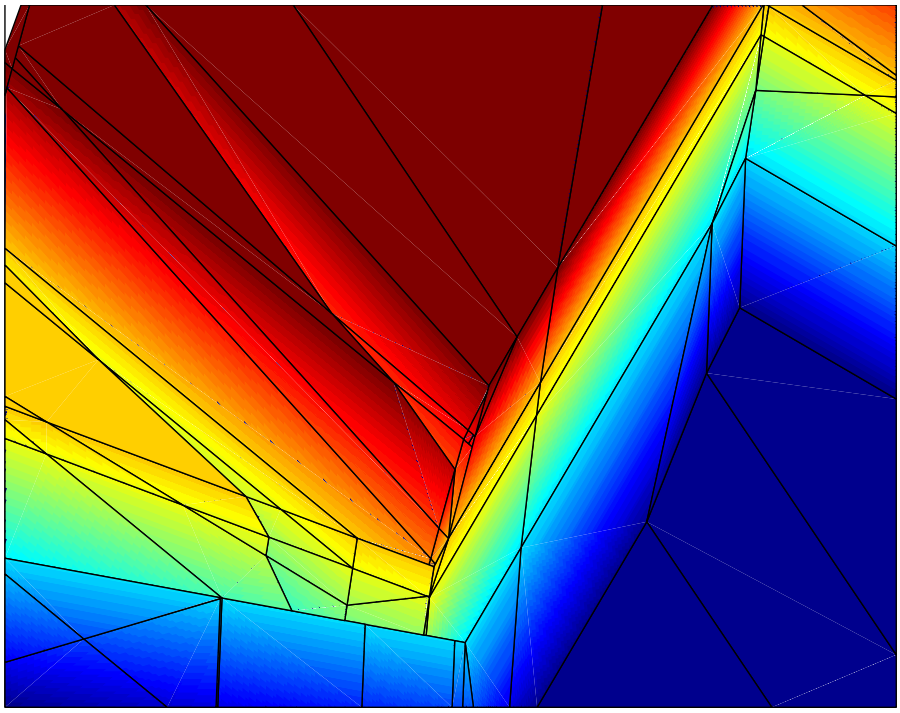
Low Complexity Model Predictive Control in Power Electronics and Power Systems



Tobias Geyer

PhD Thesis ETH No. 15953

Diss. ETH No. 15953

Low Complexity Model Predictive Control in Power Electronics and Power Systems

A dissertation submitted to the
SWISS FEDERAL INSTITUTE OF TECHNOLOGY (ETH) ZURICH

for the degree of
Doctor of Sciences

presented by

Tobias Geyer
Dipl. El.-Ing. ETH Zürich
born 25.5.1975
citizen of Biberach/Riß, Germany

accepted on the recommendation of

Prof. Dr. Manfred Morari, examiner
Prof. Dr. Göran Andersson, co-examiner
Prof. Dr. Ian A. Hiskens, co-examiner

2005

To the ONE

Acknowledgements

First of all and most importantly, I am deeply indebted to my supervisor Prof. Manfred Morari for his support, motivation and inspiring discussions. What I appreciated most were the many opportunities he gave me and the freedom he has offered me that allowed me to collaborate with the people I want, to find the research topics I am most interested in and to travel a lot. Moreover, by providing an outstanding research environment and by attracting so many different and skilled people from all over the world, the lab is turned into a unique and highly rewarding place to work at.

In the beginning, a PhD is never easy. Hence, I am grateful to Domenico Mignone for helping me in my initial phase and for providing me with many control tools, and Gianni Ferrari-Trecate for getting me started in working on ABB's case study on power systems. I would like to thank Mats Larsson from ABB's Corporate Research in Baden-Dättwil, Switzerland, for always having time to answer my numerous questions about power systems in general and with respect to the ABB case study. Furthermore, I am grateful to Andrea G. Beccuti for taking over the main work load on power systems and collaborating with me on this topic.

Looking back, the productive phase of my PhD was triggered by Fabio D. Torrisi, who proposed me to work on the mode enumeration problem of hybrid systems, introduced me to the field of Computational Geometry, affected and inspired me with his creativity, taught me how to write, and showed me how to focus on the important aspects of research. I am deeply grateful for that.

During the second part of my PhD, I have mainly worked with George Papafotiou on the optimal control of power electronics. His deep insight in the field of power electronics and our long and creative discussions have made this collaboration particularly fruitful. Without him, many results couldn't have been obtained and the part on power electronics would be significantly shorter and weaker. The project on Direct Torque Control was carried out in close collaboration with ABB. I would like to express my appreciation to Christian Stulz, Pieder Jörg and Petri Schroderus from ABB ATDD in Turgi, Switzerland, and to Andreas Poncet from ABB's Corporate Research in Baden-Dättwil, Switzerland, for their cooperation.

Moreover, I would like to thank Prof. Adolf H. Glattfelder, Prof. Pablo A. Parrilo, Prof. Göran Andersson and Prof. Ian A. Hiskens for their help, discussions and advice.

Mato Baotić helped me significantly with all my questions about virtually everything related to control and optimization. Indeed, I have discussed most of my ideas first with him due to his invaluable feedback. Thanks go also to Frank J. Christophersen, who is not only the lab's LaTeX guru, but also proved me that "LaTeX is your friend".

I am also glad to have had the opportunity to work with Lars F.S. Larsen from Danfoss Central R&D in Nordborg, Denmark, who stayed at the lab for a few month, and my two students Gaby and Martin who pursued their master thesis with me in collaboration with Rittmeyer in Zug, Switzerland, and achieved excellent results that got later awarded, patented and published.

