w_0^a, w_0^p, \ldots, w_{N-1}^a, w_{N-1}^p\}$.

For this reason the player U has the duty of counteracting *any* feasible disturbance realization with just *one* single sequence $\{u_0, \ldots, u_{N-1}\}$. This prediction model does not consider that at the next time step, the payer can measure the state $x(1)$ and "adjust" his input $u(1)$ based on the current measured state. By not considering this fact, the effect of the uncertainty may grow over the prediction horizon and may easily lead to infeasibility of the min problem (12.41)–(12.45).

On the contrary, in the closed-loop prediction scheme presented next, the optimization scheme *predicts* that the disturbance and the controller play one move at a time.

Closed-Loop Predictions

The constrained robust optimal control problem based on closed-loop predictions (CROC-CL) is defined as follows:

$$J_j^*(x_j) \triangleq \min_{u_j} J_j(x_j, u_j) \tag{12.47}$$

$$\text{subj. to} \left\{ \begin{array}{l} x_j \in \mathcal{X},\ u_j \in \mathcal{U} \\ A(w_j^p)x_j + B(w_j^p)u_j + Ew_j^a \in \mathcal{X}_{j+1} \end{array} \right\} \forall w_j^a \in \mathcal{W}^a, w_j^p \in \mathcal{W}^p \tag{12.48}$$

$$J_j(x_j, u_j) \triangleq \max_{w_j^a \in \mathcal{W}^a,\ w_j^p \in \mathcal{W}^p} \left\{ q(x_j, u_j) + J_{j+1}^*(A(w_j^p)x_j + B(w_j^p)u_j + Ew_j^a) \right\}, \tag{12.49}$$

for $j = 0, \dots, N-1$ and with boundary conditions

$$J_N^*(x_N) = p(x_N) \tag{12.50}$$

$$\mathcal{X}_N = \mathcal{X}_f, \tag{12.51}$$

where $\mathcal{X}_j$ denotes the set of states x for which (12.47)–(12.49) is feasible

$$\mathcal{X}_j = \{x \in \mathcal{X} :\ \exists u \in \mathcal{U} \text{ s.t. } A(w^p)x + B(w^p)u + Ew^a \in \mathcal{X}_{j+1}\ \forall w^a \in \mathcal{W}^a, w^p \in \mathcal{W}^p\}. \tag{12.52}$$

The reason for including constraints (12.48) in the minimization problem and not in the maximization problem is that in (12.49) w_j^a and w_j^p are free to act regardless of the state constraints. On the other hand, the input u_j has the duty of keeping the state within the constraints (12.48) for all possible disturbance realization.

Again, the optimal control problem (12.47)–(12.49) can be viewed as a deterministic zero-sum dynamic game between two players: the controller U and the disturbance W. The game is played as follows. At the generic time j player U observes x_j and responds with $u_j(x_j)$. Player W observes $(x_j, u_j(x_j))$ and responds with w_j^a and w_j^p.

Note that player U does *not* need to reveal his action u_j to player W (the disturbance). This happens for instance in games where U and W play at the same time, e.g. rock-paper-scissors.

The player W will always play the worst case action only if it has knowledge of both x_j and $u_j(x_j)$. In fact, w_j^a and w_j^p in (12.49) are a function of x_j and u_j. If U does *not* reveal his action to player W, then we can only claim that the player W *might* play the worst case action. Problem (12.47)–(12.49) is meaningful in both cases. Robust constraint satisfaction and worst case minimization will always be guaranteed.

Example 12.6. Consider the system

$$x_{k+1} = x_k + u_k + w_k \tag{12.53}$$

where x, u and w are state, input and disturbance, respectively. Let $u_k \in \{-1, 0, 1\}$ and $w_k \in \{-1, 0, 1\}$ be feasible input and disturbance. Here $\{-1, 0, 1\}$ denotes the set with three elements: -1, 0 and 1. Let $x(0) = 0$

be the initial state. The objective for player U is to play two moves in order to keep the state x_2 at time 2 in the set $[-1, 1]$. If U is able to do so for any possible disturbance, then he will win the game.

The open-loop formulation (12.44)–(12.45) is infeasible. In fact, in open-loop U can choose from nine possible sequences: (0,0), (1,1), (-1,-1), (-1,1) (1,-1), (-1,0), (1,0), (0,1) and (0,-1). For any of those sequence there will always exist a disturbance sequence w_0, w_1 which will bring x_2 outside the feasible set [-1,1].

The closed-loop formulation (12.47)–(12.49) is feasible and has a simple solution: $u_k = -x_k$. In this case system (12.53) becomes $x_{k+1} = w_k$ and $x_2 = w_1$ lies in the feasible set $[-1, 1]$ for all admissible disturbances w_1.

Explicit Solutions

In the following sections we will describe how to compute the solution to CROC-OL and CROC-CL problems. In particular we will show that the solution to CROC-OL and CROC-CL problem with worst-case perforce based on 1- or ∞-norm can be expressed in feedback form where $u^*(k)$ is a continuous piecewise affine function on polyhedra of the state $x(k)$, i.e., $u^*(k) = f_k(x(k))$ where

$$f_k(x) = F_k^i x + g_k^i \quad \text{if} \quad H_k^i x \leq K_k^i, \; i = 1, \ldots, N_k^r. \tag{12.54}$$

H_k^i and K_k^i in equation (12.54) are the matrices describing the i-th polyhedron $CR_k^i = \{x \in \mathbb{R}^n \; : \; H_k^i x \leq K_k^i\}$ inside which the feedback optimal control law $u^*(k)$ at time k has the affine form $F_k^i x + g_k^i$.

If CROC-OL problems are considered, the set of polyhedra CR_k^i, $i = 1, \ldots, N_k^r$ is a *polyhedral partition* of the set of feasible states $\mathcal{X}_k^{OL}$ of problem (12.41)–(12.45) at time k.

If CROC-CL problems are considered, the set of polyhedra CR_k^i, $i = 1, \ldots, N_k^r$ is a *polyhedral partition* of the set of feasible states $\mathcal{X}_k$ of problem (12.47)–(12.49) at time k.

The difference between the feasible sets $\mathcal{X}_k^{OL}$ and $\mathcal{X}_k$ associated with *open-loop* prediction and *closed-loop* prediction, respectively, are discussed in the next section.

12.4 Feasible Solutions

As in the nominal case (Section 10.3), there are two ways to rigourously define and compute the robust feasible sets: the *batch approach* and the *recursive approach*. While for systems without disturbances, both approaches yield the same result, in the robust case the *batch approach* provides the feasible set $\mathcal{X}_i^{OL}$ of CROC with open-loop predictions and the *recursive approach* provides the feasible set $\mathcal{X}_i$ of CROC with closed-loop predictions. From the discussion in the previous sections, clearly we have $\mathcal{X}_i^{OL} \subseteq \mathcal{X}_i$. We will detail the *batch approach* and the *recursive approach* next and at the end of the section will show how to modify the batch approach in order to compute $\mathcal{X}_i$.

Batch Approach: Open-Loop Prediction

Consider the set $\mathcal{X}_i^{OL}$ (12.46) of feasible states x_i at time i for which (12.44)–(12.45) is feasible, for $i = 0, \dots, N$. Consider the definition of $\mathcal{X}_i^{OL}$ in (12.46) rewritten below

$$\begin{aligned}\mathcal{X}_i^{OL} = \{x_i \in \mathcal{X} :\ \exists (u_i, \dots, u_{N-1}) \text{ such that } x_k \in \mathcal{X},\ u_k \in \mathcal{U},\ k = i, \dots, N-1,\ x_N \in \mathcal{X}_f \\ \forall\ w_k^a \in \mathcal{W}^a, w_k^p \in \mathcal{W}^p\ k = i, \dots, N-1, \text{ where } x_{k+1} = A(w_k^p)x_k + B(w_k^p)u_k + Ew_k^a\}\end{aligned}$$

For any initial state $x_i \in \mathcal{X}_i^{OL}$ there exists a feasible sequence of inputs $U_i \triangleq [u_i', \dots, u_{N-1}']$ which keeps the state evolution in the feasible set $\mathcal{X}$ at future time instants $k = i+1, \dots, N-1$ and forces x_N into $\mathcal{X}_f$ at time N for all feasible disturbance sequence $w_k^a \in \mathcal{W}^a,\ w_k^p \in \mathcal{W}^p,\ k = i, \dots, N-1$. Clearly $\mathcal{X}_N^{OL} = \mathcal{X}_f$.

Next we show how to compute $\mathcal{X}_i^{OL}$ for $i = 0, \dots, N-1$. Let the state and input constraint sets $\mathcal{X}$, $\mathcal{X}_f$ and $\mathcal{U}$ be the $\mathcal{H}$-polyhedra $A_x x \le b_x$, $A_f x \le b_f$, $A_u u \le b_u$, respectively. Recall that the disturbance sets are defined as $\mathcal{W}^a = \text{conv}\{w^{a,1}, \dots, w^{a,n_{\mathcal{W}^a}}\}$ and $\mathcal{W}^p = \text{conv}\{w^{p,1}, \dots, w^{p,n_{\mathcal{W}^p}}\}$. Define $U_i \triangleq [u_i', \dots, u_{N-1}']$ and the polyhedron $\mathcal{P}_i$ of robustly feasible states and input sequences at time i, defined as

$$\mathcal{P}_i = \{(U_i, x_i) \in \mathbb{R}^{m(N-i)+n} :\ G_i U_i - E_i x_i \le W_i\}. \tag{12.55}$$

In (12.55) G_i, E_i and W_i are obtained by collecting all the following inequalities:

- Input Constraints

$$A_u u_k \le b_u, \quad k = i, \dots, N-1$$

- State Constraints

$$A_x x_k \le b_x, \quad k = i, \dots, N-1 \text{ for all } \ w_l^a \in \mathcal{W}^a,\ w_l^p \in \mathcal{W}^p,\ l = i, \dots, k-1. \tag{12.56}$$

- Terminal State Constraints

$$A_f x_N \le b_f, \quad \text{for all } \ w_l^a \in \mathcal{W}^a,\ w_l^p \in \mathcal{W}^p,\ l = i, \dots, N-1. \tag{12.57}$$

Constraints (12.56)-(12.57) are enforced for all feasible disturbance sequences. In order to do so, we rewrite constraints (12.56)-(12.57) as a function of x_i and the input sequence U_i. Since we assumed that that $A(\cdot)$, $B(\cdot)$ are affine functions of w^p (cf. equation (12.40)) and since the composition of a convex constraint with an affine map generates a convex constraint (Section 1.2), we can use Proposition 12.25 to rewrite constraints (12.56) as

$$\begin{aligned} &A_x\left((\Pi_{j=i}^{k-1}A(w_j^p))x_i+\textstyle\sum_{l=i}^{k-1}(\Pi_{j=i}^{l-1}A(w_j^p)(B(w_{k-1-l}^p)u_{k-1-l}+Ew_{k-1-l}^a))\right)\le b_x,\\ &\text{for all } \ w_j^a\in\{w^{a,i}\}_{i=1}^{n_{\mathcal{W}^a}},\ w_j^p\in\{w^{p,i}\}_{i=1}^{n_{\mathcal{W}^p}},\ \forall\, j=i,\dots,k-1,\\ &\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad k=i,\dots,N-1. \end{aligned} \tag{12.58}$$

and imposing the constraints at all the vertices of the sets $\underbrace{\mathcal{W}^a\times\mathcal{W}^a\times\ldots\times\mathcal{W}^a}_{i,\dots,N-1}$ and $\underbrace{\mathcal{W}^p\times\mathcal{W}^p\times\ldots\times\mathcal{W}^p}_{i,\dots,N-1}$. Note that the constraints (12.58) are now linear in x_i and U_i. The same procedure can be repeated for constraint (12.57).

Once the matrices $G_i,\ E_i$ and W_i have been computed, the set $\mathcal{X}_i^{OL}$ is a polyhedron and can be computed by projecting the polyhedron $\mathcal{P}_i$ in (12.55) on the x_i space.

Recursive Approach: Closed-Loop Prediction

In the *recursive approach* we have

$$\begin{aligned} \mathcal{X}_i&=\{x\in\mathcal{X}:\ \exists u\in\mathcal{U}\text{ such that } A(w_i^p)x+B(w_i^p)u+Ew_i^a\in\mathcal{X}_{i+1}\ \forall w_i^a\in\mathcal{W}^a,\ w_i^p\in\mathcal{W}^p\},\\ &\quad i=0,\dots,N-1\\ \mathcal{X}_N&=\mathcal{X}_f. \end{aligned} \tag{12.59}$$

The definition of $\mathcal{X}_i$ in (12.59) is recursive and it requires that for any feasible initial state $x_i\in\mathcal{X}_i$ there exists a feasible input u_i which keeps the next state $A(w_i^p)x+B(w_i^p)u+Ew_i^a$ in the feasible set $\mathcal{X}_{i+1}$ for all feasible disturbances $w_i^a\in\mathcal{W}^a,\ w_i^p\in\mathcal{W}^p$.

Initializing $\mathcal{X}_N$ to $\mathcal{X}_f$ and solving (12.59) backward in time yields the feasible set $\mathcal{X}_0$ for the CROC-CL (12.47)–(12.49) which, as shown in example 12.6, is different from $\mathcal{X}_0^{OL}$.

Let $\mathcal{X}_i$ be the $\mathcal{H}$-polyhedron $A_{\mathcal{X}_i}x\le b_{\mathcal{X}_i}$. Then the set $\mathcal{X}_{i-1}$ is the projection of the following polyhedron

$$\begin{bmatrix}A_u\\0\\A_{\mathcal{X}_i}B(w_i^p)\end{bmatrix}u_i+\begin{bmatrix}0\\A_x\\A_{\mathcal{X}_i}A(w_i^p)\end{bmatrix}x_i\le\begin{bmatrix}b_u\\b_x\\b_{\mathcal{X}_i}-Ew_i^a\end{bmatrix}\ \text{ for all }\ w_i^a\in\{w^{a,i}\}_{i=1}^{n_{\mathcal{W}^a}},\ w_i^p\in\{w^{p,i}\}_{i=1}^{n_{\mathcal{W}^p}} \tag{12.60}$$

on the x_i space (note that we have used Proposition 12.25 and imposed state constraints at all the vertices of the sets $\mathcal{W}^a \times \mathcal{W}^p$).

The backward evolution in time of the feasible sets $\mathcal{X}_i$ enjoys the properties described by Theorems 10.2 and 10.3 for the nominal case. In particular if a robust controlled invariant set is chosen as terminal constraint $\mathcal{X}_f$, then set $\mathcal{X}_i$ grows as i becomes smaller and stops growing when it becomes the maximal robust stabilizable set.

Proposition 12.3. *Let the terminal constraint set $\mathcal{X}_f$ be a robust control invariant subset of $\mathcal{X}$. Then,*

1. *The feasible set $\mathcal{X}_i$, $i = 0, \dots, N-1$ is equal to the $(N-i)$-step robust stabilizable set:*
$$\mathcal{X}_i = \mathcal{K}_{N-i}(\mathcal{X}_f, \mathcal{W})$$
2. *The feasible set $\mathcal{X}_i$, $i = 0, \dots, N-1$ is robust control invariant and contained within the maximal robust control invariant set:*
$$\mathcal{X}_i \subseteq \mathcal{C}_\infty$$
3. *$\mathcal{X}_i \supseteq \mathcal{X}_j$ if $i < j$, $i = 0, \dots, N-1$. The size of the feasible $\mathcal{X}_i$ set stops increasing (with decreasing i) if and only if the maximal robust stabilizable set is finitely determined and $N-i$ is larger than its determinedness index $\bar{N}$, i.e.*
$$\mathcal{X}_i \supset \mathcal{X}_j \text{ if } N - \bar{N} < i < j < N$$
Furthermore,
$$\mathcal{X}_i = \mathcal{K}_\infty(\mathcal{X}_f, \mathcal{W}) \text{ if } i \le N - \bar{N}$$

Batch Approach: Closed-Loop Prediction

The batch approach can be modified in order to obtain $\mathcal{X}_i$ instead of $\mathcal{X}_i^{OL}$. The main idea is to augment the number of inputs by allowing one input sequence for each vertex i of the disturbance sets $\underbrace{\mathcal{W}^a \times \mathcal{W}^a \times \ldots \times \mathcal{W}^a}_{1,\ldots,N-1}$ and $\underbrace{\mathcal{W}^p \times \mathcal{W}^p \times \ldots \times \mathcal{W}^p}_{1,\ldots,N-1}$. The generic k-th input sequence can be written as

$$U_k = [u_0, \tilde{u}_1^{j_1}, \tilde{u}_2^{j_2}, \ldots \tilde{u}_{N-1}^{j_{N-1}}]$$

where $j_1 \in \{1, \ldots, n_{\mathcal{W}^a} n_{\mathcal{W}^p}\}$, $j_2 \in \{1, \ldots, (n_{\mathcal{W}^a} n_{\mathcal{W}^p})^2\}$,..., $j_i \in \{1, \ldots, (n_{\mathcal{W}^a} n_{\mathcal{W}^p})^{N-1}\}$. This yields $(n_{\mathcal{W}^a} n_{\mathcal{W}^p})(n_{\mathcal{W}^a} n_{\mathcal{W}^p})^2 \cdot (n_{\mathcal{W}^a} n_{\mathcal{W}^p})^{N-1}$ free control moves.

Since the approach is computationally demanding, we prefer to present the main idea through a very simple example rather than including all the tedious details.

Example 12.7. Consider the system

$$x_{k+1} = x_k + u_k + w_k \tag{12.61}$$

where x, u and w are state, input and disturbance, respectively. Let $u_k \in [-1, 1]$ and $w_k \in [-1, 1]$ be the feasible input and disturbance. The objective for player U is to play two moves in order to keep the state at time three x_3 in the set $\mathcal{X}_f = [-1, 1]$.

Batch approach

We rewrite the terminal constraint as

$$\begin{array}{c} x_3 = x_0 + u_0 + u_1 + u_2 + w_0 + w_1 + w_2 \in [-1,1] \\ \text{for all } \; w_0 \in [-1,1], \; w_1 \in [-1,1], \; w_2 \in [-1,1] \end{array} \tag{12.62}$$

which by Proposition 12.25 becomes

$$\begin{array}{c} -1 \le x_0 + u_0 + u_1 + u_2 + 3 \le 1 \\ -1 \le x_0 + u_0 + u_1 + u_2 + 1 \le 1 \\ -1 \le x_0 + u_0 + u_1 + u_2 - 1 \le 1 \\ -1 \le x_0 + u_0 + u_1 + u_2 - 3 \le 1 \end{array} \tag{12.63}$$

which by removing redundant constraints becomes the (infeasible) constraint

$$2 \le x_0 + u_0 + u_1 + u_2 \le -2$$

The set $\mathcal{X}_0^{OL}$ is the projection on the x_0 space of the polyhedron $\mathcal{P}_0$

$$\mathcal{P}_0 = \{(u_0, u_1, u_2, x_0) \in \mathbb{R}^4 : \begin{bmatrix} 1 & 0 & 0 \\ -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & -1 \\ 1 & 1 & 1 \\ -1 & -1 & -1 \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ u_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ -1 \end{bmatrix} x_0 \le \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ -2 \\ -2 \end{bmatrix} \} \tag{12.64}$$

which is empty since the terminal state constraint is infeasible.

Recursive approach

By using the recursive approach we obtain $\mathcal{X}_3 = \mathcal{X}_f = [-1, 1]$. We rewrite the terminal constraint as

$$x_3 = x_2 + u_2 + w_2 \in [-1, 1] \;\; \text{for all } w_2 \in [-1, 1] \tag{12.65}$$

which by Proposition 12.25 becomes

$$\begin{array}{l} -1 \leq x_2 + u_2 + 1 \leq 1 \\ -1 \leq x_2 + u_2 - 1 \leq 1 \end{array} \tag{12.66}$$

which by removing redundant constraints becomes

$$0 \leq x_2 + u_2 \leq 0$$

The set $\mathcal{X}_2$ is the projection on the x_2 space of the polyhedron

$$\begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 1 & 1 \\ -1 & -1 \end{bmatrix} [u_2,\ x_2] \leq \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} \tag{12.67}$$

which yields $\mathcal{X}_2 = [-1, 1]$. Since $\mathcal{X}_2 = \mathcal{X}_3$ one can conclude that $\mathcal{X}_2$ is the maximal controllable robust invariant set and $\mathcal{X}_0 = \mathcal{X}_1 = \mathcal{X}_2 = [-1, 1]$.

Batch approach with closed-loop predictions

The horizon is three and therefore we consider the disturbance set over the horizon 3-1=2, $\mathcal{W} \times \mathcal{W} = [-1, 1] \times [-1, 1]$. Sich a set has four vertices: $\{\tilde{w}_0^1 = 1,\ \tilde{w}_1^1 = 1\}$, $\{\tilde{w}_0^2 = -1,\ \tilde{w}_1^2 = 1\}$, $\{\tilde{w}_0^1 = 1,\ \tilde{w}_1^3 = -1\}$, $\{\tilde{w}_0^2 = -1,\ \tilde{w}_1^4 = -1\}$. We introduce an input sequence $u_0,\ \tilde{u}_1^i,\ \tilde{u}_2^j$, where the index $i \in \{1, 2\}$ is associated with the vertices $\tilde{w}_0^1$ and $\tilde{w}_0^2$ and the index $j \in \{1, 2, 3, 4\}$ is associated with the vertices $\tilde{w}_1^1,\ \tilde{w}_1^2,\ \tilde{w}_1^3,\ \tilde{w}_1^4$. The terminal constraint is thus rewritten as

$$x_3 = x_0 + u_0 + \tilde{u}_1^i + \tilde{u}_2^j + \tilde{w}_0^i + \tilde{w}_1^j + w_2 \in [-1, 1],\ i = 1, 2,\ j = 1, \ldots, 4,\ \forall\, w_2 \in [-1, 1] \tag{12.68}$$

which becomes

$$\begin{array}{ll} -1 \leq x_0 + u_0 + \tilde{u}_1^1 + \tilde{u}_2^1 + 0 + w_2 \leq 1, & \forall\, w_2 \in [-1, 1] \\ -1 \leq x_0 + u_0 + \tilde{u}_1^2 + \tilde{u}_2^2 + 0 + w_2 \leq 1, & \forall\, w_2 \in [-1, 1] \\ -1 \leq x_0 + u_0 + \tilde{u}_1^1 + \tilde{u}_2^3 + 2 + w_2 \leq 1, & \forall\, w_2 \in [-1, 1] \\ -1 \leq x_0 + u_0 + \tilde{u}_1^2 + \tilde{u}_2^4 - 2 + w_2 \leq 1, & \forall\, w_2 \in [-1, 1] \\ -1 \leq u_0 \leq 1 & \\ -1 \leq \tilde{u}_1^1 \leq 1 & \\ -1 \leq \tilde{u}_1^2 \leq 1 & \\ -1 \leq \tilde{u}_2^1 \leq 1 & \\ -1 \leq \tilde{u}_2^2 \leq 1 & \\ -1 \leq \tilde{u}_2^3 \leq 1 & \\ -1 \leq \tilde{u}_2^4 \leq 1 & \end{array} \tag{12.69}$$

The set $\mathcal{X}_0$ can be obtained by using Proposition 12.25 for the polyhedron (12.69) and projecting the resulting polyhedron in the $(x_0,\ u_0,\ \tilde{u}_1^1, \tilde{u}_1^2,\ \tilde{u}_2^1,\ \tilde{u}_2^2,\ \tilde{u}_2^3,\ \tilde{u}_2^4)$-space on the x_0 space. This yields $\mathcal{X}_0 = [-1, 1]$.

12.5 State Feedback Solution, 1-Norm and ∞-Norm Case

In this chapter we show how to find a *state feedback* solution to CROC problems, namely an explicit function $u_k^*(x_k)$ mapping the state x_k to its corresponding optimal input u_k^*, $\forall k = 0, \ldots, N-1$. *The reader is assumed to be familiar with the concept of multiparametric programming presented in Chapter 6.*

Multiparametric Programming with Piecewise-Linear Cost

Consider the optimization problem

$$\begin{array}{rl} J^*(x) = \min\limits_z & J(z,x) \\ \text{subj. to} & Gz \le W + Sx, \end{array} \tag{12.70}$$

where $z \in \mathbb{R}^s$ are the optimization variables, $x \in \mathbb{R}^n$ is the vector of parameters, $J(z,x)$ is the objective function and $G \in \mathbb{R}^{m\times s}$, $W \in \mathbb{R}^m$, and $S \in \mathbb{R}^{m\times n}$. Problem (12.70) with $J(z,x) = c'z$ is a multiparametric linear program (mp-LP) (see Chapter 6) .

For a given polyhedral set $\mathcal{K} \subseteq \mathbb{R}^n$ of parameters, solving (12.70) amounts to determining the set $\mathcal{K}^* \subseteq \mathcal{K}$ of parameters for which (12.70) is feasible, the value function $J^* : \mathcal{K}^* \to \mathbb{R}$, and the optimizer function[1] $z^* : \mathcal{K}^* \to \mathbb{R}^s$.

The properties of $J^*(\cdot)$ and $z^*(\cdot)$ have been analyzed in Chapter 6 and summarized in Theorem 6.4. Below we give some results based on this theorem.

Lemma 12.4. *Let $J : \mathbb{R}^s \times \mathbb{R}^n \to \mathbb{R}$ be a continuous convex piecewise affine function of (z,x) of the form*

$$J(z,x) = L_i z + H_i x + K_i \text{ for } \left[\begin{smallmatrix} z \\ x \end{smallmatrix}\right] \in \mathcal{R}_i \tag{12.71}$$

where $\{\mathcal{R}_i\}_{i=1}^{n_J}$ are polyhedral sets with disjoint interiors, $\mathcal{R} \triangleq \bigcup_{i=1}^{n_J} \mathcal{R}_i$ is a polyhedron and L_i, H_i and K_i are matrices of suitable dimensions. Then the multiparametric optimization problem (12.70) is an mp-LP.

Proof: As J is a convex piecewise affine function, it follows that $J(z,x)$ can be rewritten as $J(z,x) = \max_{i=1,\ldots,s} \{L_i z + H_i x + K_i\}$ (see Section 4.1.5). Then, it is easy to show that (12.70) is equivalent to the following mp-LP: $\min_{z,\varepsilon} \varepsilon$ subject to $Cz \le c + Sx$, $L_i z + H_i x + K_i \le \varepsilon$, $i = 1, \ldots, s$. $\square$

Lemma 12.5. *Let $f : \mathbb{R}^s \times \mathbb{R}^n \times \mathbb{R}^{n_w} \to \mathbb{R}$ and $g : \mathbb{R}^s \times \mathbb{R}^n \times \mathbb{R}^{n_w} \to \mathbb{R}^{n_g}$ be functions of (z,x,w) convex in w for each (z,x). Assume that the variable*

[1] In case of multiple solutions, we define $z^*(x)$ as one of the optimizers.

w belongs to the polyhedron $\mathcal{W}$ with vertices $\{\bar{w}_i\}_{i=1}^{N_{\mathcal{W}}}$. Then the min-max multiparametric problem

$$\begin{array}{rl} J^*(x) = \min_z & \max_{w\in\mathcal{W}} f(z,x,w) \\ & \textit{subj. to } g(z,x,w) \ \leq 0 \ \forall w \in \mathcal{W} \end{array} \tag{12.72}$$

is equivalent to the multiparametric optimization problem

$$\begin{array}{rl} J^*(x) = \min_{\mu,z} & \mu \\ & \textit{subj. to } \mu \geq f(z,x,\bar{w}_i), \ i=1,\dots,N_{\mathcal{W}} \\ & \qquad g(z,x,\bar{w}_i) \leq 0, \ i=1,\dots,N_{\mathcal{W}}. \end{array} \tag{12.73}$$

Proof: Easily follows from the fact that the maximum of a convex function over a convex set is attained at an extreme point of the set, cf. also [232].
□

Corollary 12.1. *If f is convex and piecewise affine in (z,x), i.e. $f(z,x,w) = \max_{i=1,\dots,n_f} \{L_i(w)z + H_i(w)x + K_i(w)\}$ and g is linear in (z,x) for all $w \in \mathcal{W}$, $g(z,x,w) = K_g(w) + L_g(w)x + H_g(w)z$ (with $K_g(\cdot)$, $L_g(\cdot)$, $H_g(\cdot)$, $L_i(\cdot)$, $H_i(\cdot)$, $K_i(\cdot)$, $i=1,\dots,n_f$, convex functions), then the min-max multiparametric problem (12.72) is equivalent to the mp-LP problem*

$$\begin{array}{rl} J^*(x) = min_{\mu,z} & \mu \\ & \textit{subj. to } \mu \geq K_j(\bar{w}_i) + L_j(\bar{w}_i)z + H_j(\bar{w}_i)x, \ i=1,\dots,N_{\mathcal{W}}, \ j=1,\dots,n_f \\ & \qquad L_g(\bar{w}_i)x + H_g(\bar{w}_i)z \leq -K_g(\bar{w}_i), \ i=1,\dots,N_{\mathcal{W}} \end{array} \tag{12.74}$$

Remark 12.5. As discussed in Proposition 12.2, in the case $g(x,z,w) = g_1(x,z)+g_2(w)$, the second constraint in (12.73) can be replaced by $g(z,x) \leq -\bar{g}$, where $\bar{g} \triangleq \left[\bar{g}_1,\dots,\bar{g}_{n_g}\right]'$ is a vector whose i-th component is

$$\bar{g}_i = \max_{w\in\mathcal{W}} \ g_2^i(w), \tag{12.75}$$

and $g_2^i(w)$ denotes the i-th component of $g_2(w)$. Similarly, if $f(x,z,w) = f_1(x,z) + f_2(w)$, the first constraint in (12.73) can be replaced by $f(z,x) \leq -\bar{f}$, where

$$\bar{f}_i = \max_{w\in\mathcal{W}} \ f_2^i(w). \tag{12.76}$$

Clearly, this has the advantage of reducing the number of constraints in the multiparametric program from $N_{\mathcal{W}}n_g$ to n_g for the second constraint in (12.73) and from $N_{\mathcal{W}}n_f$ to n_f for the first constraint in (12.73).

In the following subsections we show how to solve CROC problems in state feedback form by using multiparametric linear programming.

12.5.1 Batch Approach: Open-Loop Predictions

Theorem 12.2. *Consider the CROC-OL (12.41)–(12.45) with $p = 1$ or $p = \infty$. Assume that the parametric uncertainties are in the B matrix only ($A(w^p) \equiv A$). Then, there exists a solution $u^*(0) = f_0(x(0))$, $f_0 : \mathbb{R}^n \to \mathbb{R}^m$, which is continuous and PPWA*

$$f_0(x) = F_0^i x + g_0^i \quad \text{if} \quad x \in CR_0^i,\ i = 1, \ldots, N_0^r \tag{12.77}$$

where the polyhedral sets $CR_0^i \triangleq \{H_0^i x \leq k_0^i\}$, $i = 1, \ldots, N_0^r$, are a partition of the feasible set $\mathcal{X}_0^{OL}$. Moreover f_0 can be found by solving an mp-LP.

Proof: Since $x_k = A^k x_0 + \sum_{k=0}^{k-1} A^i [B(w_{k-1-i}^p) u_{k-1-i} + E w_{k-1-i}^a]$ is a linear function of the disturbances $\mathbf{w}^a \triangleq \{w_0^a, \ldots, w_{N-1}^a\}$, and $\mathbf{w}^p \triangleq \{w_0^p, \ldots, w_{N-1}^p\}$ for a fixed input sequence and x_0, the cost function in the maximization problem (12.41) is convex and piecewise affine with respect to the optimization vectors $\mathbf{w}^a$ and $\mathbf{w}^p$ and the parameters U_0, x_0. The constraints in (12.44) are linear in U_0 and x_0, for any $\mathbf{w}^a$ and $\mathbf{w}^p$. Therefore, by Lemma 12.5, problem (12.41)–(12.45) can be solved by solving an mp-LP through the enumeration of all the vertices of the sets $\mathcal{W}^a \times \mathcal{W}^a \times \ldots \times \mathcal{W}^a$ and $\mathcal{W}^p \times \mathcal{W}^p \times \ldots \times \mathcal{W}^p$. The theorem follows from the mp-LP properties described Theorem 6.4. □

Remark 12.6. In case of OL-CROC with additive disturbances only ($w(t) = 0$) the number of constraints in (12.45) can be reduced as explained in Remark 12.5.

12.5.2 Recursive Approach: Closed-Loop Predictions

Theorem 12.3. *There exists a state-feedback control law $u^*(k) = f_k(x(k))$, $f_k : \mathcal{X}_k \subseteq \mathbb{R}^n \to \mathcal{U} \subseteq \mathbb{R}^m$, solution of the CROC-CL (12.47)–(12.51) with cost (12.42) and $k = 0, \ldots, N-1$ which is time-varying, continuous and piecewise affine on polyhedra*

$$f_k(x) = F_k^i x + g_k^i \quad \text{if} \quad x \in CR_k^i, \quad i = 1, \ldots, N_k^r \tag{12.78}$$

where the polyhedral sets $CR_k^i = \{x \in \mathbb{R}^n\ :\ H_k^i x \leq K_k^i\}$, $i = 1, \ldots, N_k^r$ are a partition of the feasible polyhedron $\mathcal{X}_k$. Moreover f_i, $i = 0, \ldots, N-1$ can be found by solving N mp-LPs.

Proof: Consider the first step $j = N-1$ of dynamic programming applied to the CROC-CL problem (12.47)–(12.49) with cost (12.42)

$$J^*_{N-1}(x_{N-1}) \triangleq \min_{u_{N-1}} J_{N-1}(x_{N-1}, u_{N-1}) \tag{12.79}$$

$$\text{subj. to} \begin{cases} Fx_{N-1} + Gu_{N-1} \leq f \\ A(w^p_{N-1})x_{N-1} + B(w^p_{N-1})u_{N-1} + Ew^a_{N-1} \in \mathcal{X}_f \\ \forall w^a_{N-1} \in \mathcal{W}^a, w^p_{N-1} \in \mathcal{W}^p \end{cases} \tag{12.80}$$

$$J_{N-1}(x_{N-1}, u_{N-1}) \triangleq \max_{w^a_{N-1} \in \mathcal{W}^a,\ w^p_{N-1} \in \mathcal{W}^p} \left\{ \begin{array}{l} \|Qx_{N-1}\|_p + \|Ru_{N-1}\|_p + \\ +\|P(A(w^p_{N-1})x_{N-1} + \\ +B(w^p_{N-1})u_{N-1} + Ew^a_{N-1})\|_p \end{array} \right\}. \tag{12.81}$$

The cost function in the maximization problem (12.81) is piecewise affine and convex with respect to the optimization vector w^a_{N-1}, w^p_{N-1} and the parameters u_{N-1}, x_{N-1}. Moreover, the constraints in the minimization problem (12.80) are linear in (u_{N-1}, x_{N-1}) for all vectors w^a_{N-1}, w^p_{N-1}. Therefore, by Corollary 12.1, $J^*_{N-1}(x_{N-1})$, $u^*_{N-1}(x_{N-1})$ and $\mathcal{X}_{N-1}$ are computable via the mp-LP:

$$J^*_{N-1}(x_{N-1}) \triangleq \min_{\mu, u_{N-1}} \mu \tag{12.82a}$$

$$\begin{aligned} \text{subj. to } & \mu \geq \|Qx_{N-1}\|_p + \|Ru_{N-1}\|_p + \\ & + \|P(A(\bar{w}^p_h)x_{N-1} + B(\bar{w}^p_h)u_{N-1} + E\bar{w}^a_i)\|_p \qquad (12.82\text{b}) \\ & Fx_{N-1} + Gu_{N-1} \leq f \qquad (12.82\text{c}) \\ & A(\bar{w}^p_h)x_{N-1} + B(\bar{w}^p_h)u_{N-1} + E\bar{w}^a_i \in \mathcal{X}_N \qquad (12.82\text{d}) \\ & \forall i = 1, \dots, n_{\mathcal{W}^a},\ \forall h = 1, \dots, n_{\mathcal{W}^p}. \end{aligned}$$

where $\{\bar{w}^a_i\}_{i=1}^{n_{\mathcal{W}^a}}$ and $\{\bar{w}^p_h\}_{h=1}^{n_{\mathcal{W}^p}}$ are the vertices of the disturbance sets $\mathcal{W}^a$ and $\mathcal{W}^p$, respectively. By Theorem 6.4, J^*_{N-1} is a convex and piecewise affine function of x_{N-1}, the corresponding optimizer u^*_{N-1} is piecewise affine and continuous, and the feasible set $\mathcal{X}_{N-1}$ is a convex polyhedron. Therefore, the convexity and linearity arguments still hold for $j = N-2, \dots, 0$ and the procedure can be iterated backwards in time j, proving the theorem. The theorem follows from the mp-LP properties described in Theorem 6.4. □

Remark 12.7. Let n_a and n_b be the number of inequalities in (12.82b) and (12.82d), respectively, for any i and h. In case of additive disturbances only ($n_{\mathcal{W}^p} = 0$) the total number of constraints in (12.82b) and (12.82d) for all i and h can be reduced from $(n_a + n_b)n_{\mathcal{W}^a}n_{\mathcal{W}^p}$ to $n_a + n_b$ as shown in Remark 12.5.

Remark 12.8. The closed-loop solution $u^*(k) = f_k(x(k))$ can be also obtained by using the modified batch approach with closed-loop prediction as discussed in Section 12.4. The idea there is to augment the number of free inputs by allowing one free sequence $\tilde{u}^i_0, \dots \tilde{u}^i_{N-1}$ for each vertex i of the sets

$\underbrace{\mathcal{W}^a \times \mathcal{W}^a \times \ldots \times \mathcal{W}^a}_{0,\ldots,N-1} \times \underbrace{\mathcal{W}^p \times \mathcal{W}^p \times \ldots \times \mathcal{W}^p}_{0,\ldots,N-1}$. The high number of extreme points of such set and the consequent high number of inputs and constraints make this approach not computationally attractive.

12.5.3 Solution to CROC-CL and CROC-OL via mp-MILP*

Consider the multiparametric mixed-integer linear program (mp-MILP)

$$\begin{array}{rl} J^*(x) = \min\limits_z & \{J(z,x) = c'z\} \\ & \text{subj. to } Gz \leq W + Sx. \end{array} \tag{12.83}$$

where $z \triangleq [z_c, z_d]$, $z_c \in \mathbb{R}^{n_c}$, $z_d \in \{0,1\}^{n_d}$, $s \triangleq n_c + n_d$ is the optimization vector, $x \in \mathbb{R}^n$ is the vector of parameters, and where z is the optimization vector, $x \in \mathbb{R}^s$ is the vector of parameters.

For a given polyhedral set $\mathcal{K} \subseteq \mathbb{R}^n$ of parameters, solving (12.83) amounts to determining the set $\mathcal{K}^* \subseteq \mathcal{K}$ of parameters for which (12.83) is feasible, the value function $J : \mathcal{K}^* \to \mathbb{R}$, and the optimizer function[2] $z^* : \mathcal{K}^* \to \mathbb{R}^s$.

The properties of $J^*(\cdot)$ and $z^*(\cdot)$ have been analyzed in Chapter 6 and summarized in Theorem 6.10. Below we state some properties based on these theorems.

Lemma 12.6. *Let $J : \mathbb{R}^s \times \mathbb{R}^n \to \mathbb{R}$ be a continuous piecewise affine (possibly nonconvex) function of (z,x),*

$$J(z,x) = L_i z + H_i x + K_i \text{ for } \left[\begin{smallmatrix} z \\ x \end{smallmatrix}\right] \in \mathcal{R}_i, \tag{12.84}$$

where $\{\mathcal{R}_i\}_{i=1}^{n_J}$ are polyhedral sets with disjoint interiors, $\mathcal{R} \triangleq \bigcup_{i=1}^{n_J} \mathcal{R}_i$ is a (possibly non-convex) polyhedral set and L_i, H_i and K_i are matrices of suitable dimensions. Then the multiparametric optimization problem

$$\begin{array}{ll} J^*(x) \triangleq \min_z & J(z,x) \\ \text{subj. to } Cz \leq c + Sx. & \end{array} \tag{12.85}$$

is an mp-MILP.

Proof: By following the approach of [37] to transform piecewise affine functions into a set of mixed-integer linear inequalities, introduce the auxiliary binary optimization variables $\delta_i \in \{0,1\}$, defined as

$$[\delta_i = 1] \;\leftrightarrow\; \left[\left[\begin{smallmatrix} z \\ x \end{smallmatrix}\right] \in \mathcal{R}_i\right], \tag{12.86}$$

where δ_i, $i = 1,\ldots,n_J$, satisfy the exclusive-or condition $\sum_{i=1}^{n_J} \delta_i = 1$, and set

[2] In case of multiple solutions, we define $z^*(x)$ as one of the optimizers.

$$J(z,x) = \sum_{i=1}^{n_J} q_i \tag{12.87}$$

$$q_i \triangleq [L_i z + H_i x + K_i]\delta_i \tag{12.88}$$

where q_i are auxiliary continuous optimization vectors. By transforming (12.86)–(12.88) into mixed-integer linear inequalities [37], it is easy to rewrite (12.85) as a multiparametric MILP. □

Theorem 12.4. *By solving 2N mp-MILPs, the solution of the CROC-CL (12.47)–(12.51) problem with additive disturbances only ($n_{\mathcal{W}^p} = 0$) can be obtained in state feedback piecewise affine form (12.78)*

Proof: Consider the first step $j = N-1$ of the dynamic programming solution (12.47)–(12.49) to CROC-CL. From the terminal conditions (12.51) it follows that the cost function in the maximization problem is piecewise affine with respect to both the optimization vector w^a_{N-1} and the parameters u_{N-1}, x_{N-1}. By Lemma 12.6, (12.47)-(12.49) can be computed via mp-MILP, and, by Theorem 6.10, it turns out that $J_{N-1}(u_{N-1},\, x_{N-1})$ is a piecewise affine and continuous function. Then, since constraints (12.80) are linear with respect to w^a_{N-1} for each u_{N-1}, x_{N-1}, we can apply Lemma 12.5 by solving LPs of the form (12.26). Then, by Lemma 12.6, $J^*_{N-1}(x_{N-1})$ is again computable via mp-MILP, and by Theorem 6.10 it is a piecewise affine and continuous function of x_{N-1}. By virtue of Theorem 6.10, $\mathcal{X}_{N-1}$ is a (possible non-convex) polyhedral set and therefore the above maximization and minimization procedures can be iterated to compute the solution (12.78) to the CROC-CL problem. □

Theorem 12.5. *By solving two mp-MILPs, the solution $u^*(x_0)$ to the CROC-OL (12.41)–(12.45) with additive disturbances only ($w(t) = 0$) can be computed in explicit piecewise affine form (12.77).*

Proof: The objective function in the maximization problem (12.41) is convex and piecewise affine with respect to the optimization vector $\mathbf{w}^a = \{w^a_0, \ldots, w^a_{N-1}\}$ and the parameters $U = \{u_0, \ldots, u_{N-1}\}$, x_0. By Lemma 12.6, it can be solved via mp-MILP. By Theorem 6.10, the value function J is a piecewise affine function of U and x_0 and the constraints in (12.44) are a linear function of the disturbance $\mathbf{w}^a$ for any given U and x_0. Then, by Lemma 12.6 and Lemma 12.5 the minimization problem is again solvable via mp-MILP, and the optimizer $U^* = \{u^*_0,\ldots,u^*_{N-1}\}$ is a piecewise affine function of x_0. □

Remark 12.9. Theorems 12.4, 12.5 and Theorems 12.2, 12.3 propose two different ways of finding the PPWA solution to constrained robust optimal control by using dynamic programming. The solution approach of Theorems 12.4, 12.5 is more general than the one of Theorems 12.2, 12.3 as it does not exploit convexity, so that it may be also used in other contexts, for instance in CROC-CL of hybrid systems.