Many of my colleagues became (very) good friends – in particular George, Mato and Frank, Philipp, Lars, Rainer, and most importantly Ele with whom I spent wonderful years. Then, there were the photo people Frank, Ele, Vale, Marc, Helfried and Miroslav who turned the lab into a photographic hot spot. But also all the other members of the lab provided a great and pleasant atmosphere. These include, just to mention a few, Kristian, Pascal, Gültekin, Dominik, Tobias, Andreas and the administrative staff members – in particular Alice and Esther. I would like to express special thanks to Domenico for raising my sports addiction and to Chantal, Iris, Markus and Claudia for providing amazing opportunities at ETH Zürich to live it out.

Yet, in the end, I am most grateful to my parents who were always there to support me and to share my thoughts, and to my friends in Zürich Farshad, Iris, Rainer, Bettina, Andreas and Joana who reminded me that there is a life beyond the lab.

Tobias Geyer
Zürich, March 2005

Abstract

This thesis focuses on Model Predictive Control (MPC) of discrete-time hybrid systems. Hybrid systems contain continuous and discrete valued components, and are located at the intersection between the fields of control theory and computer science. MPC uses an internal model of the controlled plant to predict the future evolution of the controlled variables over a prediction horizon. A cost function is minimized to obtain the optimal control input sequence, which is applied to the plant by means of a receding horizon policy. The latter implies that only the first control input of the input sequence is implemented, the horizon is shifted by one time-step and the above procedure is repeated at the next sampling instant. Most importantly, theory and tools are available to off-line derive the piecewise affine (PWA) state-feedback control law. Hence, any time-consuming on-line computation of the control input is avoided and plants with high sampling frequencies can be controlled.

The thesis is divided into two parts: The first part is devoted to theory and algorithms, whereas the second part tackles applications in the fields of power electronics and power systems. In the first part, using the notion of cell enumeration in hyperplane arrangements from computational geometry, we propose an algorithm that efficiently enumerates all feasible modes of a composition of hybrid systems. This technique allows the designer to evaluate the complexity of the compound model, to efficiently translate the model into a PWA representation, and to reduce the computational burden of optimal control schemes by adding cuts that prune infeasible modes from the model.

With respect to implementation, an important issue is the complexity reduction of PWA state-feedback controllers. Hence, we propose two algorithms that solve the problem of deriving a PWA representation that is both equivalent to the given one and minimal in the number of regions. As both algorithms refrain from solving additional Linear Programs, they are not only optimal but also computationally feasible. In many cases, the optimal complexity reduction constitutes an enabling technique when implementing the optimal controllers as look-up tables in hardware.

In the second part of the thesis, we consider the field of power electronics that is intrinsically hybrid, since the positions of semiconductor switches are described by binary variables. The fact that the methodologies of MPC and hybrid systems are basically unknown in the power electronics community has motivated us to consider such problems, namely

switch-mode DC-DC converters and induction machines driven by three-phase inverters using the notion of Direct Torque Control (DTC). For these problems, we propose novel modelling and control schemes that are conceptually simple, easy to devise, understand and tune, and most importantly, implementable.

Specifically for DTC, we present a low complexity modelling approach of the induction machine, based on which we propose three novel Model Predictive Control (MPC) approaches to tackle the DTC problem, namely *MPC based on Priority Levels*, *MPC based on Feasibility and Move Blocking*, and *MPC based on Extrapolation*. In particular the third control scheme is expected to be implementable, what has motivated our industrial partner to protect the scheme by a patent.

Considering the synchronous step-down DC-DC converter as an illustrative example for DC-DC converters, we derive a hybrid model of the converter that is valid for the whole operating regime, and for which we formulate and solve off-line an MPC problem leading to a state-feedback control law parameterized over the whole state-space. The analysis of the controller shows that the considered state-space is control invariant, and that the nominal closed-loop system is globally exponentially stable what is proved by a piecewise quadratic (PWQ) Lyapunov function. Moreover, the controller rejects large disturbances in the input voltage and the load.

Alike power electronics, power systems possess many hybrid features including integer manipulated variables such as load-shedding and capacitor switching, and internal controllers based on logic and finite state machines such as tap changers in transformers. Motivated by the recent severe blackouts in the US and Europe, we propose an emergency voltage control scheme that stabilizes the voltages in spite of major outages in order to prevent a voltage collapse and a blackout. To avoid unnecessary disruptive control actions, the control moves are classified into nominal and emergency control actions, and corresponding penalty levels are used in the objective function triggering disruptive control moves such as load-shedding only if absolutely necessary.

Zusammenfassung

Der Fokus dieser Dissertation liegt auf der Modellprädiktiven Regelung von zeitdiskreten hybriden Systemen. Hybride Systeme, welche sowohl wertekontinuierliche als auch wertediskrete Komponenten enthalten, sind an der Schnittstelle zwischen den Bereichen der Regelungstechnik und der Informatik angesiedelt. Die Modellprädiktive Regelung verwendet ein internes Modell der zu regelnden Strecke, um das zukünftige Verhalten der Regelgrößen innerhalb eines Horizontes vorherzusagen. Die Minimierung einer Zielfunktion unter Einbezug des Modells und weiterer Beschränkungen ergibt die optimale Stellgrößensequenz. Von dieser wird nur das erste Element auf die Strecke angewendet, der Horizont wird anschließend um einen Zeitschritt verschoben, und die obige Prozedur wird beim nächsten Abtastzeitpunkt wiederholt. Dies wird als *Receding Horizon* Verfahren bezeichnet. Darüber hinaus ist die Tatsache entscheidend, daß theoretische Ergebnisse und Softwaretools existieren, um offline den entsprechenden abschnittsweisen affinen Zustandsregler zu berechnen. Dadurch wird eine zeitaufwendige Berechnung der Stellgröße vermieden und die Regelung von Strecken mit hohen Abtastfrequenzen ermöglicht.

Die Dissertation gliedert sich in zwei Teile auf: Der erste Teil beschäftigt sich mit Theorie und Algorithmen, während der zweite Teil Anwendungen in der Leistungselektronik und in Energiesystemen beschreibt. Basierend auf dem Aufzählen aller Zellen eines Arrangements von Hyperebenen, welches ein Konzept aus dem Gebiet der Algorithmischen Geometrie darstellt, entwickeln wir im ersten Teil der Dissertation einen Algorithmus, der alle zulässigen Modi einer Zusammenschaltung hybrider Systeme bestimmt. Dies ermöglicht dem Designer, die Komplexität der Zusammenschaltung zu bestimmen, diese effizient in eine abschnittsweise affine Darstellung zu transformieren, und außerdem den Rechenaufwand insbesondere von Modellprädiktiver Regelung zu verringern. Dazu werden Schnitte (so genannte *Cuts*) dem Modell hinzugefügt, die die nicht zulässigen Modi vom Modell entfernen.

Die Komplexitätsreduktion der abschnittsweisen affinen Zustandsregler ist für die Reglerimplementierung von großer Bedeutung. Daher stellen wir zwei Algorithmen vor, die eine abschnittsweise affine Darstellung herleiten, welche äquivalent zum gegebenen Regler ist und die kleinstmögliche Anzahl von Regionen besitzt. Beide Algorithmen sind sowohl optimal als auch vom Rechenaufwand her praktikabel. In vielen Fällen stellt die optimale Komplexitätsreduktion die einzige Möglichkeit dar, die Komplexität der Regler so weit

zu verringern, so daß diese in Form von Look-up Tables in Hardware realisiert werden können.

Im zweiten Teil der Dissertation widmen wir uns dem Gebiet der Leistungselektronik, welches von inhärent hybrider Natur ist, da die Schaltzustände der Halbleiterschalter durch binäre Variablen beschrieben werden können. Die Tatsache, daß die Methoden der Modellprädiktiven Regelung und der hybriden Systeme hier im wesentlichen unbekannt sind, hat uns motiviert, solche Probleme zu bearbeiten – insbesondere getaktete Gleichspannungswandler und Asynchronmaschinen, wobei letztere von Dreiphasen-Wechselrichtern unter Verwendung des Konzeptes der Direkten Drehmomentregelung angesteuert werden. Für diese Anwendungen stellen wir neuartige Modellierungsansätze und Reglerschemata vor, welche nicht nur konzeptionell einfach, sondern auch leicht zu entwerfen, zu verstehen und zu tunen sind und, was entscheidend ist, praktisch anwendbar sind.

Für die Direkte Drehmomentregelung leiten wir ein Modell geringer Komplexität her, basierend auf dem wir drei neuartige Modellprädiktive Regelungskonzepte entwickeln – insbesondere ein Verfahren beruhend auf einer Zielfunktion mit verschiedenen Prioritäten, ein weiteres Verfahren basierend auf der Realisierbarkeit und dem Einfrieren von Stellgrößenänderungen, und ein drittes Verfahren, das auf Extrapolation beruht. Vor allem beim dritten Regelungsverfahren ist davon auszugehen, daß es praktisch anwendbar ist, was unseren Industriepartner veranlaßt hat, dieses durch ein Patent zu schützen.

Für den synchronen Tiefsetzsteller, den wir als anschauliches Beispiel für einen Gleichspannungswandler betrachten, leiten wir ein hybrides Modell her, das für alle Arbeitspunkte Gültigkeit besitzt. Unter Verwendung dieses Modells formulieren und lösen wir offline ein Modellprädiktives Regelungsproblem, das einen Zustandsregler ergibt, welcher über den gesamten betrachteten Zustandsraum parametrisiert ist. Eine nachfolgende Regleranalyse zeigt, daß der gesamte betrachtete Zustandsraum regelungsinvariant ist und daß der nominelle geschlossene Regelkreis exponentiell stabil ist, was durch eine abschnittsweise quadratische Lyapunovfunktion bewiesen wird. Darüber hinaus kompensiert der Regler große Störungen in der Eingangsspannung und der Last.

Ebenso wie leistungselektronische Systeme besitzen Energiesysteme zahlreiche hybride Eigenschaften. Hierzu gehören ganzzahlige Stellgrößen, die das Abtrennen von Verbrauchern oder das Hinzuschalten von Kondensatorenbänken beschreiben, ebenso wie interne Regelschleifen, welche häufig Logikkomponenten und Zustandsautomaten besitzen, wie bspw. Transformatoren mit stufbarem Windungsverhältnis. Angeregt durch die in den letzten Jahren stattgefundenen weiträumigen Stromausfälle in den USA und in Europa, stellen wir ein Notfallspannungsregelungskonzept vor, das die Spannung trotz schwerer Ausfälle stabilisiert und so einen Spannungszusammenbruch und einen Stromausfall verhindert. Um unnötige und für den Verbraucher unangenehme Reglervorgänge zu minimieren, unterteilen wir die Regleraktionen in Standard- und in Notfallmaßnahmen. In der

Zielfunktion des Modellprädiktiven Reglers werden entsprechende Gewichte verwendet, so daß Maßnahmen wie das Abtrennen von Verbrauchern vom Netz nur dann ausgelöst werden, wenn diese unabdingbar für den Erhalt der Spannungsstabilität im Netz sind.