12.6 Parametrizations of the Control Policies

From the previous sections it is clear that CROC-OL (12.41)–(12.45) is conservative. Since are optimizing an open-loop control sequence that has to cope with all possible future disturbance realizations, without taking future measurements into account. The CROC-CL (12.47)–(12.51) formulation overcomes this issue but it can quickly lead to an intractable problem.

This section presents an alternative approach which introduces feedback in the system and, in some cases, can be more efficient than CROC-CL. The idea is to parameterize the control sequence in the states vector and optimize over these parameters. The approach is described next for systems with additive uncertainties.

Consider the worst case cost function as

$$J_0(x(0), U_0) \triangleq \max_{w_0^a, \ldots, w_{N-1}^a} p(x_N) + \sum_{k=0}^{N-1} q(x_k, u_k)$$

$$\text{subj. to} \begin{cases} x_{k+1} = Ax_k + Bu_k + Ew_k^a \\ w_k^a \in \mathcal{W}^a, \\ k = 0, \ldots, N-1 \end{cases} \tag{12.89}$$

where N is the time horizon and $U_0 \triangleq [u_0', \ldots, u_{N-1}']' \in \mathbb{R}^s$, $s \triangleq mN$ the vector of the input sequence. Consider the robust optimal control problem

$$J_0^*(x_0) \triangleq \min_{U_0} J_0(x_0, U_0) \tag{12.90}$$

$$\text{subj. to} \left\{ \begin{array}{l} x_k \in \mathcal{X},\ u_k \in \mathcal{U} \\ x_{k+1} = Ax_k + Bu_k + Ew_k^a \\ x_N \in \mathcal{X}_f \\ k = 0, \ldots, N-1 \end{array} \right\} \begin{array}{l} \forall w_k^a \in \mathcal{W}^a \\ \forall k = 0, \ldots, N-1 \end{array} \tag{12.91}$$

Consider the parametrization of the control sequence

$$u_k = \sum_{i=0}^{k} L_{k,i} x_i + g_i, \quad k \in \{0, \ldots, N-1\} \tag{12.92}$$

with the compact notation:

$$U_0 = Lx + g,$$

where $x = [x_0', x_1', \ldots, x_N']'$ and

$$L = \begin{bmatrix} L_{0,0} & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ L_{N-1,0} & \cdots & L_{N-1,N-1} & 0 \end{bmatrix}, \quad g = \begin{bmatrix} g_0 \\ \vdots \\ g_{N-1} \end{bmatrix} \tag{12.93}$$

where $L \in \mathbb{R}^{mN \times nN}$ and $g \in \mathbb{R}^{mN}$ are unknown feedback control gain and offset, respectively. With the parametrization (12.92) the robust control problem (12.89)-(12.91) becomes

$$J_0^{Lg}(x(0), L, g) \triangleq \max_{w_0^a, \ldots, w_{N-1}^a} p(x_N) + \sum_{k=0}^{N-1} q(x_k, u_k)$$

$$\text{subj. to } \begin{cases} x_{k+1} = Ax_k + Bu_k + Ew_k^a \\ w_k^a \in \mathcal{W}^a, \\ u_k = \sum_{i=0}^{k} L_{k,i} x_i + g_i \\ k = 0, \ldots, N-1 \end{cases} \tag{12.94}$$

$$J_0^{Lg^*}(x_0) \triangleq \min_{L,g} J_0^{Lg}(x_0, L, g) \tag{12.95}$$

$$\text{subj. to } \left. \begin{cases} x_k \in \mathcal{X}, \ u_k \in \mathcal{U} \\ x_{k+1} = Ax_k + Bu_k + Ew_k^a \\ u_k = \sum_{i=0}^{k} L_{k,i} x_i + g_i \\ x_N \in \mathcal{X}_f \\ k = 0, \ldots, N-1 \end{cases} \right\} \begin{array}{l} \forall w_k^a \in \mathcal{W}^a \\ \forall k = 0, \ldots, N-1 \end{array} \tag{12.96}$$

We denote with $\mathcal{X}_0^{Lg} \subseteq \mathcal{X}$ the set of states x_0 for which the robust optimal control problem (12.95)-(12.96) is feasible, i.e.,

$$\begin{array}{rl} \mathcal{X}_0^{Lg} = & \left\{ x_0 \in \mathbb{R}^n : \ \mathcal{P}_0^{Lg}(x_0) \neq \emptyset \right\} \\ \mathcal{P}_0^{Lg}(x_0) = & \{ L, g : x_k \in \mathcal{X}, \ u_k \in \mathcal{U}, \ k = 0, \ldots, N-1, \ x_N \in \mathcal{X}_f \\ & \forall \ w_k^a \in \mathcal{W}^a \ k = 0, \ldots, N-1, \text{ where } x_{k+1} = Ax_k + Bu_k + Ew_k^a, \ u_k = \sum_{i=0}^{k} L_{k,i} x_i \} \end{array} \tag{12.97}$$

Problem (12.94) looks for the worst value $J_0^{Lg}(x_0, L, g)$ of the performance index and the corresponding worst sequences $\mathbf{w}^{p*}$ as a function of x_0 and the controller gain L and offset g.

Problem (12.95)–(12.96) minimizes (over L and g) the worst performance subject to the constraint that the input sequence $U_0 = Lx + g$ must be feasible *for all* possible disturbance realizations. Notice that formulation (12.94)–(12.96) is based on an *closed-loop* prediction. Unfortunately the set $\mathcal{P}_0^{Lg}(x_0)$ is non–convex, in general [177]. Therefore, finding L and g for a given x_0 may be difficult.

Consider now the parametrization of the control sequence in past disturbances

$$u_k = \sum_{i=0}^{k-1} M_{k,i} w_i + v_i, \quad k \in \mathbb{N}_{[0,N-1]}, \tag{12.98}$$

which can be compactly written as:

$$u = Mw + v$$

where

$$M = \begin{bmatrix} 0 & \cdots & \cdots & 0 \\ M_{1,0} & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ M_{N-1,0} & \cdots & M_{N-1,N-2} & 0 \end{bmatrix}, \quad v = \begin{bmatrix} v_0 \\ \vdots \\ \vdots \\ v_{N-1} \end{bmatrix}. \tag{12.99}$$

Notice that since

$$w_k = x_{k+1} - Ax_k - Bu_k, \quad k \in \{0, \ldots, N-1\}.$$

the parametrization (12.98) can also be interpreted as a parametrization in the state space. The advantages of using (12.98) are explained next.

With the parametrization (12.98) the robust control problem (12.89-(12.91) becomes

$$J_0^{Mv}(x(0), M, v) \triangleq \max_{w_0^a, \ldots, w_{N-1}^a} p(x_N) + \sum_{k=0}^{N-1} q(x_k, u_k)$$

$$\text{subj. to} \begin{cases} x_{k+1} = Ax_k + Bu_k + Ew_k^a \\ w_k^a \in \mathcal{W}^a, \\ u_k = \sum_{i=0}^{k-1} M_{k,i} w_i + v_i \\ k = 0, \ldots, N-1 \end{cases} \tag{12.100}$$

$$J_0^{Mv^*}(x_0) \triangleq \min_{M,v} J_0^{Mv}(x_0, M, v) \tag{12.101}$$

$$\text{subj. to} \left\{ \begin{array}{l} x_k \in \mathcal{X}, \ u_k \in \mathcal{U} \\ x_{k+1} = Ax_k + Bu_k + Ew_k^a \\ u_k = \sum_{i=0}^{k-1} M_{k,i} w_i + v_i \\ x_N \in \mathcal{X}_f \\ k = 0, \ldots, N-1 \end{array} \right\} \begin{array}{l} \forall w_k^a \in \mathcal{W}^a \\ \forall k = 0, \ldots, N-1 \end{array} \tag{12.102}$$

We denote with $\mathcal{X}_0^{Mv} \subseteq \mathcal{X}$ the set of states x_0 for which the robust optimal control problem (12.101)-(12.102) is feasible, i.e.,

$$\begin{array}{ll} \mathcal{X}_0^{Mv} = & \left\{x_0 \in \mathbb{R}^n : \ \mathcal{P}_0^{Mv}(x_0) \neq \emptyset\right\} \\ \mathcal{P}_0^{Mv}(x_0) = & \{M, v : x_k \in \mathcal{X}, \ u_k \in \mathcal{U}, \ k = 0, \ldots, N-1, \ x_N \in \mathcal{X}_f \\ & \forall \ w_k^a \in \mathcal{W}^a \ k = 0, \ldots, N-1, \text{ where } x_{k+1} = Ax_k + Bu_k + Ew_k^a, \ u_k = \sum_{i=0}^{k-1} M_{k,i} w_i + v_i\} \end{array} \tag{12.103}$$

The following result has been proven in [177].

Theorem 12.6. *Consider the control parameterizations (12.92), (12.98) and the corresponding feasible sets $\mathcal{X}_0^{Lg}$ in (12.97) and $\mathcal{X}_0^{Mv}$ in (12.103). Then,*

$$\mathcal{X}_0^{Lg} = \mathcal{X}_0^{Mv}$$

and $\mathcal{P}_N^{Mv}(x_0)$ *is convex in* M *and* v*.*

Note that in general $\mathcal{X}_0^{Mv}$ and $J_0^{Mv*}(x_0)$ are different from the corresponding CROC-CL solutions $\mathcal{X}_0$ and $J_0^*(x_0)$. In particular $\mathcal{X}_0^{Mv} \subseteq \mathcal{X}_0$ and $J_0^{Mv*}(x_0) \geq J_0^*(x_0)$.

The idea of the parametrization (12.98) appears in the work of Gartska & Wets in 1974 in the context of stochastic optimization [110]. Recently, it re-appeared in robust optimization work by Guslitzer and Ben-Tal [127, 42], and in the context of robust MPC in the wok of van Hessem & Bosgra, Löfberg and Goulart & Kerrigan [255, 177, 121].

12.7 Example

Example 12.8. Consider the problem of robustly regulating to the origin the system

$$x(t+1) = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t) + w^a(t)$$

subject to the input constraints

$$\mathcal{U} = \{u \in \mathbb{R} \;:\; -3 \leq u \leq 3\}$$

and the state constraints

$$\mathcal{X} = \{x \in \mathbb{R}^2 \;:\; -10 \leq x \leq 10 \; k = 0, \ldots, 3\}$$

The two-dimensional disturbance w^a is restricted to the set $\mathcal{W}^a = \{v : \|w^a\|_\infty \leq 1.5\}$.

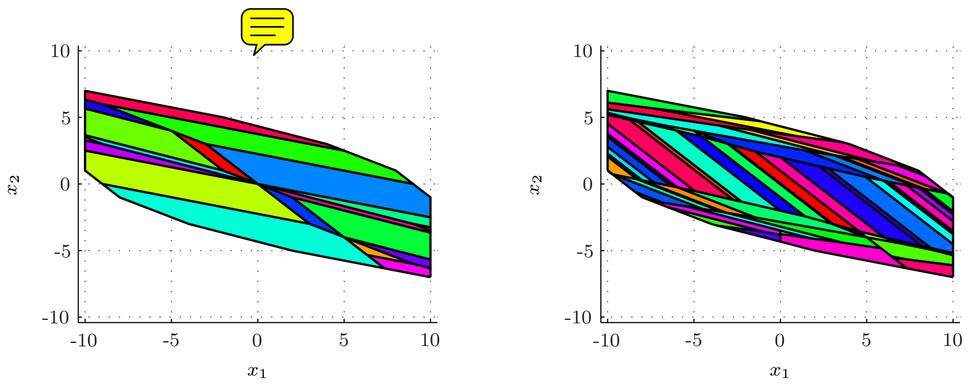
Next we compute the state feedback control law (12.104) obtained by solving the CROC-CL (12.47)–(12.51) and the CROC-OL (12.41)–(12.45). We use the cost function

$$\|Px_N\|_\infty + \sum_{k=0}^{N-1} (\|Qx_k\|_\infty + |Ru_k|)$$

with $N = 5$, $P = Q = \left[\begin{smallmatrix} 1 & 1 \\ 0 & 1 \end{smallmatrix}\right]$, $R = 1.8$ and we set $\mathcal{X}_f = \mathcal{X}$.

CROC-OL. The min-max problem is formulated as in (12.41)–(12.45). The resulting polyhedral partition consists of 22 regions and it is depicted in Figure 12.7(a). We remark that the CROC-OL problem (12.41)–(12.45) is infeasible for horizon N greater than five.

CROC-CL. The min-max problem is formulated as in (12.47)–(12.49) and solved using the approach of Theorem 12.3. The resulting polyhedral partition consists of 64 regions and is depicted in Figure 12.7(b).



(a) CROC-OL (b) CROC-CL

Fig. 12.7 Polyhedral partition of the state-space corresponding to the explicit solution of CROC-OL and CROC-CL

12.8 Robust Receding Horizon Control

A robust receding horizon controller for system (12.38)-(12.40) which enforces the constraints (12.39) at each time t in spite of additive and parametric uncertainties can be obtained immediately by setting

$$u(t) = f_0^*(x(t)), \tag{12.104}$$

where $f_0^*(x_0) : \mathbb{R}^n \to \mathbb{R}^m$ is the solution to the CROC-OL or CROC-CL problems discussed in the previous sections. In this way we obtain a state feedback strategy defined at all time steps $t = 0, 1, \ldots$, from the associated finite time CROC problems.

If f_0 is computed by solving CROC-CL (12.47)–(12.51) (CROC-OL (12.41)–(12.45)), then the RHC law (12.104) is called a robust receding horizon controller with (open-loop) closed-loop predictions. The closed-loop system obtained by controlling (12.38)-(12.40) with the RHC (12.104) is

$$x(k+1) = A(w^p)x(k) + B(w^p)f_0(x(k)) + Ew^a \triangleq f_{cl}(x(k), w^p, w^a),\ k \geq 0 \tag{12.105}$$

If $p = 1$ or $p = \infty$ in the CROC-CL (12.47)–(12.51) (CROC-OL (12.41)–(12.45)), then from Theorem 12.3 (Theorem 12.2) we can immediately conclude that the robust RHC law (12.104) is piecewise affine and thus its on-line computation consists of a simple function evaluation.

Convergence and persistent feasibility of the robust receding horizon controller are not guaranteed as discussed in Chapter 11 for nominal receding horizon controllers. In the robust RHC case is it desirable to obtain robust convergence to a set $\mathcal{O} \subseteq \mathcal{X}_f$ (rather than the convergence to an equilibrium point) for all $\mathcal{X}_0$. In other words, the goal is to design a RHC control law which drives any feasible state in $\mathcal{X}_0$ into the set $\mathcal{O}$ for all admissible disturbances and keeps the states inside the set $\mathcal{O}$ for all future time and for all admissible disturbances. Clearly this is possible only if $\mathcal{O}$ is a robust control invariant set for system (12.38)-(12.40).

We define a distance of a point $x \in \mathbb{R}^n$ from a nonempty set $\mathcal{Y} \subset \mathbb{R}^n$ as:

$$d(x, \mathcal{Y}) \triangleq \inf_{y \in \mathcal{Y}} d(x, y) \tag{12.106}$$

The following theorem presents sufficient conditions for convergence and persistent feasibility in the robust case. It can be proven by using the arguments of Theorem 11.2.

Theorem 12.7. *Consider system (12.38)-(12.40), the closed-loop RHC law (12.104) where $f_0(x)$ is obtained by solving the CROC-CL (12.47)–(12.51) for $x(0) = x$ and the closed-loop system (12.105). Assume that*

(A1) The sets $\mathcal{X}$, $\mathcal{X}_f$, $\mathcal{U}$, $\mathcal{W}^a$, $\mathcal{W}^p$ are compact.
(A2) $\mathcal{X}_f$ and $\mathcal{O}$ are robust control invariants, $\mathcal{O} \subseteq \mathcal{X}_f \subseteq \mathcal{X}$.
(A3) $J^p(x) \leq 0 \ \forall x \in \mathcal{X}_f$ where

$$J^p(x) = \min_u J^c(x, u)$$
$$\text{subj. to } \begin{cases} u \in \mathcal{U} \\ A(w^p)x + B(w^p)u + Ew^a \in \mathcal{X}_f \ \forall \ w^a \in \mathcal{W}^a \ \ w^p \in \mathcal{W}^p \end{cases} \tag{12.107}$$

and

$$J^c(x, u) = \max_{w^a, w^p} p(x^+) - p(x) + q(x, u)$$
$$\text{subj. to } \begin{cases} w^a \in \mathcal{W}^a, \ \ w^p \in \mathcal{W}^p \\ x^+ = A(w^p)x + B(w^p)u + Ew^a \end{cases} \tag{12.108}$$

(A4) There exist constants $c_1, c_2 > 0$ such that

$$c_1 d(x, \mathcal{O}) \leq p(x) \leq c_2 d(x, \mathcal{O}) \ \forall x \in \mathcal{X}_0 \tag{12.109}$$

(A5) There exist constants $c_3, c_4 > 0$ such that

$$c_3 d(x, \mathcal{O}) \leq q(x, u) \leq c_4 d(x, \mathcal{O}) \ \forall \ (x, u) \in \mathcal{X}, \times \mathcal{U} \tag{12.110}$$

Then, for all $x \in \mathcal{X}_0$, $\lim_{k \to \infty} d(x(k), \mathcal{O}) = 0$. $\mathcal{X}_0$ is called the region of attraction.

Remark 12.10. Theorem 12.7 holds also for the CROC-OL (12.41)–(12.45). The region of attraction will be $\mathcal{X}_0^{OL}$.

Remark 12.11. Assumptions A4 and A5 in Theorem 12.7 imply that stage cost $q(x, u)$ and terminal cost $p(x)$ in (12.41) need to be zero in $\mathcal{O}$. Note that this cannot be obtained with $p = 2$ in (12.41).

Remark 12.12. Robust control invariance of the set $\mathcal{X}_f$ in Assumption (A2) of Theorem 12.7 implies that problem (12.107) in Assumption (A3) is always feasible.

Remark 12.13. Robust control invariance of the set $\mathcal{O}$ in Assumption (A2) of Theorem 12.7 is also implicitly guaranteed by Assumption (A3) as shown next.

From Remark 12.11, Assumption (A3) in $\mathcal{O}$ becomes $\min_u \max_{w^a,w^p}(p(A(w^p)x+B(w^p)u + Ew^a) \leq 0$ for all $x \in \mathcal{O}$. From Assumption (A4), this can be verified only if it is equal to zero, i.e. if it exists a $u \in \mathcal{U}$ such that $A(w^p)x + B(w^p)u + Ew^a \in \mathcal{O}$ for all $w^a \in \mathcal{W}^a$ $w^p \in \mathcal{W}^p$ (since $\mathcal{O}$ is the only place where $p(x)$ can be zero). This implies the robust control invariance of $\mathcal{O}$.

12.9 Literature Review

An extensive treatment of robust invariant sets can be found in [51, 52, 49, 53]. The proof to Theorem 12.1 can be fond in [158, 90] For the derivation of the algorithms 12.1, 12.2 for computing robust invariant sets (and their finite termination) see [10, 45, 158, 115, 151].

Min-max robust constrained optimal control was originally proposed by Witsenhausen [264]. In the context of robust MPC, the problem was tackled by Campo and Morari [71], and further developed in [5] for SISO FIR plants. Kothare *et al.* [164] optimize robust performance for polytopic/multi-model and linear fractional uncertainty, Scokaert and Mayne [232] for additive disturbances, and Lee and Yu [172] for linear time-varying and time-invariant state-space models depending on a vector of parameters $\theta \in \Theta$, where Θ is either an ellipsoid or a polyhedron. Other suboptimal CROC-Cl strategies have been proposed in [164, 25, 165]. For stability and feasibility of the robust RHC (12.38), (12.104) we refer the reader to [38, 180, 188, 21].

Chapter 13
On-line Control Computation

In previous chapters we have shown how to compute the solution to the constrained finite time optimal control problem as an explicit piecewise affine function of the initial state. This method reveals its effectiveness when applied to *Receding Horizon Control* (RHC). Having a precomputed solution as an explicit piecewise affine on polyhedra function of the state vector reduces the on-line computation of the RHC control law to a function evaluation, therefore avoiding the on-line solution of a quadratic or linear program. The main drawback of such explicit optimal 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.

13.1 Introduction

In Chapter 10 we have shown how to compute the solution to the constrained finite time optimal control (CFTOC) problem as an explicit piecewise affine function of the initial state. Such a function is computed off-line by using a multiparametric program solver, 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 applied to *Receding Horizon Control* (RHC). Having a precomputed solution as an explicit piecewise affine on polyhedra (PPWA) function of the state vector reduces the on-line computation of the RHC control law to a function evaluation, therefore avoiding the on-line solution of a quadratic or linear program.

The main drawback of such explicit optimal 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 way to implement the piecewise affine feedback laws is to store the polyhedral cells $\{H^i x \leq K^i\}$, perform on-line a linear search through them to locate the one which contains $x(t)$ and then look up the corresponding feedback gain (F^i, g^i) (note that this procedure can be easily parallelized). This chapter presents implementation techniques which avoid the storage and the evaluation of the polyhedral and can significantly reduce the on-line storage demands and computational complexity of RHC. They exploit the properties of the value function and the piecewise affine optimal control law of the constrained finite time optimal control problem.

13.2 Efficient On-Line Algorithms

Let the explicit optimal control law be:

$$u^*(x) = F^i x + G^i, \quad \forall x \in \mathcal{P}_i, \quad i = 1, \dots, N^r \tag{13.1}$$

where $F^i \in \mathbb{R}^{m\times n}$, $G^i \in \mathbb{R}^m$, and $\mathcal{P}_i = \left\{ x \in \mathbb{R}^n \ : \ H^i x \leq K^i, \ H^i \in \mathbb{R}^{N_c^i \times n}, \ K^i \in \mathbb{R}^{N_c^i} \right\}$, $i = 1, \dots, N^r$ is a polyhedral partition of $\mathcal{X}$. In the following H_j^i denotes the j-row of the matrix H^i and N_c^i is the numbers of constraints defining the i-th polyhedron. The on-line implementation of the control law (13.1) is simply executed according to the following steps:

Algorithm 13.2.1

```
1   Measure the current state $x(t)$
2   Search for the $j$-th polyhedron that contains $x(t)$, $(H^j x(t) \leq K^j)$
3   Implement the $j$-th control law $(u(t) = F^j x(t) + G^j)$
```

In Algorithm 13.2.1, step (2) is critical and it is the only step whose efficiency can be improved. 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 13.2.2

Input: Current state $x(t)$ and polyhedral partion $\{\mathcal{P}_i\}_{i=1}^{N^r}$ of the control law (13.1)

Output: Index j of the polyhedron $\mathcal{P}_j$ in the control law (13.1) containing the current state $x(t)$

```
1   $i = 0$, notfound=1;
2   while $i \leq N^r$ and notfound
3       $j = 0$, stillfeasible=1
4       while $j \leq N_c^i$ and stillfeasible
5           if $H_j^i x(t) > K_j^i$ then stillfeasible=0
6           else $j = j + 1$
7       end
8       if stillfeasible=1 then notfound=0
9   end
```

In Algorithm 13.2.2 H_j^i denotes the j-row of the matrix H^i, K_j^i denotes the j-th element of the vector K^i and N_c^i is the number of constraints defining the i-th polyhedron $\mathcal{P}_i$. Algorithm 13.2.2 requires the storage of all polyhedra $\mathcal{P}_i$, i.e., $(n+1)N_C$ real numbers (n numbers for each row of the matrix H^i plus one number for the corresponding element in the matrix K^i), $N_C \triangleq \sum_{i=1}^{N^r} N_c^i$, and in the worst case (the state is contained in the last region of the list) it will give a solution after nN_C multiplications, $(n-1)N_C$ sums and N_C comparisons.

Remark 13.1. Note that Algorithm 13.2.2 can also deduce if the point x is not inside of the feasible set $\mathcal{X}_0 = \cup_{i=1}^{N^r}\mathcal{P}_i$. As opposed to that, in the following sections we implicitly assume that x belongs to $\mathcal{X}_0$. If this (reasonable) assumptions does not hold, it is always possible to include set of *boundaries* of feasible parameter space $\mathcal{X}_0$. Then, before using any of proposed algorithms, we should first check if the point x is inside the boundaries of $\mathcal{X}_0$.

By using the properties of the value function, in this section we show how Algorithm 13.2.2 can be replaced by more efficient algorithms that *avoid storing the polyhedral regions* $\mathcal{P}_i,\ i = 1,\dots,N^r$, therefore reducing significantly the storage demand, and that have a smaller computational complexity.

In the following we will distinguish between optimal control based on LP and optimal control based on QP.

13.2.1 Efficient Implementation, $1,\infty$-Norm Case

From Corollary 10.5, the value function $J^*(x)$ corresponding to the solution of the CFTOC problem (11.45) with $1,\infty$-norm is convex and PWA:

$$J^*(x) = T^{i'}x + V^i, \quad \forall x \in \mathcal{P}_i, \quad i = 1,\dots,N^r. \tag{13.2}$$

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 (see Section 4.1.5), the function $J^*(x)$ in equation (13.2) can be represented alternatively as

$$J^*(x) = \max\left\{T^{i'}x + V^i,\ i = 1,\dots,N^r\right\} \text{ for } x \in \mathcal{X} = \cup_{i=1}^{N^r}\mathcal{P}_i. \tag{13.3}$$

Thus, the polyhedral region P_j containing x can be simply identified by searching for the maximum number in the list $\{T^{i'}x + V^i\}_{i=1}^{N^r}$:

$$x \in \mathcal{P}_j \Leftrightarrow T^{j'}x + V^j = \max\left\{T^{i'}x + V^i,\ i = 1,\dots,N^r\right\}. \tag{13.4}$$

Therefore, instead of searching for the polyhedron j that contains the point x via Algorithm 13.2.2, 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 (see Figure 13.1):

Algorithm 13.2.3

```
Input: Current state $x$ and value function 13.2
Output: Index $j$ of the polyhedron $\mathcal{P}_j$ containing the current state $x(t)$
  in the control law (13.1)
1    Compute the list $\mathcal{L} = \{n_i \triangleq T^{i\prime}x + V^i,\ i = 1, \dots, N^r\}$
2    Find $j$ such that $n_j = \max_{n_i \in \mathcal{L}} n_i$
```

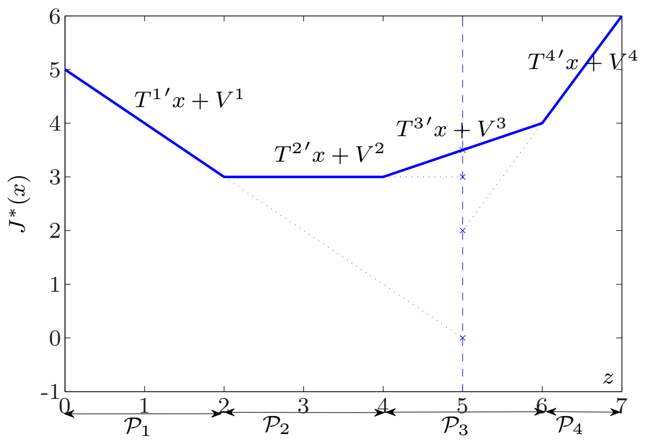


Fig. 13.1 Example for Algorithm 13.2.3 in one dimension: For a given point $x \in \mathcal{P}_3$ $(x = 5)$ we have $J^*(x) = \max(T^{1\prime}x + V^1, \dots, T^{4\prime}x + V^4)$.

Algorithm 13.2.3 requires the storage of $(n + 1)N^r$ real numbers and will give a solution after nN^r multiplications, $(n-1)N^r$ sums, and $N^r - 1$ comparisons. In Table 13.1 we compare the complexity of Algorithm 13.2.3 against Algorithm 13.2.2 in terms of storage demand and number of flops.

Table 13.1 Complexity comparison of Algorithm 13.2.2 and Algorithm 13.2.3

	Algorithm 13.2.2	Algorithm 13.2.3
Storage demand (real numbers)	$(n+1)N_C$	$(n+1)N^r$
Number of flops (worst case)	$2nN_C$	$2nN^r$

Remark 13.2. Algorithm 13.2.3 will outperform Algorithm 13.2.2 since typically $N_C \gg N^r$.

13.2.2 Efficient Implementation, 2-Norm Case

Consider the state feedback solution (10.38) of the CFTOC problem (10.32) with $p = 2$. Theorem 10.2 states that the value function $J^*(x)$ is convex and piecewise quadratic on polyhedra and the simple Algorithm 13.2.3 described in the previous subsection cannot be used here. Instead, a different approach is described below. It uses a surrogate of the value function to uniquely characterize the polyhedral partition of the optimal control law.

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 "appropriate" PWA function as *PWA descriptor function*. First we outline the properties of the PWA descriptor function and then we describe the search procedure itself.

Let $\{\mathcal{P}_i\}_{i=1}^{N^r}$ be the polyhedral partition obtained by solving the mp-QP (10.52) and denote by $C_i = \{j \ : \ \mathcal{P}_j$ is a neighbor of $\mathcal{P}_i,\ j = 1, \dots, N^r, j \neq i\}$ the list of all neighboring polyhedra of $\mathcal{P}_i$. The list C_i has N_c^i elements and we denote by $C_i(k)$ its k-th element.

Definition 13.1 (PWA descriptor function). A continuous real-valued PWA function

$$f(x) = f_i(x) \triangleq {A^i}' x + B^i, \ \text{if } x \in \mathcal{P}_i, \tag{13.5}$$

is called descriptor function if

$$A^i \neq A^j, \ \forall j \in C_i, \ i = 1, \dots, N_r. \tag{13.6}$$

Theorem 13.1. *Let $f(x)$ be a PWA descriptor function on the polyhedral partition $\{\mathcal{P}_i\}_{i=1}^{N_r}$.*

Let $O^i(x) \in \mathbb{R}^{N_c^i}$ be a vector associated with region $\mathcal{P}_i$, and let the j-th element of $O^i(x)$ be defined as

$$O_j^i(x) = \begin{cases} +1 & f_i(x) \geqslant f_{C_i(j)}(x) \\ -1 & f_i(x) < f_{C_i(j)}(x) \end{cases} \tag{13.7}$$

Then $O^i(x)$ has the following properties:

(i)$O^i(x) = S^i = const, \quad \forall x \in \mathcal{P}_i, \quad i = 1, \dots, N_r$.
(ii)$O^i(x) \neq S^i, \quad \forall x \notin \mathcal{P}_i, \quad i = 1, \dots, N_r$.

Proof: Let $\mathcal{F} = \mathcal{P}_i \cap \mathcal{P}_{C_i(j)}$ be the common facet of $\mathcal{P}_i$ and $\mathcal{P}_{C_i(j)}$. Define the linear function

$$g_j^i(x) = f_i(x) - f_{C_i(j)}(x). \tag{13.8}$$

From the continuity of $f(x)$ it follows that $g_j^i(x) = 0, \forall x \in \mathcal{F}$. As $\mathcal{P}_i$ and $\mathcal{P}_{C_i(j)}$ are disjoint convex polyhedra and $A^i \neq A^{C_i(j)}$ it follows that $g_j^i(\xi_i) > 0$

(or $g_j^i(\xi_i) < 0$, but not both) for any interior point ξ_i of $\mathcal{P}_i$. Similarly for any interior point $\xi_{C_i(j)}$ of $\mathcal{P}_{C_i(j)}$ we have $g_j^i(\xi_{C_i(j)}) < 0$ (or $g_i^j(\xi_i) > 0$, but not both). Consequently, $g_j^i(x) = 0$ is the separating hyperplane between $\mathcal{P}_i$ and $\mathcal{P}_{C_i(j)}$.

(i) Because $g_j^i(x) = 0$ is a separating hyperplane, the function $g_j^i(x)$ does not change its sign for all $x \in \mathcal{P}_i$, i.e., $O_j^i(x) = s_j^i, \forall x \in \mathcal{P}_i$ with $s_j^i = +1$ or $s_j^i = -1$. The same reasoning can be applied to all neighbors of $\mathcal{P}_i$ to get the vector $S^i = \{s_j^i\} \in \mathbb{R}^{N_c^i}$.

(ii) $\forall x \notin \mathcal{P}_i,\ \exists j \in C_i$ such that $H_j^i x > K_j^i$. Since $g_j^i(x) = 0$ is a separating hyperplane then $O_j^i(x) = -s_j^i$. $\square$

Equivalently, Theorem 13.1 states that

$$x \in \mathcal{P}_i \Leftrightarrow O^i(x) = S^i, \tag{13.9}$$

which means that the function $O^i(x)$ and the vector S^i uniquely characterize $\mathcal{P}_i$. Therefore, to check on-line if the polyhedral region i contains the state x it is sufficient to compute the binary vector $O^i(x)$ and compare it with S^i. Vectors S^i are calculated off-line for all $i = 1, \ldots, N^r$, by comparing the values of $f_i(x)$ and $f_{C_i(j)}(x)$ for $j = 1, \ldots, N_c^i$, for a point x belonging to $\mathcal{P}_i$, for instance, the Chebychev center of $\mathcal{P}_i$.

In Figure 13.2 a one dimensional example illustrates the procedure with $N^r = 4$ regions. The list of neighboring regions C_i and the vector 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 $O^3(x) = [-1\ \ -1]' \neq S^3$, $O^1(x) = 1 \neq S^1$, $O^4(x) = 1 \neq S^4$. 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., one should check the neighboring region $\mathcal{P}_j$ for which $O_j^i(x) \neq s_j^i$.

The overall procedure is composed of two parts:

1. *(off-line)* Construction of the PWA function $f(x)$ in (13.5) satisfying (13.6) and computation of the list of neighbors C_i and the vector S^i,
2. *(on-line)* Execution of the following algorithm

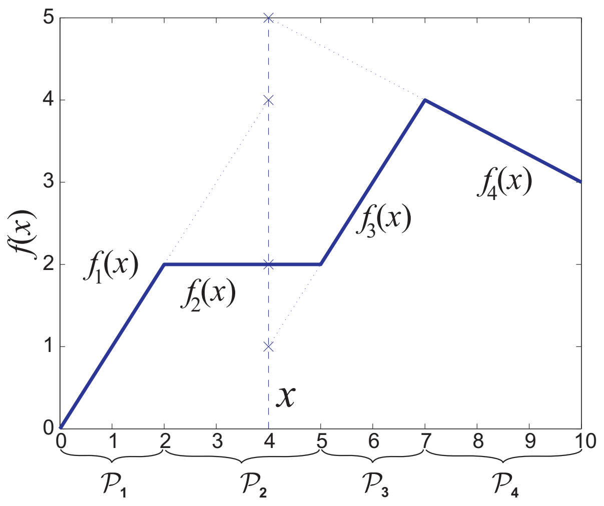


Fig. 13.2 Example for Algorithm 13.2.4 in one dimension: For a given point $x \in \mathcal{P}_2$ ($x = 4$) we have $O_2(x) = [-1\ 1]' = S_2$, while $O_1(x) = 1 \neq S_1 = -1$, $O_3(x) = [-1\ -1]' \neq S_3 = [1\ -1]'$, $O_4(x) = 1 \neq S_4 = -1$.

Algorithm 13.2.4

```
Input: Current state $x$, the list of neighboring regions $C_i$ and the vectors
  $S^i$
Output: Index $i$ of the polyhedron $\mathcal{P}_j$ containing the current state $x(t)$
  in the control law (13.1)
1    $i = 1$, notfound=1;
2    while notfound
3        compute $O^i(x)$
4        if $O^i(x) = S^i$ then notfound=0
5        else $i = C_i(q)$, where $O^i_q(x) \neq s^i_q$.
6    end
```

Algorithm 13.2.4 does not require the storage of the polyhedra $\mathcal{P}_i$, but only the storage of one linear function $f_i(x)$ per polyhedron, i.e., $N^r(n+1)$ real numbers and the list of neighbors C_i which requires N_C integers. In the worst case, Algorithm 13.2.4 terminates after $N^r n$ multiplications, $N^r(n-1)$ sums and N_C comparisons.

In Table 13.2 we compare the complexity of Algorithm 13.2.4 against the standard Algorithm 13.2.2 in terms of storage demand and number of flops.

Remark 13.3. Note that the computation of $O^i(x)$ in Algorithm 13.2.4 requires the evaluation of N_c^i linear functions, but the overall computation never exceeds N^r linear function evaluations. Consequently, Algorithm 13.2.4 will outperform Algorithm 13.2.2, since typically $N_C \gg N^r$.

Table 13.2 Complexity comparison of Algorithm 13.2.2 and Algorithm 13.2.4

	Algorithm 13.2.2	Algorithm 13.2.4
Storage demand (real numbers)	$(n+1)N_C$	$(n+1)N^r$
Number of flops (worst case)	$2nN_C$	$(2n-1)N^r + N_C$

Now that we have shown how to locate polyhedron in which 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 descriptor function can be generate from a vector-valued function $m : \mathbb{R}^n \to \mathbb{R}^s$. This general result will be used in the next subsections.

Definition 13.2 (Vector valued PWA descriptor function). A continuous vector-valued PWA function

$$m(x) = \bar{A}^i x + \bar{B}^i, \text{ if } x \in \mathcal{P}_i, \tag{13.10}$$

is called vector-valued PWA descriptor function if

$$\bar{A}^i \neq \bar{A}^j, \ \forall j \in C_i, \ i = 1, \ldots, N_r. \tag{13.11}$$

where $\bar{A}^i \in \mathbb{R}^{s\times n}$, $\bar{B}^i \in \mathbb{R}^s$.

Theorem 13.2. *Given a vector-valued PWA descriptor function $m(x)$ defined over a polyhedral partition $\{\mathcal{P}_i\}_{i=1}^{N_r}$ it is possible to construct PWA descriptor function $f(x)$ over the same polyhedral partition.*

Proof: Let $\mathcal{N}_{i,j}$ be the null-space of $(\bar{A}^i - \bar{A}^j)'$. Since by the definition $\bar{A}^i - \bar{A}^j \neq \mathbf{0}$ it follows that $\mathcal{N}_{i,j}$ is not full dimensional, i.e., $\mathcal{N}_{i,j} \subseteq \mathbb{R}^{s-1}$. Consequently, it is always possible to find a vector $w \in \mathbb{R}^s$ such that $w(\bar{A}^i - \bar{A}^j) \neq \mathbf{0}$ holds for all $i = 1, \ldots, N_r$ 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 13.2, 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 PWA descriptor function. But, from a numerical point of view, we would like to obtain w that is as far away as possible from the null-spaces $\mathcal{N}_{i,j}$. We show one algorithm for finding such a vector w.

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

Algorithm 13.2.5

Input: Vectors $a_i \in \mathbb{R}^s$, $i = 1, \dots, N$.
Output: The vector $w \in \mathbb{R}^s$ satisfying the set of equations $w' a_i \neq 0$, $i = 1, \dots, N$

```
1   $w \leftarrow [1, \dots, 1]'$, $R \leftarrow 1$
2   while $k \leq N_C/2$
3       $d \leftarrow w' a_k$
4       if $0 \leq d \leq R$ then $w \leftarrow w + \frac{1}{2}(R - d) a_k$, $R \leftarrow \frac{1}{2}(R + d)$
5       if $-R \leq d < 0$ then $w \leftarrow w - \frac{1}{2}(R + d) a_k$, $R \leftarrow \frac{1}{2}(R - d)$
6   end
```

Algorithm 13.2.5 is based on a construction of a sequence of balls $\mathcal{B} = \{x \ : \ x = w + r, \ \|r\|_2 \leq R\}$. As depicted in Figure 13.3, Algorithm 13.2.5 starts with the initial ball of radius $R = 1$, centered at $w = [1, \dots, 1]'$. Iteratively one hyperplane $a_k' x = 0$ at the time is introduced and the largest ball $\mathcal{B}' \subseteq \mathcal{B}$ that does not intersect this hyperplane is designed. The center w of the final ball is the vector w we wanted to construct, while R gives an 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 13.2.5 for the construction of the PWA descriptor function.

13.2.2.1 Generating a PWA descriptor function from the value function

Let $J^*(x)$ be the convex and piecewise quadratic (CPWQ) value function obtained as a solution of the CFTOC (10.32) problem for $p = 2$:

$$J^*(x) = q_i(x) \triangleq x' Q^i x + T^{i\prime} x + V^i, \quad \text{if } x \in \mathcal{P}_i, \ i = 1, \dots, N^r. \tag{13.12}$$

[1] Index k goes to $N_C/2$ since the term $(\bar{A}^j - \bar{A}^i)$ is the same as $(\bar{A}^i - \bar{A}^j)$ and thus there is no need to consider it twice.

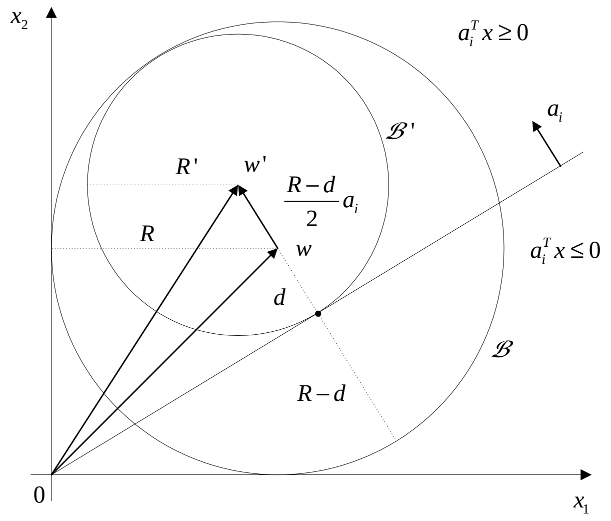


Fig. 13.3 Illustration for Algorithm 13.2.5 in two dimensions.

In Section 6.3.4 we have proven that for non-degenerate problems the value function $J^*(x)$ is a $C^{(1)}$ function. We can obtain a vector-valued PWA descriptor function by differentiating $J^*(x)$.