Contents

Acknowledgements i

Abstract iii

Zusammenfassung v

1 Introduction 1
- 1.1 Mode Enumeration and Optimal Complexity Reduction . . . 2
- 1.2 Direct Torque Control . . . 5
- 1.3 DC-DC Converters . . . 8
- 1.4 Power Systems . . . 9

I Background 13

2 Background 15
- 2.1 Polyhedra . . . 15
- 2.2 Discrete-Time Hybrid Modelling . . . 16
  - 2.2.1 Discrete Hybrid Automata . . . 17
  - 2.2.2 Mixed Logical Dynamical Systems . . . 19
  - 2.2.3 Piecewise Affine Systems . . . 20
  - 2.2.4 HYSDEL . . . 21
  - 2.2.5 Summary . . . 21
- 2.3 Optimization Problems . . . 22
- 2.4 Constrained Optimal Control . . . 23
  - 2.4.1 On-Line Computation of Control Input . . . 24
  - 2.4.2 Off-Line Computation of State-Feedback Control Law . . . 25
  - 2.4.3 Receding Horizon Control . . . 28

II Mode Enumeration and Optimal Complexity Reduction 29

3 Mode Enumeration of Compositional Hybrid Systems 31
3.1 Introduction . . . 31
3.2 Cell Enumeration in Hyperplane Arrangements . . . 33
3.3 Equivalence of DHAs and PWA Systems . . . 35
3.4 Mode Enumeration Algorithm . . . 36
3.4.1 Single DHA . . . 36
3.4.2 Composition of DHAs . . . 37
3.5 Examples and Applications . . . 46
3.5.1 Car Example . . . 46
3.5.2 Paperboy Example . . . 47
3.6 Cuts for Model Predictive Control . . . 50
3.7 Conclusions and Future Research . . . 53

4 Optimal Complexity Reduction of Piecewise Affine Systems 55
4.1 Introduction . . . 55
4.2 Problem Statement and Properties . . . 57
4.2.1 Target Systems . . . 57
4.2.2 Problem Statement . . . 58
4.2.3 Convexity of Unions of Polyhedra . . . 60
4.2.4 Connectivity of Polyhedra . . . 61
4.3 Disjoint Optimal Complexity Reduction . . . 61
4.4 Non-Disjoint Optimal Complexity Reduction . . . 65
4.5 Local and Global Optimality . . . 69
4.5.1 Derivation of Global Hyperplane Arrangement . . . 69
4.5.2 Optimality of Algorithms . . . 72
4.6 Extensions . . . 73
4.6.1 Simplified Hyperplane Arrangement . . . 74
4.6.2 Divide and Conquer . . . 75
4.7 Examples . . . 77
4.7.1 Paperboy Problem . . . 77
4.7.2 Constrained Infinite Time Optimal Control Law . . . 78
4.7.3 Optimal Direct Torque Control Law . . . 80
4.8 Conclusions and Future Research . . . 82

III Direct Torque Control 85

5 Problem Description 87

5.1 Introduction . . . 87
5.2 The dq0 Reference Frame . . . 89
5.3 Nonlinear Continuous-Time Modelling . . . 90
5.3.1 Two-Level Inverter . . . 90
5.3.2 Three-Level Inverter . . . 90
5.3.3 Induction Motor . . . 94
5.4 Control Problem . . . 96
5.5 Classic DTC . . . 97
5.6 Review of Academic State of the Art . . . 98
5.7 Summary of Research Contributions . . . 100

6 Discrete-Time Modelling of DTC Drives 103

6.1 Physical Properties of DTC Drives . . . 103
6.2 Nonlinear Model of DTC Drive with Three-Level Inverter . . . 105
6.3 Low Complexity Modelling . . . 107
6.3.1 Stator Flux Dynamics . . . 107
6.3.2 Three-Level Inverter . . . 112
6.4 PWA Approximation of Nonlinearities . . . 113
6.4.1 Bounds on State Space . . . 113
6.4.2 Stator Flux Dynamics . . . 115
6.4.3 Three-Level Inverter . . . 116
6.5 MLD Model . . . 116
6.5.1 DTC Drive with Two-Level Inverter . . . 117
6.5.2 DTC Drive with Three-Level Inverter . . . 117
6.6 PWA Model . . . 118
6.6.1 DTC Drive with Two-Level Inverter . . . 118

7 Model Predictive Direct Torque Control 121

7.1 Introduction . . . 121
7.2 MPC based on Priority Levels . . . 123
7.2.1 Priority Levels . . . 124
7.2.2 Objective Function . . . 125
7.2.3 On-Line Computation of Control Input . . . 126
7.2.4 Off-Line Computation of State-Feedback Control Law . . . 127
7.2.5 Performance Evaluation . . . 128
7.3 MPC based on Feasibility and Move Blocking . . . 135

7.3.1 On-Line Computation of Control Input . . . 136
7.3.2 Off-Line Computation of State-Feedback Control Law . . . 137
7.3.3 Performance Evaluation . . . 144
7.3.4 Extensions . . . 146
7.4 MPC based on Extrapolation . . . 146
7.4.1 Horizon $N > 1$ . . . 147
7.4.2 Horizon $N = 1$ . . . 154
7.4.3 Subfunctions of Pseudo Codes . . . 158
7.4.4 Performance Evaluation . . . 161
7.5 Infeasibilities . . . 165
7.6 Case Studies . . . 169
7.7 Conclusions and Future Research . . . 170

IV DC-DC Converters 175

8 Switch-Mode DC-DC Converters 177
8.1 Introduction . . . 177
8.1.1 Switch-Mode DC-DC Conversion . . . 177
8.1.2 Review of the State of the Art . . . 178
8.1.3 Outlook . . . 182
8.2 Modelling the Synchronous Converter . . . 182
8.2.1 Continuous-Time Model . . . 182
8.2.2 ν-Resolution Discrete-Time Hybrid Model . . . 186
8.2.3 ν-Resolution Model in MLD Form . . . 189
8.2.4 ν-Resolution Model in PWA Form . . . 189
8.3 Constrained Optimal Control . . . 190
8.3.1 Objective Function . . . 191
8.3.2 On-Line Computation of Control Input . . . 192
8.3.3 Off-Line Computation of State-Feedback Control Law . . . 192
8.3.4 Load Variations . . . 194
8.4 Analysis . . . 196
8.5 Simulation Results . . . 198
8.5.1 Nominal Start-Up . . . 199
8.5.2 Step Change in Input Voltage . . . 201
8.5.3 Step Change in Output Resistance . . . 203
8.6 Conclusions and Future Research . . . 207