Theorem 13.3. *Consider the value function $J^*(x)$ in (13.12) and assume that the CFTOC (10.32) problem leads to a non-degenerate mp-QP (10.54). Then the gradient $m(x) \triangleq \nabla J^*(x)$, is a vector-valued PWA descriptor function.*

Proof: From Theorem 6.9 we see that $m(x)$ is continuous vector-valued PWA function, while from equation (6.63) we get

$$m(x) \triangleq \nabla J^*(x) = 2Q_i x + T_i \tag{13.13}$$

Since from Theorem 6.7 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 13.3 and Theorem 13.2 it follows that by using Algorithm 13.2.5 we can construct a PWA descriptor function from the gradient of the value function $J^*(x)$.

13.2.2.2 Generating a PWA descriptor function from the optimal inputs

Another way to construct descriptor function $f(x)$ emerges naturally if we look at the properties of the optimizer $U_0^*(x)$ corresponding to the state feedback solution of the CFTOC problem (10.32). From Theorem 6.6 it follows that the optimizer $U_0^*(x)$ is continuous in x and piecewise affine on polyhedra:

$$U_0^*(x) = l_i(x) \triangleq F^i x + G^i, \text{ if } x \in \mathcal{P}_i, \ i = 1, \ldots, N^r, \tag{13.14}$$

where $F^i \in \mathbb{R}^{s \times n}$ and $G^i \in \mathbb{R}^s$. We will assume that $\mathcal{P}_i$ are critical regions (as defined in Section 6.1.2) Before going further we need the following lemma.

Lemma 13.1. *Consider the state feedback solution (13.14) of the CFTOC problem (10.32) and assume that the CFTOC (10.32) leads to a non-degenerate mp-QP (10.54). Let $\mathcal{P}_i$, $\mathcal{P}_j$ be two neighboring polyhedra, then $F^i \neq F^j$.*

Proof: The proof is a simple consequence of Theorem 6.7. As in Theorem 6.7, without loss of generality we can assume that the set of active constraints $\mathcal{A}_i$ associated with the critical region $\mathcal{P}_i$ is empty, i.e., $\mathcal{A}_i = \emptyset$. Suppose that the optimizer is the same for both polyhedra, i.e., $[F^i \ G^i] = [F^j \ G^j]$. Then, the cost functions $q_i(x)$ and $q_j(x)$ are also equal. From the proof of Theorem 6.7 this implies that $\mathcal{P}_i = \mathcal{P}_j$, which is a contradiction. Thus we have $[F^i \ G^i] \neq [F^j \ G^j]$. Note that $F^i = F^j$ cannot happen since, from the continuity of $U_0^*(x)$, this would imply $G^i = G^j$. Consequently we have $F^i \neq F^j$. □

From Lemma 13.1 and Theorem 13.2 it follows that an appropriate PWA descriptor function $f(x)$ can be calculated from the gradient of the optimizer $U^*(x)$ by using Algorithm 13.2.5.

Remark 13.4. Note that even if we are implementing 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 13.5. In some cases the use of the optimal control profile $U^*(x)$ for the construction of descriptor function $f(x)$ can be extremely simple. If there is a row r, $r \leq m$ (m is the dimension of u) for which $(F^i)^r \neq (F^j)^r$, $\forall i = 1 \ldots, N^r$, $\forall j \in C_i$, it is enough to set $A^{i'} = (F^i)^r$ and $B^i = (G^i)^r$, where $(F^i)^r$ and $(G^i)^r$ denote the r-th row of the matrices F^i and G^i, respectively. In this way we *avoid* the storage of the descriptor function altogether, since it is equal to one component of the control law, which is stored anyway.

13.3 Example

As an example, we compare the performance of Algorithm 13.2.2, 13.2.3 and 13.2.4 on CFTOC problem for the discrete-time system

$$\begin{cases} x(t+1) = \begin{bmatrix} 4 & -1.5 & 0.5 & -0.25 \\ 4 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 0.5 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 0.5 \\ 0 \\ 0 \\ 0 \end{bmatrix} u(t) \\ y(t) = \begin{bmatrix} 0.083 & 0.22 & 0.11 & 0.02 \end{bmatrix} x(t) \end{cases} \tag{13.15}$$

resulting from the linear system

$$y = \frac{1}{s^4} u \tag{13.16}$$

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

$$-1 \leq u(t) \leq 1 \tag{13.17}$$

and the output constraint

$$-10 \leq y(t) \leq 10. \tag{13.18}$$

13.3.1 CFTOC based on LP

To regulate (13.15), we design an receding horizon controller based on the optimization problem (11.45) where $p = \infty$, $N = 2$, $Q = \text{diag}\{5, 10, 10, 10\}$, $R = 0.8$, $P = 0$. The PPWA solution of the mp-LP problem was computed consists of 136 regions. In Table 13.3 we report the comparison between the complexity of Algorithm 13.2.2 and Algorithm 13.2.3 for this example.

The average on-line evaluation of the PPWA solution for a set of 1000 random points in the state space is 2259 flops (Algorithm 13.2.2), and 1088 flops (Algorithm 13.2.3). We point out that the solution using Matlab's LP solver (function linprog.m with interior point algorithm and LargeScale set to 'off') takes 25459 flops on average.

Table 13.3 Complexity comparison of Algorithm 13.2.2 and Algorithm 13.2.3 for the example in Section 13.3.1

	Algorithm 13.2.2	Algorithm 13.2.3
Storage demand (real numbers)	5690	680
Number of flops (worst case)	9104	1088
Number of flops (average for 1000 random points)	2259	1088

13.3.2 CFTOC based on QP

To regulate (13.15), we design a receding horizon controller based on the optimization problem (11.45) where $p = 2$, $N = 7$, $Q = I$, $R = 0.01$, $P = 0$. The PPWA solution of the mp-QP problem consists of 213 regions. We obtained a descriptor function from the value function and for this example the choice of $w = [1\ 0\ 0\ 0]'$ is satisfactory. In Table 13.4 we report the comparison between the complexity of Algorithm 13.2.2 and Algorithm 13.2.4 for this example.

The average computation of the PPWA solution for a set of 1000 random points in the state space is 2114 flops (Algorithm 13.2.2), and 175 flops (Algorithm 13.2.4). 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 13.4 Complexity comparison of Algorithm 13.2.2 and Algorithm 13.2.4 for the example in Section 13.3.2

	Algorithm 13.2.2	Algorithm 13.2.4
Storage demand (real numbers)	9740	1065
Number of flops (worst case)	15584	3439
Number of flops (average for 1000 random points)	2114	175

13.4 Literature Review

The problem considered in this Chapter has been approached by several other researchers. For instance in in [249] the authors propose to organize the controller gains of the PWA control law on a balanced search tree. By doing so, the search for the region containing the current state has a logarithmic average computation complexity although less efficient in terms of memory requirements. At the expense of the optimality of the solution a similar computational complexity can be achieved with an approximative point location algorithm described in [148].

The comparison of the proposed algorithms with other very efficient solvers appeared in the literature [19, 50, 229, 96, 193, 196, 260] requires the simultaneous analysis of several issues such as speed of computation, storage demand and real time code verifiability. This is an involved study and as such is outside of the scope of this book.

Part V
Constrained Optimal Control of Hybrid Systems

Chapter 14
Models of Hybrid Systems

Hybrid systems describe the dynamical interaction between continuous and discrete signals in one common framework (see Figure 14.1). In this chapter we focus our attention on mathematical models of hybrid systems that are particularly suitable for solving finite time constrained optimal control problems.

14.1 Hybrid models

The mathematical model of a dynamical system is traditionally associated with differential or difference equations, typically derived from physical laws governing the dynamics of the system under consideration. Consequently, most of the control theory and tools are based on models describing the evolution of real-valued signals according to smooth linear or nonlinear state transition functions, typically differential or difference equations. In many applications, however, the system to be controlled also contains discrete-valued signals satisfying Boolean relations, if-then-else conditions, on/off conditions, etc., that also involve the real-valued signals. An example would be an on/off alarm signal triggered by an analog variable passing over a given threshold. *Hybrid systems* describe in a common framework the dynamics of real-valued variables, the dynamics of discrete variables, and their interaction.

In this chapter we will focus on discrete-time hybrid systems, that we will call *discrete hybrid automata* (DHA), whose continuous dynamics is described by linear difference equations and whose discrete dynamics is described by finite state machines, both synchronized by the same clock [251]. A particular case of DHA is the popular class of *piecewise affine* (PWA) systems [241]. Essentially, PWA systems are switched affine systems whose mode depends on the current location of the state vector, as depicted in Figure 14.2. PWA and DHA systems can be translated into a form, denoted as *mixed logical dynamical* (MLD) form, that is more suitable for solving optimization problems. In

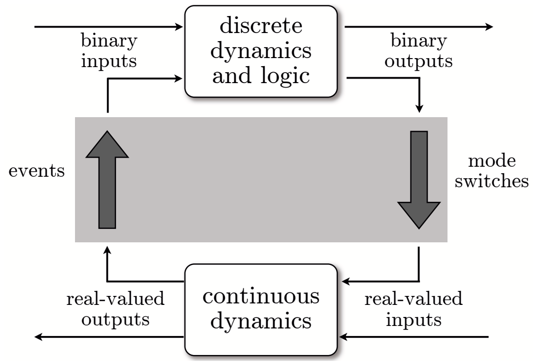


Fig. 14.1 Hybrid systems. Logic-based discrete dynamics and continuous dynamics interact through events and mode switches

particular, complex finite time hybrid dynamical optimization problems can be recast into mixed-integer linear or quadratic programs as will be shown in Chapter 15.

In the book we will often refer to the tool HYSDEL (HYbrid Systems DEscription Language), a high level language for modeling and simulating DHA. Therefore, DHA will represent for us the starting point for modeling hybrid systems. We will show that DHA, PWA, and MLD systems are equivalent model classes, and in particular that DHA systems can be converted to an equivalent PWA or MLD form for solving optimal control problems.

After introducing PWA systems, we will go through the steps needed for modeling a system as a DHA. We will first detail the process of translating propositional logic involving Boolean variables and linear threshold events over continuous variables into mixed-integer linear inequalities, generalizing several results available in the literature, in order to get an equivalent MLD form of a DHA system. Finally, we will briefly present the tool HYSDEL, that allows describing the DHA in a textual form and obtain equivalent MLD and PWA representations in MATLAB$^{\mathrm{TM}}$.

14.2 Piecewise Affine Systems

Piecewise Affine (PWA) systems [241, 135] are defined by partitioning the space of states and inputs into polyhedral regions (cf. Figure 14.2) and associating with each region different affine state-update and output equations:

$$x(t+1) = A^{i(t)}x(t) + B^{i(t)}u(t) + f^{i(t)} \tag{14.1a}$$

$$y(t) = C^{i(t)}x(t) + D^{i(t)}u(t) + g^{i(t)} \tag{14.1b}$$

$$i(t) \text{ such that } H^{i(t)}x(t) + J^{i(t)}u(t) \leq K^{i(t)} \tag{14.1c}$$

where $x(t) \in \mathbb{R}^n$ is the state vector at time $t \in \mathbb{T}$ and $\mathbb{T} \triangleq \{0, 1, \ldots\}$ is the set of nonnegative integers, $u(t) \in \mathbb{R}^m$ is the input vector, $y(t) \in \mathbb{R}^p$ is the output vector, $i(t) \in \mathcal{I} \triangleq \{1, \ldots, s\}$ is the current *mode* of the system, the matrices $A^{i(t)}$, $B^{i(t)}$, $f^{i(t)}$, $C^{i(t)}$, $D^{i(t)}$, $g^{i(t)}$, $H^{i(t)}$, $J^{i(t)}$, $K^{i(t)}$ are constant and have suitable dimensions, and the inequalities in (14.1c) should be interpreted component-wise. Each linear inequality in (14.1c) defines a half-space in $\mathbb{R}^n$ and a corresponding hyperplane, that will be also referred to as *guardline*. Each vector inequality (14.1c) defines a polyhedron $\mathcal{C}^i = \{[\begin{smallmatrix} x \\ u \end{smallmatrix}] \in \mathbb{R}^{n+m} : H^i x + J^i u \leq K^i\}$ in the state+input space $\mathbb{R}^{n+m}$ that will be referred to as *cell*, and the union of such polyhedral cells as *partition*. We assume that $\mathcal{C}^i$ are full dimensional sets of $\mathbb{R}^{n+m}$, for all $i = 1, \ldots, s$.

A PWA system is called *well-posed* if it satisfies the following property [35]:

Definition 14.1. Let P be a PWA system of the form (14.1) and let $\mathcal{C} = \cup_{i=1}^{s} \mathcal{C}^i \subseteq \mathbb{R}^{n+m}$ be the polyhedral partition associated with it. System P is called *well-posed* if for all pairs $(x(t), u(t)) \in \mathcal{C}$ there exists only one index $i(t)$ satisfying (14.1).

Definition 14.1 implies that $x(t+1)$, $y(t)$ are single-valued functions of $x(t)$ and $u(t)$, and therefore that state and output trajectories are uniquely determined by the initial state and input trajectory. A relaxation of definition 14.1 is to let polyhedral cells $\mathcal{C}^i$ sharing one or more hyperplanes. In this case the index $i(t)$ is not uniquely defined, and therefore the PWA system is not well-posed. However, if the mappings $(x(t), u(t)) \to x(t+1)$ and $(x(t), u(t)) \to y(t)$ are continuous across the guardlines that are facets of two or more cells (and, therefore, they are continuous on their domain of definition), such mappings are still single valued.

14.2.1 Modeling Discontinuities

Discontinuous dynamical behaviors can be modeled by disconnecting the domain. For instance, the state-update equation

$$x(t+1) = \begin{cases} \frac{1}{2}x(t) + 1 & \text{if } x(t) \leq 0 \\ 0 & \text{if } x(t) > 0 \end{cases} \tag{14.2a}$$

is discontinuous across $x = 0$. It can be modeled as

$$x(t+1) = \begin{cases} \frac{1}{2}x(t) + 1 & \text{if } x(t) \leq 0 \\ 0 & \text{if } x(t) \geq \epsilon \end{cases} \tag{14.2b}$$

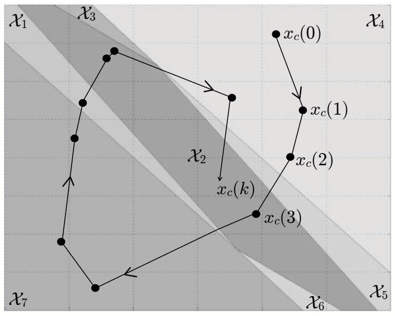


Fig. 14.2 Piecewise affine (PWA) systems. Mode switches are only triggered by linear threshold events

where $\epsilon > 0$ is an arbitrarily small number, for instance the machine precision. Clearly, system (14.2) is not defined for $0 < x(t) < \epsilon$, i.e., for the values of the state that cannot be represented in the machine. However, the trajectories produced by (14.2a) and (14.2b) are identical as long as $x(t) > \epsilon$ or $x(t) \le 0$, $\forall t \in \mathbb{N}$.

As remarked above, multiple definitions of the state-update and output functions over common boundaries of sets $\mathcal{C}^i$ is a technical issue that arises only when the PWA mapping is discontinuous. Rather than disconnecting the domain, another way of dealing with discontinuous PWA mappings is to allow strict inequalities in the definition of the polyhedral cells in (14.1), or by dealing with open polyhedra and boundaries separately as in [241]. We prefer to assume that in the definition of the PWA dynamics (14.1) the polyhedral cells $\mathcal{C}^{i(t)}$ are closed sets: As will be clear in the next chapter, the closed-polyhedra description is mainly motivated by the fact that numerical solvers cannot handle open sets.

Example 14.1. The following PWA system

$$\left\{\begin{array}{rl} x(t+1) &= 0.8\begin{bmatrix}\cos\alpha(t) & -\sin\alpha(t)\\ \sin\alpha(t) & \cos\alpha(t)\end{bmatrix}x(t)+\begin{bmatrix}0\\1\end{bmatrix}u(t)\\ y(t) &= \begin{bmatrix}0 & 1\end{bmatrix}x(t)\\ \alpha(t) &= \begin{cases}\frac{\pi}{3} & \text{if } \begin{bmatrix}1 & 0\end{bmatrix}x(t)\ge 0\\ -\frac{\pi}{3} & \text{if } \begin{bmatrix}1 & 0\end{bmatrix}x(t)< 0\end{cases}\end{array}\right. \tag{14.3}$$

is discontinuous at $x = \begin{bmatrix}0\\x_2\end{bmatrix}$, $\forall x_2 \neq 0$. It can be described in form (14.1) as

$$\left\{\begin{array}{l} x(t+1) = \begin{cases}0.4\begin{bmatrix}1 & -\sqrt{3}\\ \sqrt{3} & 1\end{bmatrix}x(t)+\begin{bmatrix}0\\1\end{bmatrix}u(t) \text{ if } \begin{bmatrix}1 & 0\end{bmatrix}x(t)\ge 0\\[2ex] 0.4\begin{bmatrix}1 & \sqrt{3}\\ -\sqrt{3} & 1\end{bmatrix}x(t)+\begin{bmatrix}0\\1\end{bmatrix}u(t) \text{ if } \begin{bmatrix}1 & 0\end{bmatrix}x(t)\le -\epsilon\end{cases}\\ y(t) = \begin{bmatrix}0 & 1\end{bmatrix}x(t)\end{array}\right. \tag{14.4}$$

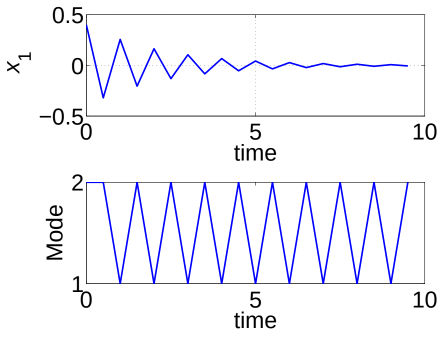


Fig. 14.3 Free response of state x_1 in Example 14.1 for $x(0) = [1\ 0]$

for all $x_1 \in (-\infty, -\epsilon] \cup [0, +\infty)$, $x_2 \in \mathbb{R}$, $u \in \mathbb{R}$, and $\epsilon > 0$.

Figure 14.3 shows the free response of the systems (open-loop simulation of the system for a constant input $u_1 = 0$) starting from the initial condition $x(0) = [1\ 0]$ with sampling time equal to 0.5s and $\epsilon = 10^{-6}$.

In case the partition $\mathcal{C}$ does not cover the whole space $\mathbb{R}^{n+m}$, well-posedness does not imply that trajectories are *persistent*, i.e., that for all $t \in \mathbb{N}$ a successor state $x(t+1)$ and an output $y(t)$ are defined. A typical case of $\mathcal{C} \neq \mathbb{R}^{n+m}$ is when we are dealing with bounded inputs and bounded states $u_{\min} \leq u(t) \leq u_{\max}$, $x_{\min} \leq x(t) \leq x_{\max}$. By embedding such ranges in the inequalities (14.1c), the system becomes undefined outside the bounds, as no index i exists that satisfies any of the set of inequalities (14.1c).

As will be clearer in the next chapter, when model (14.1) is used in an optimal control formulation, any input sequence and initial state that are feasible for the related optimization problem automatically define unique trajectories over the whole optimal control horizon.

PWA systems can model a large number of physical processes, as they can model static nonlinearities through a piecewise affine approximation, or approximate nonlinear dynamics via multiple linearizations at different operating points. Moreover, tools exist nowadays for obtaining piecewise affine approximations automatically (see Section 14.8).

When the mode $i(t)$ is an exogenous variable, condition (14.1c) disappears and we refer to (14.1) as a *switched affine system* (SAS), see Section 14.3.1.

14.2.2 Binary States, Inputs, and Outputs

When dealing with hybrid systems, quite often one encounters some signals that can only assume a binary value, namely either 0 or 1. In the most general form, let us assume that the state vector $x = \left[\begin{smallmatrix} x_c \\ x_\ell \end{smallmatrix}\right]$ where $x_c \in \mathbb{R}^{n_c}$ are the continuous states, $x_\ell \in \mathbb{R}^{n_\ell}$ are the binary states, and set $n \triangleq n_c + n_\ell$.

Similarly, let $y \in \mathbb{R}^{p_c} \times \{0,1\}^{p_\ell}$, $p \triangleq p_c + p_\ell$, $u \in \mathbb{R}^{m_c} \times \{0,1\}^{m_\ell}$, $m \triangleq m_c + m_\ell$. By defining a polyhedral partition $\{\mathcal{C}^i\}_{i=0}^{s-1}$ of the sets of state and input space $\mathbb{R}^{n+m}$, for any $x_\ell \in \{0,1\}$ and $u_\ell \in \{0,1\}$ a sufficient condition for the PWA system to be well posed is that the rows and columns of matrices A^i, B^i, C^i, D^i corresponding to binary states and binary outputs are zero and that the corresponding rows of matrices f^i, g^i are either 0 or 1, i.e., that the binary state update and output equations are binary piecewise constant functions.

In the following sections we will treat 0-1 binary variables both as numbers (over which arithmetic operations are defined) and as Boolean variables (over which Boolean functions are defined, see Section 14.3.3). The variable type will be clear from the context.

As an example, it is easy to verify that the hybrid dynamical system

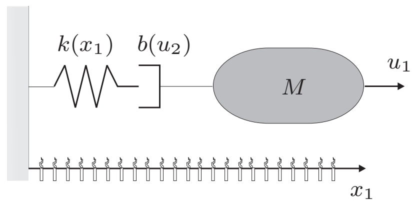
$$x_c(t+1) = 2x_c(t) + u_c(t) - 3u_\ell(t) \tag{14.5a}$$

$$x_\ell(t+1) = x_\ell(t) \wedge u_\ell(t) \tag{14.5b}$$

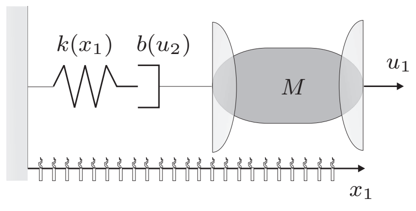
where "$\wedge$" represents the logic operator "and", can be represented in the PWA form

$$\begin{bmatrix} x_c \\ x_\ell \end{bmatrix}(t+1) = \begin{cases} \begin{bmatrix} 2x_c(t) + u_c(t) \\ 0 \end{bmatrix} & \text{if } x_\ell \le \frac{1}{2}, u_\ell \le \frac{1}{2} \\ \begin{bmatrix} 2x_c(t) + u_c(t) - 3 \\ 0 \end{bmatrix} & \text{if } x_\ell \le \frac{1}{2}, u_\ell \ge \frac{1}{2} + \epsilon \\ \begin{bmatrix} 2x_c(t) + u_c(t) \\ 0 \end{bmatrix} & \text{if } x_\ell \ge \frac{1}{2} + \epsilon, u_\ell \le \frac{1}{2} \\ \begin{bmatrix} 2x_c(t) + u_c(t) - 3 \\ 1 \end{bmatrix} & \text{if } x_\ell \ge \frac{1}{2} + \epsilon, u_\ell \ge \frac{1}{2} + \epsilon. \end{cases} \tag{14.5c}$$

by associating $x_\ell = 0$ with $x_\ell \le \frac{1}{2}$ and $x_\ell = 1$ with $x_\ell \ge \frac{1}{2} + \epsilon$ for any $0 < \epsilon \le \frac{1}{2}$. Note that, by assuming $x_\ell(0) \in \{0,1\}$ and $u_\ell(t) \in \{0,1\}$ for all $t \in \mathbb{T}$, $x_\ell(t)$ will be in $\{0,1\}$ for all $t \in \mathbb{T}$.



(a) System with low viscous friction ($u_2 = 1$)



(b) System with high viscous friction ($u_2 = 0$)

Fig. 14.4 Spring mass system of Example 14.2

Example 14.2. Consider the spring-mass system depicted in Figure 14.4(a), where the spring has the nonlinear characteristics described in Figure 14.5. The viscous friction coefficient can be instantaneously switched from one value b_1 to another different value b_2 by instantaneously changing the geometrical shape of the mass through a binary input u_2 (see Figure 14.4(b)).

The system dynamics can be described in continuous-time as:

$$M\dot{x}_2 = u_1 - k(x_1) - b(u_2)x_2$$

where x_1 and $x_2 = \dot{x}_1$ denote the position and the speed of the mass, respectively, u_1 a continuous force input, and u_2 binary input switching the friction coefficient. The spring coefficient is

$$k(x_1) = \begin{cases} k_1x_1 + d_1 \text{ if } x_1 \leq x_m \\ k_2x_1 + d_2 \text{ if } x_1 > x_m, \end{cases}$$

and the viscous friction coefficient is

$$b(u_2) = \begin{cases} b_1 \text{ if } u_2 = 1 \\ b_2 \text{ if } u_2 = 0. \end{cases}$$

Assume the system description is valid for $-5 \leq x_1, x_2 \leq 5$, and for $-10 \leq u_2 \leq 10$ (all units will be omitted here for simplicity).

The system has four modes, depending on the binary input and the region of linearity. Assuming that the system parameters are $M = 1$, $b_1 = 1$, $b_2 = 50$, $k_1 = 1$, $k_2 = 3$, $d_1 = 1$, $d_2 = 7.5$, $x_m = 1$, after discretizing the dynamics in each mode with a sampling time of 0.5 time units we obtain the following discrete-time PWA system

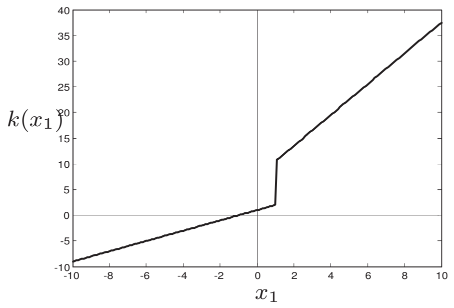


Fig. 14.5 Nonlinear spring force $k(x_1)$

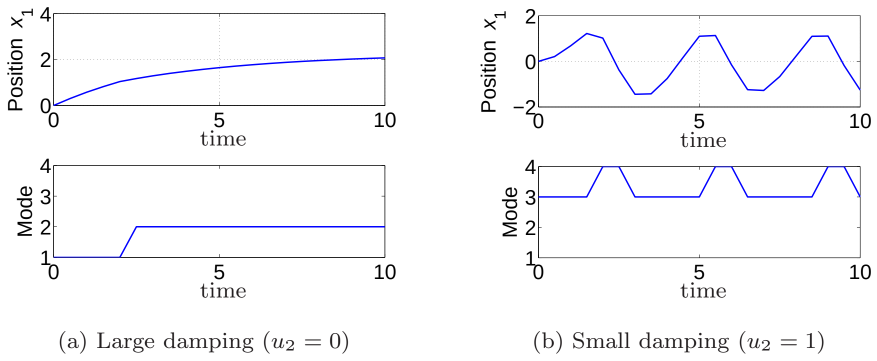


Fig. 14.6 Open-loop simulation of system (14.6) for $u_1 = 3$ and zero initial conditions

$$x(t+1) = \begin{cases} \begin{bmatrix} 0.8956 & 0.0198 \\ -0.0198 & -0.0004 \end{bmatrix} x(t) + \begin{bmatrix} 0.1044 \\ 0.0198 \end{bmatrix} u_1(t) + \begin{bmatrix} -0.0096 \\ -0.0198 \end{bmatrix} & \begin{array}{l}\text{Mode 1} \\ \text{if } x_1(t) \leq 1, u_2(t) \leq 0.5\end{array} \\ \begin{bmatrix} 0.8956 & 0.0195 \\ -0.0584 & -0.0012 \end{bmatrix} x(t) + \begin{bmatrix} 0.1044 \\ 0.0195 \end{bmatrix} u_1(t) + \begin{bmatrix} -0.0711 \\ -0.1459 \end{bmatrix} & \begin{array}{l}\text{Mode 2} \\ \text{if } x_1(t) \geq 1+\epsilon, u_2(t) \leq 0.5\end{array} \\ \begin{bmatrix} 0.8956 & 0.3773 \\ -0.3773 & 0.5182 \end{bmatrix} x(t) + \begin{bmatrix} 0.1044 \\ 0.3773 \end{bmatrix} u_1(t) + \begin{bmatrix} -0.1044 \\ -0.3773 \end{bmatrix} & \begin{array}{l}\text{Mode 3} \\ \text{if } x_1(t) \leq 1, u_2(t) \geq 0.5\end{array} \\ \begin{bmatrix} 0.8956 & 0.3463 \\ -1.0389 & 0.3529 \end{bmatrix} x(t) + \begin{bmatrix} 0.1044 \\ 0.3463 \end{bmatrix} u_1(t) + \begin{bmatrix} -0.7519 \\ -2.5972 \end{bmatrix} & \begin{array}{l}\text{Mode 4} \\ \text{if } x(t) \geq 1+\epsilon, u_2(t) \geq 0.5\end{array} \end{cases} \tag{14.6}$$

for $x_1(t) \in [-5, 1] \cup [1 + \epsilon, 5]$, $x_2(t) \in [-5, 5]$, $u(t) \in [-10, 10]$, and for any arbitrary small $\epsilon > 0$.

Figure 14.6 shows the open-loop simulation of the system for a constant continuous input $u_1 = 3$, starting from zero initial conditions, under different viscous coefficients, for $\epsilon = 10^{-6}$.

Example 14.3. Consider the following SISO system:

$$x_1(t+1) = ax_1(t) + bu(t). \tag{14.7}$$

A logic state $x_2 \in [0,1]$ stores the information whether the state of system (14.7) has ever gone below a certain lower bound x_{lb} or not:

$$x_2(t+1) = x_2(t) \bigvee [x_1(t) \leq x_{lb}], \tag{14.8}$$

Assume that the input coefficient is a function of the logic state:

$$b = \begin{cases} b_1 \text{ if } x_2 = 0 \\ b_2 \text{ if } x_2 = 1. \end{cases} \tag{14.9}$$

The system can be described by the PWA model:

$$x(t+1) = \begin{cases} \begin{bmatrix} a & 0 \\ 0 & 0 \end{bmatrix} x(t) + \begin{bmatrix} b_2 \\ 0 \end{bmatrix} u(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} & \text{if } \begin{bmatrix} 1 & 0 \end{bmatrix} x(t) \leq x_{lb} \\ \begin{bmatrix} a & 0 \\ 0 & 1 \end{bmatrix} x(t) + \begin{bmatrix} b_1 \\ 0 \end{bmatrix} u(t) & \text{if } \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} x(t) \geq \begin{bmatrix} x_{lb} + \epsilon \\ -0.5 \end{bmatrix} \\ \begin{bmatrix} a & 0 \\ 0 & 1 \end{bmatrix} x(t) + \begin{bmatrix} b_2 \\ 0 \end{bmatrix} u(t) & \text{if } x(t) \geq \begin{bmatrix} x_{lb} + \epsilon \\ 0.5 \end{bmatrix} \end{cases} \tag{14.10}$$

for $u(t) \in \mathbb{R}$, $x_1(t) \in (-\infty, x_{lb}] \cup [x_{lb} + \epsilon, +\infty)$, $x_2 \in \{0,1\}$, and for any $\epsilon > 0$.

Figure 14.7 shows two open-loop simulations of the system, for $a = 0.5$, $b_1 = 0.1$, $b_2 = 0.3$, $x_{lb} = -1$, $\epsilon = 10^{-6}$. Note that when the continuous state $x_1(t)$ goes below $x_{lb} = -1$ at time t, then $x_\ell(t+1)$ triggers to 1 and the input has a stronger effect on the states from time $t+2$ on. Indeed, the steady state of x_1 is a function of the logic state x_2.

14.3 Discrete Hybrid Automata

As shown in Fig. 14.8, a *discrete hybrid automaton* (DHA) is formed by generating the mode $i(t)$ of a switched affine system through a *mode selector* function that depends on (i) the discrete state of a *finite state machine*, (ii) *discrete events* generated by the continuous variables of the switched affine system exceeding given linear thresholds (the guardlines), (iii) exogenous discrete inputs [251]. We will detail each of the four blocks in the next sections.

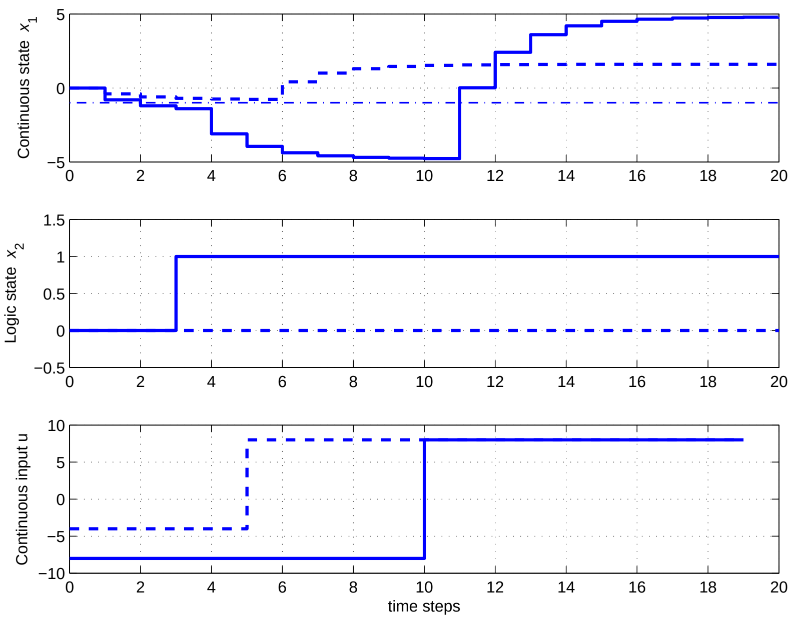


Fig. 14.7 Open-loop simulation of system (14.10) for different input excitations

14.3.1 Switched Affine System (SAS)

A switched affine system is a collection of affine systems:

$$x_c(t+1) = A^{i(t)}x_c(t) + B^{i(t)}u_c(t) + f^{i(t)}, \tag{14.11a}$$
$$y_c(t) = C^{i(t)}x_c(t) + D^{i(t)}u_c(t) + g^{i(t)}, \tag{14.11b}$$

where $t \in \mathbb{T}$ is the time indicator, $x_c \in \mathbb{R}^{n_c}$ is the continuous state vector, $u_c \in \mathbb{R}^{m_c}$ is the exogenous continuous input vector, $y_c \in \mathbb{R}^{p_c}$ is the continuous output vector, $\{A^i, B^i, f^i, C^i, D^i, g^i\}_{i\in\mathcal{I}}$ is a collection of matrices of suitable dimensions, and the mode $i(t) \in \mathcal{I} \triangleq \{1,\ldots,s\}$ is an input signal that determines the affine state update dynamics at time t. An SAS of the form (14.11) preserves the value of the state when a mode switch occurs, but it is possible to implement reset maps on an SAS as shown in [251].

14.3.2 Event Generator (EG)

An event generator is an object that generates a binary vector $\delta_e(t) \in \{0,1\}^{n_e}$ of *event conditions* according to the satisfaction of a linear (or affine) threshold condition. Let $h : \mathbb{R}^{n_c} \times \mathbb{R}^{n_c} \to \{0,1\}^{n_e}$ be a vector function defined

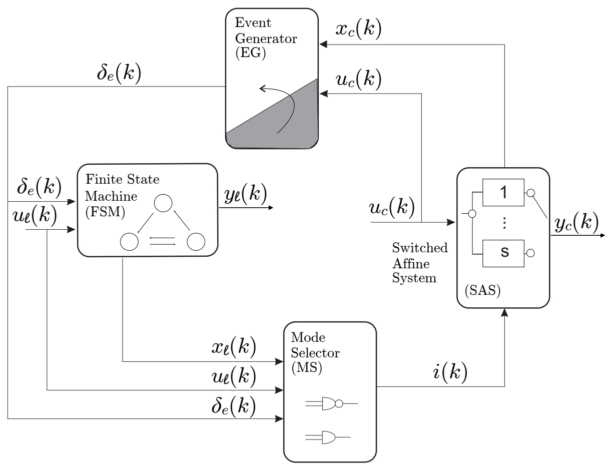


Fig. 14.8 A discrete hybrid automaton (DHA) is the connection of a finite state machine (FSM) and a switched affine system (SAS), through a mode selector (MS) and an event generator (EG). The output signals are omitted for clarity

as

$$h^i(x_c, u_c) = \begin{cases} 1 \text{ if } (H^i)'x_c + (J^i)'u_c + K^i \leq 0 \\ 0 \text{ if } (H^i)'x_c + (J^i)'u_c + K^i > 0 \end{cases}$$

where ^{i} denotes the ith component of a vector or the ith row of a matrix, and A, B, C are constant matrices of suitable dimensions. Then, events are defined as

$$\delta_e(t) = h(x_c(t), u_c(t)) \tag{14.12}$$

In particular, *state events* are modeled as $[\delta_e(t) = 1] \leftrightarrow [a'x_c(t) \leq b]$. Note that *time events* can be modeled as in (14.12) by adding the continuous time as an additional continuous and autonomous state variable, $\tau(t+1) = \tau(t) + T_s$, where T_s is the sampling time, and by letting $[\delta_e(t) = 1] \leftrightarrow [tT_s \geq \tau_0]$, where τ_0 is a given time. By doing so, the hybrid model can be written as a time-invariant one. Clearly the same approach can be used for time-varying events $\delta_e(t) = h(x_c(t), u_c(t), t)$, by using time-varying event conditions $h : \mathbb{R}^{n_c} \times \mathbb{R}^{n_c} \times \mathbb{T} \to \{0, 1\}^{n_e}$.

14.3.3 Boolean Algebra

Before dealing in detail with the other blocks constituting the DHA and introduce further notation, we recall here some basic definitions of Boolean algebra[1].

A variable δ is a Boolean variable if $\delta \in \{0,1\}$, where "$\delta = 0$" means something is false, "$\delta = 1$" that is true. A Boolean expression is obtained by combining Boolean variables through the logic operators $\neg$ (not), $\vee$ (or), $\wedge$ (and), $\leftarrow$ (implied by), $\rightarrow$ (implies), and $\leftrightarrow$ (iff). A Boolean function $f : \{0,1\}^{n-1} \mapsto \{0,1\}$ is used to define a Boolean variable δ_n as a logic function of other variables $\delta_1, \dots, \delta_{n-1}$:

$$\delta_n = f(\delta_1, \delta_2, \dots, \delta_{n-1}) \tag{14.13}$$

Given n Boolean variables $\delta_1, \dots, \delta_n$, a Boolean formula F defines a relation

$$F(\delta_1, \dots, \delta_n) \tag{14.14}$$

that must hold true. Every Boolean formula $F(\delta_1, \delta_2, \dots, \delta_n)$ can be rewritten in the conjunctive normal form (CNF)

$$\begin{gathered}\text{(CNF)} \quad \bigwedge_{j=1}^{m} \left(\left(\bigvee_{i \in P_j} \delta_i \right) \bigvee \left(\bigvee_{i \in N_j} \sim \delta_i \right) \right) \\ N_j, P_j \subseteq \{1, \dots, n\}, \ \forall j = 1, \dots, m.\end{gathered} \tag{14.15}$$

As mentioned in Section 14.2.2, often we will use the term binary variable and Boolean variable without distinction. An improper arithmetic operation over Boolean variables should be understood as an arithmetic operation over corresponding binary variables and, vice versa, an improper Boolean function of binary variables should be interpreted as the same function over the corresponding Boolean variables. Therefore, from now on we will call "binary" variables both 0-1 variables and Boolean variables.

14.3.4 Finite State Machine (*FSM*)

A finite state machine[2] (or automaton) is a discrete dynamic process that evolves according to a Boolean state update function:

[1] A more comprehensive treatment of Boolean calculus can be found in digital circuit design texts, e.g. [80, 132]. For a rigorous exposition see e.g. [192].

[2] In this text we will only refer to synchronous finite state machines, where the transitions may happen only at sampling times. The adjective synchronous will be omitted for brevity.

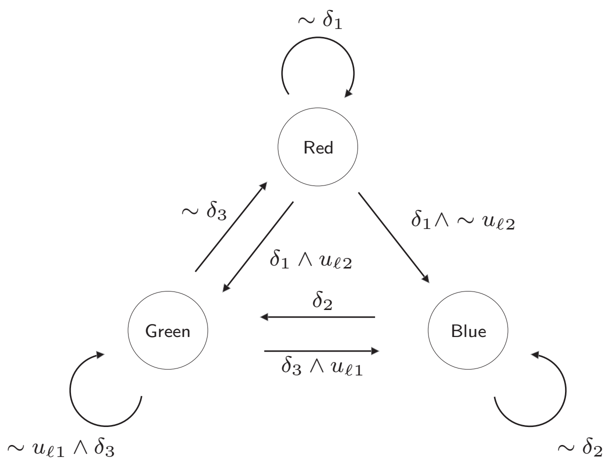

$\sim\delta_1$
Red
$\sim\delta_3$
$\delta_1\wedge\sim u_{\ell 2}$
$\delta_1\wedge u_{\ell 2}$
δ_2
Green
Blue
$\delta_3\wedge u_{\ell 1}$
$\sim u_{\ell 1}\wedge\delta_3$
$\sim\delta_2$

Fig. 14.9 Example of finite state machine

$$x_\ell(t+1) = f_\ell(x_\ell(t), u_\ell(t), \delta_e(t)), \tag{14.16a}$$

where $x_\ell \in \{0,1\}^{n_\ell}$ is the binary state, $u_\ell \in \{0,1\}^{m_\ell}$ is the exogenous binary input, $\delta_e(t)$ is the endogenous binary input coming from the EG, and $f_\ell : \{0,1\}^{n_\ell} \times \{0,1\}^{m_\ell} \times \{0,1\}^{n_e} \to \{0,1\}^{n_\ell}$ is a deterministic Boolean function.

An FSM can be conveniently represented using an oriented graph. An FSM may also have an associated binary output

$$y_\ell(t) = g_\ell(x_\ell(t), u_\ell(t), \delta_e(t)), \tag{14.16b}$$

where $y_\ell \in \{0,1\}^{p_\ell}$ and $g_\ell : \{0,1\}^{n_\ell} \times \{0,1\}^{m_\ell} \times \{0,1\}^{n_e} \mapsto \{0,1\}^{p_\ell}$.

Example 14.4. Figure 14.9 shows a finite state machine where $u_\ell = [u_{\ell 1}\ u_{\ell 2}]'$ is the input vector, and $\delta = [\delta_1 \ldots \delta_3]'$ is a vector of signals coming from the event generator. The Boolean state update function (also called *state transition function*) is:

$$x_\ell(t+1) = \begin{cases} \mathsf{Red} \text{ if } ((x_\ell(t) = \mathsf{Green}) \wedge \sim\delta_3) \vee \\ \quad ((x_\ell(t) = \mathsf{Red}) \wedge \sim\delta_1), \\ \mathsf{Green} \text{ if } ((x_\ell(t) = \mathsf{Red}) \wedge \delta_1 \wedge u_{\ell 2}) \vee \\ \quad ((x_\ell(t) = \mathsf{Blue}) \wedge \delta_2) \vee \\ \quad ((x_\ell(t) = \mathsf{Green}) \wedge \sim u_{\ell 1} \wedge \delta_3), \\ \mathsf{Blue} \text{ if } ((x_\ell(t) = \mathsf{Red}) \wedge \delta_1 \wedge \sim u_{\ell 2}) \vee \\ \quad ((x_\ell(t) = \mathsf{Green}) \wedge (\delta_3 \wedge u_{\ell 1})) \vee \\ \quad ((x_\ell(t) = \mathsf{Blue}) \wedge \sim\delta_2)). \end{cases} \tag{14.17}$$

By associating a binary vector $x_\ell = \left[\begin{smallmatrix} x_{\ell 1} \\ x_{\ell 2} \end{smallmatrix}\right]$ to each state ($\mathsf{Red} = \left[\begin{smallmatrix} 0 \\ 0 \end{smallmatrix}\right]$, $\mathsf{Green} = \left[\begin{smallmatrix} 0 \\ 1 \end{smallmatrix}\right]$, and $\mathsf{Blue} = \left[\begin{smallmatrix} 1 \\ 0 \end{smallmatrix}\right]$), one can rewrite (14.17) as:

$$\begin{aligned}
x_{\ell 1}(t+1) &= (\sim x_{\ell 1} \wedge \sim x_{\ell 2} \wedge \delta_1 \wedge \sim u_{\ell 2}) \vee \\
&\quad (x_{\ell 1} \wedge \sim \delta_2) \vee (x_{\ell 2} \wedge \delta_3 \wedge u_{\ell 1}), \\
x_{\ell 2}(t+1) &= (\sim x_{\ell 1} \wedge \sim x_{\ell 2} \wedge \delta_1 \wedge u_{\ell 2}) \vee \\
&\quad (x_{\ell 1} \wedge \delta_2) \vee (x_{\ell 2} \wedge \delta_3 \wedge \sim u_{\ell 1}),
\end{aligned}$$

where the time index (t) has been omitted for brevity.

Since the Boolean state update function is deterministic, for each state the conditions associated with all the outgoing arcs are mutually exclusive.

14.3.5 Mode Selector

In a DHA, the dynamic mode $i(t) \in \mathcal{I} = \{1, \ldots, s\}$ of the SAS is a function of the binary state $x_\ell(t)$, the binary input $u_\ell(t)$, and the events $\delta_e(t)$. With a slight abuse of notation, let us indicate the mode $i(t)$ through its binary encoding, $i(t) \in \{0,1\}^{n_s}$ where $n_s = \lceil \log_2 s \rceil$, so that $i(t)$ can be treated as a vector of Boolean variables[3]. Then, we define the *mode selector* by the Boolean function $f_{\mathrm{M}} : \{0,1\}^{n_\ell} \times \{0,1\}^{m_\ell} \times \{0,1\}^{n_e} \to \{0,1\}^{n_s}$. The output of this function

$$i(t) = \mu(x_\ell(t), u_\ell(t), \delta_e(t)) \tag{14.18}$$

is called the *active mode* of the DHA at time t. We say that a *mode switch* occurs at step t if $i(t) \neq i(t-1)$. Note that, in contrast to continuous-time hybrid models where switches can occur at any time, in our discrete-time setting, as mentioned earlier, a mode switch can only occur at sampling instants.

14.3.6 DHA Trajectories

For a given initial condition $\left[\begin{smallmatrix} x_c(0) \\ x_\ell(0) \end{smallmatrix}\right] \in \mathbb{R}^{n_c} \times \{0,1\}^{n_\ell}$, and inputs $\left[\begin{smallmatrix} u_c(t) \\ u_\ell(t) \end{smallmatrix}\right] \in \mathbb{R}^{m_c} \times \{0,1\}^{m_\ell}$, $t \in \mathbb{T}$, the state $x(t)$ of the system is computed for all $t \in \mathbb{T}$ by recursively iterating the set of equations:

[3] Any discrete variable $\alpha \in \{\alpha_1, \ldots, \alpha_j\}$ admits a Boolean encoding $a \in \{0,1\}^{d(j)}$, where $d(j)$ is the number of bits used to represent $\alpha_1, \ldots, \alpha_j$. For example, $\alpha \in \{0,1,2\}$ may be encoded as $a \in \{0,1\}^2$ by associating $\left[\begin{smallmatrix} 0 \\ 0 \end{smallmatrix}\right] \to 0$, $\left[\begin{smallmatrix} 0 \\ 1 \end{smallmatrix}\right] \to 1$, $\left[\begin{smallmatrix} 1 \\ 0 \end{smallmatrix}\right] \to 2$.

$$\delta_e(t) = h(x_c(t), u_c(t), t) \tag{14.19a}$$
$$i(t) = \mu(x_\ell(t), u_\ell(t), \delta_e(t)) \tag{14.19b}$$
$$y_c(t) = C^{i(t)}x_c(t) + D^{i(t)}u_c(t) + g^{i(t)} \tag{14.19c}$$
$$y_\ell(t) = g_\ell(x_\ell(t), u_\ell(t), \delta_e(t)) \tag{14.19d}$$
$$x_c(t+1) = A^{i(t)}x_c(t) + B^{i(t)}u_c(t) + f^{i(t)} \tag{14.19e}$$
$$x_\ell(t+1) = f_\ell(x_\ell(t), u_\ell(t), \delta_e(t)) \tag{14.19f}$$

A definition of well-posedness of DHA can be given similarly to Definition 14.1 by requiring that the successor states $x_c(t+1)$, $x_\ell(t+1)$ and the outputs $y_c(t)$, $y_\ell(t)$ are uniquely defined functions of $x_c(t)$, $x_\ell(t)$, $u_c(t)$, $u_\ell(t)$ defined by the DHA equations (14.19).

DHA can be considered as a subclass of *hybrid automata* (HA) [7]. The main difference is in the time model: DHA are based on discrete time, HA on continuous time. Moreover DHA models do not allow instantaneous transitions, and are deterministic, opposed to HA where any enabled transition may occur in zero time. This has two consequences (i) DHA do not admit live-locks (infinite switches in zero time), (ii) DHA do not admit Zeno behaviors (infinite switches in finite time). Finally in DHA models, guards, reset maps and continuous dynamics are limited to linear (or affine) functions.

14.4 Logic and Mixed-Integer Inequalities

Despite the fact that DHA are rich in their expressiveness and are therefore quite suitable for modeling and simulating a wide class of hybrid dynamical systems, they are not directly suitable for solving optimal control problems, because of their heterogeneous discrete and continuous nature. In this section we want to describe how DHA can be translated into different hybrid models that are more suitable for optimization. We highlight the main techniques of the translation process, by generalizing several results that appeared in the literature [117, 221, 262, 197, 76, 37, 143, 74, 252, 261, 202].

14.4.1 Transformation of Boolean Relations

Boolean formulas can be equivalently represented as integer linear inequalities. For instance, $\delta_1 \vee \delta_2 = 1$ is equivalent to $\delta_1 + \delta_2 \geq 1$ [262]. Some equivalences are reported in Table 14.1. The results of the table can be generalized as follows.

Lemma 14.1. *For every Boolean formula $F(\delta_1, \delta_2, \dots, \delta_n)$ there exists a polyhedral set P such that a set of binary values $\{\delta_1, \delta_2, \dots, \delta_n\}$ satisfies the Boolean formula F if and only if $\delta = [\delta_1 \; \delta_2 \; \dots \; \delta_n]' \in P$.*

relation	Boolean	linear constraints
AND	$\delta_1 \wedge \delta_2$	$\delta_1 = 1$, $\delta_2 = 1$ **or** $\delta_1 + \delta_2 \geq 2$
OR	$\delta_1 \vee X_2$	$\delta_1 + \delta_2 \geq 1$
NOT	$\sim \delta_1$	$\delta_1 = 0$
XOR	$\delta_1 \oplus \delta_2$	$\delta_1 + \delta_2 = 1$
IMPLY	$\delta_1 \to \delta_2$	$\delta_1 - \delta_2 \leq 0$
IFF	$\delta_1 \leftrightarrow \delta_2$	$\delta_1 - \delta_2 = 0$
ASSIGNMENT $\delta_3 = \delta_1 \wedge \delta_2$	$\delta_3 \leftrightarrow \delta_1 \wedge \delta_2$	$\delta_1 + (1 - \delta_3) \geq 1$ $\delta_2 + (1 - \delta_3) \geq 1$ $(1 - \delta_1) + (1 - \delta_2) + \delta_3 \geq 1$

Table 14.1 Basic conversion of Boolean relations into mixed-integer inequalities. Relations involving the inverted literals $\sim \delta$ can be obtained by substituting $(1 - \delta)$ for δ in the corresponding inequalities. More conversions are reported in [194], or can be derived by (14.15)–(14.20)

Proof: Given a formula F, one way of constructing one of such polyhedra P is to rewrite F in the conjunctive normal form (14.15), and then simply define P as

$$P = \left\{ \delta \in \{0,1\}^n : \begin{array}{c} 1 \leq \sum_{i\in P_1} \delta_i \; + \sum_{i \in N_1} (1 - \delta_i) \\ \vdots \\ 1 \leq \sum_{i\in P_m} \delta_i + \sum_{i \in N_m} (1 - \delta_i) \end{array} \right\} \tag{14.20}$$

□

The smallest polyhedron P associated with formula F has the following geometric interpretation: Assume to list all the 0-1 combinations of δ_i's satisfying F (namely, to generate the truth table of F), and think of each combination as an n-dimensional binary vector in $\mathbb{R}^n$, then P is the convex hull of such vectors [76, 142, 195]. For methods to compute convex hulls, we refer the reader to [101].

14.4.2 Translating DHA Components into Linear Mixed-Integer Relations

Events of the form (14.12) can be expressed equivalently as

$$h^i(x_c(t), u_c(t), t) \leq M^i(1 - \delta_e^i), \tag{14.21a}$$

$$h^i(x_c(t), u_c(t), t) > m^i \delta_e^i, \qquad i = 1, \ldots, n_e, \tag{14.21b}$$

where M^i, m^i are upper and lower bounds, respectively, on $h^i(x_c(t), u_c(t), t)$. As we have pointed out in Section 14.2.1, from a computational viewpoint it is convenient to avoid strict inequalities. As suggested in [262], we modify the strict inequality (14.21b) into

$$h^i(x_c(t), u_c(t), t) \geq \epsilon + (m^i - \epsilon)\delta_e^i \tag{14.21c}$$

where ϵ is a small positive scalar (e.g., the machine precision). Clearly, as for the case of discontinuous PWA discussed in Section 14.2.1, Equations (14.21) or (14.12) are equivalent to (14.21c) only for $h^i(x_c(t), u_c(t), t) \leq 0$ and $h^i(x_c(t), u_c(t), t) \geq \epsilon$.

Regarding switched affine dynamics, we first rewrite the state-update equation (14.11a) as the following combination of affine terms and *if-then-else* conditions:

$$z_1(t) = \begin{cases} A^1 x_c(t) + B^1 u_c(t) + f^1, & \text{if } (i(t) = 1), \\ 0, & \text{otherwise}, \end{cases} \tag{14.22a}$$

$$\vdots$$

$$z_s(t) = \begin{cases} A^s x_c(t) + B^s u_c(t) + f^s, & \text{if } (i(t) = s), \\ 0, & \text{otherwise}, \end{cases} \tag{14.22b}$$

$$x_c(t+1) = \sum_{i=1}^{s} z_i(t), \tag{14.22c}$$

where $z_i(t) \in \mathbb{R}^{n_c}$, $i = 1, \ldots, s$. The output equation (14.11b) admits a similar transformation.

A generic if-then-else construct of the form

$$\text{IF } \delta \text{ THEN } z = a^{1'}x + b^{1'}u + f^1 \text{ ELSE } z = a^{2'}x + b^{2'}u + f^2, \tag{14.23}$$

where $\delta \in \{0, 1\}$, $z \in \mathbb{R}$, $x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$, and $a^1, b^1, f^1, a^2, b^2, f^2$ are constants of suitable dimensions, can be translated into [41]

$$(m_2 - M_1)\delta + z \leq a^{2'}x + b^{2'}u + f^2, \tag{14.24a}$$

$$(m_1 - M_2)\delta - z \leq -a^{2'}x - b^{2'}u - f^2, \tag{14.24b}$$

$$(m_1 - M_2)(1 - \delta) + z \leq a^{1'}x + b^{1'}u + f^1, \tag{14.24c}$$

$$(m_2 - M_1)(1 - \delta) - z \leq -a^{1'}x - b^{1'}u - f^1, \tag{14.24d}$$

where M_i, m_i are upper and lower bounds on $a^i x + b^i u + f^i$, $i = 1, 2$.

Finally, the mode selector function and binary state-update function of the automaton are Boolean functions that can be translated into integer linear inequalities as described in Section 14.4.1. The idea of transforming a well-posed FSM into a set of Boolean equalities was also presented in [207] where the authors performed model checking using (mixed) integer optimization on an equivalent set of integer inequalities.

14.5 Mixed Logical Dynamical Systems

Given a DHA representation of a hybrid process, by following the techniques described in the previous section for converting logical relations into inequalities we obtain an equivalent representation of the DHA as a *mixed logical dynamical* (MLD) system [37] described by the following relations:

$$x(t+1) = Ax(t) + B_1u(t) + B_2\delta(t) + B_3z(t) + B_5, \tag{14.25a}$$
$$y(t) = Cx(t) + D_1u(t) + D_2\delta(t) + D_3z(t) + D_5, \tag{14.25b}$$
$$E_2\delta(t) + E_3z(t) \leq E_1u(t) + E_4x(t) + E_5, \tag{14.25c}$$

where $x \in \mathbb{R}^{n_c} \times \{0,1\}^{n_\ell}$ is a vector of continuous and binary states, $u \in \mathbb{R}^{m_c} \times \{0,1\}^{m_\ell}$ are the inputs, $y \in \mathbb{R}^{p_c} \times \{0,1\}^{p_\ell}$ the outputs, $\delta \in \{0,1\}^{r_\ell}$ are auxiliary binary variables, $z \in \mathbb{R}^{r_c}$ are continuous auxiliary variables which arise in the transformation (see Example 14.5), and A, B_1, B_2, B_3, C, D_1, D_2, D_3, E_1,...,E_5 are matrices of suitable dimensions. Given the current state $x(t)$ and input $u(t)$, the time-evolution of (14.25) is determined by finding a feasible value for $\delta(t)$ and $z(t)$ satisfying (14.25c), and then by computing $x(t+1)$ and $y(t)$ from (14.25a)–(14.25b).

As MLD models consist of a collection of linear difference equations involving both real and binary variables and a set of linear inequality constraints, they are model representations of hybrid systems that can be easily used in optimization algorithms, as will be described in Chapter 15.

A definition of well-posedness of the MLD system (14.25) can be given similarly to Definition 14.1 by requiring that for all $x(t)$ and $u(t)$ within a given bounded set the pair of variables $\delta(t)$, $z(t)$ satisfying (14.25c) is unique, so that the successor state $x(t+1)$ and output $y(t)$ are also uniquely defined functions of $x(t)$, $u(t)$ through (14.25a)–(14.25b)[4]. Such a well-posedness assumption is usually guaranteed by the procedure described in Section 14.3.3 used to generate the linear inequalities (14.25c). A numerical test for well-posedness is reported in [37, Appendix 1].

Note that the constraints (14.25c) allow one to specify additional linear constraints on continuous variables (e.g., constraints over physical variables of the system), and logical constraints over Boolean variables. The ability to include constraints, constraint prioritization, and heuristics adds to the expressiveness and generality of the MLD framework. Note also that despite the fact that the description (14.25) seems to be linear, clearly the nonlinearity is concentrated in the integrality constraints over binary variables.

Example 14.5. Consider the following simple switched linear system [37]

$$x(t+1) = \begin{cases} 0.8x(t) + u(t) \text{ if } x(t) \geq 0 \\ -0.8x(t) + u(t) \text{ if } x(t) < 0 \end{cases} \tag{14.26}$$

[4] For a more general definition of well-posedness of MLD systems see [37].

where $x(t) \in [-10, 10]$, and $u(t) \in [-1, 1]$. The condition $x(t) \geq 0$ can be associated to an event variable $\delta(t) \in \{0, 1\}$ defined as

$$[\delta(t) = 1] \ \leftrightarrow \ [x(t) \geq 0] \tag{14.27}$$

By using the transformations (14.21a)–(14.21c), equation (14.27) can be expressed by the inequalities

$$-m\delta(t) \leq x(t) - m \tag{14.28a}$$
$$-(M + \epsilon)\delta(t) \leq -x(t) - \epsilon \tag{14.28b}$$

where $M = -m = 10$, and ϵ is an arbitrarily small positive scalar. Then (14.26) can be rewritten as

$$x(t + 1) = 1.6\delta(t)x(t) - 0.8x(t) + u(t) \tag{14.29}$$

By defining a new variable $z(t) = \delta(t)x(t)$ which, by (14.23)–(14.24) can be expressed as

$$z(t) \leq M\delta(t) \tag{14.30a}$$
$$z(t) \geq m\delta(t) \tag{14.30b}$$
$$z(t) \leq x(t) - m(1 - \delta(t)) \tag{14.30c}$$
$$z(t) \geq x(t) - M(1 - \delta(t)) \tag{14.30d}$$

the evolution of system (14.26) is ruled by the linear equation

$$x(t + 1) = 1.6z(t) - 0.8x(t) + u(t) \tag{14.31}$$

subject to the linear constraints (14.28) and (14.30). Therefore, the MLD equivalent representation of (14.26) for $x \in [-10, -\epsilon] \cup [0, 10]$ and $u \in [-1, 1]$ is given by collecting Equations (14.31), (14.28) and (14.30).

14.6 Model Equivalence

In the previous chapters we have presented three different classes of discrete-time hybrid models: PWA systems, DHA, and MLD systems. For what we described in Section 14.3.3, under the assumption that the set of valid states and inputs is bounded, DHA systems can always be equivalently described as MLD systems. Also, a PWA system is a special case of a DHA whose threshold events and mode selector function are defined by the PWA partition (14.1c). Therefore, a PWA system with bounded partition $\mathcal{C}$ can always be described as an MLD system (an efficient way of modeling PWA systems in MLD form is reported in [37]). The converse result, namely that MLD systems (and therefore DHA) can be represented as PWA systems, is less obvious. For any choice of δ, model (14.25) represented an affine system defined over a polyhedral domain. Under the assumption of well posedness these domains do not overlap. This result was proved formally in [31, 26, 113].

Such equivalence results are of interest because DHA are most suitable in the modeling phase, but MLD systems are most suitable for solving open-loop finite time optimal control problems, and PWA systems are most suitable for solving finite time optimal control problems in state-feedback form, as will be described in Chapter 15.

14.7 The HYSDEL Modeling Language

A modeling language was proposed in [251] to describe DHA models, called HYbrid System DEscription Language (HYSDEL). The HYSDEL description of a DHA is an abstract modeling step. The associated HYSDEL compiler then translates the description into several computational models, in particular into an MLD using the technique presented in Section 14.4, and PWA form using either the approach of [113] or the approach of [26]. HYSDEL can generate also a simulator that runs as a function in MATLAB$^{\text{TM}}$. Both the HYSDEL compiler [5] and the Hybrid Toolbox[6] can import and convert HYSDEL models.

In this section we illustrate the functionality of HYSDEL through a set of examples. For more examples and the detailed syntax we refer the interested reader to [250].

Example 14.6. Consider the DHA system:

$$\text{SAS: } x_c'(t) = \begin{cases} x_c(t) + u_c(t) - 1, & \text{if } i(t) = 1, \\ 2x_c(t), & \text{if } i(t) = 2, \\ 2, & \text{if } i(t) = 3, \end{cases} \tag{14.32a}$$

$$\text{EG: } \begin{cases} \delta_e(t) = [x_c(t) \geq 0], \\ \delta_f(t) = [x_c(t) + u_c(t) - 1 \geq 0], \end{cases} \tag{14.32b}$$

$$\text{MS: } i(t) = \begin{cases} 1, \text{ if } \begin{bmatrix} \delta_e(t) \\ \delta_f(t) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \\ 2, \text{ if } \delta_e(t) = 1, \\ 3, \text{ if } \begin{bmatrix} \delta_e(t) \\ \delta_f(t) \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \end{cases} \tag{14.32c}$$

The corresponding HYSDEL list is reported in Table 14.2.

The HYSDEL list is composed of two parts. The first one, called INTERFACE, contains the declaration of all variables and parameters, so that it is possible to make the proper type checks. The second part, IMPLEMEN-

[5] http://control.ee.ethz.ch/~hybrid/hysdel

[6] http://www.dii.unisi.it/hybrid/toolbox

```
SYSTEM sample {
INTERFACE {
  STATE {
    REAL xr [-10, 10]; }
  INPUT {
    REAL ur [-2, 2]; }
}
IMPLEMENTATION {
  AUX {
    REAL z1, z2, z3;
    BOOL de, df, d1, d2, d3; }
  AD {
    de = xr >= 0;
    df = xr + ur - 1 >= 0; }
  LOGIC {
    d1 = ~de & ~df;
    d2 = de;
    d3 = ~de & df; }
  DA {
    z1 = {IF d1 THEN xr + ur - 1 };
    z2 = {IF d2 THEN 2 * xr };
    z3 = {IF (~de & df)  THEN 2 }; }
  CONTINUOUS {
    xr = z1 + z2 + z3; }
}}
```

Table 14.2 Sample HYSDEL list of system (14.32)

TATION, is composed of specialized sections where the relations among the variables are described. These sections are described next.

The HYSDEL section AUX contains the declaration of the auxiliary variables used in the model. The HYSDEL section AD allows one to define Boolean variables from continuous ones, and is based exactly on the semantics of the event generator (EG) described earlier. The HYSDEL section DA defines continuous variables according to if-then-else conditions. This section models part of the switched affine system (SAS), namely the variables z_i defined in (14.22a)–(14.22b). The CONTINUOUS section describes the linear dynamics, expressed as difference equations. This section models (14.22c). The section LOGIC allows one to specify arbitrary functions of Boolean variables.

Example 14.7. Consider again the PWA system described in Example 14.1. Assume that $[-5, 5] \times [-5, 5]$ is the set of states $x(t)$ of interest and $u(t) \in [-1, 1]$. By using HYSDEL the PWA system (14.3) is described as in Table 14.3 and the equivalent MLD form is obtained

```
/* 2x2 PWA system */

SYSTEM pwa {

INTERFACE {
    STATE { REAL x1 [-5,5];
            REAL x2 [-5,5];
        }
    INPUT { REAL u [-1,1];
        }
    OUTPUT{ REAL y;
        }
    PARAMETER {
        REAL alpha = 60*pi/180;
        REAL C = cos(alpha);
        REAL S = sin(alpha);
        REAL MLD_epsilon = 1e-6;    }
    }

IMPLEMENTATION {
        AUX { REAL z1,z2;
              BOOL sign; }
        AD  { sign = x1<=0; }

        DA  { z1 = {IF sign THEN 0.8*(C*x1+S*x2)
                    ELSE 0.8*(C*x1-S*x2) };
              z2 = {IF sign THEN 0.8*(-S*x1+C*x2)
                    ELSE 0.8*(S*x1+C*x2) };  }

        CONTINUOUS { x1 = z1;
                     x2 = z2+u; }

        OUTPUT { y = x2;  }
      }
}
```

Table 14.3 HYSDEL model of the PWA system described in Example 14.1

$$
\begin{aligned}
&x(t+1) = \left[\begin{smallmatrix}1 & 0\\ 0 & 1\end{smallmatrix}\right] z(t) + \left[\begin{smallmatrix}0\\ 1\end{smallmatrix}\right] u(t)\\
&y(t) = \left[\begin{smallmatrix}0 & 1\end{smallmatrix}\right] x(t)\\
&\begin{bmatrix}-5-\epsilon\\ 5\\ c_1\\ c_1\\ -c_1\\ -c_1\\ c_1\\ c_1\\ -c_1\\ -c_1\end{bmatrix}\delta(t) + \begin{bmatrix}0 & 0\\ 0 & 0\\ -1 & 0\\ 1 & 0\\ -1 & 0\\ 1 & 0\\ 0 & -1\\ 0 & 1\\ 0 & -1\\ 0 & 1\end{bmatrix} z(t) \le \begin{bmatrix}1 & 0\\ -1 & 0\\ -0.4 & -c_2\\ 0.4 & c_2\\ -0.4 & c_2\\ 0.4 & -c_2\\ c_2 & -0.4\\ -c_2 & 0.4\\ -c_2 & -0.4\\ c_2 & 0.4\end{bmatrix} x(t) + \begin{bmatrix}-\epsilon\\ 5\\ c_1\\ c_1\\ 0\\ 0\\ c_1\\ c_1\\ 0\\ 0\end{bmatrix}.
\end{aligned}
$$

where $c_1 = 4(\sqrt{3}+1)$, $c_2 = 0.4\sqrt{3}$, $\epsilon = 10^{-6}$. Note that in Table 14.3 the OUTPUT section allows to specify a linear map for the output vector y.

Example 14.8. Consider again the hybrid spring-mass system described in Example 14.2. Assume that $[-5,5]\times[-5,5]$ is the set of states x and $[-10,10]$ the set of continuous inputs u_1 of interest. By using HYSDEL, system (14.6) is

```
 /* Spring-Mass System
 */

SYSTEM springmass  {

INTERFACE { /* Description of variables and constants */

    STATE {
            REAL x1 [-5,5];
            REAL x2 [-5,5];
    }

    INPUT { REAL u1 [-10,10];
            BOOL u2;
        }


    PARAMETER {
            /* Spring breakpoint */
            REAL xm;

            /* Dynamic coefficients */
            REAL A111,A112,A121,A122,A211,A212,A221,A222;
            REAL A311,A312,A321,A322,A411,A412,A421,A422;
            REAL B111,B112,B121,B122,B211,B212,B221,B222;
            REAL B311,B312,B321,B322,B411,B412,B421,B422;
    }
}

IMPLEMENTATION {
    AUX {
            REAL zx11,zx12,zx21,zx22,zx31,zx32,zx41,zx42;
            BOOL region;
    }

    AD {    /* spring region */
            region = x1-xm <= 0;
    }

    DA {
            zx11 = { IF region  & u2  THEN A111*x1+A112*x2+B111*u1+B112};
            zx12 = { IF region  & u2  THEN A121*x1+A122*x2+B121*u1+B122};
            zx21 = { IF region  & ~u2 THEN A211*x1+A212*x2+B211*u1+B212};
            zx22 = { IF region  & ~u2 THEN A221*x1+A222*x2+B221*u1+B222};
            zx31 = { IF ~region & u2  THEN A311*x1+A312*x2+B311*u1+B312};
            zx32 = { IF ~region & u2  THEN A321*x1+A322*x2+B321*u1+B322};
            zx41 = { IF ~region & ~u2 THEN A411*x1+A412*x2+B411*u1+B412};
            zx42 = { IF ~region & ~u2 THEN A421*x1+A422*x2+B421*u1+B422};
    }

    CONTINUOUS {    x1=zx11+zx21+zx31+zx41;
                    x2=zx12+zx22+zx32+zx42;
    }
  }
}
```

Table 14.4 HYSDEL model of the spring-mass system described in Example 14.2

described as in Table 14.4 and an equivalent MLD model with 2 continuous states, 1 continuous input, 1 binary input, 9 auxiliary binary variables, 8 continuous variables, and 58 mixed-integer inequalities is obtained.

```
/* System with logic state */

SYSTEM SLS  {
INTERFACE { /* Description of variables and constants */
    STATE {
            REAL x1 [-1000,1000];
            BOOL x2;
    }
    INPUT { REAL u [-1000,1000];
        }
    PARAMETER {

            /* Lower Bound Point */
            REAL xlb = -1;

            /* Dynamic coefficients */
            REAL a = .5;
            REAL b1 =.1;
            REAL b2 =.3;
    }
}

IMPLEMENTATION {
    AUX {BOOL region;
         REAL zx1;
    }
    AD {  /* PWA Region */
            region = x1-xlb <= 0;
    }
    DA { zx1={IF x2 THEN a*x1+b2*u  ELSE a*x1+b1*u};
    }
    CONTINUOUS {    x1=zx1;
    }
    AUTOMATA { x2= x2 | region;
    }
  }
}
```

Table 14.5 HYSDEL model of the system with logic state described in Example 14.3

Example 14.9. Consider again the system with a logic state described in Example 14.3. The MLD model obtained by compiling the HYSDEL list of Table 14.5 is

$$x(t+1) = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \delta(t) + \begin{bmatrix} 1 \\ 0 \end{bmatrix} z(t)$$

$$\begin{bmatrix} -9 & 0 \\ 11 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & -1 \\ 1 & -1 \\ -1 & 1 \end{bmatrix} \delta(t) + \begin{bmatrix} 0 \\ 0 \\ -1 \\ 1 \\ -1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} z(t) \leq \begin{bmatrix} 0 \\ 0 \\ -0.3 \\ 0.3 \\ -0.1 \\ 0.1 \\ 0 \\ 0 \\ 0 \end{bmatrix} u(t) + \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ -0.5 & -14 \\ 0.5 & -14 \\ -0.5 & 14 \\ 1 & 14 \\ 0 & -1 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} x(t) + \begin{bmatrix} 1 \\ 10 \\ 14 \\ 14 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

where we have assumed that $[-10, 10]$ is the set of states x_1 and $[-10, 10]$ the set of continuous inputs u of interest. In Table 14.5 the AUTOMATA section specifies the state transition equations of the finite state machine (FSM) as a collection of Boolean functions.

14.8 Literature Review

The lack of a general theory and of systematic design tools for systems having such a heterogeneous dynamical discrete and continuous nature led to a considerable interest in the study of *hybrid systems*. After the seminal work published in 1966 by Witsenhausen [263], who examined an optimal control problem for a class of hybrid-state continuous-time dynamical systems, there has been a renewed interest in the study of hybrid systems. The main reason for such an interest is probably the recent advent of technological innovations, in particular in the domain of embedded systems, where a logical/discrete decision device is "embedded" in a physical dynamical environment to change the behavior of the environment itself. Another reason is the availability of several software packages for simulation and numerical/symbolic computation that support the theoretical developments.

Several modelling frameworks for hybrid systems have appeared in the literature. We refer the interested reader to [9, **?**] and the references therein. Each class is usually tailored to solve a particular problem, and many of them look largely dissimilar, at least at first sight. Two main categories of hybrid systems were successfully adopted for analysis and synthesis [58]: *hybrid control systems* [178, 179], where continuous dynamical systems and discrete/logic automata interact (see Fig. 14.1), and *switched systems* [241, 59, 145, 266, 235], where the state-space is partitioned into regions, each one being associated with a different continuous dynamics (see Fig. 14.2).

Today, there is a widespread agreement in defining hybrid systems as dynamical systems that switch among many operating modes, where each mode is governed by its own characteristic dynamical laws, and 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) [9]. In the literature, systems whose mode only depends on external inputs are usually called *switched systems*, the others *switching systems*.

Complex systems organized in a hierarchial manner, where, for instance, discrete planning algorithms at the higher level interact with continuous control algorithms and processes at the lower level, are another example of hybrid systems. In these systems, the hierarchical organization helps to manage the complexity of the system, as higher levels in the hierarchy require less detailed models (also called *abstractions*) of the lower levels functions.

Hybrid systems arise in a large number of application areas and are attracting increasing attention in both academic theory-oriented circles as well as in industry, for instance in the automotive industry [15, 146, 55, 116, 69, 198]. Moreover, many physical phenomena admit a natural hybrid description, like circuits involving relays or diodes [24], biomolecular networks [6], and TCP/IP networks in [139].

In this book we have worked exclusively with dynamical systems formulated in discrete time. This chapter therefore focused on hybrid models formulated in discrete time. Though the effects of sampling can be neglected in most applications, some interesting mathematical phenomena occurring in hybrid systems, such as Zeno behaviors [147] do not exist in discrete time, as switches can only occur at sampling instants. On the other hand, most of these phenomena are usually a consequence of the continuous-time switching model, rather than the real behavior. Our main motivation for concentrating on discrete-time models is that optimal control problems are easier to formulate and to solve numerically than continuous-time formulations.

In the theory of hybrid systems, several problems were investigated in the last few years. Besides the issues of existence and computation of trajectories described in Section 14.1, several other issues were considered. These include: equivalence of hybrid models, stability and passivity analysis, reachability analysis and verification of safety properties, controller synthesis, observability analysis, state estimation and fault detection schemes, system identification, stochastic and event-driven dynamics. We will briefly review some of these results in the next paragraphs and provide pointers to some relevant literature references.

Equivalence of Linear Hybrid Systems

Under the condition that the MLD system is well-posed, the result showing that an MLD systems admits an equivalent PWA representation was proved in [31]. A slightly different and more general proof is reported in [26], where the author also provides efficient MLD to PWA translation algorithms. A different algorithm for obtaining a PWA representation of a DHA is reported in [113].

The fact that PWA systems are equivalent to interconnections of linear systems and finite automata was pointed out by Sontag [242]. In [138, 35] the authors proved the equivalence of discrete-time PWA/MLD systems with other classes of discrete-time hybrid systems (under some assumptions) such as linear complementarity (LC) systems [136, 254, 137], extended linear complementarity (ELC) systems [85], and max-min-plus-scaling (MMPS) systems [86].

Stability Analysis

Piecewise quadratic Lyapunov stability is becoming a standard in the stability analysis of hybrid systems [145, 87, 93, 209, 210]. It is a deductive way to prove the stability of an equilibrium point of a subclass of hybrid systems (piecewise affine systems). The computational burden is usually low, at the price of a convex relaxation of the problem, that leads to possibly conservative results. Such conservativeness can be reduced by constructing piecewise polynomial Lyapunov functions via semidefinite programming by means of the sum of squares (SOS) decomposition of multivariate polynomials [208]. SOS methods for analyzing stability of continuous-time hybrid and switched systems are described in [214]. For the general class of switched systems of the form $\dot{x} = f_i(x)$, $i = 1, \ldots, s$, an extension of the Lyapunov criterion based on multiple Lyapunov functions was introduced in [59]. The reader is also referred to the book of Liberzon [174].

The research on stability criteria for PWA systems has been motivated by the fact that the stability of each component subsystem is not sufficient to guarantee stability of a PWA system (and vice versa). Branicky [59], gives an example where stable subsystems are suitably combined to generate an unstable PWA system. Stable systems constructed from unstable ones have been reported in [253]. These examples point out that restrictions on the switching have to be imposed in order to guarantee that a PWA composition of stable components remains stable.

Passivity analysis of hybrid models has received very little attention, except for the contributions of [70, 182, 269] and [212], in which notions of passivity for continuous-time hybrid systems are formulated, and of [27], where passivity and synthesis of passifying controllers for discrete-time PWA systems are investigated.

Reachability Analysis and Verification of Safety Properties

Although simulation allows to probe a model for a certain initial condition and input excitation, any analysis based on simulation is likely to miss the subtle phenomena that a model may generate, especially in the case of hybrid models. Reachability analysis (also referred to as "safety analysis" or "formal verification"), aims at detecting if a hybrid model will eventually reach unsafe state configurations or satisfy a temporal logic formula [7] for *all* possible initial conditions and input excitations within a prescribed set. Reachability analysis relies on a reach set computation algorithm, which is strongly related to the mathematical model of the system. In the case of MLD systems, for example, the reachability analysis problem over a finite time horizon (also referred to as *bounded model checking*) can be cast as a mixed-integer feasibility problem. Reachability analysis was also investigated via *bisimulation* ideas, namely by analyzing the properties of a simpler, more abstract system instead of those of the original hybrid dynamics [206].

Timed automata and hybrid automata have proved to be a successful modeling framework for formal verification and have been widely used in the literature. The starting point for both models is a finite state machine equipped with continuous dynamics. In the theory of *timed automata*, the dynamic part is the continuous-time flow $\dot{x} = 1$. Efficient computational tools complete the theory of timed automata and allow one to perform verification and scheduling of such models. Timed automata were extended to *linear hybrid automata* [7], where the dynamics is modeled by the differential inclusion $a \leq \dot{x} \leq b$. Specific tools allow one to verify such models against safety and liveness requirements. Linear hybrid automata were further extended to *hybrid automata* where the continuous dynamics is governed by differential equations. Tools exist to model and analyze those systems, either directly or by approximating the model with timed automata or linear hybrid automata (see e.g. the survey paper [239]).

Control

The majority of the control approaches for hybrid systems is based on optimal control ideas (see the survey [265]). For continuous-time hybrid systems, most authors either studied necessary conditions for a trajectory to be optimal [211, 245], or focused on the computation of optimal/suboptimal solutions by means of dynamic programming or the maximum principle [119, 222, 133, 134, 73, 266, 175, 234, 238, 241, 247, 62, 131, 179, 119, 61, 225, 67] The hybrid optimal control problem becomes less complex when the dynamics is expressed in discrete-time, as the main source of complexity becomes the combinatorial (yet finite) number of possible switching sequences. As will be shown in Chapter 15, optimal control problems can be solved for discrete-time hybrid systems using either the PWA or the MLD models described in this chapter. The solution to optimal control problems for discrete-time hybrid systems was first outlined by Sontag in [241]. In his plenary presentation [186] 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 piecewise linear norms. The detailed exposition presented in the first part of Chapter 15 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.

Model Predictive Control for discrete-time PWA systems and their stability and robustness properties have been studied in [60, 54, 28, 171, 168, 169]. Invariant sets computation for PWA systems has also been studied in [170, 4].

Observability and State Estimation

Observability of hybrid systems is a fundamental concept for understanding if a state observer can be designed for a hybrid system and how well it will perform. In [31] the authors show that observability properties (as well as reachability properties) can be very complex and present a number of counterexamples that rule out obvious conjectures about inheriting observability/controllability properties from the composing linear subsystems. They also provide observability tests based on linear and mixed-integer linear programming.

State estimation is the reconstruction of the value of unmeasurable state variables based on output measurements. While state estimation is primarily required for output-feedback control, it is also important in problems of monitoring and fault detection [36, 16]. Observability properties of hybrid systems were directly exploited for designing convergent state estimation schemes for hybrid systems in [94].

Identification

Identification techniques for piecewise affine systems were recently developed [95, 228, 150, 34, 152, 258, 205], that allow one to derive models (or parts of models) from input/output data.

Extensions to Event-driven and Stochastic Dynamics

The discrete-time methodologies described in this chapter were employed in [68] to tackle event-based continuous-time hybrid systems with integral continuous dynamics, called *integral continuous-time hybrid automata* (icHA).

The hybrid dynamics is translated into an equivalent MLD form, where continuous-time is an additional state variable and the index t counts events rather than time steps. Extensions of DHA to discrete-time stochastic hybrid dynamics was proposed in [29], where discrete-state transitions depend on both deterministic and stochastic events.

Chapter 15
Optimal Control of Hybrid Systems

In this chapter we study the finite time, infinite time and receding horizon optimal control problem for the class of hybrid systems presented in the previous chapter. We establish the structure of the optimal control law and derive algorithms for its computation. For finite time problems with linear and quadratic objective functions we show that the time varying feedback law is piecewise affine. We present several approaches for the computation of the optimal control law.

15.1 Problem Formulation

Consider the PWA system (14.1) subject to hard input and state constraints

$$Ex(t) + Lu(t) \leq M \tag{15.1}$$

for $t \geq 0$, and rewrite its restriction over the set of states and inputs defined by (15.1) as

$$x(t+1) = A^i x(t) + B^i u(t) + f^i \quad \text{if } \left[\begin{smallmatrix} x(t) \\ u(t) \end{smallmatrix}\right] \in \tilde{\mathcal{C}}^i \tag{15.2}$$

where $\{\tilde{\mathcal{C}}^i\}_{i=0}^{s-1}$ is the new polyhedral partition of the sets of state+input space $\mathbb{R}^{n+m}$ obtained by intersecting the sets $\mathcal{C}^i$ in (14.1) with the polyhedron described by (15.1). In this chapter we will assume that the sets $\mathcal{C}^i$ are polytopes.

Define the cost function

$$J_0(x(0), U_0) \triangleq p(x_N) + \sum_{k=0}^{N-1} q(x_k, u_k) \tag{15.3}$$

where x_k denotes the state vector at time k obtained by starting from the state $x_0 = x(0)$ and applying to the system model

$$x_{k+1} = A^i x_k + B^i u_k + f^i \quad \text{if } \left[\begin{smallmatrix} x_k \\ u_k \end{smallmatrix}\right] \in \tilde{\mathcal{C}}^i \tag{15.4}$$

the input sequence $U_0 \triangleq [u_0', \ldots, u_{N-1}']'$.

If the 1-norm or ∞-norm is used in the cost function (15.3), then we set $p(x_N) = \|Px_N\|_p$ and $q(x_k, u_k) = \|Qx_k\|_p + \|Ru_k\|_p$ with $p = 1$ or $p = \infty$ and P, Q, R full column rank matrices. Cost (15.3) is rewritten as

$$J_0(x(0), U_0) \triangleq \|Px_N\|_p + \sum_{k=0}^{N-1} \|Qx_k\|_p + \|Ru_k\|_p \tag{15.5}$$

If the squared euclidian norm is used in the cost function (15.3), then we set $p(x_N) = x_N' P x_N$ and $q(x_k, u_k) = x_k' Q x_k + u_k' R u_k$ with $P \succeq 0$, $Q \succeq 0$

and $R \succ 0$. Cost (15.3) is rewritten as

$$J_0(x(0),U_0) \triangleq x_N' P x_N + \sum_{k=0}^{N-1} x_k' Q x_k + u_k' R u_k \tag{15.6}$$

Consider the

Constrained Finite Time Optimal Control problem (CFTOC)

$$J_0^*(x(0)) \triangleq \min_{U_0} J_0(x(0),U_0) \tag{15.7a}$$

$$\text{subj. to } \begin{cases} x_{k+1} = A^i x_k + B^i u_k + f^i \ \text{ if } \left[\begin{smallmatrix} x_k \\ u_k \end{smallmatrix}\right] \in \tilde{\mathcal{C}}^i \\ x_N \in \mathcal{X}_f \\ x_0 = x(0) \end{cases} \tag{15.7b}$$

where the column vector $U_0 \triangleq [u_0', \ldots, u_{N-1}']' \in \mathbb{R}^{m_c N} \times \{0,1\}^{m_\ell N}$, is the optimization vector, N is the time horizon and $\mathcal{X}_f$ is the terminal region.

In general, the optimal control problem (15.3)-(15.7) may not have a minimizer for some feasible $x(0)$. This is caused by discontinuity of the PWA system in the input space. We will assume, however, that a minimizer ${U_0}^*(x(0))$ exists for all feasible $x(0)$. Also the optimizer function U_0^* may not be uniquely defined if the optimal set of problem (15.3)-(15.7) is not a singleton for some $x(0)$. In this case U_0^* denotes one of the optimal solutions.