V Power Systems 211

9 Emergency Voltage Control in Power Systems 213

9.1 Introduction . . . 213
9.1.1 Voltage Stability in Power Systems . . . 213
9.1.2 Review of Emergency Voltage Control . . . 214
9.1.3 Outlook . . . 218
9.1.4 Notation . . . 219
9.2 Nonlinear Model . . . 219
9.2.1 Overview . . . 220
9.2.2 Component Models . . . 221
9.2.3 Fault Scenario . . . 227
9.2.4 Characteristics and Summary . . . 228
9.2.5 Parameters . . . 228
9.3 Hybrid Modelling . . . 230
9.3.1 Decomposition . . . 230
9.3.2 MLD Model . . . 231
9.4 Constrained Optimal Control . . . 232
9.4.1 Control Objectives . . . 232
9.4.2 Cascaded Controller Scheme . . . 233
9.4.3 Objective Function . . . 233
9.4.4 On-Line Computation of Control Input . . . 235
9.5 Simulation Results . . . 237
9.6 Conclusions and Future Research . . . 241

VI Appendix 243

A HYSDEL Code 245

A.1 Paperboy Example . . . 245
A.2 DTC Drive with Two-Level Inverter . . . 247
A.3 DTC Drive with Three-Level Inverter . . . 252
A.4 Synchronous Step-Down DC-DC Converter . . . 256
A.5 Power System . . . 259

B Nomenclature 265

C Curriculum Vitae 275

Bibliography 277

1

Introduction

This thesis focuses on the constrained optimal control of hybrid systems. Hybrid systems are heterogenous systems incorporating both continuous-valued components, which are governed by differential or difference equations, and discrete-valued components, such as finite state machines, if-then-else rules, on/off switches, digital circuits and software code. Hybrid systems switch between different operating modes, where each mode is governed by a dynamical law. Mode transitions are triggered by variables crossing specific thresholds, by the elapse of certain time periods, or by external inputs. With respect to hybrid systems, modelling, controller synthesis, monitoring schemes for estimation and fault detection, verification of safety properties, and analysis of stability, robustness and performance have recently become a very active area of research, both in control engineering and computer science. This is due to the fact that hybrid systems not only pose many theoretical challenges, but also offer novel ways and opportunities to improve on traditional control schemes. Most important, enabled by the rapid and steady progress in the computational power available, many intrinsically "hard" yet practically relevant and traditionally unsystematically solved problems related to the modelling, analysis and control of hybrid systems can be nowadays successfully tackled. We will consider linear hybrid systems in the discrete-time domain given in Mixed Logical Dynamical (MLD) [BM99a] or Piecewise Affine (PWA) [Son81] form.

For hybrid systems, control schemes are proposed that are based on discrete-time constrained finite-time optimal control with a receding horizon policy, often referred to as Model Predictive Control (MPC) [Mac02]. In MPC, the current control input is obtained by solving at each sampling instant an open-loop constrained optimal control problem over a finite horizon using the current state of the plant as the initial state. The underlying optimization procedure yields an optimal control sequence that minimizes a given objective function. The receding horizon policy refers to only applying the first control input of this sequence and to recomputing the control sequence at the next sampling instant over a shifted horizon, thus providing feedback and closing the control loop. The significant advantages of MPC, including its ability to systematically cope with hard

constraints on manipulated variables, states and outputs, and to easily address systems with multiple inputs and outputs, have led to its success and widespread use, which initiated in the process industry more than two decades ago. Recent survey papers on MPC include [MRRS00] and [QB03], while the monographs [CB99] and [Mac02] provide introductions to the various classes of MPC.

As introduced in [BM99a], the MLD framework can be straightforwardly embedded in MPC allowing one to use hybrid models given in MLD form as prediction models for MPC. The underlying optimization problem is a Mixed-Integer Program, for which efficient off-the-shelf solvers exist [ILO02]. In practical applications, however, the computation time for solving the optimal control problem on-line often exceeds the sampling interval thus prohibiting the direct implementation of the controller. This obstacle is overcome by pre-computing off-line the solution to the optimal control problem for the whole state-space using Dynamic Programming and multi-parametric programming, where the state vector is treated as a parameter. For hybrid systems, such a method has been recently introduced, which is based on a PWA description of the controlled system [Bor03]. The result is a PWA state-feedback control law that can be easily implemented in form of a look-up table.

1.1 Mode Enumeration and Optimal Complexity Reduction

Introduction

When deriving models and designing controllers, in particular for hybrid systems, two issues commonly arise: The derivation of the set of (feasible) modes of the model, and the complexity reduction of PWA systems comprising both PWA models and PWA state-feedback control laws.

The modelling stage is often and most easily performed using the HYbrid Systems Description Language (Hysdel). Hybrid models given in Hysdel can be considered as compositions of Discrete Hybrid Automata (DHA) [TB04], which are a mathematical abstraction of the features provided by other computation oriented and domain specific hybrid system frameworks. In general, compositions of hybrid models are very complex, as the number of different operational modes depends exponentially on the number of component systems. The explosion of the number of possible modes leads to computational difficulties as the time and space complexity of most algorithms depends on it. Yet, most of these modes are infeasible due to the model dynamics, their interaction and additional constraints. To compute the set of (feasible) modes of a composition of models is beneficial for several reasons, among them being that these modes not only allow for reducing the

computational burden of related algorithms, but also form a basis to efficiently translate HYSDEL code into PWA form.

On the other hand, the complexity of PWA systems, which is approximately given by the number of polyhedra, is often required to be as small as possible. As an example consider the case where the state-feedback control law is computed off-line based on a PWA model. Due to the combinatorial nature of the problem, both the computation time and the controller complexity are in the worst case exponential in the number of polyhedra of the PWA model [Bor03]. On the other hand, once the PWA state-feedback control law has been derived and is implemented as a look-up table in hardware, the memory requirement and the on-line computation time are linear in the number of polyhedra of the feedback law when using standard search techniques. Hence it is often of major importance to obtain a control law of minimal complexity.

Contributions

Both problems are addressed here. Using the notion of cell enumeration in hyperplane arrangements from computational geometry, we propose in Chapter 3 an algorithm that efficiently enumerates all feasible modes of a composition of DHAs. The impact of those techniques on applications is threefold. At the modelling stage, the enumeration of modes allows the designer to understand the real complexity of the compound model. After the modelling, the model can be efficiently translated into a PWA representation, which the model is generally required to be in when deriving the PWA state-feedback control law. Compared to a recently published related algorithm for deriving the PWA model [Bem02], the one presented here is of one to two orders of magnitudes faster. During the computational stage (i.e. analysis and control), the explicit computation of the set of feasible modes of the compound system can be used as structural information to prune infeasible modes from the resulting model and thus to reduce the computational burden of related algorithms, like optimal control schemes. Furthermore, the presented algorithm is able to deal with loops that may be present in compositions, and to determine if a composition is well-posed or not.

The information provided by the mode enumeration algorithm, namely the so-called markings, can be also used to determine *a priori* – i.e. without solving any Linear Program (LP) – if a given combination of polyhedra is convex. Exploiting this fact, we propose in Chapter 4 two algorithms that solve the problem of deriving a PWA model that is both equivalent to the former and minimal in the number of regions. The first algorithm executes a branch and bound on the markings yielding a new set of disjoint polyhedra, where additional heuristics on the branching strategy are employed to reduce the computation time. The second approach relies on the fact that the optimal complexity reduction problem can be reformulated as a logic minimization problem by replacing the markings

by Boolean variables and minterms [Kat94]. Logic minimization is a fundamental problem in digital circuit, and efficient tools have been developed to successfully tackle these problems, which often encounter hundreds or thousands of variables [BHMS84]. The resulting polyhedra are in general not disjoint and thus overlapping. As both algorithms refrain from solving additional LPs, they are not only optimal but also computationally feasible. The applicability of the algorithms can be extended to general PWA systems lacking the hyperplane arrangement (like PWA state-feedback control laws) by first computing the hyperplane arrangement. In many cases, the optimal complexity reduction enables the implementation of the optimal controllers as look-up tables in hardware.

Publications

Chapter 3 is almost entirely based on

[GTM03a] Geyer, T., F.D. Torrisi and M. Morari: *Efficient Mode Enumeration of Compositional Hybrid Models*. Technical Report AUT03-01, Automatic Control Laboratory ETH Zurich, http://control.ee.ethz.ch/, 2003.

Preliminary results have appeared in

[GTM03b] Geyer, T., F.D. Torrisi and M. Morari: *Efficient Mode Enumeration of Compositional Hybrid Systems*. In Pnueli, A. and O. Maler (editors): *Hybrid Systems: Computation and Control*, volume 2623 of *Lecture Notes in Computer Science*, pages 216–232. Springer-Verlag, 2003.

Apart from the logic minimization scheme, which is an extension, Chapter 4 is based on

[GTM04] Geyer, T., F.D. Torrisi and M. Morari: *Optimal Complexity Reduction of Piecewise Affine Models Based on Hyperplane Arrangements*. In *Proceedings of the American Control Conference*, pages 1190–1195, Boston, MA, June 2004.

Software Codes

The mode enumeration algorithm is implemented in Matlab and assumes that the composition of DHAs is given as Hysdel code. The latest version of the algorithm can be downloaded from http://control.ethz.ch/~hybrid/hysdel. Also the complexity reduction algorithms are written in Matlab. They are included in the multi-parametric toolbox (MPT) [KGBM04], which is freely available from http://control.ee.ethz.ch/~mpt/.

1.2 Direct Torque Control

Introduction

The rapid development of power semiconductor devices led to the increased use of adjustable speed induction motor drives in a variety of applications. In these systems, DC-AC inverters are used to drive induction motors as variable frequency three-phase voltage or current sources. One methodology for controlling the torque and speed of induction motor drives is Direct Torque Control (DTC) [TN86], which features very favorable control performance and implementation properties.

The basic principle of DTC is to exploit the fast dynamics of the motor's stator flux and to directly manipulate the stator flux vector such that the desired torque is produced. This is achieved by choosing an inverter switch combination that drives the stator flux vector to the desired position by directly applying the appropriate voltages to the motor windings. This choice is made usually with a sampling time $T_s = 25\,\mu$s using a pre-designed switching table that is derived in a heuristic way and, depending on the particularities of the application, addresses a number of different control objectives. These primarily concern the induction motor – more specifically, the stator flux and the electromagnetic torque need to be kept within pre-specified bounds around their references. In high power applications, where three-level inverters with Gate Turn-Off (GTO) thyristors are used, the control objectives are extended to the inverter and also include the minimization of the average switching frequency and the balancing of the inverter's neutral point potential around zero. Due to the discrete switch positions of the inverter, the DTC problem is a hybrid control problem, which is complicated by the nonlinear behavior of the torque, length of stator flux and the neutral point potential.

Contributions

We aim at deriving MPC schemes that are conceptually and computationally simple yet yield a significant performance improvement with respect to the state of the art. More specifically, the term *conceptually simple* refers to controllers allowing for straightforward tuning of the controller parameters or even a lack of such parameters, and easy adaptation to different physical setups and drives, whereas *computationally simple* implies that the control scheme does not require excessive computational power to allow the implementation on DTC hardware that is currently available or at least will be so within a few years.