We will denote by $\mathcal{X}_k \subseteq \mathbb{R}^{n_c} \times \{0,1\}^{n_\ell}$ the set of states x_k that are feasible for (15.3)-(15.7):

$$\begin{aligned} \mathcal{X}_k = & \left\{ x \in \mathbb{R}^{n_c} \times \{0,1\}^{n_\ell} \;\middle|\; \begin{array}{l} \exists u \in \mathbb{R}^{m_c} \times \{0,1\}^{m_\ell}, \\ \exists i \in \{1,\ldots,s\} \\ \left[\begin{smallmatrix} x \\ u \end{smallmatrix}\right] \in \tilde{\mathcal{C}}^i \text{ and} \\ A^i x + B^i u + f^i \in \mathcal{X}_{k+1} \end{array} \right\}, \\ & k = 0,\ldots,N-1, \\ \mathcal{X}_N = & \mathcal{X}_f. \end{aligned} \tag{15.8}$$

The definition of $\mathcal{X}_i$ requires that for any initial state $x_i \in \mathcal{X}_i$ there exists a feasible sequence of inputs $U_i \triangleq [u_i', \ldots, u_{N-1}']$ which keeps the state evolution in the feasible set $\mathcal{X}$ at future time instants $k = i+1, \ldots, N-1$ and forces x_N into $\mathcal{X}_f$ at time N. The sets $\mathcal{X}_k$ for $i = 0, \ldots, N$ play an important role in the the solution of the optimal control problem. They are independent of the cost function and of the algorithm used to compute the solution to problem (15.3)-(15.7). As in the case of linear systems (see Chapter 10.3) there are two ways to rigourously define and compute the sets $\mathcal{X}_i$: the *batch approach* and the *recursive approach*. In this chapter we will not discuss the details on the computation of $\mathcal{X}_i$. Also, we will not discuss invariant and reachable sets for hybrid systems. While the basic concepts are identical to the one presented in Section 10.1 for linear systems, the discussion of efficient algorithms requires a careful treatment which goes beyond the scope of this book. The interested reader is referred to the work in [122, 112] for a detailed discussion on reachable and invariant sets for hybrid systems.

In the following we need to distinguish between optimal control based on the squared 2-norm and optimal control based on the 1-norm or ∞-norm. Note that the results of this chapter also hold when the number of switches is weighted in the cost function (15.3), if this is meaningful in a particular situation.

In this chapter we will make use of the following definition. Consider system (15.2) and recall that, in general, $x = \left[\begin{smallmatrix} x_c \\ x_\ell \end{smallmatrix}\right]$ where $x_c \in \mathbb{R}^{n_c}$ are the continuous states and $x_\ell \in \mathbb{R}^{n_\ell}$ are the binary states and $u \in \mathbb{R}^{m_c} \times \{0,1\}^{m_\ell}$ where $u_c \in \mathbb{R}^{m_c}$ are the continuous inputs and $u_\ell \in \mathbb{R}^{m_\ell}$ are the binary inputs (Section 14.2.2). We will require the following assumption.

Assumption 15.1 *For the discrete-time PWA system (15.2) is, the mapping $(x_c(t), u_c(t)) \mapsto x_c(t+1)$ is continuous.*

Assumption 15.1 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 $\tilde{\mathcal{C}}^i$ without the need of introducing multi-valued state update equations. With abuse of notation in the next sections $\tilde{\mathcal{C}}^i$ will always denote the closure of $\tilde{\mathcal{C}}^i$. Discontinuous PWA systems will be discussed in Section 15.7.

15.2 Properties of the State Feedback Solution, 2-Norm Case

Theorem 15.1. *Consider the optimal control problem (15.7) with cost (15.6) and let Assumption 15.1 hold. Then, there exists a solution in the form of a PWA state-feedback control law*

$$u_k^*(x(k)) = F_k^i x(k) + G_k^i \quad \text{if } x(k) \in \mathcal{R}_k^i, \tag{15.9}$$

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

$$\begin{aligned}\bar{\mathcal{R}}_k^i \triangleq \{x: \ & x(k)' L_k^i(j) x(k) + M_k^i(j) x(k) \leq N_k^i(j)\ , \\ & j = 1, \ldots, n_k^i\}\ , \ \ k = 0, \ldots, N-1,\end{aligned} \tag{15.10}$$

and

$$\begin{aligned} x(k+1) = & A^i x(k) + B^i u_k^*(x(k)) + f^i \\ & \text{if } \begin{bmatrix} x(k) \\ u_k^*(x(k)) \end{bmatrix} \in \tilde{\mathcal{C}}^i, \ \ i = \{1, \ldots, s\}. \end{aligned} \tag{15.11}$$

Proof: The piecewise linearity of the solution was first mentioned by Sontag in [240]. In [184] Mayne sketched a proof. In the following we will give the proof for $u_0^*(x(0))$, the same arguments can be repeated for $u_1^*(x(1)), \dots, u_{N-1}^*(x(N-1))$.

- *Case 1: ($m_l = n_l = 0$) no binary inputs and states*
  Depending on the initial state $x(0)$ and on the input sequence $U = [u_0', \dots, u_k']$, the state x_k is either infeasible or it belongs to a certain polyhedron $\tilde{\mathcal{C}}^i$, $k = 0, \dots, N-1$. The number of all possible locations of the state sequence $x_0, \dots, x_{N-1}$ is equal to s^N. Denote by $\{v_i\}_{i=1}^{s^N}$ the set of all possible switching sequences over the horizon N , and by v_i^k the k-th element of the sequence v_i, i.e., $v_i^k = j$ if $x_k \in \tilde{\mathcal{C}}^j$.
  Fix a certain v_i and constrain the state to switch according to the sequence v_i. Problem (15.3)-(15.7) becomes

$$J_{v_i}^*(x(0)) \triangleq \min_{\{U_0\}} J_0(U_0, x(0)) \tag{15.12a}$$

$$\text{subj. to } \begin{cases} x_{k+1} = A^{v_i^k} x_k + B^{v_i^k} u_k + f^{v_i^k} \\ \left[\begin{smallmatrix} x_k \\ u_k \end{smallmatrix}\right] \in \tilde{\mathcal{C}}^{v_i^k} \\ \qquad k = 0, \dots, N-1 \\ x_N \in \mathcal{X}_f \\ x_0 = x(0) \end{cases} \tag{15.12b}$$

  Problem (15.12) 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 described in Chapter 10. The first move u_0 of its solution is the PPWA feedback control law

$$u_0^i(x(0)) = \tilde{F}^{i,j} x(0) + \tilde{G}^{i,j}, \quad \forall x(0) \in \mathcal{T}^{i,j}, \quad j = 1, \dots, N^{ri} \tag{15.13}$$

  where $\mathcal{D}^i = \bigcup_{j=1}^{N^{ri}} \mathcal{T}^{i,j}$ is a polyhedral partition of the convex set $\mathcal{D}^i$ of feasible states $x(0)$ for problem (15.12). N^{ri} is the number of regions of the polyhedral partition of the solution and it is a function of the number of constraints in problem (15.12). The upper index i in (15.13) denotes that the input $u_0^i(x(0))$ is optimal when the switching sequence v_i is fixed.
  The set $\mathcal{X}_0$ of all feasible states at time 0 is $\mathcal{X}_0 = \bigcup_{i=1}^{s^N} \mathcal{D}^i$ and in general it is not convex. Indeed, as some initial states can be feasible for different switching sequences, the sets $\mathcal{D}^i$, $i = 1, \dots, s^N$, in general, can overlap. The solution $u_0^*(x(0))$ to the original problem (15.3)-(15.7) can be computed in the following way. For every polyhedron $\mathcal{T}^{i,j}$ in (15.13),

1. If $\mathcal{T}^{i,j} \cap \mathcal{T}^{l,m} = \emptyset$ for all $l \neq i$, $l = 1, \ldots, s^N$, and for all $m \neq j$, $m = 1, \ldots, N^{rl}$, then the switching sequence v_i is the only feasible one for all the states belonging to $\mathcal{T}^{i,j}$ and therefore the optimal solution is given by (15.13), i.e.

$$u_0^*(x(0)) = \tilde{F}^{i,j} x(0) + \tilde{G}^{i,j}, \quad \forall x \in \mathcal{T}^{i,j}. \tag{15.14}$$

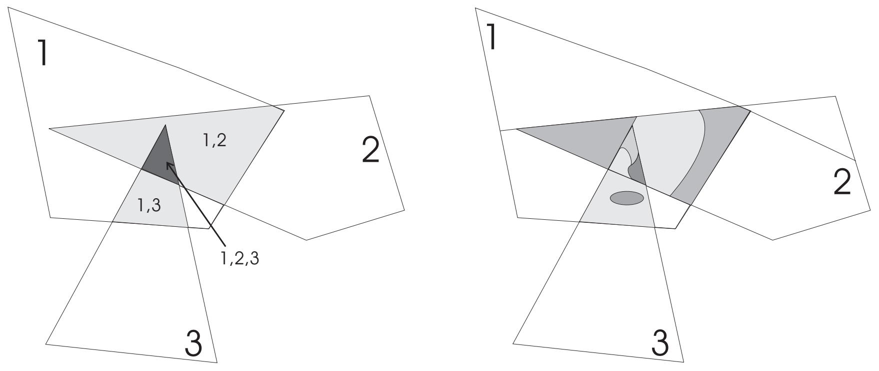
2. If $\mathcal{T}^{i,j}$ intersects one or more polyhedra $\mathcal{T}^{l_1,m_1}$,$\mathcal{T}^{l_2,m_2}$, …, the states belonging to the intersection are feasible for more than one switching sequence v_i, v_{l_1}, v_{l_2}, … and therefore the corresponding value functions $J^*_{v_i}$, $J^*_{v_{l_1}}$,$J^*_{v_{l_2}}$, … in (15.12a) have to be compared in order to compute the optimal control law.
Consider the simple case when only two polyhedra overlap, i.e. $\mathcal{T}^{i,j} \cap \mathcal{T}^{l,m} \triangleq \mathcal{T}^{(i,j),(l,m)} \neq \emptyset$. We will refer to $\mathcal{T}^{(i,j),(l,m)}$ as a *double feasibility polyhedron*. For all states belonging to $\mathcal{T}^{(i,j),(l,m)}$ the optimal solution is:

$$u_0^*(x(0)) = \begin{cases} \tilde{F}^{i,j} x(0) + \tilde{G}^{i,j}, & \begin{array}{l}\forall x(0) \in \mathcal{T}^{(i,j),(l,m)} : \\ J^*_{v_i}(x(0)) < J^*_{v_l}(x(0))\end{array} \\ \tilde{F}^{l,m} x(0) + \tilde{G}^{l,m}, & \begin{array}{l}\forall x(0) \in \mathcal{T}^{(i,j),(l,m)} : \\ J^*_{v_i}(x(0)) > J^*_{v_l}(x(0))\end{array} \\ \begin{cases} \tilde{F}^{i,j} x(0) + \tilde{G}^{i,j} \text{ or} \\ \tilde{F}^{l,m} x(0) + \tilde{G}^{l,m} \end{cases} & \begin{array}{l}\forall x(0) \in \mathcal{T}^{(i,j),(l,m)} : \\ \\ J^*_{v_i}(x(0)) = J^*_{v_l}(x(0))\end{array} \end{cases} \tag{15.15}$$

Because $J^*_{v_i}$ and $J^*_{v_l}$ are quadratic functions of $x(0)$ on $\mathcal{T}^{i,j}$ and $\mathcal{T}^{l,m}$ respectively, we find the expression (15.10) of the control law domain. The sets $\mathcal{T}^{i,j} \setminus \mathcal{T}^{l,m}$ and $\mathcal{T}^{l,m} \setminus \mathcal{T}^{i,j}$ are two *single feasibility non-Euclidean polyhedra* which can be partitioned into a set of *single feasibility polyhedra*, and thus be described through (15.10) with $L^i_k = 0$.

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, …, $(n-1)^{th}$-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 (15.10) where in each one of them a certain feasible value function is greater than all the others. The procedure is depicted in Figure 15.1 when $n = 3$.

- *Case 2: binary inputs, $m_\ell \neq 0$*
The proof can be repeated in the presence of binary inputs, $m_\ell \neq 0$. In this case the switching sequences v_i are given by all combinations of region indices and *binary inputs*, i.e., $i = 1, \ldots, (s \cdot m_\ell)^N$. The continuous component of the optimal input is given by (15.14) or (15.15). Such an optimal continuous component of the input has an associated optimal se-



(a) Step 1 (b) Step 2 and Step 3

Fig. 15.1 Graphical illustration of the main steps for the proof of Theorem 15.1 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 neither open nor closed 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.

quence v_i, whose components provide the remaining binary components of the optimal input.

- *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.

$\square$

From the result of the theorem above one immediately concludes that the value function J_0^* is piecewise quadratic:

$$J_0^*(x(0)) = x(0)' H_1^i x(0) + H_2^i x(0) + H_3^i \text{ if } x(0) \in \mathcal{R}_0^i, \tag{15.16}$$

The proof of Theorem 15.1 gives useful insights into the properties of the sets $\mathcal{R}_k^i$ in (15.10). We will summarize them next.

Each set $\mathcal{R}_k^i$ has an associated multiplicity j which means that j switching sequences are feasible for problem (15.3)-(15.7) starting from a state $x(k) \in \mathcal{R}_k^i$. If $j = 1$, then $\mathcal{R}_k^i$ is a polyhedron. In general, if $j > 1$ the boundaries of $\mathcal{R}_k^i$ 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 across 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. Across 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 15.1. *The value function J_k^**

1. *is a quadratic function of the states inside each $\mathcal{R}_k^i$*
2. *is continuous on quadratic and affine boundaries of type **b** and **c***
3. *may be discontinuous on affine boundaries of type **a**,*

and the optimizer u_k^*

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

Based on Proposition 15.1 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 15.1. *J_0^* is a lower-semicontinuous PWQ function on $\mathcal{X}_0$.*

Proof: The proof follows from the result on the minimization of lower-semicontinuous point-to-set maps in [43]. 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 15.1 it follows that the feasible switching sequences for a given state $x(0)$ are all the feasible switching sequences associated with any set $\mathcal{R}_0^j$ whose closure $\bar{\mathcal{R}}_0^j$ contains $x(0)$. Consider a state $x(0)$ belonging to boundaries of type **a** and the proof of Theorem 15.1. The only case of discontinuity can occur when *(i)* a j-ple feasible set $\mathcal{P}_1$ intersects an i-ple feasible set $\mathcal{P}_2$ with $i<j$, *(ii)* there exists a point $x(0)\in\mathcal{P}_1$, $\mathcal{P}_2$ and a neighborhood $\mathcal{N}(x(0))$ with $x,y\in\mathcal{N}(x(0))$, $x\in\mathcal{P}_1$, $x\notin\mathcal{P}_2$ and $y\in\mathcal{P}_2$, $y\notin\mathcal{P}_1$. The proof follows from the previous statements and the fact that $J_0^*(x(0))$ is the minimum of all $J_{v_i}^*(x(0))$ for all feasible switching sequences v_i. $\square$

The result of Corollary 15.1 will be used extensively in the next sections. Even if value function and optimizer are discontinuous, one can work with the closure $\bar{\mathcal{R}}_k^j$ of the original sets $\mathcal{R}_k^j$ without explicitly considering their boundaries. In fact, if a given state $x(0)$ belongs to several regions $\bar{\mathcal{R}}_0^1,\dots,\bar{\mathcal{R}}_0^p$, then the minimum value among the optimal values (15.16) associated with each region $\bar{\mathcal{R}}_0^1,\dots,\bar{\mathcal{R}}_0^p$ allows us to identify the region of the set $\mathcal{R}_0^1,\dots,\mathcal{R}_0^p$ 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 15.2. *Assume that the PWA system (15.2) is continuous, and that $E = 0$ in (15.1) and $\mathcal{X}_f = \mathbb{R}^n$ in (15.7) (which means that there are no state constraints, i.e., $\tilde{P}$ is unbounded in the x-space). Then, the value function J_0^* in (15.7) is continuous.*

Proof: Problem (15.3)-(15.7) 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 $u_0,\dots,u_{k-1}$, i.e., $x_k = f_{PWA}((\cdots(f_{PWA}(x_0,u_0),u_1),\dots,u_{k-2}),u_{k-1})$. J_0 in (15.3) will be a continuous function of x_0 and $u_0,\dots,u_{N-1}$ since it is the composition of continuous functions. By assumptions the input constraints on $u_0,\dots,u_{N-1}$ are convex and the resulting set is compact. The proof follows from the continuity of J and Theorem 5.2.

□

Note that $E = 0$ is a sufficient condition for ensuring that constraints (15.1) are convex in the optimization variables $u_0,\dots,u_n$. 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_0^* to be continuous can be found in [99]. In general, they require the convexity (or a relaxed form of it) of the cost $J_0(U_0, x(0))$ in U_0 for each $x(0)$ and the convexity of the constraints in (15.7) in U_0 for each $x(0)$. Such conditions are clearly very restrictive since the cost and the constraints in problem (15.7) 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 (15.3)-(15.7) is a PPWA state-feedback control law.

Theorem 15.2. *Assume that the optimizer ${U_0}^*(x(0))$ of (15.3)-(15.7) is unique for all $x(0)$. Then the solution to the optimal control problem (15.3)-(15.7) is a PPWA state feedback control law of the form*

$$u_k^*(x(k)) = F_k^i x(k) + G_k^i \quad \text{if } x(k) \in \mathcal{P}_k^i \quad k = 0,\dots,N-1, \tag{15.17}$$

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

Proof: : In Proposition 15.1 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 15.1 implies that $\tilde{F}^{i,j}x(0) + \tilde{G}^{i,j} = \tilde{F}^{l,m}x(0) + \tilde{G}^{l,m}$, $\forall x(0)$ 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 $u_k^*(x(k))$, $k = 1, \ldots, N-1$. $\square$

Remark 15.1. Theorem 15.2 relies on a rather strong uniqueness assumption. Sometimes, problem (15.3)-(15.7) can be modified in order to obtain uniqueness of the solution and use the result of Theorem 15.2 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.

Example 15.1. Consider the following simple system

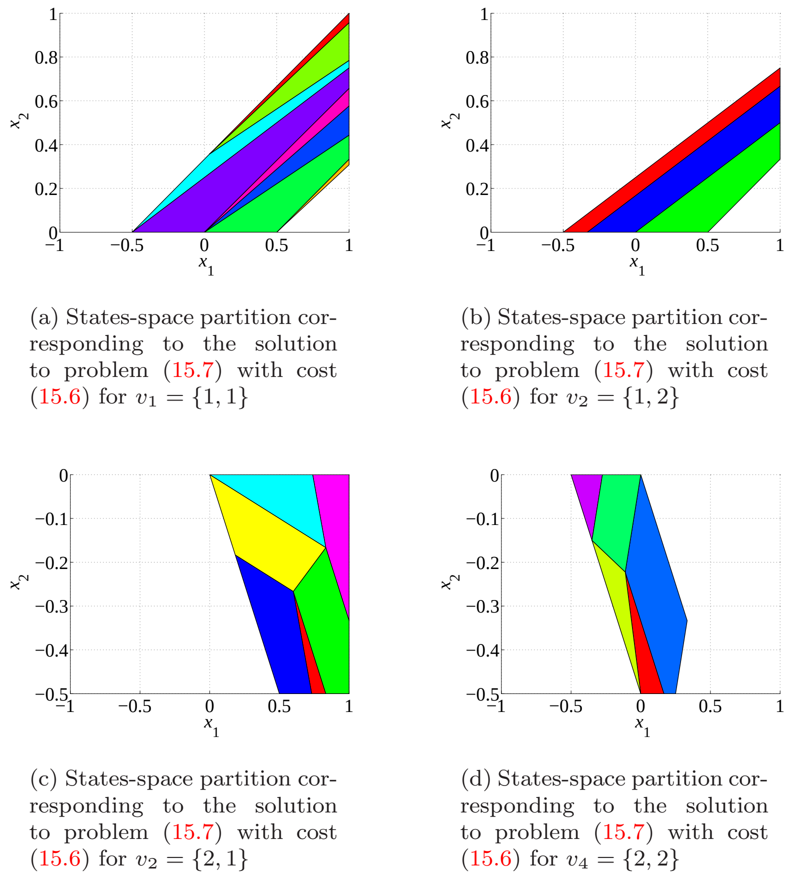
$$\begin{cases} x(t+1) = \begin{cases} \begin{bmatrix} -1 & 2 \\ 2 & -3 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t) \text{ if } x(t) \in \mathcal{C}^1 = \{x : [0\ 1]x \geq 0\} \\ \begin{bmatrix} 1 & -2 \\ 1 & 1 \end{bmatrix} x(t) + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u(t) \text{ if } x(t) \in \mathcal{C}_2 = \{x : [0\ 1]x < 0\} \end{cases} \\ \quad x(t) \in [-1, -0.5] \times [1, 1] \\ \quad u(t) \in [-1, 1] \end{cases} \tag{15.18}$$

and the optimal control problem (15.7) with cost (15.6), $N = 2$, $Q = \begin{bmatrix} 1 & 1 \\ 1 & 15 \end{bmatrix}$, $R = 10$, $P = Q$, $\mathcal{X}_f = \mathcal{X}$.

The possible switching sequences are $v_1 = \{1, 1\}$, $v_2 = \{1, 2\}$, $v_3 = \{2, 1\}$, $v_4 = \{2, 2\}$. The solution to problem (15.12) is depicted in Figure (15.2). In Figure 15.3(a) the four solutions are intersected, the white region corresponds to polyhedra of multiple feasibility. In Figure 15.3(b) we plot with different colors the regions of the state-space partition where the switching sequences $v_1 = \{1, 1\}$, $v_2 = \{1, 2\}$, $v_3 = \{2, 1\}$, $v_4 = \{2, 2\}$ are optimal.

15.3 Properties of the State Feedback Solution, $1, \infty$-Norm Case

The results of the previous section can be extended to piecewise linear cost functions, i.e., cost functions based on the 1-norm or the ∞-norm.



(a) States-space partition corresponding to the solution to problem (15.7) with cost (15.6) for $v_1 = \{1, 1\}$

(b) States-space partition corresponding to the solution to problem (15.7) with cost (15.6) for $v_2 = \{1, 2\}$

(c) States-space partition corresponding to the solution to problem (15.7) with cost (15.6) for $v_2 = \{2, 1\}$

(d) States-space partition corresponding to the solution to problem (15.7) with cost (15.6) for $v_4 = \{2, 2\}$

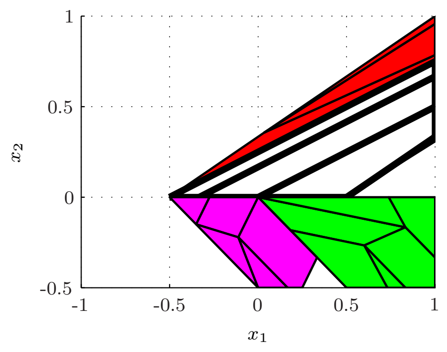
Fig. 15.2 First step for the solution of Example 15.1. Problem (15.12) with cost (15.6) is solved for different v_i, $i = 1, \ldots, 4$

Theorem 15.3. *Consider the optimal control problem (15.7) with cost (15.5), $p = 1, \infty$ and let Assumption 15.1 hold. Then there exists a solution in the form of a PPWA state-feedback control law*

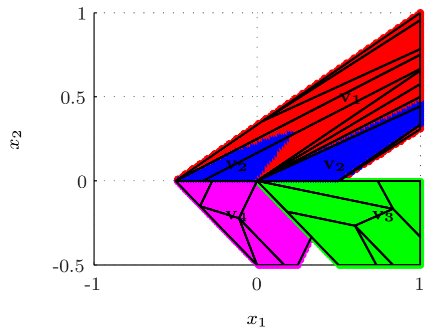
$$u_k^*(x(k)) = F_k^i x(k) + G_k^i \quad \textit{if } x(k) \in \mathcal{P}_k^i, \tag{15.19}$$

where $\mathcal{P}_k^i$, $i = 1, \ldots, N_k^r$ 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 15.1. Fix a certain switching sequence v_i, consider the problem (15.3)-(15.7) and constrain the state to switch according to the sequence v_i to obtain problem (15.12). Problem (15.12) can be viewed as a finite time optimal control problem with a



(a) Feasibility domain corresponding to the solution of Example 15.1 obtained by joining the solutions plotted in Figure 15.2. The white region corresponds to polyhedra of multiple feasibility.



(b) Regions of the state-space partition where the switching sequences $v_1 = \{1,1\}$, $v_2 = \{1,2\}$, $v_3 = \{2,1\}$, $v_4 = \{2,2\}$ are optimal.

Fig. 15.3 State-space partition corresponding to the optimal control law of Example 15.1

performance index based on 1-norm or ∞-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 Chapter 10.5. Its solution is a PPWA feedback control law

$$u_0^i(x(0)) = \tilde{F}^{i,j}x(0) + \tilde{G}^{i,j}, \quad \forall x \in \mathcal{T}^{i,j}, \quad j = 1, \dots, N^{ri}, \tag{15.20}$$

and the value function $J_{v_i}^*$ is piecewise affine on polyhedra and convex. The rest of the proof follows the proof of Theorem 15.1. Note that in this case the value functions to be compared are piecewise affine and not piecewise quadratic. □

15.4 Computation of the Optimal Control Input via Mixed Integer Programming

In the previous section the properties enjoyed by the solution to hybrid optimal control problems were investigated. Despite the fact that the proofs are constructive (as shown in the figures), they are 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 (15.17)), more efficient methods than enumeration are desirable. Here we show that the MLD framework can be used to avoid the complete enumeration. In fact, when the model of the system is an MLD model and the performance index is quadratic, the optimization problem can be cast as a Mixed-Integer Quadratic Program (MIQP). Similarly, 1-norm and ∞-norm performance indices lead to Mixed-Integer Linear Programs (MILPs) problems. In the following we detail the translation of problem (15.7) with cost (15.5) or (15.6) into a mixed integer linear or quadratic program, respectively, for which efficient branch and bound algorithms exist.

Consider the equivalent MLD representation (14.25) of the PWA system (15.2). Problem (15.7) is rewritten as:

$$J_0^*(x(0)) = \min_{U_0} J_0(x(0), U_0) \tag{15.21a}$$

$$\text{subj. to} \quad \begin{cases} x_{k+1} = Ax_k + B_1 u_k + B_2 \delta_k + B_3 z_k \\ y(t) = Cx(t) + D_1 u(t) + D_2 \delta(t) + D_3 z(t) + D_5 \\ E_2 \delta_k + E_3 z_k \leq E_1 u_k + E_4 x_k + E_5 \\ x_N \in \mathcal{X}_f \\ x_0 = x(0) \end{cases} \tag{15.21b}$$

Note that the cost function (15.21a) is of the form

$$J_0(x(0), U_0) \triangleq \|Px_N\|_p + \sum_{k=0}^{N-1} \|Qx_k\|_p + \|Ru_k\|_p + \|Q_\delta \delta_k\|_p + \|Q_z z_k\|_p \tag{15.22}$$

when $p = 1$ or $p = \infty$ or

$$J_0(x(0), U_0) \triangleq x_N' P x_N + \sum_{k=0}^{N-1} x_k' Q x_k + u_k' R u_k + \delta_k' Q_\delta \delta_k + z_k' Q_z z_k \tag{15.23}$$

when p=2.

The optimal control problem (15.21) with the cost (15.22) can be formulated as a *Mixed Integer Linear Program* (MILP). The optimal control problem (15.21) with the cost (15.23) can be formulated as a *Mixed Integer Quadratic Program* (MIQP). The compact form for both cases is

$$\begin{array}{ll} \min\limits_{\varepsilon} & \varepsilon' H_1 \varepsilon + \varepsilon' H_2 x(0) + x(0)' H_3 x(0) + c_1' \varepsilon + c_2' x(0) + c \\ \text{subj. to } G\varepsilon \leq W + Sx(0) & \end{array} \tag{15.24}$$

where H_1, H_2, H_3, c_1, c_2, G, W, S are matrices of suitable dimensions, $\varepsilon = [\varepsilon_c', \varepsilon_d']$ where ε_c, ε_d represent continuous and discrete variables, respectively and H_1, H_2, H_3, are null matrices if problem (15.24) is an MILP.

The translation of (15.21) with cost (15.23) into (15.24) is simply obtained by substituting the state update equation

$$x_k = A^k x_0 + \sum_{j=0}^{k-1} A^j(B_1 u_{k-1-j} + B_2\delta_{k-1-j} + B_3 z_{k-1-j}) \tag{15.25}$$

The optimization vector ε in (15.24) is $\varepsilon = \{u_0, \dots, u_{N-1}, \delta_0, \dots, \delta_{N-1}, z_0, \dots, z_{N-1}\}$.

The translation of (15.21) with cost (15.22) into (15.24) requires the introductions of slack variables as shown in Section 10.5 for the case of linear systems. In particular, for $p = \infty$, $Q_z = 0$ and $Q_\delta = 0$, the sum of the components of any vector $\{\varepsilon_0^u, \dots, \varepsilon_{N-1}^u, \varepsilon_0^x, \dots, \varepsilon_N^x\}$ that satisfies

$$\begin{array}{l} -\mathbf{1}_m\varepsilon_k^u \le Ru_k,\ k = 0,1,\dots,N-1 \\ -\mathbf{1}_m\varepsilon_k^u \le -Ru_k,\ k = 0,1,\dots,N-1 \\ -\mathbf{1}_n\varepsilon_k^x \le Qx_k,\ k = 0,1,\dots,N-1 \\ -\mathbf{1}_n\varepsilon_k^x \le -Qx_k,\ k = 0,1,\dots,N-1 \\ -\mathbf{1}_n\varepsilon_N^x \le Px_N, \\ -\mathbf{1}_n\varepsilon_N^x \le Px_N, \end{array} \tag{15.26}$$

represents an upper bound on $J_0^*(x(0))$, where $\mathbf{1}_k$ is a column vector of ones of length k, and where $x(k)$ is expressed as in (15.25). Similarly to what was shown in [71], it is easy to prove that the vector $\varepsilon \triangleq \{\varepsilon_0^u, \dots, \varepsilon_{N-1}^u, \varepsilon_0^x, \dots, \varepsilon_N^x, u(0), \dots, u(N-1)\}$ that satisfies equations (15.26) and simultaneously minimizes

$$J(\varepsilon) = \varepsilon_0^u + \dots + \varepsilon_{N-1}^u + \varepsilon_0^x + \dots + \varepsilon_N^x \tag{15.27}$$

also solves the original problem, i.e., the same optimum $J_0^*(x(0))$ is achieved. Therefore, problem (15.21) with cost (15.22) can be reformulated as the following MILP problem

$$\begin{array}{rl} \min\limits_{\varepsilon} & J(\varepsilon) \\ \text{subj. to} & -\mathbf{1}_m\varepsilon_k^u \le \pm Ru_k,\ k = 0,1,\dots,N-1 \\ & -\mathbf{1}_n\varepsilon_k^x \le \pm Q(A^k x_0 + \sum_{j=0}^{k-1} A^j(B_1 u_{k-1-j} + \\ & \qquad B_2\delta_{k-1-j} + B_3 z_{k-1-j}))\ k = 0,\dots,N-1 \\ & -\mathbf{1}_n\varepsilon_N^x \le \pm P(A^N x_0 + \sum_{j=0}^{N-1} A^j(B_1 u_{k-1-j} + \\ & \qquad B_2\delta_{k-1-j} + B_3 z_{k-1-j})) \\ & x_{k+1} = Ax_k + B_1 u_k + B_2\delta_k + B_3 z_k,\ k \ge 0 \\ & E_2\delta_k + E_3 z_k \le E_1 u_k + E_4 x_k + E_5,\ k \ge 0 \\ & x_N \in \mathcal{X}_f \\ & x_0 = x(0) \end{array} \tag{15.28}$$

where the variable $x(0)$ in (15.28) appears only in the constraints as a parameter vector.

Given a value of the initial state $x(0)$, the MIQP (15.24) or the MILP (15.28) can be solved to get the optimizer $\varepsilon^*(x(0))$ and therefore the optimal input $U_0^*(0)$. In the next Section 15.5 we will show how multiparametric programming can be used to efficiently compute the piecewise affine state feedback optimal control law (15.9) or (15.19).

Example 15.2. Consider the problem of steering in three steps the simple piecewise affine system presented in Example 14.1 to a small region around the origin. The system state update equations are

$$\begin{cases} x(t+1) = \begin{cases} 0.4 \begin{bmatrix} 1 & -\sqrt{3} \\ \sqrt{3} & 1 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t) \text{ if } \begin{bmatrix} 1 & 0 \end{bmatrix} x(t) \geq 0 \\ 0.4 \begin{bmatrix} 1 & \sqrt{3} \\ -\sqrt{3} & 1 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t) \text{ if } \begin{bmatrix} 1 & 0 \end{bmatrix} x(t) \leq -\epsilon \end{cases} \\ y(t) = \begin{bmatrix} 0 & 1 \end{bmatrix} x(t) \end{cases} \tag{15.29}$$

subject to the constraints

$$\begin{array}{l} x(t) \in [-5,5] \times [-5,5] \\ u(t) \in [-1,1] \end{array} \tag{15.30}$$

The MLD representation of system (15.29)-(15.30) was reported in Example 14.7.

The finite time constrained optimal control problem (15.7) with cost (15.5) with $p = \infty$, $N = 3$, $P = Q = \left[\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right]$, $R = 1$, and $\mathcal{X}_f = [-0.01,\ 0.01] \times [-0.01,\ 0.01]$, can be solved by considering the optimal control problem (15.21) with cost (15.22) for the equivalent MLD representation and solving the associated MILP problem (15.24).

The resulting MILP has 27 variables ($|\varepsilon| = 27$) which are $x \in \mathbb{R}^2$, $\delta \in \{0,1\}$, $z \in \mathbb{R}^2$, $u \in \mathbb{R}$, $y \in \mathbb{R}$ over 3 steps plus the real-valued slack variables ε_0^u, ε_1^u, ε_2^u and ε_1^x, ε_2^x, ε_3^x (note that ε_0^x is not needed). The number of mixed-integer equality constraints is 9 resulting from the 3 equality constraints of the MLD systems over 3 steps. The number of mixed-integer inequality constraints is 110.

The finite time constrained optimal control problem (15.7) with cost (15.6), $N = 3$, $P = Q = \left[\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right]$, $R = 1$, and $\mathcal{X}_f = [-0.01\ 0.01] \times [-0.01\ 0.01]$, can be solved by considering the optimal control problem (15.21) with cost (15.23) for the equivalent MLD representation and solving the associated MIQP problem (15.24).

The resulting MIQP has 21 variables ($|\varepsilon| = 21$) which are $x \in \mathbb{R}^2$, $\delta \in \{0,1\}$, $z \in \mathbb{R}^2$, $u \in \mathbb{R}$, $y \in \mathbb{R}$ over 3 steps. The number of mixed-integer equality constraints is 9 resulting from the 3 equality constraints of the MLD systems over 3 steps. The number of mixed-integer inequality constraints is 74.

15.5 State Feedback Solution via Batch Approach

Multiparametric programming [106, 91, 39, 56] can be used to efficiently compute the PWA form of the optimal state feedback control law $u^*(x(k))$.

By generalizing the results for linear systems of previous chapters to hybrid systems, the state vector $x(0)$, which appears in the objective function and in the linear part of the right-hand-side of the constraints (15.24), can be considered as a vector of parameters. Then, for performance indices based on the ∞-norm or 1-norm, the optimization problem can be treated as a *multiparametric MILP* (mp-MILP), while for performance indices based on the 2-norm, the optimization problem can be treated as a *multiparametric MIQP* (mp-MIQP). Solving an mp-MILP (mp-MIQP) amounts to expressing the solution of the MILP (MIQP) (15.24) as a function of the parameters $x(0)$.

In Section 6.4.1 we have presented an algorithm for solving mp-MILP problems, while, to the authors' knowledge, there does not exist an efficient method for solving general mp-MIQPs. In Section 15.6 we will present an algorithm that efficiently solves the specific mp-MIQPs derived from optimal control problems for discrete-time hybrid systems.

15.6 State Feedback Solution via Recursive Approach

In this section we propose an efficient algorithm for computing the solution to the finite time optimal control problem for discrete-time linear hybrid systems. It is based on a dynamic programming recursion and a multiparametric linear or quadratic programming solver. The approach represents an alternative to the mixed-integer parametric approach presented in the previous section.

The PWA solution (15.9) 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.

15.6.1 Preliminaries and Basic Steps

Consider the PWA map ζ defined as

$$\zeta: \ x \in \mathcal{R}_i \mapsto F_i x + G_i \quad \text{for} \quad i = 1, \dots, N_{\mathcal{R}}, \tag{15.31}$$

where $\mathcal{R}_i, \ i = 1, \dots, N_{\mathcal{R}}$ are subsets of the $x-$space. Note that if there exist $l, m \in \{1, \dots, N_{\mathcal{R}}\}$ 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 ζ (15.31) is not single valued.

Definition 15.1. Given a PWA map (15.31) we define $f_{PWA}(x) = \zeta_o(x)$ as the *ordered region single-valued* function associated with (15.31) when

$$\begin{aligned}\zeta_o(x) = F_j x + G_j \mid x \in \mathcal{R}_j \text{ and } \forall i < j : x \notin \mathcal{R}_i,\\ j \in \{1, \ldots, N_{\mathcal{R}}\},\end{aligned}$$

and write it in the following form

$$\zeta_o(x) = \left| \begin{array}{ll} F_1 x + G_1 & \text{if } \; x \in \mathcal{P}_1 \\ & \vdots \\ F_{N_{\mathcal{R}}} x + G_{N_{\mathcal{R}}} & \text{if } \; x \in \mathcal{P}_{N_{\mathcal{R}}}. \end{array} \right. \downarrow$$

Note that given a PWA map (15.31) the corresponding *ordered region single-valued* function ζ_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 15.4, where $x \in \mathbb{R}$, $N_{\mathcal{R}} = 2$, $F_1 = 0$, $G_1 = 0$, $\mathcal{R}_1 = [-2, 1]$, $F_2 = 1$, $G_2 = 0$, $\mathcal{R}_2 = [0, 2]$.

In the following, we assume that the sets $\mathcal{R}_i^k$ in the optimal solution (15.9) can overlap. When we refer to the PWA function $u_k^*(x(k))$ in (15.9) we will implicitly mean the ordered region single-valued function associated with the map (15.9).

Example 15.3. Let $J_1^* : \mathcal{P}_1 \to \mathbb{R}$ and $J_2^* : \mathcal{P}_2 \to \mathbb{R}$ be two quadratic functions, $J_1^*(x) \triangleq x' L_1 x + M_1 x + N_1$ and $J_2^*(x) \triangleq x' L_2 x + M_2 x + N_2$, 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 $\mathcal{P}_1 \cap \mathcal{P}_2 \triangleq \mathcal{P}_3 \neq \emptyset$ and define

$$J^*(x) \triangleq \min\{J_1^*(x), J_2^*(x)\} \tag{15.32}$$

$$u^*(x) \triangleq \begin{cases} u_1^*(x) & \text{if } J_1^*(x) \leq J_2^*(x) \\ u_2^*(x) & \text{if } J_1^*(x) \geq J_2^*(x) \end{cases} \tag{15.33}$$

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

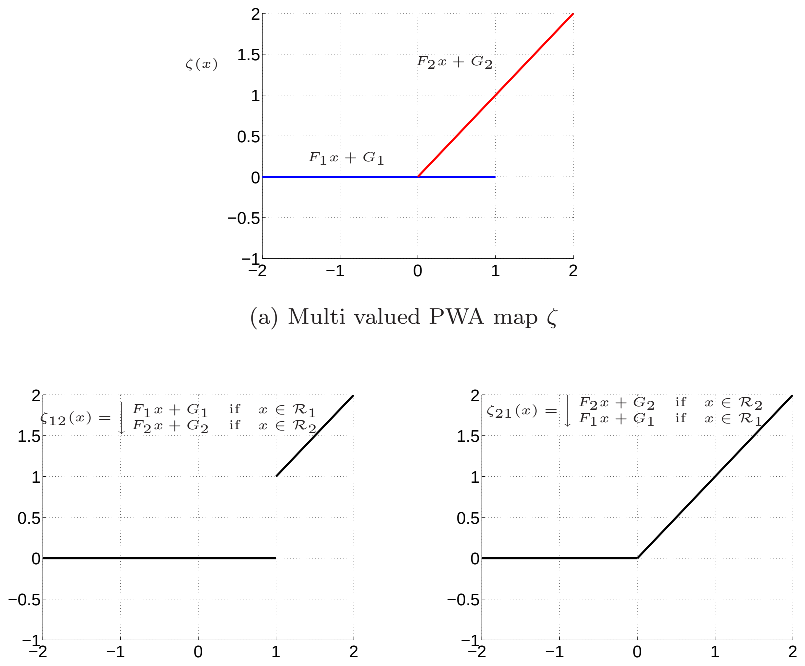
(i) $J_1^*(x) \leq J_2^*(x) \;\forall x \in \mathcal{P}_3$
(ii) $J_1^*(x) \geq J_2^*(x) \;\forall x \in \mathcal{P}_3$
(iii) $\exists x_1, x_2 \in \mathcal{P}_3 | J_1^*(x_1) < J_2^*(x_1)$ and $J_1^*(x_2) > J_2^*(x_2)$

the expressions (15.32) and a real-valued function that can be extracted from (15.33) can be written equivalently as:

(i)

$$J^*(x) = \left| \begin{array}{ll} J_1^*(x) & \text{if } x \in \mathcal{P}_1 \\ J_2^*(x) & \text{if } x \in \mathcal{P}_2 \end{array} \right. \downarrow \tag{15.34}$$

$$u^*(x) = \left| \begin{array}{ll} u_1^*(x) & \text{if } x \in \mathcal{P}_1 \\ u_2^*(x) & \text{if } x \in \mathcal{P}_2 \end{array} \right. \downarrow \tag{15.35}$$



(a) Multi valued PWA map ζ

(b) Ordered region single valued function ζ_{12}

(c) Ordered region single valued function ζ_{21}

Fig. 15.4 Illustration of the ordered region single valued function.

(ii) as in (15.34) and (15.35) by switching the indices 1 and 2
(iii)

$$J^*(x) = \begin{cases} \min\{J_1^*(x), J_2^*(x)\} & \text{if } x \in \mathcal{P}_3 \\ J_1^*(x) & \text{if } x \in \mathcal{P}_1 \\ J_2^*(x) & \text{if } x \in \mathcal{P}_2 \end{cases} \tag{15.36}$$

$$u^*(x) = \begin{cases} u_1^*(x) & \text{if } x \in \mathcal{P}_3 \bigcap \{x \mid x'L_3x + M_3x + N_3 \geq 0\} \\ u_2^*(x) & \text{if } x \in \mathcal{P}_3 \bigcap \{x \mid x'L_3x + M_3x + N_3 \leq 0\} \\ u_1^*(x) & \text{if } x \in \mathcal{P}_1 \\ u_2^*(x) & \text{if } x \in \mathcal{P}_2 \end{cases} \tag{15.37}$$

where (15.34), (15.35), (15.36), and (15.37) have to be considered as PWA and PPWQ functions in the *ordered region* sense.

Example 15.3 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 points listed above will be the three basic ingredients for storing and simplifying the optimal control law (15.9). Next we will show how to compute it.