To achieve this, we exploit in Chapter 6 a number of physical properties of DTC drives to derive discrete-time models of DTC drives with two- or three-level inverters tailored to our needs, more specifically, models that are of low complexity yet of sufficient accuracy

to serve as prediction models for our model-based control schemes. These properties are the (compared with the stator flux) slow rotor flux and speed dynamics, the symmetry of the voltage vectors, and the invariance of the motor outputs under flux rotation. The low-complexity models are derived by assuming constant speed within the prediction horizon, mapping the states (the fluxes) into a 60 degree sector, and aligning the rotor flux vector with the d-axis of a reference frame rotating with the rotational speed of the rotor. The benefits of doing this are a reduction of the number of states from five to three, and a highly reduced domain on which the nonlinear functions need to be approximated by PWA functions.

Based on the hybrid models of the DTC drive, we propose in Chapter 7 three novel control approaches to tackle the DTC problem, which are inspired by the principles of MPC and tailored to the peculiarities of DTC. The first scheme uses soft constraints to model the hysteresis bounds on the torque, stator flux and neutral point potential, and approximates the average switching frequency (over an infinite horizon) by the number of switch transitions over a short horizon. To make this approximation meaningful and to avoid excessive switching, the *Late Switching Strategy* has to be added, which favors the postponement of switch transitions. Three penalty levels with corresponding penalties of different orders of magnitude provide clear controller priorities and make the fine-tuning of the objective function obsolete, and the *Multiple Prediction Model Approach* allows us to extend the prediction interval without increasing the computational burden. This control scheme not only leads to short commissioning times for DTC drives, but it also leads to a performance improvement in terms of a reduction of the switching frequency in the range of 20 % with respect to the industrial state of the art, while simultaneously reducing the torque and flux ripples. Yet the complexity of the control law is rather excessive.

The second scheme exploits the fact that the control objectives only weakly relate to optimality but rather to feasibility, in the sense that the main objective is to find a control input that keeps the controlled variables within their bounds, i.e. a control input that is feasible. The second, weaker objective is to select among the set of feasible control inputs the one that minimizes the average switching frequency, which is again approximated by the number of switch transitions over the (short) horizon. We therefore propose an MPC scheme based on feasibility in combination with a move blocking strategy, where we allow for switching only at the current time-step. For each input sequence, we determine the number of steps the controlled variables are kept within their bounds, i.e. remain feasible. The switching frequency is emulated by the cost function, which is defined as the number of switch transitions divided by the number of predicted time-steps an input remains feasible, and the control input is chosen so that it minimizes this cost function. The simplicity of the control methodology translates into a state-feedback control law with a complexity that is of an order of magnitude lower than the one of the first scheme, while

the performance is improved.

The third scheme can be considered as a combination of the two preceding concepts. Specifically, we use a rather short horizon and compute for the input sequences over the horizon the evolution of the controlled variables using the prediction model. To emulate a long horizon, the "promising" trajectories are extrapolated and the number of steps is determined when the first controlled variable hits a bound. The cost of each input sequence is then determined by dividing the total number of switch transitions in the sequence by the length of the extrapolated trajectory. Minimizing this cost yields the optimal input sequence and the next control input to be applied. The major benefits of this scheme are its superior performance in terms of switching frequency, which is reduced over the whole range of operating points by up to 50 %, with an average reduction of 25 %. Furthermore, the controller needs no tuning parameters. As the computation of an explicit solution is avoided, all quantities may be time-varying including model parameters, set points and bounds. Those can be adapted on-line, making the concept applicable to the whole range of operating points. As all computations are performed on-line, the prediction model is not restricted to be PWA, allowing us to use the nonlinear (and more accurate) discrete-time model.

Summing up, all control schemes are based on minimizing an approximate of the average switching frequency, they use an internal model of the DTC drive to predict the output response to input sequences, and they are tailored to the specific DTC problem set-up. Starting from the first scheme, the complexity of the controllers in terms of computation times and the memory requirement is steadily reduced by several orders of magnitude, while the performance is steadily improved. In particular the last control scheme is expected to be implementable on the currently available DTC hardware.

Publications

Chapters 5, 6 and 7 are based on a re-arrangement and slight extension of

[PGM04c] Papafotiou, G., T. Geyer and M. Morari: *Optimal Direct Torque Control of Three-Phase Symmetric Induction Motors*, 2004. submitted to journal.

[GP05] Geyer, T. and G. Papafotiou: *Direct Torque Control for Induction Motor Drives: A Model Predictive Control Approach based on Feasibility.* In Morari, M. and L. Thiele (editors): *Hybrid Systems: Computation and Control*, volume 3414 of *Lecture Notes in Computer Science*, pages 274–290. Springer-Verlag, 2005.

[GPM04a] Geyer, T., G. Papafotiou and M. Morari: *Model Predictive Direct Torque Control Minimizing Switching Frequency*, 2004, unpublished.

Preliminary results of the modelling and the first control scheme have appeared in

[PGM04b] Papafotiou, G., T. Geyer and M. Morari: *Optimal Direct Torque Control of Three-Phase Symmetric Induction Motors.* In *Proceedings of the 43th IEEE Conference on Decision and Control*, Atlantis, Bahamas, December 2004.

Some concepts, in particular the third control scheme, have been protected by the following patent.

[GPM04c] Geyer, T., G. Papafotiou and M. Morari: *Verfahren zum Betrieb einer rotierenden elektrischen Maschine*, European Patent application pending, EP 04 405 767.7, 2004.