Remark 15.2. In Example 15.3 the description (15.36)-(15.37) of Case (iii) can always be used but the on-line evaluation of the control $u^*(x)$ is rather involved requiring a series of set membership and comparison evaluations. To assess if the simpler description of $u^*(x)$ of Case (i) or Case (ii) could be used instead, one needs to solve indefinite quadratic programs of the form

$$\begin{array}{ll} \min_x & x'L_3x + M_3x + N_3 \\ \text{subj. to} & x \in \mathcal{P}_3 \end{array} \tag{15.38}$$

which are nontrivial, in general.

15.6.2 Multiparametric Programming with Multiple Quadratic Functions

Consider the multi-parametric program

$$\begin{array}{rl} J^*(x) \triangleq & \min_u \; l(x,u) + q(f(x,u)) \\ & \text{s.t.} \quad f(x,u) \in \mathcal{P}, \end{array} \tag{15.39}$$

where $\mathcal{P} \subseteq \mathbb{R}^n$ is a compact set, $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$, $q : \mathcal{P} \to \mathbb{R}$, and $l : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ is a convex quadratic function of x and u. We aim at determining the region $\mathcal{X}$ of variables x such that the program (15.39) 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 (15.39).

Lemma 15.1 (one to one problem). *Problem (15.39) 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 Chapter 6.3.1.

Lemma 15.2 (one to one problem of multiplicity d). *Problem (15.39) 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

$$\begin{array}{rll} J^*(x) = \min_u & \{l(x,u) + \min\{q_1(f(x,u)), \ldots, q_d(f(x,u))\}\} \\ & \text{subj. to } \ f(x,u) \in \mathcal{P} \end{array} \tag{15.40}$$

and it is equivalent to

$$J^*(x) = \min \left\{ \begin{array}{l} \min_u l(x,u) + q_1(f(x,u)), \\ \text{subj. to } f(x,u) \in \mathcal{P}, \\ \vdots, \\ \min_u l(x,u) + q_d(f(x,u))\} \\ \text{subj. to } f(x,u) \in \mathcal{P} \end{array} \right\}. \tag{15.41}$$

The i^{th} sub-problems in (15.41)

$$J_i^*(x) \triangleq \min_u l(x,u) + q_i(f(x,u)) \tag{15.42}$$

$$\text{subj. to } f(x,u) \in \mathcal{P} \tag{15.43}$$

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

$$u^i(x) = \tilde{F}^{i,j} x + \tilde{G}^{i,j}, \quad \forall x \in \mathcal{T}^{i,j}, \quad j = 1, \ldots, N^{ri}, \tag{15.44}$$

where $\mathcal{T}^i = \bigcup_{j=1}^{N^{ri}} \mathcal{T}^{i,j}$ is a polyhedral partition of the convex set $\mathcal{T}^i$ of feasible x for the ith sub-problem and N^{ri} 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 (15.40) is obtained by comparing and storing the solution of d mp-QP subproblems (15.42)-(15.43) as explained in Example 15.3. Consider the case $d = 2$, and consider the intersection of the polyhedra $\mathcal{T}^{1,i}$ and $\mathcal{T}^{2,l}$ for $i = 1, \ldots, N^{r1}$, $l = 1, \ldots, N^{r2}$. For all $\mathcal{T}^{1,i} \cap \mathcal{T}^{2,l} \triangleq \mathcal{T}^{(1,i),(2,l)} \neq \emptyset$ the optimal solution is stored in an ordered way as described in Example 15.3, 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

$$\begin{array}{l} u^*(x) = F^i x + G^i \ \text{if} \ \ x \in \mathcal{R}^i, \\ \mathcal{R}^i \triangleq \{x : \ x' L^i(j) x + M^i(j) x \le N^i(j), \ j = 1, \ldots, n^i\}, \end{array} \tag{15.45}$$

where $\mathcal{R} = \bigcup_{j=1}^{N_{\mathcal{R}}} \mathcal{R}^j$ is a polyhedron and the value function

$$J^*(x) = \tilde{J}_j^*(x) \text{ if } x \in \mathcal{D}^j, j = 1, \ldots, N^{\mathcal{D}}, \tag{15.46}$$

where $\tilde{J}_j^*(x)$ are multiple quadratic functions defined over the convex polyhedra $\mathcal{D}^j$. The polyhedron $\mathcal{D}^j$ can contain several regions $\mathcal{R}^i$ or can coincide with one of them. Note that (15.45) and (15.46) have to be considered as PWA and PPWQ functions in the *ordered region* sense.

If $d > 2$ then the value function in (15.46) 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 (15.45)– (15.46). $\square$

Lemma 15.3 (one to r problem). *Problem (15.39) 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 $q(x) \triangleq q_i$ 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

$$J^*(x) = \min \left\{ \begin{array}{l} \min_u l(x,u) + q_1(f(x,u)), \\ \text{subj. to } f(x,u) \in \bar{\mathcal{P}}_1 \\ \qquad\qquad f(x,u) \in \mathcal{P} \\ \vdots, \\ \min_u l(x,u) + q_r(f(x,u))\} \\ \text{subj. to } f(x,u) \in \bar{\mathcal{P}}_r \\ \qquad\qquad f(x,u) \in \mathcal{P} \end{array} \right\}. \tag{15.47}$$

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 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 (15.47).The cost function in problem (15.39) is lower-semicontinuous since it is a composition of a lower-semicontinuous function and a continuous function. Then, since the domain is compact, problem (15.47) admits a minimum. Therefore for a given x, there exists one mp-QP in problem (15.47) which yields the optimal solution. The procedure based on solving mp-QPs and storing the results as in Example 15.3 will be the same as in Lemma 15.2 but the domain $\mathcal{R} = \bigcup_{j=1}^{N^{\mathcal{R}}} \mathcal{R}^j$ of the solution can be a non-Euclidean polyhedron. $\square$

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 15.4. *Problem (15.39) 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.

15.6.3 Algorithmic Solution of the Bellman Equations

In the following we will substitute the PWA system equations (15.2) with the shorter form

$$x(k+1) = \tilde{f}_{PWA}(x(k), u(k)) \tag{15.48}$$

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 $\left[\begin{smallmatrix} x \\ u \end{smallmatrix}\right] \in \tilde{\mathcal{C}}^i$, $i = 1, \ldots, s$, and $\left\{\tilde{\mathcal{C}}^i\right\}$ is a polyhedral partition of $\tilde{\mathcal{C}}$.

Consider the dynamic programming formulation of the CFTOC problem (15.6)-(15.7),

$$J_j^*(x(j)) \triangleq \ \min_{u_j} \quad x_j' Q x_j + u_j' R u_j + J_{j+1}^*(\tilde{f}_{PWA}(x(j), u_j)) \tag{15.49}$$

$$\text{subj. to} \quad \tilde{f}_{PWA}(x(j), u_j) \in \mathcal{X}_{j+1} \tag{15.50}$$

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

$$\mathcal{X}_N \quad = \mathcal{X}_f \tag{15.51}$$

$$J_N^*(x) = x' P x, \tag{15.52}$$

where $\mathcal{X}_j$ is the set of all states $x(j)$ for which problem (15.49)–(15.50) is feasible:

$$\mathcal{X}_j = \{x \in \mathbb{R}^n | \ \exists u, \ \tilde{f}_{PWA}(x,u) \in \mathcal{X}_{j+1}\}. \tag{15.53}$$

Equations (15.49)-(15.53) 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 Bellman equations (15.49)–(15.52) 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 (15.49)–(15.52)

$$J_{N-1}^*(x_{N-1}) \triangleq \min_{\{u_{N-1}\}} x_{N-1}' Q x_{N-1} + u_{N-1}' R u_{N-1} + J_N^*(\tilde{f}_{PWA}(x_{N-1}, u_{N-1})) \tag{15.54}$$

$$\text{subj. to} \quad \tilde{f}_{PWA}(x_{N-1}, u_{N-1}) \in \mathcal{X}_f. \tag{15.55}$$

The cost to go function $J_N^*(x)$ in (15.54) is quadratic, the terminal region $\mathcal{X}_f$ is a polyhedron and the constraints are piecewise affine. Problem (15.54)–(15.55) is an *s to one problem* that can be solved by solving s mp-QPs 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 (15.49)–(15.52) is an *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 $sN_{j+1}^r d_{j+1}$ mp-QPs (Lemma 15.4). The resulting optimal solution will have the form (15.9) 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 *one to one problem* becomes a 2^{m_ℓ} *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., $\mathbf{R} := \{\mathcal{R}_i\}_{i=1,\dots,|\mathbf{R}|}$, where $\mathcal{R}_i$ is a polyhedron and $|\mathbf{R}|$ is the cardinality of the set $\mathbf{R}$. Furthermore, when we say 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 15.5

```
INPUT CFTOC problem (15.3)−−(15.7)
OUTPUT Solution (15.9) in the ordered region sense.

LET $\mathbf{R}_N = \{\mathcal{X}_f\}$
LET $J^*_{N,1}(x) := x'Px$
FOR $k = N-1, \ldots, 1$,
    FOR $i = 1, \ldots, |\mathbf{R}_{k+1}|$,
        FOR $j = 1, \ldots, s$,
            LET $\mathcal{S}_{k,i,j} = \{\}$
            SOLVE the mp−QP

$$\begin{array}{ll} \mathcal{S}_{k,i,j} \leftarrow & \min_{u_k} \; x_k'Qx_k + u_k'Ru_k + J^*_{k+1,i}(A_jx_k + B_ju_k + f_j) \\ & \text{subj. to} \; \begin{cases} A_jx_k + B_ju_k + f_j \in \mathcal{R}_{k+1,i} \\ \left[\begin{smallmatrix} x_k \\ u_k \end{smallmatrix}\right] \in \tilde{\mathcal{C}}^j. \end{cases} \end{array}$$

        END
    END

    LET $\mathbf{R}_k = \{\mathcal{R}_{k,i,j,l}\}_{i,j,l}$. Denote by $\mathcal{R}_{k,h}$ its
    elements, and by $J^*_{k,h}$ and $u^*_{k,h}(x)$ the associated
    costs and optimizers, with $h \in \{1, \ldots, |\mathbf{R}_k|\}$

    KEEP only triplets $(J^*_{k,h}(x), \; u^*_{k,h}(x), \; \mathcal{R}_{k,h})$
    for which
    $\exists x \in \mathcal{R}_{k,h} : x \notin \mathcal{R}_{k,d}, \forall d \neq h$ OR
    $\exists x \in \mathcal{R}_{k,h} : J^*_{k,h}(x) < J^*_{k,d}(x), \forall d \neq h$

    CREATE multiplicity information and additional
    regions for an ordered region solution as
    explained in Example 15.3
END.
```
□

In Algorithm 15.5, 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 $\mathcal{R}_{k,i,j,l}$, for all l:

$$S_{k,i,j} = \bigcup_l \left\{ \left(J^*_{k,i,j,l}(x), \; u^*_{k,i,j,l}(x), \; \mathcal{R}_{k,i,j,l} \right) \right\}. \tag{15.56}$$

where the indices in (15.56) 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.

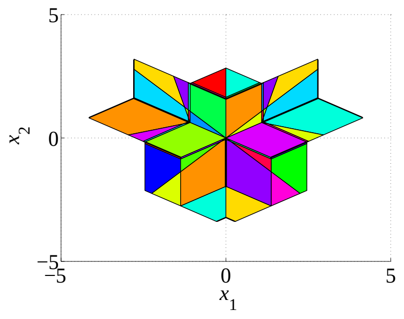
The "KEEP only triplets" Step of Algorithm 15.5 aims at discarding regions $\mathcal{R}_{k,h}$ that are completely covered by some other regions that have lower cost. Obviously, if there are some parts of the region $\mathcal{R}_{k,h}$ that are not covered at all by other regions (first condition) we need to keep them. Note that comparing the cost functions 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.

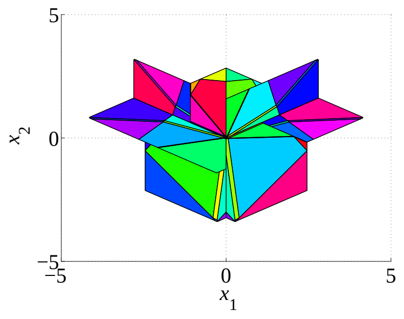
The output of Algorithm 15.5 is the state-feedback control law (15.9) considered in the ordered region sense. The online implementation of the control law requires simply the evaluation of the PWA controller (15.9) in the ordered region sense.

15.6.4 Examples

Example 15.4. Consider the control problem of steering the piecewise affine system (15.29) to a small region around the origin. The constraints and cost function of Example (15.2) are used. The state-feedback solution $u_0^*(x(0))$ was determined by using the approach presented in the previous section. When the infinity norm is used in the cost function, the feasible state-space $\mathcal{X}_0$ at time 0 is divided into 84 polyhedral regions and it is depicted in Fig. 15.5(a). When the squared Euclidean norm is used in the cost function, the feasible state-space $\mathcal{X}_0$ at time 0 is divided into 92 polyhedral regions and is depicted in Fig. 15.5(b).



(a) Partition of the feasible state-space $\mathcal{X}_0$ when the infinity norm is used in the cost function ($N^r{}_3 = 84$)



(b) Partition of the feasible state-space $\mathcal{X}_0$ when the squared Euclidian norm is used in the cost function ($N^r{}_3 = 92$)

Fig. 15.5 Example 15.4: state space control partition for $u_0^*(x(0))$

Note that as explained in Section 15.6.2 for the case of the squared Euclidian norm, the optimal control law is stored in a special data structure where:

1. The ordering of the regions is important.

2. The polyhedra can overlap.
3. The polyhedra can have an associated value function of multiplicity $d > 1$. Thus, d quadratic functions have to be compared on-line in order to compute the optimal control action.

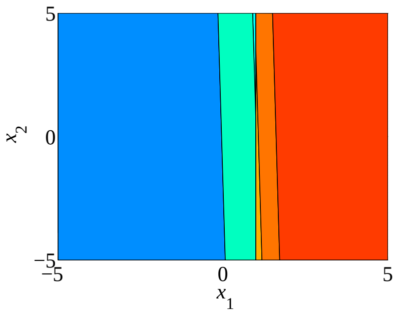
Example 15.5. Consider the hybrid spring-mass system described in Example 14.2

$$x(t+1) = \begin{cases} \left[\begin{smallmatrix} 0.90 & 0.02 \\ -0.02 & -0.00 \end{smallmatrix}\right] x(t) + \left[\begin{smallmatrix} 0.10 \\ 0.02 \end{smallmatrix}\right] u_1(t) + \left[\begin{smallmatrix} -0.01 \\ -0.02 \end{smallmatrix}\right] & \text{if } x_1(t) \leq 1, u_2(t) \leq 0.5 \\ \left[\begin{smallmatrix} 0.90 & 0.02 \\ -0.06 & -0.00 \end{smallmatrix}\right] x(t) + \left[\begin{smallmatrix} 0.10 \\ 0.02 \end{smallmatrix}\right] u_1(t) + \left[\begin{smallmatrix} -0.07 \\ -0.15 \end{smallmatrix}\right] & \text{if } x_1(t) \geq 1 + \epsilon, u_2(t) \leq 0.5 \\ \left[\begin{smallmatrix} 0.90 & 0.38 \\ -0.38 & 0.52 \end{smallmatrix}\right] x(t) + \left[\begin{smallmatrix} 0.10 \\ 0.38 \end{smallmatrix}\right] u_1(t) + \left[\begin{smallmatrix} -0.10 \\ -0.38 \end{smallmatrix}\right] & \text{if } x_1(t) \leq 1, u_2(t) \geq 0.5 \\ \left[\begin{smallmatrix} 0.90 & 0.35 \\ -1.04 & 0.35 \end{smallmatrix}\right] x(t) + \left[\begin{smallmatrix} 0.10 \\ 0.35 \end{smallmatrix}\right] u_1(t) + \left[\begin{smallmatrix} -0.75 \\ -2.60 \end{smallmatrix}\right] & \text{if } x(t) \geq 1 + \epsilon, u_2(t) \geq 0.5 \end{cases} \tag{15.57}$$

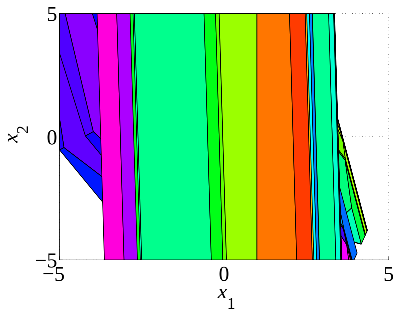
subject to the constraints

$$\begin{aligned} x(t) &\in [-5, 5] \times [-5, 5] \\ u(t) &\in [-1, 1] \end{aligned} \tag{15.58}$$

We solve the finite time constrained optimal control problem (15.7) with cost (15.6) with $N = 3$, $P = Q = \left[\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right]$, $R = \left[\begin{smallmatrix} 0.2 & 0 \\ 0 & 1 \end{smallmatrix}\right]$. The state-feedback solution was determined by using the approach presented in the previous section in two cases: without terminal constraint (Figure 15.6(a)) and with terminal constraint $\mathcal{X}_f = [-0.01\ 0.01] \times [-0.01\ 0.01]$ (Figure 15.6(b)).



(a) Partition with no terminal constraint ($N^r{}_3 = 183$)



(b) Partition with terminal constraint ($N^r{}_3 = 181$)

Fig. 15.6 Example 15.5: state space optimal control partition

15.7 Discontinuous PWA systems

Without Assumption 15.1 the optimal control problem (15.3)-(15.7) may be feasible but may not admit an optimizer for some $x(0)$ (the problem in this case would 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 (15.2) 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 15.3 cannot by used. Therefore, when considering the closure of polyhedral domains in multi-parametric programming (15.47), 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 is to introduce gaps between the boundaries of any two polyhedra belonging to the PWA domain (or, equivalently, to shrink by a quantity ε the size of every polyhedron of the original PWA system). This way, one deals with PWA systems defined over a disconnected union of closed polyhedra. By doing so, one can use the approach discussed in this chapter for continuous PWA systems. However, the optimal controller will not be defined at the points in the gaps. Also, the computed solution might be arbitrarily different from the original solution to problem (15.3)-(15.7) at any feasible point x. Despite this, if the dimension ε of the gaps is close to the machine precision and comparable to sensor/estimation errors, such an approach very appealing in practice. To the best of our knowledge in some cases this approach is the only one that is computationally tractable for computing controllers for discontinuous hybrid systems fulfilling state and input constraints that are implementable in real-time.

Without Assumption 15.1, problem (15.3)-(15.7) 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 book.

15.8 Receding Horizon Control

Consider the problem of regulating to the origin the PWA system (15.2). Receding Horizon Control (RHC) can be used to solve such a constrained regulation problem. The control algorithm is identical to the one outlined in Chapter 11 for linear systems. Assume that a full measurement of the state $x(t)$ is available at the current time t. Then, the finite time optimal control problem

$$J_t^*(x(t)) \triangleq \min_{U_{t\to t+N|t}} J_t(x(t), U_{t\to t+N|t}) \tag{15.59a}$$

$$\text{subj. to } \begin{cases} x_{t+k+1|t} = A^i x_{t+k|t} + B^i u_{t+k|t} + f^i \text{ if } \left[\begin{smallmatrix} x_{t+k|t} \\ u_{t+k|t} \end{smallmatrix}\right] \in \tilde{\mathcal{C}}^i \\ x_{t+N|t} \in \mathcal{X}_f \\ x_{t|t} = x(t) \end{cases} \tag{15.59b}$$

is solved at each time t, where $U_{t\to t+N|t} = \{u_{t|t}, \ldots, u_{t+N-1|t}\}$. Let $U_t^* = \{u_{t|t}^*, \ldots, u_{t+N-1|t}^*\}$ be the optimal solution of (15.59) at time t. Then, the first sample of U_t^* is applied to system (15.2):

$$u(t) = u_{t|t}^*. \tag{15.60}$$

The optimization (15.59) is repeated at time $t+1$, based on the new state $x_{t+1|t+1} = x(t+1)$, yielding a *moving* or *receding horizon* control strategy.

Based on the results of previous sections the state feedback receding horizon controller (15.59)–(15.60) can be immediately obtained in two ways: (i) solve the MIQP/MILP (15.24) for $x_{t|t} = x(t)$ or (ii) by setting

$$u(t) = f_0^*(x(t)), \tag{15.61}$$

where $f_0^* : \mathbb{R}^n \to \mathbb{R}^{n_u}$ is the piecewise affine solution to the CFTOC (15.59) computed as explained in Section 15.6. Clearly, the explicit form (15.61) has the advantage of being easier to implement, and provides insight on the type of action of the controller in different regions of the state space.

15.8.1 Stability and Feasibility Issues

As discussed in Chapter 11 the feasibility and stability of the receding horizon controller (15.59)–(15.60) is, in general, not guaranteed.

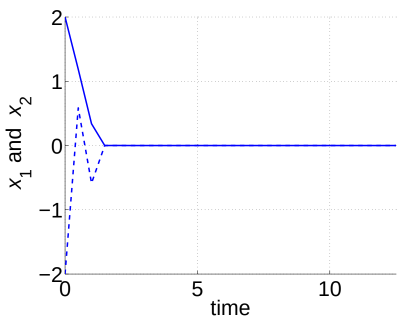
Theorem 11.2 in Chapter 11 can be immediately modified for the hybrid case: persistent feasibility and Lyapunov stability are guaranteed if the terminal constraint $\mathcal{X}_f$ is a control invariant set (assumption (A2) in Theorem 11.2) and the terminal cost $p(x_N)$ is a control Lyapunov function (assumption (A3) in Theorem 11.2). In the hybrid case the computation of control invariant sets and control Lyapunov functions is computationally more involved. As in the linear case, $\mathcal{X}_f = 0$ satisfies the aforementioned properties and it thus represents a very simple, yet restrictive, choice for guaranteeing persistent feasibility and Lyapunov stability.

15.8.2 Examples

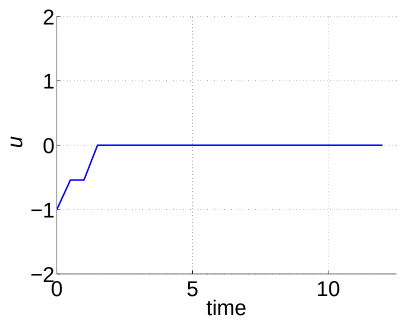
Example 15.6. Consider the problem of regulating the piecewise affine system (15.29) to the origin. The finite time constrained optimal control (15.7) is solved with $N = 3$, $P = Q = \left[\begin{smallmatrix}1 & 0\\ 0 & 1\end{smallmatrix}\right]$, $R = 1$, and $\mathcal{X}_f = [-0.01\ 0.01] \times [-0.01\ 0.01]$. Its state feedback solution (15.9) $u^*(x(0)) = f_0^*(x(0))$ at time 0 is implemented in a receding horizon fashion, i.e. $u(x(k)) = f_0^*(x(k))$.

The state feedback control law with cost (15.5) with $p = \infty$ has been computed in Example 15.4 and it consists of 84 polyhedral regions. None of them has multiplicity higher than 1. Figure 15.7 shows the corresponding closed-loop trajectories starting from the initial state $x(0) = [-2\ 2]'$.

The state feedback control law with cost (15.6) has been computed in Example 15.4 and it consists of 92 polyhedral regions, some of them have multiplicity higher than 1. Figure 15.8 shows the corresponding closed-loop trajectories starting from the initial state $x(0) = [-2\ 2]'$.



(a) State Trajectories (x_1 dashed line and x_2 solid line)



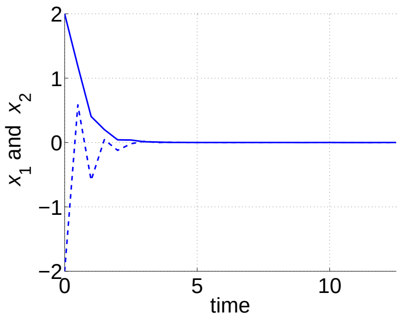
(b) Optimal Input

Fig. 15.7 Example 15.6: MPC control of system (15.29) when infinity norm is used

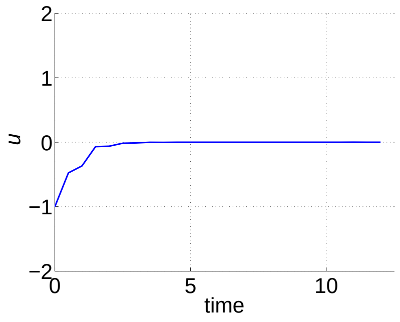
Example 15.7. Consider the problem of regulating the hybrid spring-mass system (15.57) described in Examples 14.2 and 15.5 to the origin. The finite time constrained optimal control problem (15.7) with cost (15.6) is solved with $N = 3$, $P = Q = \left[\begin{smallmatrix}1 & 0\\ 0 & 1\end{smallmatrix}\right]$, $R = \left[\begin{smallmatrix}0.2 & 0\\ 0 & 1\end{smallmatrix}\right]$. Its state feedback solution (15.9) $u^*(x(0)) = f_0^*(x(0))$ at time 0 is implemented in a receding horizon fashion, i.e. $u(x(k)) = f_0^*(x(k))$.

The state-feedback solution was determined in Example 15.5 for the case of no terminal constraint (Figure 15.5(a)). Figure 15.9 depicts the corresponding closed-loop trajectories starting from the initial state $x(0) = [3\ 4]'$.

The state-feedback solution was determined in Example 15.5 for terminal constraint $\mathcal{X}_f = [-0.01,\ 0.01] \times [-0.01,\ 0.01]$ (Figure 15.5(b)). Figure 15.10



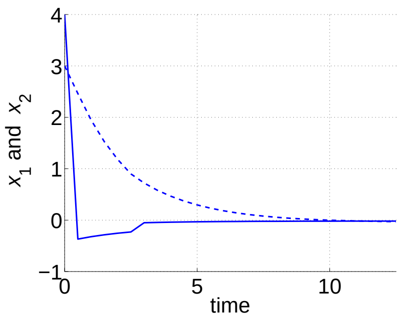
(a) State Trajectories (x_1 dashed line and x_2 solid line)



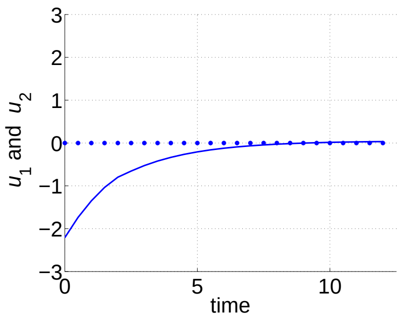
(b) Optimal Input

Fig. 15.8 Example 15.6: MPC control of system (15.29) when squared Euclidian norm is used

depicts the corresponding closed-loop trajectories starting from the initial state $x(0) = [3\ 4]'$.

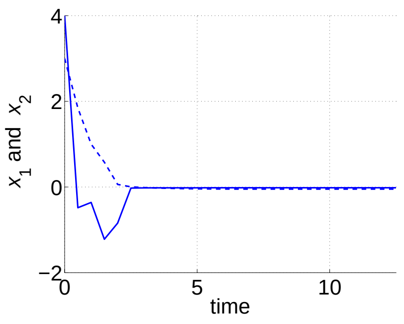


(a) State Trajectories (x_1 dashed line and x_2 solid line)

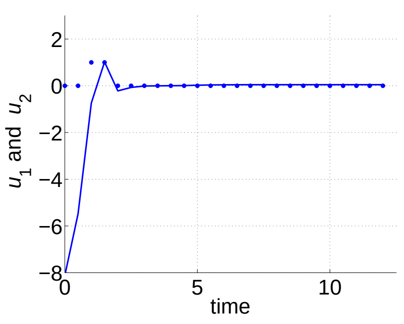


(b) Optimal Inputs (u_1 solid line and the binary input u_2 with a starred line)

Fig. 15.9 Example 15.7: MPC control of system (15.57) without terminal constraint



(a) State Trajectories (x_1 dashed line and x_2 solid line)



(b) Optimal Inputs (u_1 solid line and the binary input u_2 with a starred line)

Fig. 15.10 Example 15.6: MPC control of system (15.57) with terminal constraint

Appendix A
Polycover Algorithms

A.1 Polycover: MILP formulation

We note that $\mathcal{P}$ is not fully covered by $\mathcal{Q}$, i.e.

$$\mathcal{P} \nsubseteq (\cup_{i=1}^{N_Q} \mathcal{Q}_i) \tag{A.1}$$

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

$$\exists x \in \mathcal{P} : \exists j_i \in \{1, \ldots, n_{Q_i}\},\ [Q_i^x]_{(j_i)} x - [Q_i^c]_{(j_i)} > 0,\ i = 1, \ldots, N_Q. \tag{A.2}$$

To express this violation of constraints we introduce slack variables

$$y_{i,j}(x) = \begin{cases} [Q_i^x]_{(j)} x - [Q_i^c]_{(j)} & \text{if} \quad [Q_i^x]_{(j)} x - [Q_i^c]_{(j)} \geq 0, \\ 0 & \text{if} \quad [Q_i^x]_{(j)} x - [Q_i^c]_{(j)} \leq 0, \end{cases}, \quad j = 1, \ldots, n_{Q_i}, \quad i = 1, \ldots, N_Q. \tag{A.3}$$

The expression (A.2) can now be posed as a feasibility question in x and $y_{i,j}$

$$\begin{array}{c} P^x x \leq P^c, \\ \sum_{j=1}^{n_{Q_i}} y_{i,j} > 0, \quad i = 1, \ldots, N_Q \end{array} \tag{A.4}$$

Checking the condition (A.4) is still not possible with standard solvers, since the relation (A.3) describes a non-linear function. However, by introducing auxiliary binary variables one can rewrite (A.3) as the following equivalent set of linear inequalities

$$\begin{bmatrix} 0 & -m \\ 0 & -M \\ 1 & -M \\ -1 & m \\ 1 & -m \\ -1 & M \end{bmatrix} \begin{bmatrix} y_{i,j} \\ \delta_{i,j} \end{bmatrix} \leqslant \begin{bmatrix} [Q_i^x]_{(j)}x - [Q_i^c]_{(j)} - m \\ -[Q_i^x]_{(j)}x + [Q_i^c]_{(j)} \\ 0 \\ 0 \\ [Q_i^x]_{(j)}x - [Q_i^c]_{(j)} - m \\ -[Q_i^x]_{(j)}x + [Q_i^c]_{(j)} + M \end{bmatrix} \tag{A.5}$$
$$\delta_{i,j} \in \{0,1\}, \quad j = 1, \ldots, n_{Q_i}, \quad i = 1, \ldots, N_Q,$$

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

$$\begin{aligned} m \leq \quad \min_{x,i,j} \quad & [Q_i^x]_{(j)}x - [Q_i^c]_{(j)} \\ \text{subj. to } & P^x x \leq P^c \\ & j \in \{1, \ldots, n_{Q_i}\} \\ & i \in \{1, \ldots, N_Q\} \end{aligned} \tag{A.6}$$

$$\begin{aligned} M \geq \quad \min_{x,i,j} \quad & [Q_i^x]_{(j)}x - [Q_i^c]_{(j)} \\ \text{subj. to } & P^x x \leq P^c \\ & j \in \{1, \ldots, n_{Q_i}\} \\ & i \in \{1, \ldots, N_Q\} \end{aligned} \tag{A.7}$$

Actually, in terms of the number of inequalities that are used, (A.5) can be further simplified to

$$\begin{bmatrix} -1 & 0 \\ -1 & 0 \\ 1 & -m \\ 1 & -M \end{bmatrix} \begin{bmatrix} y_{i,j} \\ \delta_{i,j} \end{bmatrix} \leqslant \begin{bmatrix} 0 \\ -[Q_i^x]_{(j)}x + [Q_i^c]_{(j)} \\ [Q_i^x]_{(j)}x - [Q_i^c]_{(j)} - m \\ 0 \end{bmatrix} \tag{A.8}$$
$$\delta_{i,j} \in \{0,1\}, \quad j = 1, \ldots, n_{Q_i}, \quad i = 1, \ldots, N_Q$$

Since (A.4) and (A.8) describe a Mixed Integer Linear Programming (MILP) feasibility problem it follows that we can check if $\mathcal{P} \nsubseteq (\cup_{i=1}^{N_Q} \mathcal{Q}_i)$ by solving an MILP feasibility problem.

However, instead of solving a feasibility MILP problem with (A.4) it may be more useful (and numerically robust) to solve the following optimality MILP problem

$$\begin{aligned} \max_{\lambda, x, \delta_{i,j}, y_{i,j}} \quad & \lambda \\ \text{subj. to} \quad & \begin{cases} P^x x \leq P^c, \\ \sum_{j=1}^{n_{Q_i}} y_{i,j} \geq \lambda, \quad i = 1, \ldots, N_Q \\ \sum_{j=1}^{n_{Q_i}} d_{i,j} \geq 1, \quad i = 1, \ldots, N_Q \\ \text{Eq.(A.8)} \end{cases} \end{aligned} \tag{A.9}$$

Effectively, the optimal value λ^* is related to the size of a largest non-covered part of $\mathcal{P}$.

Theorem A.1. *Let λ^* be the solution to the problem (A.9), then $\mathcal{P} \nsubseteq (\cup_{i=1}^{N_Q} \mathcal{Q}_i)$ if and only if $\lambda^* > 0$.*

Proof: Follows from the construction of the MILP problem (A.9). □ ■

Remark A.1. Strictly speaking, condition $\sum_{j=1}^{n_{Q_i}} d_{i,j} \geq 1$ in (A.9) is redundant, but it reduces the problems with the integrality tolerances in existing MILP solvers. Also note that when solving (A.9) there is no need for condition $y_{i,j} \geq 0$ (first row in Eq. (A.8)).

The MILP problem (A.9) 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.

A.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 $\mathcal{Q}_i = \{x \in \mathbb{R}^{n_x} \mid Q_i^x x \leq Q_i^c\}$ be N_Q polytopes in $\mathbb{R}^{n_x}$ given as the intersection of n_{Q_i} half-spaces (i.e. Q_i^x is a $n_{Q_i} \times n_x$ matrix). We want to determine if $\mathcal{P} \subseteq (\cup_{i=1}^{N_Q} \mathcal{Q}_i)$. Here we consider an exact solution to the problem by using the algorithm that (as an extension) computes the set $\mathcal{R} = \mathcal{P} \setminus (\cup_{i=1}^{N_Q} \mathcal{Q}_i)$, 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 A.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: $\mathcal{P} = \{x \mid P^x x \leq P^c\}$, $\mathcal{Q}_i = \{x \mid Q_i^x x \leq Q_i^c\}$, $P^x \in \mathbb{R}^{n_P \times n_x}$, $P^c \in \mathbb{R}^{n_P}$, $Q_i^x \in \mathbb{R}^{n_{Q_i} \times n_x}$, $Q_i^c \in \mathbb{R}^{n_{Q_i}}$, 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 3.4.4) and to remove those $\mathcal{Q}_i$ that do not intersect $\mathcal{P}$, e.g. by checking if Chebyshev Ball (see Section 3.4.5) of a joint polyhedra $\{x \mid P^x x \leq P^c, Q_i^x x \leq Q_i^c\}$ is nonempty.

Algorithm A.1 (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 A.1
OUTPUT  $\mathcal{R} = \{\mathcal{R}_i\}_{i=1}^{N_R} :=$ 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
```

$$\mathcal{A}_i = \{j \mid \exists x \in \mathbb{R}^{n_x} : P^x x \leq P^c,\ [Q_i^x]_{(j)} x > [Q_i^c]_{(j)}, j \in \{1, \ldots, n_{Q_i}\}\}. \quad (A.10)$$

2. *IF* $\exists i : \mathcal{A}_i = \emptyset$ *THEN LET* $\mathcal{R} \leftarrow \emptyset$ *and EXIT*3. *Remove non-active constraints from* $\mathcal{Q}_i$: $Q_i^x \leftarrow [Q_i^x]_{(\mathcal{A}_i)}$, $\quad Q_i^c \leftarrow [Q_i^c]_{(\mathcal{A}_i)}$, $\quad i = 1, \ldots, N_Q$.4. *LET* $\mathcal{R} =$ *feaspoly*$(\mathcal{P}, \{\mathcal{Q}_i\}_{i=1}^{N_Q})$5. *Remove redundant constraints from* $\mathcal{R}_i$: $\mathcal{R}_i \leftarrow$ *minrep*$(\mathcal{R}_i)$, $\quad i = 1, \ldots, N_R$. □

Algorithm A.2 (feaspoly$(\mathcal{P}, \{\mathcal{Q}_i\}_{i=1}^{N_Q})$)

```
INPUT   Polyhedron $\mathcal{P} = \{x \mid P^x x \le P^c\}$ and $\mathcal{Q} = \{\mathcal{Q}_i\}_{i=1}^{N_Q}$ satisfying Assumption A.1
OUTPUT  Set of feasible polyhedra $\mathcal{R} = \{\mathcal{R}_i\}_{i=1}^{N_R} :=$ feaspoly$(\mathcal{P}, \{\mathcal{Q}_i\}_{i=1}^{N_Q})$

1. LET $\mathcal{R} = \emptyset$, $k = 1$
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 $\mathcal{R} \leftarrow \mathcal{P}$ and EXIT, ELSE go to Step 2
4. FOR $j = 1$ TO $n_{Q_k}$
    4.1. IF $\exists x : P^x x \le P^c,\ [Q_k^x]_{(j)} x > [Q_k^c]_{(j)}$
        4.1.1. LET $\tilde{\mathcal{P}} = \mathcal{P} \cap \{x \mid [Q_k^x]_{(j)} x \ge [Q_k^c]_{(j)}\}$
        4.1.2. IF $N_Q > k$ THEN $\mathcal{R} \leftarrow \mathcal{R} \cup$ feaspoly$(\tilde{\mathcal{P}}, \{\mathcal{Q}_i\}_{i=k+1}^{N_Q})$, ELSE $\mathcal{R} \leftarrow \mathcal{R} \cup \tilde{\mathcal{P}}$
    4.2. LET $\mathcal{P} \leftarrow \mathcal{P} \cap \{x \mid [Q_k^x]_{(j)} x \le [Q_k^c]_{(j)}\}$
5. EXIT
```
□

Remark A.2. Steps 1 of Algorithm A.1 and Step 2 and Step 4.1 of Algorithm A.2 are simple feasibility LP's. Note that Algorithm A.2 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 A.3. The **regiondiff** algorithm (Algorithm A.1) basically implements a branch & bound search for feasible constraint combinations where each branch corresponds to the inversion of a facet of $\mathcal{Q}_i$ obtained in Step 3 of Algorithm A.1.

The most demanding part of the set-difference computation lies in Algorithm A.2 (and it's recursive calls). Therefore in this analysis we neglect the cost of removal of the *non-active constraints* from $\mathcal{Q}_i$ or computing the minimal representation of the regions $\mathcal{R}_i$ that describe the set difference.

The worst case complexity of Algorithm A.2 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 $\mathcal{Q}_i$, 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 $\mathcal{Q}_i$. 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 A.2 is bounded with

$$\sum_{i=1}^{N_Q} \left[lp(n_x + 1, n_P + \sum_{j=1}^{i} n_{Q_j}) \prod_{j=1}^{i} n_{Q_i} \right], \tag{A.11}$$

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 [66] for the hyperplane arrangement problem that gives an upper bound[1] for the maximal number of cells created by M hyperplanes in $\mathbb{R}^{n_x}$

$$\sum_{i=0}^{n_x} \binom{M}{i}. \tag{A.12}$$

By letting M be equal to the total number of active-constraints, i.e. $M = \sum_i n_{Q_i}$, from (A.12) follows

$$N_R \le \sum_{i=1}^{n_x} \binom{M}{i} = \mathcal{O}(M^{n_x}). \tag{A.13}$$

Remark A.4. In practice the complexity estimate given by (A.11) is very conservative. Each polytope $\mathcal{R}_i$, $i = 1, \dots, N_R$, corresponds to the leaf (bottom) node of the exploration tree in Algorithm A.1 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 (A.13) than with the expression (A.11).

In a special case when we only want to check if $\mathcal{P} \subseteq (\cup_{i=1}^{N_Q} \mathcal{Q}_i)$ finding any feasible $\mathcal{R}_j$ in Algorithm A.2 provides a negative answer and we can abort further search as is shown in the following algorithm.

Algorithm A.3 (iscover($\mathcal{P}, \{\mathcal{Q}_i\}_{i=1}^{N_Q}$))

```
INPUT   $\mathcal{P}$ and $\mathcal{Q} = \{\mathcal{Q}_i\}_{i=1}^{N_Q}$ satisfying Assumption A.1
OUTPUT  $R \in \{$TRUE, FALSE$\}$, $R :=$ iscover$(\mathcal{P}, \{\mathcal{Q}_i\}_{i=1}^{N_Q})$

1. LET $k = 1$, $R =$ 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 $n_{Q_k}$
```

[1] The upper bound is obtained by the hyperplanes in the so-called general position.

```
    4.1. **IF** $\exists x : P^x x \leq P^c,\ [Q_k^x]_{(j)} x > [Q_k^c]_{(j)}$
        4.1.1. **IF** $k = N_Q$ **THEN EXIT**
        4.1.2. **LET** $\tilde{\mathcal{P}} = \mathcal{P} \cap \{x \mid [Q_k^x]_{(j)} x \geq [Q_k^c]_{(j)}\}$
        4.1.3. **IF** $iscover(\tilde{\mathcal{P}}, \{\mathcal{Q}_i\}_{i=k+1}^{N_Q}) =$ **FALSE THEN EXIT**
    4.2. **LET** $\mathcal{P} \leftarrow \mathcal{P} \cap \{x \mid [Q_k^x]_{(j)} x \leq [Q_k^c]_{(j)}\}$
5. **LET** $R =$ **TRUE** *and* **EXIT**                                   □
```

Similarly to Algorithm A.2 the worst case complexity of Algorithm A.3 is bounded by (A.11) (see also Remark A.4).

Appendix B
Merging of P-collections

B.0.1 Preliminaries

Definition B.1 (Separating Hyperplane). Suppose $\mathcal{P}_1$ and $\mathcal{P}_2$ are two (convex) polyhedra that do not intersect, i.e. $\mathcal{P}_1 \cap \mathcal{P}_2 = \emptyset$. A hyperplane $\{x \mid c^T x = d\}$ with $c \neq 0$ and d, such that $c^T x \leq d$ for all $x \in \mathcal{P}_1$ and $c^T x \geq d$ for all $x \in \mathcal{P}_2$ is called a *separating hyperplane* for the polyhedra $\mathcal{P}_1$ and $\mathcal{P}_2$.

Consider the markings defined in Section 3.5.3.1. The proof of the following lemma follows directly from the definition of the markings.

Lemma B.1 (Separating Hyperplane). *Given the hyperplane arrangement $\{H_i\}_{i=1,\dots,n}$ consisting of n distinct hyperplanes, the set of markings $M(\mathcal{R})$, and the two polyhedra $\mathcal{P}_1$ and $\mathcal{P}_2$ with the corresponding markings $m_1, m_2 \in M(\mathcal{R})$ that differ in the j-th component, then H_j is a separating hyperplane for $\mathcal{P}_1$ and $\mathcal{P}_2$.*

Recall the definition of envelope in Section 3.4.2

Lemma B.2. *Given the hyperplane arrangement $\{H_i\}_{i=1,\dots,n}$ consisting of n distinct hyperplanes, the set of markings $M(\mathcal{R})$, and the two polyhedra $\mathcal{P}_1$ and $\mathcal{P}_2$ with the corresponding markings $m_1, m_2 \in M(\mathcal{R})$, where $m_1(i) = m_2(i)$ for $i \in \mathcal{I}$ and $m_1(i) \neq m_2(i)$ for $i \in \mathcal{I}'$ with $\mathcal{I}' = \{1, \dots, n\} \setminus \mathcal{I}$, we construct the marking m as follows: $m(i) = m_1(i)$ for $i \in \mathcal{I}$ and $m(i) =$'$*$' for $i \in \mathcal{I}'$. Then the envelope env$(\mathcal{P}_1, \mathcal{P}_2)$ of the two polyhedra is given by the marking m.*

Proof: Recall that a '$*$' in a marking means that the corresponding hyperplane does not define the polyhedron. As all the facets of $\mathcal{P}_1$ and $\mathcal{P}_2$ are subsets of the hyperplanes in the arrangement, and as the hyperplanes with indices $\mathcal{I}'$ are separating hyperplanes for $\mathcal{P}_1$ and $\mathcal{P}_2$ according to Lemma B.1, the proof follows from the definition of the envelope. ■

The proof can be easily generalized to envelopes of more than two polyhedra.

Recall Theorem 3.2 stating that given the two polyhedra $\mathcal{P}_1$ and $\mathcal{P}_2$, their union $\mathcal{P}_1 \cup \mathcal{P}_2$ is convex if and only if $\mathcal{P}_1 \cup \mathcal{P}_2 = \text{env}(\mathcal{P}_1, \mathcal{P}_2)$. The following lemma allows us to determine the convexity of two polyhedra by only evaluating their corresponding markings. This lemma constitutes the basis for the two OCR algorithms.

The following lemma allows us to determine the convexity of two polyhedra by only evaluating their corresponding markings. This lemma constitutes the basis for the two OCR algorithms.

Lemma B.3. *Given the collection of markings $M(\mathcal{R})$, the union of the two polyhedra $\mathcal{P}_1$ and $\mathcal{P}_2$ with the markings $m_1, m_2 \in M(\mathcal{R})$, $m_1 \neq m_2$, is convex, if and only if the markings differ in exactly one component.*

Proof: As we have Theorem 3.2 at our disposal, we only need to prove that $\mathcal{P}_1 \cup \mathcal{P}_2 = \text{env}(\mathcal{P}_1, \mathcal{P}_2)$ if and only if m_1 and m_2 differ in exactly one component. The "$\Leftarrow$" part follows directly from Lemma B.2. The "$\Rightarrow$" part follows by contradiction. Recall, that $\mathcal{P}_1 \cup \mathcal{P}_2 \subseteq \text{env}(\mathcal{P}_1, \mathcal{P}_2)$, and assume that $\mathcal{P}_1 \cup \mathcal{P}_2 \neq \text{env}(\mathcal{P}_1, \mathcal{P}_2)$, i.e. there are points $x \in \text{env}(\mathcal{P}_1, \mathcal{P}_2) \setminus (\mathcal{P}_1 \cup \mathcal{P}_2)$. Then there exists at least one hyperplane that is separating x from $\mathcal{P}_1$ or x from $\mathcal{P}_2$ besides the one that is separating $\mathcal{P}_1$ from $\mathcal{P}_2$. Thus m_1 and m_2 differ in at least two components. ■

The concept of markings in a hyperplane arrangement allows us to evaluate the convexity of polyhedra by applying Lemma B.3 to their associated set of markings. The algorithms refrain from solving LPs – in fact, they extract the information from the markings that in turn summarize the result of the LPs solved to compute the cells of the hyperplane arrangement. Even though we will design algorithms assuring optimality, the computation times to solve the OCR problems are rather small making the algorithms applicable to problems of meaningful size.

Definition B.2 (Connectivity). Two polyhedra are called *neighboring* polyhedra if they share a common facet. A set of polyhedra $\{\mathcal{P}_i\}_{i \in \mathcal{I}}$ is *connected* if for each $\mathcal{P}_i$, $i \in \mathcal{I}$, there exists a $\mathcal{P}_j$, $i \neq j$, $j \in \mathcal{I}$ such that $\mathcal{P}_i$ and $\mathcal{P}_j$ are neighboring polyhedra.

Obviously, a necessary condition for the convexity of a union of a set of polyhedra is that the set of polyhedra is connected. Connectivity can be easily determined using the markings. Given the set of markings $M(\mathcal{R})$ and the set of polyhedra with markings $m_i \in M(\mathcal{R})$, the polyhedra are connected if and only if for each polyhedron $\mathcal{P}_{m_i}$ with marking $m_i \in M(\mathcal{R})$, there exists a polyhedron $\mathcal{P}_{m_j}$ with marking $m_j \in M(\mathcal{R})$, such that m_i and m_j differ in exactly one component. In order to reduce the computation time, we exploit this fact by further partitioning the set of polyhedra with the same color into connected subsets.

B.1 Disjoint Optimal Complexity Reduction

Let the set M_w denote the markings of a connected subset with the same color. We refer to the corresponding polyhedra as *white* polyhedra. As the color of the remaining polyhedra is not relevant at this stage, we assume that the remaining markings $M_b' = M(\mathcal{R})\backslash M_w$ correspond to *black* polyhedra. The basic concept of the algorithm is to derive a minimal representation of the white polyhedra by dividing their envelope sequentially into polyhedra using the hyperplanes of the hyperplane arrangement.

B.1.1 Algorithm based on Branch and Bound

Let the envelope of the white polyhedra with markings M_w be denoted by $\mathcal{P}_m$. The envelope is given by the marking m, which is constructed as in Lemma B.2. Slightly abusing the notation we will write $m = \text{env}(M_w)$. As all white polyhedra are contained in their envelope, we can formulate an equivalent problem with reduced complexity that considers only the black polyhedra contained in this envelope, i.e. $M_b = \{m_b \in M_b' \mid \mathcal{P}_{m_b} \subseteq \mathcal{P}_m\}$, where $\mathcal{P}_{m_b}$ denotes the polyhedron with marking m_b.

Let $\mathcal{I} \in \{1, \ldots, n\}$ denote the index set of hyperplanes in $\mathcal{A}$ that are separating hyperplanes for polyhedra in the envelope $\mathcal{P}_m$. According to Lemma B.1, $\mathcal{I}$ is simply the collection of indices i with $m(i) =$'$*$'. Then, we can choose any hyperplane H_i, $i \in \mathcal{I}$, to divide $\mathcal{P}_m$ into two polyhedra. H_i also divides the sets of white and black markings respectively into two subsets. We denote the subset of M_w that holds those markings whose i-th element is a '$-$' with $M_{w|m(i)=-}$, i.e. $M_{w|m(i)=-} = \{m \in M_w \mid m(i) =\text{'}-\text{'}\}$. $M_{w|m(i)=+}$ and the partition of M_b are defined accordingly. Clearly, the unions of each pair of subset equal the original sets M_w and M_b, respectively. Next, the algorithm branches on the i-th hyperplane by calling itself twice – first with the arguments M_w and M_b restricted to possessing a '$-$' as i-th element, and then correspondingly with the arguments restricted to a '+'. Both function calls return sets of markings M_m corresponding to merged white polyhedra. This is repeated for all the remaining hyperplanes with indices $i \in \mathcal{I}$.

A branch terminates if one of the following two cases occurs. First, if the set of markings corresponding to black polyhedra is empty, i.e. $M_b = \emptyset$. This implies, that at this point the envelope contains only white polyhedra. Hence, the envelope represents the union of the set of white polyhedra with markings in M_w, and it is convex by construction. We will refer to this convex set as a *merged white* polyhedron. Second, if the set of markings corresponding to white polyhedra is empty, i.e. $M_w = \emptyset$, as this implies that no more white polyhedra are available.

The algorithm uses standard bound techniques to cut off suboptimal branches by using the two global variables z and $\bar{z}$. z denotes the current

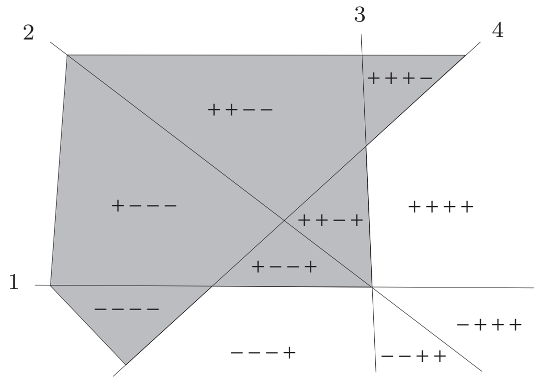


Fig. B.1 Example with four hyperplanes in $\mathcal{R} = \mathbb{R}^2$ and the corresponding markings. The polyhedra corresponding to M_w are white and the polyhedra corresponding to M'_b are grey shaded, respectively

number of merged white polyhedra and $\bar{z}$ is the local upper bound on z. Initially, z is set to 0, and $\bar{z}$ is initialized as the number of original white polyhedra. Branching is only performed if $z < \bar{z}$, as branches with $z \geqslant \bar{z}$ are either equivalent to or worse than the current optimum.

The above described branch and bound algorithm is summarized in the following.

Algorithm B.1.1

```
[] function  $M_m = \mathsf{Merge}(\ M_w,\ M'_b,\ z,\ \bar{z}\ )$
[] $m = \mathrm{env}(M_w)$
[] $M_b = \{m_b \in M'_b \mid \mathcal{P}_{m_b} \subseteq \mathcal{P}_m\}$
[] if $M_w = \emptyset$ then $M_m = \emptyset$
[] elseif $M_b = \emptyset$ then $M_m = m$
[] else
    [] $\mathcal{I} = \{i \mid m(i) = '*'\}$
    [] $M_m = \emptyset$
    [] for $i \in \mathcal{I}$
        [] if $z < \bar{z}$ then
            [] $M_{m_1} = \mathsf{Merge}\ (\ M_{w|m(i)=-}\ ,\ M_{b|m(i)=-}\ ,\ z,\ \bar{z}\ )$
            [] $M_{m_2} = \mathsf{Merge}\ (\ M_{w|m(i)=+}\ ,\ M_{b|m(i)=+}\ ,\ z + |M_{m_1}|,\ \bar{z}\ )$
            [] if  $M_m = \emptyset$  or  $|M_{m_1}| + |M_{m_2}| < |M_m|$  then
                [] $M_m = M_{m_1} \cup M_{m_2}$
                [] $\bar{z} = \min(\bar{z},\ z + |M_m|)$
[] return $M_m$
```

Example B.1. As an example with four hyperplanes in a two-dimensional space consider Fig. B.1. The envelope of the white polyhedra is given by the positive half space of H_4 and the marking $m = \quad +$. Thus, only the black polyhedra with markings $M_b = \{+{-}{-}+, ++{-}+\}$ are considered, and branching is only performed on the hyperplanes in $\mathcal{I} = \{1, 2, 3\}$. Branching on H_1 leads in one step to the two merged (white) polyhedra with $M_m = \{- \quad +, ++++\}$. This is already the optimal solution. Nevertheless, the algorithm also branches on the two remaining hyperplanes in $\mathcal{I}$ and finds two additional solutions that are equivalent to the first one in terms of the number of polyhedra.

Lemma B.4. *Algorithm B.1.1 solves the Disjoint Optimal Complexity Reduction Problem 3.1.*

Proof: The proof follows in a constructive way from the algorithm. When branching on the i-th hyperplane H_i, the set of white markings is divided into the two sets $M_{w|m(i)=-}$ and $M_{w|m(i)=+}$ according to the two half spaces defined by H_i. This operation assures that the merged polyhedra are mutually disjoint. In particular, as no white polyhedra are discarded during the operation and since $M_w = (M_{w|m(i)=-}) \cup (M_{w|m(i)=+})$, the union of the merged polyhedra equals the union of the white polyhedra. The minimality of the number of merged polyhedra is ensured by branching on all hyperplanes unless bound techniques come into effect cutting off suboptimal branches. ■

We conclude that the proposed algorithm is efficient as the convexity recognition is performed only by comparing the markings rather than by solving LPs, it is optimal as the branch and bound algorithm guarantees that the global minimum is found, and it is a top down approach based on the notion of the envelope with counterexamples.

B.1.2 Branching Heuristics

Apart from bound techniques, additional heuristics can be used to greatly reduce the computation time. These heuristics provide the hyperplanes with branching priorities according to their expected benefit in the OCR process and allow for deciding on which hyperplane to branch first. The heuristics are intended to quickly find a solution equal or close to the optimal one thus allowing for effective pruning of suboptimal branches.

Specifically, we associate to the hyperplanes the following (descending) branching order:

1. Hyperplanes that separate two non-connected groups of white polyhedra thus allowing us to divide the problem into two subproblems. Connectivity can be easily determined. Since in the subproblems only the black polyhedra within the envelope of white polyhedra are considered, any hyperplane separating the two groups of white polyhedra yields the same

subproblems. Thus, we are only interested in the first hyperplane found with this property.
2. Hyperplanes, such that one half space contains only white polyhedra[1]. If so, we choose the hyperplane yielding the maximal number of white polyhedra.
3. Any remaining hyperplane.

B.2 Non-Disjoint Optimal Complexity Reduction

B.2.1 Boolean Calculus

We start by rather informally recalling basic terminology for Boolean calculus, which can be found in any digital circuit design textbook (see e.g. [154]).

A *Boolean variable* is a $\{0, 1\}$, $\{false, true\}$ or binary variable. A *Boolean expression* is an algebraic statement containing Boolean variables and operators. To improve the readability, we consider the three binary operators '$\cdot$' (AND), '$+$' (OR) and '$\bar{\ }$' (NOT), rather than '$\wedge$', '$\vee$' and '!'. Each appearance of a Boolean variable (or its complement) in an expression is called a *literal*. A *product term* is an ANDed string of literals containing a subset of the given Boolean variables (or their complements), while a *minterm* is a particular product term, in which all variables appear exactly once (complemented or not).

A *Boolean function* uniquely maps some number of Boolean inputs into a Boolean variable using a Boolean expression. A Boolean function can be represented in two canonical forms: *sum of products* and *product of sums*. Here, we focus on sum of products, which are also known as *disjunctive normal form* or *minterm expansion*. A Boolean expression is in disjunctive normal form, if it is a disjunction (sequence of ORs) consisting of one or more disjuncts, each of which is a conjunction (AND) of one or more literals.

B.2.2 Logic Minimization

In this section, we reformulate the complexity reduction problem as a logic minimization problem. Thus, instead of the marking with $\{-, +\}$ elements, we will use a Boolean vector with $\{0, 1\}$ components. Logic minimization is commonly used in digital circuit design, where a given Boolean function is to be minimized in terms of the number of literals and product terms. The number of literals is equivalent to the total number of gate inputs in a circuit

[1] Note that the existence of hyperplanes having in one half space only black polyhedra would contradict the fact that only black polyhedra within the envelope of white polyhedra are taken into account.

and is also proportional to the amount of wiring necessary to implement the Boolean function. The number of product terms, on the other hand, relates to the number of gates and is thus a measure of the circuit area needed.

Logic minimization started in the 1950s with the work of Veitch [256] and Karnaugh [153]. They introduced the K-map to manually minimize simple two-level[2] Boolean functions with up to six variables. A few years later, Quine [218] and McCluskey [189] developed more sophisticated and systematic minimization techniques to obtain a two-level implementation of a given Boolean function with the minimum number of gates. As finding a global minimum is known to belong to the class of $\mathcal{NP}$-complete problems, the Quine-McCluskey algorithm often becomes computationally intractable even for medium sized problems with some 20 variables. To extend the applicability of logic minimization to larger problems, a number of heuristic approaches were introduced in the tools MINI [141] and PRESTO [64].

Inspired by MINI and PRESTO, ESPRESSO-II [63] was designed in the beginning of the 1980s. The aim was to build a logic minimization tool that is able to solve most of the posed problems without the usage of excessive computational power, such that the solution is close or equal to the global minimum. A number of improved heuristics are included in the tool leading to small computation times and solutions that are at least for medium sized problems globally optimal. A great flexibility is achieved by numerous options allowing one to also enforce global optimality for large problems, thus guaranteeing the minimum number of product terms while heuristically minimizing the number of literals. The tool is readily available from the University of California, Berkeley [88], and it has been employed for the examples presented in the remainder of this chapter.

B.2.3 Problem Formulation with Boolean Logic

For a hyperplane arrangement with n hyperplanes $H_i = \{x \in \mathbb{R}^d \mid a_i^T x = b_i\}$, $i \in \{1, \dots, n\}$, in the d-dimensional Euclidian space $\mathbb{R}^d$ we had defined in Section 3.5.3.1 in (3.35) the simplified sign vector $\mathrm{SV} : \mathbb{R}^d \to \{-, +\}^n$, and for a given marking m a polyhedral cell of the arrangement was $\mathcal{P}_m = \{x \in \mathbb{R}^d \mid \mathrm{SV}(x) = m\}$.

Alternatively, we redefine the sign vector as the function $\mathrm{SV}' : \mathbb{R}^d \to \{0, 1\}^n$ that maps x into a Boolean vector with components

$$\mathrm{SV}'_i(x) = \begin{cases} 0 & \text{if } a_i^T x \le b_i, \\ 1 & \text{if } a_i^T x > b_i \end{cases} \quad \text{for } i \in \{1, 2, \dots, n\}\,, \tag{B.1}$$

[2] The term *two-level* relates to the implementation of such a function using digital gates. If the function is in disjunctive normal form, for example, NOT gates constitute the zero level, AND gates the first level, and OR gates the second level.

where we use the dash to distinguish it from the original sign vector (3.35). Accordingly, a polyhedral cell is defined as $\mathcal{P}_\delta = \{x \in \mathbb{R}^d \mid \mathrm{SV}'(x) = \delta\}$ for a given Boolean vector δ, which replaces the marking m. Let $\Delta(\mathcal{R})$ be the image of $\mathrm{SV}'(x)$ for $x \in \mathcal{R} \subseteq \mathbb{R}^d$, namely the collection of all the possible Boolean vectors of all the points in $\mathcal{R}$.

The ' ' element, which extends the sign vector by denoting hyperplanes that are not a facet of the associated polyhedron, is translated into Boolean variables that are removed from the Boolean vector δ. Thus δ has in general a variable number of components.

B.2.4 Algorithm based on Logic Minimization

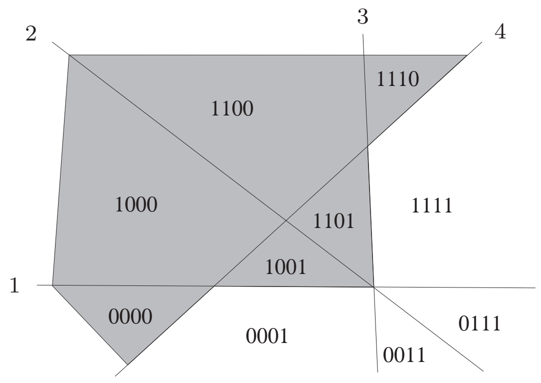
We start by introducing the Boolean function f_W that – given the Boolean vector δ – evaluates whether the color of the polyhedron is white, black or undecided. The color is undecided if the corresponding polyhedron is not a cell in the hyperplane arrangement, i.e. the corresponding δ is not contained in $\Delta(\mathcal{R})$ and the polyhedron features an empty interior. Specifically, f_W yields for δ corresponding to white polyhedra a '1', for black ones a '0' and for empty ones (with an empty interior) an 'X', which is usually referred to as a *don't care* in digital circuit design.

We write f_W in disjunctive normal form. Each minterm in f_W represents a white polyhedron, each literal refers to a facet of such a polyhedron and f_W represents the union of all white polyhedra. Logic minimization can be used to reduce the number of terms in f_W, which is equivalent to reducing the number of white polyhedra, and additionally to reduce the number of literals of each term. The latter refers to reducing the number of facets per polyhedron. These objectives lead in general to overlapping polyhedra. Overlaps not only allow for reducing the overall number of product terms and literals as will be shown in Section B.3, but in particular in digital circuit design, this is a highly desired feature as the occurrence of so-called hazards resulting from different gate propagation times can be reduced or even avoided.

Alternatively, one may represent f_W in form of a truth table[3]. Such a truth table is the preferred input of ESPRESSO-II, which we use to perform the logic minimization. With respect to a Boolean function, a truth table carries the main advantage that it allows one to provide the logic minimization tool with additional structural information, namely empty polyhedra can be specified with an 'X'[4]. During the minimization process, the tool assigns

[3] A truth table is a two-dimensional array with $n+1$ columns, where the first n columns correspond to the possible values of n (Boolean) inputs, and the last column to the Boolean function output. The rows list all possible combinations of inputs together with the corresponding outputs.

[4] The number of rows in the truth table is exponential in n (the length of the Boolean vector δ given by the number of hyperplanes in the arrangement). Yet according



(a) Four hyperplanes in $\mathcal{R} = \mathbb{R}^2$ and the corresponding Boolean vectors δ

δ_1	δ_2	δ_3	δ_4	f_W
0	0	0	0	0
0	0	0	1	1
0	0	1	0	X
0	0	1	1	1
0	1	0	0	X
0	1	0	1	X
0	1	1	0	X
0	1	1	1	1
1	0	0	0	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
1	1	1	1	1

(b) Truth table for white polyhedra

Fig. B.2 Revisited Example B.1 with the hyperplane arrangement, the corresponding Boolean variables and the truth table

to the polyhedra with don't cares a color such that the overall number of product terms and literals becomes minimal.

The result of the logic minimization is either a simplified truth table or a reduced disjunctive normal form. Both representations directly translate into the (overlapping) set of merged polyhedra $\{\mathcal{Q}_i\}_{i=1,\dots,q}$.

We refer to the logic minimization as Algorithm B.2.1. Summing up, for a given color, the truth table with the Boolean function f_W is built, a logic minimization tool (here ESPRESSO-II) is used to derive a simplified truth table minimal in the number of rows (which refer to product terms and polyhedra) and, with second priority, minimal in the number of entries per row (which refer to literals and facets).

Example B.2. Reconsider Example B.1 with the hyperplanes and markings as in Fig. B.1, where we aim at minimizing the number of white polyhedra. Here, we associate with each hyperplane a Boolean variable δ_i, which we collect in the Boolean vector δ, and restate the problem in terms of δ as shown in Fig. B.2(a). The Boolean function for the white polyhedra follows immediately to

to Buck's formula (3.36), the great majority of these rows refers to don't cares. In ESPRESSO-II, the truth table can be passed to the solver by only specifying the rows with $f_W = 0$ and $f_W = 1$, where ESPRESSO-II complements the rows with $f_W = X$ internally. This technique allows for greatly reducing the memory requirement when passing the OCR problem to ESPRESSO-II.

$$f_{\mathrm{W}} = \bar{\delta}_1\bar{\delta}_2\bar{\delta}_3\delta_4 + \bar{\delta}_1\bar{\delta}_2\delta_3\delta_4 + \bar{\delta}_1\delta_2\delta_3\delta_4 + \delta_1\delta_2\delta_3\delta_4\,. \tag{B.2}$$

Thus, a given $x \in \mathcal{R}$ determines δ via (B.1), and f_{W} answers the question, whether x belongs to a white or black polyhedron. Simplifying this function algebraically leads to $f_{\mathrm{W}} = \bar{\delta}_1\bar{\delta}_2\delta_4 + \delta_2\delta_3\delta_4$. For this, we have not exploited the don't cares.

Alternatively, we may translate Fig. B.2(a) into the truth table for white polyhedra shown in Table B.2(b). Here, the empty polyhedra are listed with an 'X'. Using ESPRESSO-II, this additional information allows one to obtain the representation $f_{\mathrm{W}} = \bar{\delta}_1\delta_4 + \delta_3\delta_4$ that is minimal in the number of product terms (polyhedra) and the number of literals (facets). In terms of markings, this result corresponds to $M_m = \{- +,\ ++\}$. Compared to Example B.1, where the disjoint OCR Algorithm based on the markings yielded $M_m = \{- +, ++++\}$, the solution here is reduced by two facets. In general, as we will see in Section B.3, allowing for non-disjoint polyhedra often leads to solutions with less polyhedra and less facets with respect to the case where we restrict ourself to disjoint polyhedra.

Lemma B.5. *Algorithm B.2.1 solves the Non-Disjoint Optimal Complexity Reduction Problem 3.2.*

Proof: Given the resulting white polyhedra $\{\mathcal{Q}_i\}_{i=1,\dots,q}$ the proof contains three parts. Firstly, we need to prove that adding additional hyperplanes to the arrangement does not improve the solution by reducing q. This follows directly from the fact that only facets separating black and white polyhedra are needed as facets for $\mathcal{Q}_i$, and that all these facets are subsets of the hyperplanes contained in the arrangement. Secondly, recall the equivalence between polyhedra and product terms, and facets and literals, respectively. As the logic minimization tool yields the minimal number of product terms (assuming that empty polyhedra are included in the minimization process as don't cares), q is minimal, too. Furthermore, the equivalence ensures that the union of the resulting polyhedra $\mathcal{Q}_i$ equals the union of the original white polyhedra. Thirdly, the minimization of the number of literals leads to the minimal number of facets. ■

B.2.5 Multiple-Valued Logic Minimization

Let a PWA system have $|C|$ different dynamics or feedback-laws, namely the colors $C = \{0, \dots, |C| - 1\}$. With each color we associate a Boolean function and derive a truth table with $|C|$ Boolean outputs. So far, we have considered only OCR problems with two colors (white and black) using Boolean logic minimization. Multiple-color problems were handled by selecting one color $c \in C$ as white color and collecting the remaining colors $C \backslash c$ as black color.

The polyhedra were merged for each color separately, namely by solving $|C|$ independent OCR problems.

Alternatively, associate with the colors the integers $c \in C$, derive one multiple-valued Boolean function (with image C), and set up a truth table with one integer output. Subsequently, consider one OCR for all colors at the same time by running one multiple-valued logic minimization, which is offered for example by ESPRESSO-II. The result is the same as before, but the computation time is in general reduced.

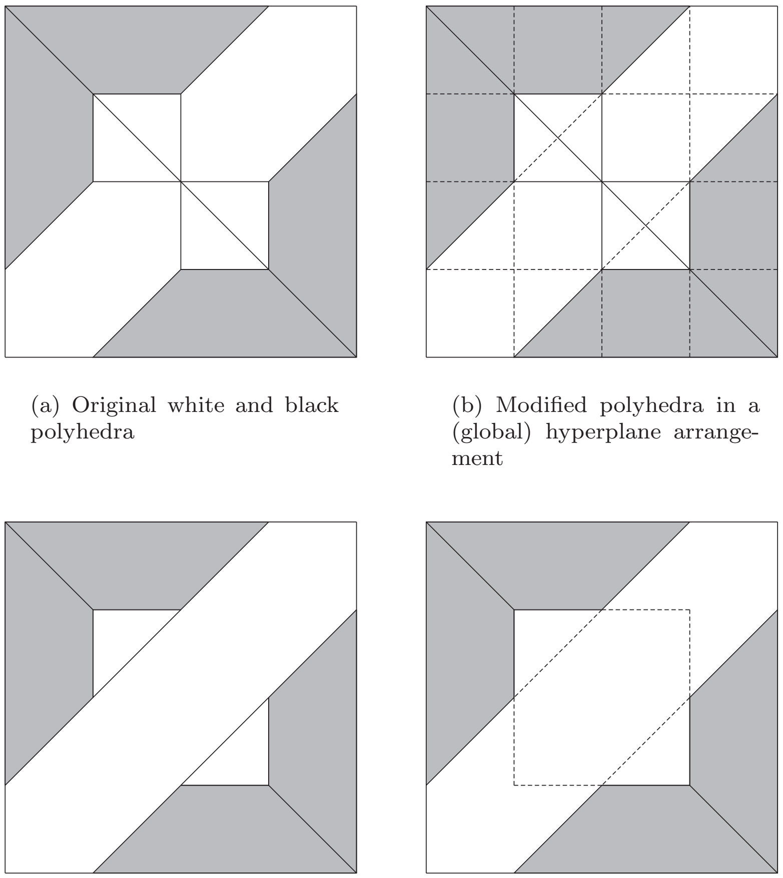
B.3 Local and Global Optimality

B.3.1 Derivation of Global Hyperplane Arrangement

The Algorithms B.1.1 and B.2.1 in the proposed form are only applicable to problems with a globally valid hyperplane arrangement. In this section, we remove Assumption 3.1 and propose two extensions that will allow us to also employ the algorithms for problems with local hyperplane arrangements, or even more general, for problems that altogether lack a hyperplane arrangement.

As mentioned before, PWA models resulting from the Mode Enumeration Algorithm often contain a collection of local hyperplane arrangements, where each one is defined in a polyhedron $\mathcal{R}$, which is a subset of the state-input space, namely $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{U}$. For a given $\mathcal{R}$, the hyperplane arrangement is readily available together with the markings. Thus, OCR can be performed for each subset $\mathcal{R}$, and the overall solution is the union of the local solutions. Even though the results are locally optimal, the overall solution is in general *suboptimal*. As an example, consider two local hyperplane arrangements that each encompass one white polyhedron and a number of black polyhedra, and assume that the union of these two white polyhedra is convex. Using Algorithm B.1.1 or B.2.1 twice (for each local hyperplane arrangement) fails to merge the two white polyhedra, and is thus clearly suboptimal. Nevertheless, if we are interested only in reducing the number of polyhedra but not necessarily in finding the minimal number, and have rather limited time and computational power at our disposal, this approach is meaningful.

If the aim is to derive the *optimal* solution, we need to compute the global hyperplane arrangement by extending the facets of the polyhedra. Here, we give only a brief outline of such an algorithm, which consists of three major steps. First, we collect the facets of all polyhedra. By removing duplicates, we obtain the hyperplane arrangement. Next, we determine the relative position of each polyhedron with respect to each hyperplane. This yields a preliminary set of markings, where we use an additional symbol to denote polyhedra whose interior intersects with a hyperplane. The algorithm resolves these markings in a last step by dividing the corresponding polyhedra into two. As



(a) Original white and black polyhedra

(b) Modified polyhedra in a (global) hyperplane arrangement

(c) Polyhedra resulting from disjoint OCR

(d) Polyhedra resulting from non-disjoint OCR

Fig. B.3 Derivation of cells defined in a global hyperplane arrangement and OCR in Example B.3

this operation involves solving LPs and increases the number of polyhedra significantly, such an algorithm is computational tractable only for problems with a limited complexity. However, a number of enhancements, namely the exploitation of parallel hyperplanes and the removal of redundant hyperplanes reduces the computation time remarkably. We refer to this approach as Algorithm B.3.1.

Example B.3. Consider the sets of white and black polyhedra in Fig. B.3(a). The above proposed algorithm identifies 13 different facets. Since the ones constraining the convex hull of the polyhedra are not considered, the hyperplane arrangement encompasses nine hyperplanes shown as dashed lines in Fig. B.3(b). As a result, the number of white polyhedra is blown up from 6 to 16. OCR restricted to disjoint polyhedra (Algorithm B.1.1) yields three white polyhedra depicted in Fig. B.3(c), whereas Algorithm B.2.1 yields only two white polyhedra that are overlapping as indicated by the dashed lines in Fig. B.3(d).

It is particularly interesting to observe that merging the original white polyhedra in Fig. B.3(a) in a optimal way without using a global hyperplane arrangement would lead to four white polyhedra. Such an approach would require to determine the convexity of each union (each pair, triple, etc.) of white polyhedra by using the algorithms in [32], which resort to solving LPs, and to choose among the convex unions a combination that yields the minimal number of unions and covers all white polyhedra. Despite the fact that such an approach is computationally intractable even for very small problems, it is also in general inferior to the OCR algorithms in terms of the number of resulting polyhedra as the example demonstrates.

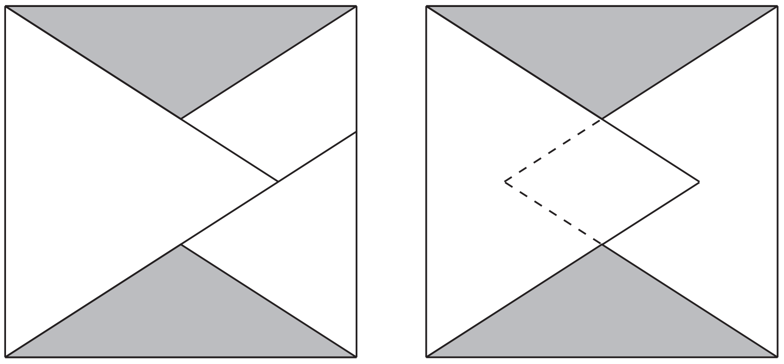
Thus deriving the global hyperplane arrangement first and reducing the complexity subsequently in an optimal way yields in general a lower number of polyhedra compared to the case, where the original polyhedra are merged optimally without the notion of a global hyperplane arrangement. This may serve as a motivation to extend the facets and to derive the hyperplane arrangement, although doing so significantly blows up the number of polyhedra to be considered.

B.3.2 Optimality of Algorithms

In the following, we compare the two OCR algorithms with each other. Both are optimal in the sense that they yield the minimum number of polyhedra for the specific problem they solve (Problems 3.1 and 3.2). Yet, as the problems differ regarding the property whether the resulting polyhedra are required to be disjoint or not, the complexity of the solution in terms of the number of polyhedra and facets differs in general, too.

In Problem 3.1, the resulting polyhedra are required to be disjoint and unions of the original polyhedra. Thus, Problem 3.1 is an optimal *merging* problem, which can be also considered as a *specific* optimal *set partitioning* problem. The problem is specific, since the hyperplanes along which the set can be partitioned are restricted to the hyperplanes given by the facets of the original polyhedra to be merged. This issue is rather subtle, yet we would like to clarify it with the following example.

Example B.4. For given sets of white and black polyhedra, assume we have derived the (global) hyperplane arrangement, split the polyhedra into cells defined in this arrangement, and run subsequently Algorithm B.1.1 that yields


(a) Polyhedra resulting from disjoint OCR

(b) Polyhedra resulting from non-disjoint OCR

Fig. B.4 OCR in Example B.4 visualizing the consequence of restricting the hyperplane arrangement to hyperplanes given by facets of the original white polyhedra

the three white polyhedra shown in Fig. B.4(a). This solution is optimal with respect to Problem 3.1. Yet, adding to the hyperplane arrangement an additional vertical hyperplane that cuts through the center of the figure would reduce the solution to only two white polyhedra. On the other hand, Algorithm B.2.1 leads to the two white polyhedra depicted in Fig. B.4(b), where the dashed lines indicate the overlaps. Adding additional hyperplanes to the arrangement before running Algorithm B.2.1 would not improve on the solution. This holds in general thanks to Lemma B.6 presented at the end of this section.

We conclude that even though Algorithm B.1.1 derives a solution that is minimal in the number of *merged* polyhedra, by introducing additional facets the number of polyhedra might be further reduced. Thus, in general, the merged polyhedra constitute only a suboptimal solution to the (more general) optimal set partitioning problem, which is not addressed here. Nevertheless, even though such a case has been constructed here, they are very rare and have so far been not encountered in applications.

In Problem 3.2, the restriction requiring the resulting polyhedra to be disjoint and unions of the original polyhedra is dropped. Hence strictly speaking, the second problem is not a merging problem but a more general optimal set covering problem. As Problem 3.2 is less restrictive than Problem 3.1, we expect Algorithm B.2.1 to yield in general a lower number of polyhedra and facets than Algorithm B.1.1. This is confirmed by the Examples B.3 and B.4. In particular, as already mentioned above, adding additional hyperplanes does not improve on the solution. This leads to the following key lemma.

Lemma B.6. *Algorithm B.3.1 followed by Algorithm B.2.1 solves the General Non-Disjoint Optimal Complexity Reduction Problem 3.3.*

Proof: The proof follows directly from Problem 3.2, Lemma B.5 and the fact that Algorithm B.2.1 minimizes (with second priority) the number of facets. ■

References

1. L. Chisci A. Bemporad and E. Mosca. On the stabilizing property of siorhc. *Automatica*, 30(12):2013–2015, 1994.
2. J. Acevedo and E.N. Pistikopoulos. A multiparametric programming approach for linear process engineering problems under uncertainty. *Ind. Eng. Chem. Res.*, 36:717–728, 1997.
3. I. Adler and R.D.C. Monteiro. A geometric view of parametric linear programming. *Algorithmica*, 8(2):161–176, 1992.
4. A. Alessio, M. Lazar, A. Bemporad, and W. P. M. H. Heemels. Squaring the circle: An algorithm for obtaining polyhedral invariant sets from ellipsoidal ones. *Automatica*, 43(12):2096–2103.
5. J.C. Allwright and G.C. Papavasiliou. On linear programming and robust model-predictive control using impulse-responses. *Systems & Control Letters*, 18:159–164, 1992.
6. R. Alur, C. Belta, F. Ivančić, V. Kumar, M. Mintz, G.J. Pappas, H. Rubin, and J. Schug. Hybrid modeling and simulation of biomolecular networks. In M.D. Di Benedetto and A. Sangiovanni Vincentelli, editors, *Hybrid Systems: Computation and Control*, volume 2034 of *Lecture Notes in Computer Science*, pages 19–33. Springer-Verlag, 2001.
7. R. Alur, C. Courcoubetis, T.A. Henzinger, and P.-H. Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In A.P. Ravn R.L. Grossman, A. Nerode and H. Rischel, editors, *Hybrid Systems*, volume 736 of *Lecture Notes in Computer Science*, pages 209–229. Springer Verlag, 1993.
8. B. D. O. Anderson and J. B. Moore. *Optimal Control: Linear Quadratic Methods*. Prentice-Hall, Englewood Cliffs, NJ, 1990.
9. P.J. Antsaklis. A brief introduction to the theory and applications of hybrid systems. *Proc. IEEE, Special Issue on Hybrid Systems: Theory and Applications*, 88(7):879–886, July 2000.
10. J. P. Aubin. *Viability theory*. Systems & Control: Foundations & Applications. Birkhaüser, 1991.
11. D. Avis. lrs: A revised implementation of the reverse search vertex enumeration algorithm. *Polytopes, Combinatorics and Computation*, pages 177—198, 2000.
12. D. Avis and K. Fukuda. Reverse search for enumeration. *Discr. App. Math.*, 65:21–46, 1996.
13. T. A. Badgwell and K. R. Muske. Disturbance model design for linear model predictive control. In *Proceedings of the American Control Conference*, volume 2, pages 1621–1626, 2002.
14. E. Balas. Projection with a minimum system of inequalities. *Computational Optimization and Applications*, 10:189–193, 1998.
15. A. Balluchi, L. Benvenuti, M. Di Benedetto, C. Pinello, and A. Sangiovanni-Vincentelli. Automotive engine control and hybrid systems: Challenges and opportunities. *Proc. IEEE*, 88(7):888–912, 2000.
16. A. Balluchi, L. Benvenuti, M. D. Di Benedetto, and A. L. Sangiovanni-Vincentelli. Design of observers for hybrid systems. In C.J. Tomlin and M.R. Greenstreet, editors, *Hybrid Systems: Computation and Control*, volume 2289 of *Lecture Notes in Computer Science*, pages 76–89. Springer-Verlag, Berlin Heidelberg New York, 2002.
17. B. Bank, J. Guddat, D. Klatte, B. Kummer, and K. Tammer. *Non-Linear Parametric Optimization*. Akademie-Verlag, Berlin, 1982.
18. M. Baotic. An efficient algorithm for multi-parametric quadratic programming. Technical Report AUT02-04, Automatic Control Laboratory, ETH Zurich, Switzerland, February 2002.

19. M. Baotic, F. Borrelli, A. Bemporad, and M. Morari. Efficient on-line computation of constrained optimal control. *SIAM Journal on Control and Optimization*, 5:2470–2489, September 2008.
20. M. Baotić and F.D. Torrisi. Polycover. Technical Report AUT03-11, Automatic Control Laboratory, ETHZ, Switzerland, 2003. Available from http://control.ee.ethz.ch/research/publications/publications.msql?
21. Miroslav Barić. *Constrained Control - Computations, Performance and Robustness*. PhD thesis, ETH Zurich, Switzerland, oct 2008.
22. T. Basar and G. J. Olsder. *Dynamic Noncooperative Game Theory, Second Edition*. Classics in Applied Mathematics. SIAM, Philadelphia, 1998.
23. M.S. Bazaraa, H.D. Sherali, and C.M. Shetty. *Nonlinear Programming — Theory and Algorithms*. John Wiley & Sons, Inc., New York, second edition, 1993.
24. A.G. Beccuti, G. Papafotiou, R. Frasca, and M. Morari. Explicit hybrid model predictive control of the DC-DC boost converter. In *IEEE PESC*, Orlando, Florida, USA, June 2007.
25. A. Bemporad. Reducing conservativeness in predictive control of constrained systems with disturbances. In *Proc. 37th IEEE Conf. on Decision and Control*, pages 1384–1391, Tampa, Florida, USA, 1998.
26. A. Bemporad. Efficient conversion of mixed logical dynamical systems into an equivalent piecewise affine form. *IEEE Trans. Automatic Control*, 49(5):832–838, 2004.
27. A. Bemporad, G. Bianchini, and F. Brogi. Passivity analysis and passification of discrete-time hybrid systems. *IEEE Trans. Automatic Control*, 54(4):1004–1009, 2008.
28. A. Bemporad, F. Borrelli, and M. Morari. Piecewise linear robust model predictive control. In *Proc. European Control Conf.*, Porto, Portugal, October 2001.
29. A. Bemporad and S. Di Cairano. Optimal control of discrete hybrid stochastic automata. In M. Morari and L. Thiele, editors, *Hybrid Systems: Computation and Control*, number 3414 in Lecture Notes in Computer Science, pages 151–167. Springer-Verlag, 2005.
30. A. Bemporad, L. Chisci, and E. Mosca. On the stabilizing property of SIORHC. *Automatica*, 30(12):2013–2015, 1994.
31. A. Bemporad, G. Ferrari-Trecate, and M. Morari. Observability and controllability of piecewise affine and hybrid systems. *IEEE Trans. Automatic Control*, 45(10):1864–1876, 2000.
32. A. Bemporad, K. Fukuda, and F.D. Torrisi. On convexity recognition of the union of polyhedra. In *Proc. Int. Conf. on Advances in Convex Analysis and Global Optimization*, Samos, Greece, 2000.
33. A. Bemporad, K. Fukuda, and F.D. Torrisi. Convexity recognition of the union of polyhedra. *Computational Geometry*, 18:141–154, 2001.
34. A. Bemporad, A. Garulli, S. Paoletti, and A. Vicino. A bounded-error approach to piecewise affine system identification. *IEEE Trans. Automatic Control*, 50(10):1567–1580, October 2005.
35. A. Bemporad, W.P.M.H. Heemels, and B. De Schutter. On hybrid systems and closed-loop MPC systems. *IEEE Trans. Automatic Control*, 47(5):863–869, May 2002.
36. A. Bemporad, D. Mignone, and M. Morari. Moving horizon estimation for hybrid systems and fault detection. In *Proc. American Contr. Conf.*, pages 2471–2475, Chicago, IL, June 1999.
37. A. Bemporad and M. Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3):407–427, March 1999.
38. A. Bemporad and M. Morari. Robust model predictive control: A survey. In A. Garulli, A. Tesi, and A. Vicino, editors, *Robustness in Identification and Control*, number 245 in Lecture Notes in Control and Information Sciences, pages 207–226. Springer-Verlag, 1999.

39. A. Bemporad, M. Morari, V. Dua, and E.N. Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38(1):3–20, 2002.
40. A. Bemporad, M. Morari, V. Dua, and E.N. Pistikopoulos. The Explicit Linear Quadratic Regulator for Constrained Systems. *Automatica*, 38(1):3–20, January 2002.
41. A. Bemporad, F.D. Torrisi, and M. Morari. Discrete-time hybrid modeling and verification of the batch evaporator process benchmark. *European Journal of Control*, 2001. to appear.
42. A. Ben-Tal, A. Goryashko, E. Guslitzer, and A. Nemirovski. Adjustable robust solutions of uncertain linear programs. Technical report, Technion (Israel Institute of Technology), Haifa, Israel, 2002.
43. Claude Berge. *Topological Spaces*. Dover Publications, inc., Mineola, New York, 1997.
44. A.B. Berkelaar, K. Roos, and T. Terlaky. The optimal set and optimal partition approach to linear and quadratic programming. In T. Gal and H.J. Greenberg, editors, *Advances in Sensitivity Analysis and Parametric Programming*, volume 6 of *International Series in Operations Research & Management Science*, chapter 6. Kluwer Academic Publishers, 1997.
45. D. P. Bertsekas. *Control of Uncertain Systems with a set–membership description of the uncertainty*. PhD thesis, MIT, 1971.
46. D. P. Bertsekas. *Dynamic Programming and Optimal Control*, volume II. Athena Scientific, Belmont, Massachusetts, 2nd edition, 2001.
47. D. P. Bertsekas and I. B. Rhodes. On the minimax reachability of target sets and target tubes. *Automatica*, 7:233–247, 1971.
48. D.P. Bertsekas. Infinite-time reachability of state-space regions by using feedback control. *IEEE Trans. Automatic Control*, 17:604–613, October 1972.
49. D.P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, Belmont, Massachusetts, 1995.
50. L.T. Biegler and V.M. Zavala. Large-scale nonlinear programming using ipopt: An integrating framework for enterprise-wide dynamic optimization. *Computers & Chemical Engineering*, 33(3):575 – 582, 2009. Selected Papers from the 17th European Symposium on Computer Aided Process Engineering held in Bucharest, Romania, May 2007.
51. F. Blanchini. Ultimate boundedness control for uncertain discrete-time systems via set-induced Lyapunov functions. *IEEE Trans. Automatic Control*, 39(2):428–433, February 1994.
52. F. Blanchini. Set invariance in control — a survey. *Automatica*, 35(11):1747–1768, November 1999.
53. F. Blanchini and S. Miano. *Set-Theoretic Methods in Control*. Birkhäauser, 2009.
54. F. Borrelli, M. Baotić, A. Bemporad, and M. Morari. Dynamic programming for constrained optimal control of discrete-time linear hybrid systems. *Automatica*, 41:1709–1721, October 2005.
55. F. Borrelli, A. Bemporad, M. Fodor, and D. Hrovat. An MPC/hybrid system approach to traction control. *IEEE Trans. Contr. Systems Technology*, 14(3):541–552, May 2006.
56. F. Borrelli, A. Bemporad, and M. Morari. A geometric algorithm for multi-parametric linear programming. *J. Opt. Theory and Applications*, 118(3), September 2003.
57. S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.
58. M.S. Branicky. *Studies in hybrid systems: modeling, analysis, and control*. PhD thesis, LIDS-TH 2304, Massachusetts Institute of Technology, Cambridge, MA, 1995.

59. M.S. Branicky. Multiple Lyapunov functions and other analysis tools for switched and hybrid systems. *IEEE Trans. Automatic Control*, 43(4):475–482, April 1998.
60. M.S. Branicky, V.S. Borkar, and S.K. Mitter. A unified framework for hybrid control: model and optimal control theory. *IEEE Trans. Automatic Control*, 43(1):31–45, 1998.
61. M.S. Branicky and S.K. Mitter. Algorithms for optimal hybrid control. In *Proc. 34th IEEE Conf. on Decision and Control*, New Orleans, USA, December 1995.
62. M.S. Branicky and G. Zhang. Solving hybrid control problems: Level sets and behavioral programming. In *Proc. American Contr. Conf.*, Chicago, Illinois USA, June 2000.
63. R.K. Brayton, G.D. Hachtel, C.T. McMullen, and A.L. Sangiovanni-Vincentelli. *Logic Minimization Algorithms for VLSI Synthesis*. Kluwer Academic Publishers, 1984.
64. D.W. Brown. A state-machine synthesizer – SMS. In *Proceedings of the 18th Design Automation Conference*, pages 301–304, June 1981.
65. R.C. Buck. Partition of space. *American Math. Monthly*, 50:541–544, 1943.
66. R.C. Buck. Partition of space. *American Mathematical Monthly*, 50:541–544, 1943.
67. M. Buss, O. von Stryk, R. Bulirsch, and G. Schmidt. Towards hybrid optimal control. *AT - Automatisierungstechnik*, 48:448–459, 2000.
68. S. Di Cairano, A. Bemporad, and J. Júlvez. Event-driven optimization-based control of hybrid systems with integral continuous-time dynamics. *Automatica*, 45:1243–1251, 2009.
69. S. Di Cairano, A. Bemporad, I. Kolmanovsky, and D. Hrovat. Model predictive control of magnetically actuated mass spring dampers for automotive applications. *Int. J. Control*, 80(11):1701–1716, 2007.
70. M.K. Camlibel, W.P.M.H. Heemels, and J.M. Schumacher. On linear passive complementarity systems. *European Journal of Control*, 8(3), 2002.
71. P.J. Campo and M. Morari. Robust model predictive control. In *Proc. American Contr. Conf.*, volume 2, pages 1021–1026, 1987.
72. P.J. Campo and M. Morari. Model predictive optimal averaging level control. *AIChE Journal*, 35(4):579–591, 1989.
73. C.G. Cassandras, D.L. Pepyne, and Y.Wardi. Optimal control of a class of hybrid systems. *IEEE Trans. Automatic Control*, 46(3):3981–415, 2001.
74. T.M. Cavalier, P.M. Pardalos, and A.L. Soyster. Modeling and integer programming techniques applied to propositional calculus. *Computers Opns Res.*, 17(6):561–570, 1990.
75. S.N. Cernikov. Contraction of finite systems of linear inequalities (in russian). *Doklady Akademiia Nauk SSSR*, 152(5):1075–1078, 1963. (English translation in Societ Mathematics Doklady, Vol. 4, No. 5 (1963), pp.1520–1524).
76. V. Chandru and J.N. Hooker. *Optimization methods for logical inference*. Wiley-Interscience, 1999.
77. B. Chazelle. Convex partitions of polyhedra: A lower bound and worst-case optimal algorithm. *SIAM Journal of Computing*, 13:488–507, August 1984.
78. H. Chen and F. Allgöwer. A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability. *Automatica*, 304(10):1205–1218, 1998.
79. D. Chmielewski and V. Manousiouthakis. On constrained infinite-time linear quadratic optimal control. *Systems & Control Letters*, 29(3):121–130, November 1996.
80. D. Christiansen. *Electronics Engineers' Handbook, 4th edition*. IEEE Press/ McGraw Hill, Inc., 1997.
81. F. J. Christophersen. *Optimal Control and Analysis for Constrained Piecewise Affine Systems*. Dr. sc. ETH Zurich thesis, ETH Zurich, Zurich, Switzerland, August 2006. Available from http://control.ee.ethz.ch.

82. F. J. Christophersen and M. Morari. Further Results on 'Infinity Norms as Lyapunov Functions for Linear Systems'. *IEEE Trans. on Automatic Control*, 52(3):547–553, March 2007.
83. C.R. Cuttler and B.C. Ramaker. Dynamic matrix control—a computer control algorithm. In *Proc. American Contr. Conf.*, volume WP5-B, San Francisco, USA, 1980.
84. G. B. Dantzig, J. Folkman, and N. Shapiro. On the continuity of the minimum set of a continuous function. *Journal of Mathematical Analysis and Applications*, 17:519–548, 1967.
85. B. De Schutter and B. De Moor. The extended linear complementarity problem and the modeling and analysis of hybrid systems. In P. Antsaklis, W. Kohn, M. Lemmon, A. Nerode, and S. Sastry, editors, *Hybrid Systems V*, volume 1567 of *Lecture Notes in Computer Science*, pages 70–85. Springer, 1999.
86. B. De Schutter and T. van den Boom. Model predictive control for max-plus-linear discrete event systems. *Automatica*, 37(7):1049–1056, July 2001.
87. R. DeCarlo, M. Branicky, S. Pettersson, and B. Lennartson. Perspectives and results on the stability and stabilizability of hybrid systems. *Proceedings of the IEEE*, 88(7):1069–1082, 2000.
88. Departement of EECS, University of California, Berkeley. webpage of ESPRESSO-II. online document, 1982. http://www-cad.eecs.berkeley.edu/Software/software.html.
89. J. A. De Doná. *Input Constrained Linear Control*. PhD thesis, Control Group, Department of Engineering, University of Cambridge, Cambridge, 2000.
90. C. E.T. Dórea and J. C. Hennet. (a,b)-invariant polyhedral sets of linear discrete-time systems. *J. Optim. Theory Appl.*, 103(3):521–542, 1999.
91. V. Dua and E.N. Pistikopoulos. An algorithm for the solution of multiparametric mixed integer linear programming problems. *Annals of Operations Research*, to appear.
92. H. Edelsbrunner. *Algorithms in Combinatorial Geometry*. Springer, 1987.
93. G. Ferrari-Trecate, F.A. Cuzzola, D. Mignone, and M. Morari. Analysis of discrete-time piecewise affine and hybrid systems. *Automatica*, 38:2139–2146, 2002.
94. G. Ferrari-Trecate, D. Mignone, and M. Morari. Moving horizon estimation for hybrid systems. *IEEE Trans. Automatic Control*, 47(10):1663–1676, 2002.
95. G. Ferrari-Trecate, M. Muselli, D. Liberati, and M. Morari. A clustering technique for the identification of piecewise affine systems. *Automatica*, 39(2):205–217, February 2003.
96. H.J. Ferreau and M. Diehl H.G. Bock. An online active set strategy to overcome the limitations of explicit mpc. *International Journal of Robust and Nonlinear Control*, 18:816–830, 2008.
97. J.A. Ferrez and K. Fukuda. Implementations of lp-based reverse search algorithms for the zonotope construction and the fixed-rank convex quadratic maximization in binary variables using the ZRAM and the cddlib libraries. Technical report, Mcgill, July 2002.
98. J.A. Ferrez, K. Fukuda, and Th.M. Liebling. Cuts, zonotopes and arrangements. Technical report, EPF Lausanne, Switzerland, November 2001.
99. A. V. Fiacco. Sensitivity analysis for nonlinear programming using penalty methods. *Mathematical Programming*, 10(3):287–311, 1976.
100. A V. Fiacco. *Introduction to sensitivity and stability analysis in nonlinear programming*. Academic Press, London, U.K., 1983.
101. K. Fukuda. *cdd/cdd+ Reference Manual*. Institute for operations Research ETH-Zentrum, ETH-Zentrum, CH-8092 Zurich, Switzerland, 0.61 (cdd) 0.75 (cdd+) edition, December 1997.

102. K. Fukuda. *Polyhedral computation FAQ*, 2000. On line document. Both html and ps versions available from http://www.ifor.math.ethz.ch/staff/fukuda.
103. K. Fukuda, T. M. Liebling, and Christine Lütolf. Extended convex hull. In *12th Canadian Conference on Computational Geometry*, page 57–64, July 2000.
104. K. Fukuda and A. Prodon. Double description method revisited. *Combinatorics and Computer Science*, 1120:91–111, 1996.
105. G.M. Ziegler Gal. *Lectures on Polytopes*. Springer, 1995.
106. T. Gal. *Postoptimal Analyses, Parametric Programming, and Related Topics*. de Gruyter, Berlin, 2nd edition, 1995.
107. T. Gal and H.J. Greenberg (Eds.). *Advances in Sensitivity Analysis and Parametric Programming*, volume 6 of *International Series in Operations Research & Management Science*. Kluwer Academic Publishers, 1997.
108. T. Gal and J. Nedoma. Multiparametric linear programming. *Management Science*, 18:406–442, 1972.
109. C.E. García and A.M. Morshedi. Quadratic programming solution of dynamic matrix control (QDMC). *Chem. Eng. Communications*, 46:73–87, 1986.
110. Stanley J. Gartska and Roger J.B. Wets. On decision rules in stochastic programming. *Mathematical Programming*, 7:117–143, 1974.
111. S.I. Gass and T.L. Saaty. The computational algorithm for the parametric objective function. *Naval Research Logistics Quarterly*, 2:39–45, 1955.
112. T. Geyer. *Low Complexity Model Predictive Control in Power Electronics and Power Systems*. PhD thesis, Swiss Federal Institute of Technology (ETH), ETH-Zentrum, CH-8092 Zurich, Switzerland, March 2005.
113. T. Geyer, F.D. Torrisi, and M. Morari. Efficient Mode Enumeration of Compositional Hybrid Models. In A. Pnueli and O. Maler, editors, *Hybrid Systems: Computation and Control*, volume 2623 of *Lecture Notes in Computer Science*, pages 216–232. Springer-Verlag, 2003.
114. E. G. Gilbert and K.T. Tan. Linear systems with state and control constraints: The theory and application of maximal output admissible sets. *IEEE Trans. Automatic Control*, 36(9):1008–1020, September 1991.
115. E.G. Gilbert and K. Tin Tan. Linear systems with state and control constraints: the theory and applications of maximal output admissible sets. *IEEE Trans. Automatic Control*, 36(9):1008–1020, 1991.
116. N. Giorgetti, A. Bemporad, H.E. Tseng, and D. Hrovat. Hybrid model predictive control application towards optimal semi-active suspension. *Int. J. Control*, 79(5):521–533, 2006.
117. F. Glover. Improved linear integer programming formulations of nonlinear integer problems. *Management Science*, 22(4):455–460, 1975.
118. R. Goebel, R. Sanfelice, and A. Teel. Hybrid dynamical systems. *Control Systems Magazine, IEEE*, 29(2):28–93, April 2009.
119. K. Gokbayrak and C.G. Cassandras. A hierarchical decomposition method for optimal control of hybrid systems. In *Proc. 38th IEEE Conf. on Decision and Control*, pages 1816–1821, Phoenix, AZ, December 1999.
120. G. C. Goodwin and K. S. Sin. *Adaptive Filtering Prediction and Control*. Prentice-Hall, Englewood Cliffs, NJ, 1984.
121. P. J. Goulart, E. C. Kerrigan, and J. M. Maciejowski. Optimization over state feedback policies for robust control with constraints. *Automatica*, 42(4):523–533, 2006.
122. P. Grieder. *Efficient Computation of Feedback Controllers for Constrained Systems*. Dr. Sc. Techn. thesis, Swiss Federal Institute of Technology (ETH), ETH-Zentrum, CH-8092 Zurich, Switzerland, 2004.
123. P. Grieder, F. Borrelli, F. Torrisi, and M. Morari. Constrained infinite horizon linear quadratic regulator. Technical Report AUT02-66, Automatic Control Laboratory, ETH Zurich, Switzerland, July 2002. http://control.ee.ethz.ch.

124. P. Grieder, F. Borrelli, F.D. Torrisi, and M. Morari. Computation of the constrained infinite time linear quadratic regulator. *Automatica*, 40:701–708, April 2004.
125. P. Grieder, M. Kvasnica, M. Baotić, and M. Morari. Low complexity control of piecewise affine systems with stability guarantee. Technical Report AUT03-13, Automatic Control Lab, ETHZ, Switzerland, 2003. Available from http://control.ee.ethz.ch/research/publications/publications.msql?
126. B. Grünbaum. *Convex Polytopes*. Springer-Verlag, second edition, 2000.
127. E. Guslitzer. Uncertainty-immunized solutions in linear programming. Master's thesis, Technion (Israel Institute of Technology), Haifa, Israel, 2002.
128. P.O. Gutman. Online use of a linear programming controller. In G. Ferrate and E.A. Puente, editors, *Software for Computer Control 1982. Proceedings of the Third IFAC/IFIP Symposium*, pages 313–318. Pergamon, Oxford, 1983.
129. P.O. Gutman and M. Cwikel. Admissible sets and feedback control for discrete-time linear dynamical systems with bounded control and states. *IEEE Trans. Automatic Control*, AC-31(4):373–376, 1986.
130. P.O. Gutman and M. Cwikel. An algorithm to find maximal state constraint for discrete-time linear dynamical systems with bounded control and states. *IEEE Trans. Automatic Control*, AC-32(3):251–254, 1987.
131. A Hassibi and S. Boyd. Quadratic stabilization and control of piecewise-linear systems. In *Proc. American Contr. Conf.*, Philadelphia, Pennsylvania USA, June 1998.
132. J.P. Hayes. *Introduction to Digital Logic Design*. Addison-Wesley Publishing Company, Inc., 1993.
133. S. Hedlund and A. Rantzer. Optimal control of hybrid systems. In *Proc. 38th IEEE Conf. on Decision and Control*, pages 3972–3976, Phoenix, AZ, December 1999.
134. S. Hedlund and A. Rantzer. Convex dynamic programming for hybrid systems. *IEEE Trans. Automatic Control*, 47(9):1536–1540, September 2002.
135. W.P.H.M Heemels, B. de Schutter, and A. Bemporad. On the equivalence of classes of hybrid dynamical models. In *Proc. 40th IEEE Conf. on Decision and Control*, pages 364–369, Orlando, Florida, 2001.
136. W.P.M.H. Heemels. *Linear complementarity systems: a study in hybrid dynamics*. PhD thesis, Dept. of Electrical Engineering, Eindhoven University of Technology, The Netherlands, 1999.
137. W.P.M.H. Heemels, J.M. Schumacher, and S. Weiland. Linear complementarity systems. *SIAM Journal on Applied Mathematics*, 60(4):1234–1269, 2000.
138. W.P.M.H. Heemels, B. De Schutter, and A. Bemporad. Equivalence of hybrid dynamical models. *Automatica*, 37(7):1085–1091, July 2001.
139. J. P. Hespanha, S. Bohacek, K. Obraczka, and J. Lee. Hybrid modeling of TCP congestion control. In M.D. Di Benedetto and A. Sangiovanni Vincentelli, editors, *Hybrid Systems: Computation and Control*, volume 2034 of *Lecture Notes in Computer Science*, pages 291–304. Springer-Verlag, 2001.
140. W. M. Hogan. Point-to-set maps in mathematical programming. *SIAM Review*, 15(3):591–603, July 1973.
141. S.J. Hong, R.G. Cain, and D.L. Ostapko. MINI: A heuristic approach for logic minimization. *IBM J. of Res. and Dev.*, 18:443–458, 1974.
142. J.N. Hooker. *Logic-Based Methods for Optimization: Combining Optimization and Constraint Satisfaction*. Wiley, New York, 2000.
143. T. Huerlimann. *Reference Manual for the LPL Modeling Language, Version 4.42*. Departement for Informatics, Université de Fribourg, Switzerland, http://www2-iiuf.unifr.ch/tcs/lpl/TonyHome.htm, 2001.
144. A. Jadbabaie and J. Jie Yu Hauser. Stabilizing receding horizon control of nonlinear systems: a control lyapunov function approach. volume 3, pages 1535 –1539 vol.3, 1999.

145. M. Johannson and A. Rantzer. Computation of piece-wise quadratic Lyapunov functions for hybrid systems. *IEEE Trans. Automatic Control*, 43(4):555–559, 1998.
146. T.A. Johansen, J. Kalkkuhl, J. Lüdemann, and I. Petersen. Hybrid control strategies in ABS. In *Proc. American Contr. Conf.*, Arlington, Virginia, June 2001.
147. K H Johansson, M Egerstedt, J Lygeros, and S Sastry. On the regularization of Zeno hybrid automata. *System & Control Letters*, 38:141–150, 1999.
148. C. Jones, P. Grieder, and S. Raković. A Logarithmic-Time Solution to the Point Location Problem for Closed-Form Linear MPC. In *IFAC World Congress*, Prague, Czech Republic, July 2005.
149. C.N. Jones, E.C. Kerrigan, and J.M. Maciejowski. Equality set projection: A new algorithm for the projection of polytopes in halfspace representation. Technical Report CUED Technical Report CUED/F-INFENG/TR.463, Department of Engineering, Cambridge University, UK, 2004. `http://www-control.eng.cam.ac.uk/ cnj22/`.
150. P. Julian, M. Jordan, and A. Desages. Canonical piecewise-linear approximation of smooth functions. *IEEE Trans. Circuits and Systems — I: Fundamental Theory and Applications*, 45(5):567–571, May 1998.
151. A.A. Julius and A. J. van der Schaft. The maximal controlled invariant set of switched linear systems. In *Proc. of the 41st IEEE Conference on Decision and Control*, Las Vegas, Nevada, USA, December 2002.
152. A. Juloski, S. Weiland, and M.Heemels. A bayesian approach to identification of hybrid systems. *Proc. 43th IEEE Conf. on Decision and Control*, 2004.
153. M. Karnaugh. A map method for synthesis of combinational logic circuits. *AIEE Transactions on Communications and Electronics*, 72:593–599, November 1953.
154. R.H. Katz. *Contemporary Logic Design*. Benjamin/Cummings Publishing Company, 1994.
155. S. Keerthi and E. Gilbert. Computation of minimum-time feedback control laws for discrete-time systems with state-control constraints. *Automatic Control, IEEE Transactions on*, 32(5):432–435, 1987.
156. S. S. Keerthi and K. Sridharan. Solution of parametrized linear inequalities by fourier elimination and its applications. *J. Opt. Theory and Applications*, 65(1):161–169, 1990.
157. S.S. Keerthi and E.G. Gilbert. Optimal infinite-horizon feedback control laws for a general class of constrained discrete-time systems: stability and moving-horizon approximations. *J. Opt. Theory and Applications*, 57:265–293, 1988.
158. E. C. Kerrigan. *Robust Constraints Satisfaction: Invariant Sets and Predictive Control*. PhD thesis, Department of Engineering, University of Cambridge, Cambridge, England, 2000.
159. E. C. Kerrigan and J. M. Maciejowski. On robust optimization and the optimal control of constrained linear systems with bounded state disturbances. In *In Proc. 2003 European Control Conference*, Cambridge, UK, September 2003.
160. H. K. Khalil. *Nonlinear Systems*. Prentice Hall, 2nd edition, 1996.
161. H. Kiendl, J. Adamy, and P. Stelzner. Vector norms as Lyapunov functions for linear systems. *IEEE Trans. Automatic Control*, 37(6):839–842, June 1992.
162. D. Klatte and G. Thiere. Error bounds for solutions of linear equations and inequalities. *ZOR - Mathematical Methods of Operations Research*, 41:191–214, 1995.
163. I. Kolmanovsky and E. G. Gilbert. Theory and computation of disturbance invariant sets for discrete-time linear systems. *Mathematical Problems in Egineering*, 4:317–367, 1998.
164. M.V. Kothare, V. Balakrishnan, and M. Morari. Robust constrained model predictive control using linear matrix inequalities. *Automatica*, 32(10):1361–1379, 1996.

165. B. Kouvaritakis, J.A. Rossiter, and J. Schuurmans. Efficient robust predictive control. *IEEE Trans. Automatic Control*, 45(8):1545–1549, 2000.
166. M. Kvasnica, P. Grieder, M. Baotić, and F.J. Christophersen. *Multi-Parametric Toolbox (MPT): User's Manual*, 2004. Available from http://control.ee.ethz.ch/~mpt/.
167. M. Kvasnica, P. Grieder, M. Baotić, and M. Morari. Multi Parametric Toolbox (MPT). In *Hybrid Systems: Computation and Control*, Lecture Notes in Computer Science, Volume 2993, pages 448–462, Pennsylvania, Philadelphia, USA, March 2004. Springer Verlag. Available from http://control.ee.ethz.ch/research/publications/publications.msql?
168. M. Lazar and W. P. M. H. Heemels. Predictive control of hybrid systems: Input-to-state stability results for sub-optimal solutions. *Automatica*, 45(1):180–185, 2009.
169. M. Lazar, W. P. M. H. Heemels, and A. R. Teel. Lyapunov functions, stability and input-to-state stability subtleties for discrete-time discontinuous systems. *IEEE Transactions on Automatic Control*, 54(10):2421–2425, 2009.
170. M. Lazar, W. P. M. H. Heemels, S. Weiland, and A. Bemporad. Stabilizing model predictive control of hybrid systems. *IEEE Transactions on Automatic Control*, 51(11):1813–1818, 2006.
171. M. Lazar, D. Muñoz de la Peña, W. P. M. H. Heemels, and T. Alamo. On input-to-state stability of min-max nonlinear model predictive control. *Systems & Control Letters*, 57:39–48, 2008.
172. J. H. Lee and Z. Yu. Worst-case formulations of model predictive control for systems with bounded parameters. *Automatica*, 33(5):763–781, 1997.
173. F. L. Lewis and V. L. Syrmos. *Optimal Control*. John Wiley & Sons, Inc., New York, 1995.
174. D. Liberzon. *Switching in Systems and Control*. Systems and Control: Foundations and Application. Birkhauser, Boston, MA, June 2003.
175. B. Lincoln and A. Rantzer. Optimizing linear system switching. In *Proc. 40th IEEE Conf. on Decision and Control*, pages 2063–2068, Orlando, FL, USA, 2001.
176. Y.-C. Liu and C. B. Brosilow. Simulation of large scale dynamic systems—I. modular integration methods. *Computers & Chemical Engineering*, 11(3):241–253, 1987.
177. J. Löfberg. *Minimax approaches to robust model predictive control*. PhD thesis, Linköping University, Sweden, April 2003.
178. J. Lygeros, D.N. Godbole, and S. Sastry. A game theoretic approach to hybrid system design. In R. Alur and T. Henzinger, editors, *Hybrid Systems III*, volume 1066 of *Lecture Notes in Computer Science*, pages 1–12. Springer Verlag, 1996.
179. J. Lygeros, C. Tomlin, and S. Sastry. Controllers for reachability specifications for hybrid systems. *Automatica*, 35(3):349–370, 1999.
180. J.M. Maciejowski. *Predictive Control with Constraints*. Prentice Hall, 2002.
181. Urban Maeder, Francesco Borrelli, and Manfred Morari. Linear offset-free model predictive control. *Automatica*, 45(10):2214 – 2222, 2009.
182. S. Mahapatra. *Stability of Hybrid Haptic Systems*. PhD thesis, University of Illinois, Chicago, Illinois, 2003.
183. O.L. Mangasarian and J.B. Rosen. Inequalities for stochastic nonlinear programming problems. *Opreations Research*, 12(1):143–154, 1964.
184. D. Q. Mayne. Constrained Optimal Control. European Control Conference, Plenary Lecture, September 2001.
185. D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36:789–814, 2000.
186. D.Q. Mayne. Constrained optimal control. *European Control Conference, Plenary Lecture*, September 2001.
187. D.Q. Mayne. Control of constrained dynamic systems. *European Jornal of Control*, 7:87–99, 2001.

188. D.Q. Mayne, J.B. Rawlings, C.V. Rao, and P.O.M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, June 2000.
189. E.J. McCluskey. Minimization of boolean functions. *Bell Systems Technical Journal*, 35:1417–1444, November 1956.
190. T.A. Meadowcroft, G. Stephanopoulos, and C. Brosilow. The Modular Multivariable Controller: 1: Steady-state properties. *AIChE Journal*, 38(8):1254–1278, 1992.
191. S. Mehrotra and R.D.C. Monteiro. Parametric and range analysis for interior point methods. Technical report, Dept. of Systems and Industrial Engineering, University of Arizona, Tucson, USA, 1992.
192. E. Mendelson. *Introduction to mathematical logic*. Van Nostrand, 1964.
193. J. A. Mendez, B. Kouvaritakis, and J. A. Rossiter. State space approach to interpolation in MPC. *Int. J. Robust Nonlinear Control*, 10(1):27–38, January 2000.
194. D. Mignone. *Control and Estimation of Hybrid Systems with Mathematical Optimization*. PhD thesis, Automatic Control Labotatory - ETH, Zurich, 2002.
195. D. Mignone, A. Bemporad, and M. Morari. A framework for control, fault detection, state estimation and verification of hybrid systems. *Proc. American Contr. Conf.*, pages 134–138, June 1999.
196. R. Milman and E.J. Davison. A fast mpc algorithm using nonfeasible active set methods. *Journal of Optimization Theory and Applications*, 139(3):591 – 616, 2008.
197. G. Mitra, C. Lucas, and Moody S. Tool for reformulating logical forms into zero-one mixed integer programs. *European Journal of Operational Research*, 71:262–276, 1994.
198. R. Möbus, M. Baotic, and M. Morari. Multi-objective adaptive cruise control. In O. Maler and A. Pnueli, editors, *Hybrid Systems: Computation and Control*, number 2623 in Lecture Notes in Computer Science, pages 359–374. Springer-Verlag, 2003.
199. M. Morari and G. Stephanopoulos. Minimizing unobservability in inferential control schemes. *Int. J. Control*, 31:367–377, 1980.
200. M. Morari and G. Stephanopoulos. Studies in the synthesis of control structures for chemical processes; Part III: Optimal selection of secondary measurements within the framework of state estimation in the presence of persistent unknown disturbances. *AIChE J.*, 26:247–260, 1980.
201. K. R. Muske and T. A. Badgwell. Disturbance modeling for offset-free linear model predictive control. *Journal of Process Control*, 12:617–632, 2002.
202. G.L. Nemhauser and L.A. Wolsey. *Integer and Combinatorial Optimization*. Wiley, 1988.
203. G. Pannocchia. Robust disturbance modeling for model predictive control with application to multivariable ill-conditioned processes. *J. Process Control*, 13(8):693–701, 2003.
204. G. Pannocchia and J. B. Rawlings. Disturbance models for offset-free model predictive control. *AIChE Journal*, 49(2):426–437, 2003.
205. S. Paoletti. *Identification of Piecewise Affine Models*. PhD thesis, Dept. Information Engineering, University of Siena, Italy, 2004.
206. G. Lafferriere G.J. Pappas and S. Sastry. Reachability analysis of hybrid systems using bisimulations. In *Proc. 37th IEEE Conf. on Decision and Control*, pages 1623–1628, Tampa, Florida, USA, 1998.
207. T. Park and P.I. Barton. Implicit model checking of logic-based control systems. *AIChE Journal*, 43(9):2246–2260, 1997.
208. P. A. Parrilo. *Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization*. Ph. D. thesis, California Institute of Technology, Pasadena, CA, USA, 2000.

209. S. Pettersson and B. Lennartson. Stability and robustness for hybrid systems. In *Proc. 35th IEEE Conf. on Decision and Control*, pages 1202–1207, Kobe, Japan, 1996.
210. S. Pettersson and B. Lennartson. Exponential stability of hybrid systems using piecewise quadratic lyapunov functions resulting in LMIs. In *Proc. IFAC World Congress*, pages 103–108, Beijing, China, July 1999.
211. B. Piccoli. Necessary conditions for hybrid optimization. In *Proc. 38th IEEE Conf. on Decision and Control*, Phoenix, Arizona USA, December 1999.
212. A. Pogromski, M. Jirstrand, and P. Spangeus. On stability and passivity of a class of hybrid systems. In *Proc. 37th IEEE Conf. on Decision and Control*, pages 3705–3710, Tampa, Florida, USA, 1998.
213. A. Polanski. On infinity norms as Lyapunov functions for linear systems. *IEEE Trans. Automatic Control*, 40(7):1270–1274, July 1995.
214. S. Prajna and A. Papachristodoulou. Analysis of switched and hybrid systems – beyond piecewise quadratic methods. In *Proceedings of the American Control Conference*, volume 4, pages 2779–2784, 2003.
215. A.I. Propoi. Use of linear programming methods for synthesizing sampled-data automatic systems. *Automation and Remote Control*, 24(7):837–844, 1963.
216. S.J. Qin and T.A. Badgwell. An overview of industrial model predictive control technology. In *Chemical Process Control - V*, volume 93, no. 316, pages 232–256. AIChe Symposium Series - American Institute of Chemical Engineers, 1997.
217. S.J. Qin and T.A. Badgwell. A survey of industrial model predictive control technology. *Control Engineering Practice*, 11:733–764, 2003.
218. W.V. Quine. A way to simplify truth functions. *American Math. Monthly*, 62:627–631, November 1955.
219. S. V. Raković, P. Grieder, M. Kvasnica, D. Q. Mayne, and M. Morari. Computation of invariant sets for piecewise affine discrete time systems subject to bounded disturbances. In *IEEE Conference on Decision and Control*, pages 1418–1423, December 2004.
220. S. V. Rakovic, E. C. Kerrigan, and D. Q. Mayne. Reachability computations for constrained discrete-time systems with state- and input-dependent disturbances. In *Proc. of the Conf. on Decision and Control, Maui, Hawaii, USA*, pages 3905–3910, December 2003.
221. R. Raman and I.E. Grossmann. Relation between MILP modeling and logical inference for chemical process synthesis. *Computers Chem. Engng.*, 15(2):73–84, 1991.
222. A. Rantzer and M. Johansson. Piecewise linear quadratic optimal control. *IEEE Trans. Automatic Control*, 45(4):629–637, April 2000.
223. J. Richalet, A. Rault, J. L. Testud, and J. Papon. Algorithmic control of industrial processes. In *Fourth IFAC symposium on identixcation and system parameter estimation*, volume WP5-B, pages 1119–1167, 1976.
224. J. Richalet, A. Rault, J. L. Testud, and J. Papon. Model predictive heuristic control-application to industrial processes. *Automatica*, 14:413–428, 1978.
225. P. Riedinger, F.Kratz, C. Iung, and C.Zanne. Linear quadratic optimization for hybrid systems. In *Proc. 38th IEEE Conf. on Decision and Control*, Phoenix, Arizona USA, December 1999.
226. S. M. Robinson. Some continuity properties of polyhedral multifunctions. *Mathematical Proggraming Study*, 14:206–214, 1981.
227. S. M. Robinson and R.H. Day. A sufficient condition for continuity of optimal sets in mathematical programming. *Journal of Mathematical Analysis and Applications*, 45:506–511, 1974.
228. J. Roll, A. Bemporad, and L. Ljung. Identification of piecewise affine systems via mixed-integer programming. *Automatica*, 40(1):37–50, 2004.

229. Lino O. Santos, Paulo A. F. N. A. Afonso, José A. A. M. Castro, Nuno M. C. Oliveira, and Lorenz T. Biegler. On-line implementation of nonlinear mpc: an experimental case study. *Control Engineering Practice*, 9(8):847 – 857, 2001.
230. M. Schechter. Polyhedral functions and multiparametric linear programming. *J. Opt. Theory and Applications*, 53(2):269–280, May 1987.
231. C. Scherer and S. Weiland. Linear Matrix Inequalities in Control. Technical report, Delft Center for Systems and Control, Delft University of Technology, Delft, The Netherlands, January 2005. Available from http://www.dcsc.tudelft.nl/~cscherer/2416/lmi05.pdf.
232. P.O.M. Scokaert and D.Q. Mayne. Min-max feedback model predictive control for constrained linear systems. *IEEE Trans. Automatic Control*, 43(8):1136–1142, 1998.
233. P.O.M. Scokaert and J.B. Rawlings. Constrained linear quadratic regulation. *IEEE Trans. Automatic Control*, 43(8):1163–1169, 1998.
234. C. Seatzu, D. Corona, A. Giua, and A. Bemporad. Optimal control of continuous-time switched affine systems. *IEEE Trans. Automatic Control*, 2003. Accepted for publication as a regular paper.
235. C. Seatzu, D. Corona, A. Giua, and A. Bemporad. Optimal control of continuous-time switched affine systems. *IEEE Trans. Automatic Control*, 51(5):726–741, 2006.
236. M.M. Seron, J.A. DeDoná, and G.C. Goodwin. Global analytical model predictive control with input constraints. In *Proc. 39th IEEE Conf. on Decision and Control*, pages 154–159, 2000.
237. J. Serra. *Image Analysis and Mathematical Morphology, Vol II: Theoretical advances*. Academic Press, 1988.
238. M.S. Shaikh and P.E. Caines. On the optimal control of hybrid systems: optimization of trajectories, switching times and location schedules. In *6th Int. Workshop on Hybrid Systems: Computation and Control*, Prague, The Czech Republic, 2003.
239. B.I. Silva, O. Stursberg, B.H. Krogh, and S. Engell. An assessment of the current status of algorithmic approaches to the verification of hybrid systems. In *Proc. 40th IEEE Conf. on Decision and Control*, pages 2867–2874, Orlando, Florida, December 2001.
240. E. D. Sontag. Nonlinear regulation: The piecewise linear approach. *IEEE Trans. Automatic Control*, 26(2):346–358, April 1981.
241. E.D. Sontag. Nonlinear regulation: The piecewise linear approach. *IEEE Trans. Automatic Control*, 26(2):346–358, April 1981.
242. E.D. Sontag. Interconnected automata and linear systems: A theoretical framework in discrete-time. In R. Alur, T.A. Henzinger, and E.D. Sontag, editors, *Hybrid Systems III - Verification and Control*, number 1066 in Lecture Notes in Computer Science, pages 436–448. Springer-Verlag, 1996.
243. J. Spjotvold, E.C. Kerrigan, C.N. Jones, P. Tondel, and T.A Johansen. On the facet-to-facet property of solutions to convex parametric quadratic programs. *Automatica*, 42(12):2209–2214, December 2006.
244. R. Suard, J. Löfberg, P. Grieder, M. Kvasnica, and M. Morari. Efficient computation of controller partitions in multi-parametric programming. In *Proc. 43th IEEE Conf. on Decision and Control*, pages 3643–3648, Bahamas, December 2004.
245. H.J. Sussmann. A maximum principle for hybrid optimal control problems. In *Proc. 38th IEEE Conf. on Decision and Control*, Phoenix, Arizona USA, December 1999.
246. M. Sznaier and M.J. Damborg. Suboptimal control of linear systems with state and control inequality constraints. In *Proc. 26th IEEE Conf. on Decision and Control*, volume 1, pages 761–762, 1987.

247. C.J. Tomlin, J. Lygeros, and S.S. Sastry. A game theoretic approach to controller design for hybrid systems. *Proceeding of IEEE*, 88, July 2000.
248. P. Tøndel, T.A. Johansen, and A. Bemporad. An algorithm for multi-parametric quadratic programming and explicit MPC solutions. In *Proc. 40th IEEE Conf. on Decision and Control*, December 2001.
249. P. Tøndel, T.A. Johansen, and A. Bemporad. Evaluation of piecewise affine control via binary search tree. *Automatica*, 39(5):945–950, 2003.
250. F. Torrisi, A. Bemporad, G. Bertini, P. Hertach, D. Jost, and Mignone D. Hysdel 2.0.5 - user manual. Technical Report AUT02-28, Automatic Control Laboratory, ETH Zurich, 2002.
251. F.D. Torrisi and A. Bemporad. HYSDEL — A tool for generating computational hybrid models. *IEEE Trans. Contr. Systems Technology*, 12(2):235–249, March 2004.
252. M.L. Tyler and M. Morari. Propositional logic in control and monitoring problems. *Automatica*, 35(4):565–582, 1999.
253. V.I. Utkin. Variable structure systems with sliding modes. *IEEE Trans. Automatic Control*, 22(2):212–222, April 1977.
254. A.J. van der Schaft and J.M. Schumacher. Complementarity modelling of hybrid systems. *IEEE Trans. Automatic Control*, 43:483–490, 1998.
255. D.H. vanHessem and O.H. Bosgra. A conic reformulation of model predictive control including bounded and stochastic disturbances under state and input constraint. In *Proc. 41th IEEE Conf. on Decision and Control*, pages 4643–4648, Las Vegas, Nevada, USA, 2002.
256. E.W. Veitch. A chart method for simplifying boolean functions. In *Proceedings of the Association for Computing Machinery*, pages 127–133, May 1952.
257. B. Vibhor and F. Borrelli. On a property of a class of offset-free model predictive controllers. In *Proceedings of the American Control Conference*, June 2008.
258. R. Vidal, S. Soatto, Y. Ma, and S. Sastry. An algebraic geometric approach to the identification of a class of linear hybrid systems. In *Proc. 42th IEEE Conf. on Decision and Control*, pages 167–172, Maui, Hawaii, 2003.
259. D. W. Walkup and R.J.-B. Wets. A lipschitzian characterizations of convex polyhedra. *Proceeding of the American Mathematical Society*, 20:167–173, 1969.
260. Y. Wang and S. Boyd. Fast model predictive control using online optimization. In *Proceedings of the 2008 IFAC World Congress*, pages 6974–6997, July 2008.
261. H.P. Williams. Logical problems and integer programming. *Bulletin of the Institute of Mathematics and Its Applications*, 13:18–20, 1977.
262. H.P. Williams. *Model Building in Mathematical Programming.* John Wiley & Sons, Third Edition, 1993.
263. H. Witsenhausen. A class of hybrid-state continuous-time dynamic systems. *IEEE Trans. Automatic Control*, 11(2):161–167, 1966.
264. H.S. Witsenhausen. A min-max control problem for sampled linear systems. *IEEE Trans. Automatic Control*, 13(1):5–21, 1968.
265. X. Xu and P.J. Antsaklis. Results and perspectives on computational methods for optimal control of switched systems. In O. Maler and A. Pnueli, editors, *Hybrid Systems: Computation and Control*, number 2623 in Lecture Notes in Computer Science, pages 540–555. Springer-Verlag, 2003.
266. X. Xu and P.J. Antsaklis. Optimal control of switched systems based on parameterization of the switching instants. *IEEE Trans. Automatic Control*, 49(1):2–16, 2004.
267. L.A. Zadeh and L.H. Whalen. On optimal control and linear programming. *IRE Trans. Automatic Control*, 7:45–46, 1962.
268. E. Zafiriou and M. Morari. A general controller synthesis methodology based on the IMC structure and the H_2-, H_∞- and μ-optimal control theories. *Computers & Chemical Engineering*, 12(7):757–765, 1988.

269. M. Zefran, F. Bullo, and M. Stein. A notion of passivity for hybrid systems. In *Proc. 40th IEEE Conf. on Decision and Control*, pages 768–773, Orlando, FL, 2001.
270. G. M. Ziegler. *Lectures on Polytopes.* Springer, 1994.