1.3 DC-DC Converters

Introduction

Switch-mode DC-DC converters are power electronic circuits that are used in a large variety of applications. The scope is to achieve output voltage regulation in the presence of input voltage and output load variations. The difficulties in controlling DC-DC converters arise from their hybrid nature. In general, these converters feature two or three different modes of operation, where each mode has an associated linear continuous-time dynamic. Furthermore, constraints are present. These result from the converter topology including the manipulated variable (duty cycle) which is bounded in between of zero and one, and from constraints on states like the discontinuous current mode where the inductor current is constrained to be non-negative. Additionally, constraints are imposed as safety measures, such as current limiting or soft-starting, where the latter constitutes a constraint on the maximal derivative of the current during start-up. The control problem is further complicated by gross operating point changes with input voltage and output load variations and model uncertainties.

Contributions

Motivated by the hybrid nature of DC-DC converters, we propose in Chapter 8 a novel approach to the modelling and controller design problem for fixed-frequency DC-DC converters, using a synchronous step-down DC-DC converter as an illustrative example. In particular, the notion of the ν-resolution model is introduced to capture the hybrid nature

of the converter, which is modelled as a hybrid system using the MLD framework. This leads to a model that is valid for the whole operating regime and captures the evolution of the state variables within the switching period. Based on the MLD model, we formulate and solve a constrained finite time optimal control problem. This allows for a systematic controller design that achieves the objective of regulating the output voltage to the reference despite input voltage and output load variations while satisfying the constraints. In particular, the control performance does not degrade for changing operating points. Furthermore, we derive the explicit PWA state-feedback control law, which can be easily stored in a look-up table and used for the practical implementation of the proposed control scheme. An a posteriori analysis step shows that the considered state space is a control invariant set. Most importantly, a Piecewise Quadratic (PWQ) Lyapunov function can be computed that proves exponential stability of the closed-loop system for the whole range of operating points.

Publications

Chapter 8 is based on

[GPM05] Geyer, T., G. Papafotiou and M. Morari: *Global Optimal Control of Switch-Mode DC-DC Converters*, 2005. submitted to journal.

The modelling framework and on-line control scheme was first proposed in

[GPM04b] Geyer, T., G. Papafotiou and M. Morari: *On the Optimal Control of Switch-Mode DC-DC Converters*. In Alur, R. and G. Pappas (editors): *Hybrid Systems: Computation and Control*, volume 2993 of *Lecture Notes in Computer Science*, pages 342–356. Springer-Verlag, 2004.

The extension to the explicit control law can be found in

[PGM04a] Papafotiou, G., T. Geyer and M. Morari: *Hybrid Modelling and Optimal Control of Switch-mode DC-DC Converters*. In *IEEE Workshop on Computers in Power Electronics (COMPEL)*, Champaing, IL, USA, 2004.

1.4 Power Systems

Introduction

An electrical power system consists of numerous components connected together to form a large, complex system generating, transmitting and distributing electrical power. Electric power systems and additional preventive control schemes are designed in such a way,

that the system should be able to withstand any single contingency, that is, outage of any single component without loss of stability and with all system variables kept within predefined ranges [Kun94]. Not all possible disturbances, however, can be foreseen at the planning stage, and these may result (in particular when multiple contingencies occur within short time) in instability leading eventually to a collapse or islanding of the system. Furthermore, because of environmental constraints on the extension of the transmission capacity, increased electricity consumption and new economic constraints imposed by the liberalized power market, power systems are operated closer and closer to their stability limits. In the past, a number of severe voltage instability incidents have occurred around the globe [Tay94, vV98], most notably the recent blackouts in northeastern US and southern Canada in August 2003, in southern Sweden and eastern Denmark in September 2003, followed only within a few days by the major blackout in Italy that has affected most of the country's 58 million people.

The design of emergency voltage control schemes is complicated by the fact that power systems are hybrid systems including local controllers based on logic rules and finite state machines. Furthermore, most control moves are inherently discrete-valued, like capacitor banks and tap changers, which must be switched using fixed step sizes, and load-shedding, which must be carried out by disconnecting whole feeders. Recent advances in computation, communication and power system instrumentation technology, more specifically Phasor Measurement Units and Wide-Area Measurement Systems [Reh01], have made coordinated and model based approaches tractable. They are highly attractive since the use of a model in combination with on-line optimization allows for optimal coordination of different control moves and automatic adaption to changing operating conditions. Thanks to this, they are less conservative than non-adaptive local schemes – even if the local schemes have been optimally tuned – and thus avoid unnecessary operation of the protection schemes in order to minimize load shedding.

Contributions

In Chapter 9 we present a novel emergency control scheme capable of predicting and preventing a voltage collapse in a power system, that is modelled as a hybrid system incorporating nonlinear dynamics, discrete events and discrete manipulated variables. We decompose the system into a discrete event system, and a continuous-valued dynamical system governed by nonlinear differential algebraic equations. To cast the model into MLD form, the nonlinearities are replaced by PWA approximations.

The control objectives are to maintain the load voltage close to its reference, while fulfilling the soft constraints on the bus voltages and while switching the manipulated variables as little as possible. We classify the control move into nominal and emergency control actions. During nominal control, the soft constraints on the bus voltages can be

fulfilled by using only "cheap" control moves, i.e. capacitor bank switching and changing the tap changer voltage reference. If the soft constraints cannot be met by only applying cheap control moves, the controller uses emergency control and the full range of available control moves including load-shedding.

MPC in connection with the MLD framework is used to successfully stabilize the voltage of a four bus example system, defined within the Control and Computation (CC) branch of the IST Research Project IST-2001-33520 as a case study. The tuning of the controller is straightforward and systematic allowing us to easily distinguish between nominal and emergency control moves. In comparison with previous work, the MLD framework allows for modelling the hybrid behavior of the power system and thus provides better accuracy since effectively multiple linearizations are used during the prediction interval. To the best of our knowledge this is the first time a general-purpose constrained optimal control framework for hybrid systems is applied successfully to the emergency voltage control problem.

Publications

Preliminary versions of Chapter 9 have appeared in

[GLM02] Geyer, T., M. Larsson and M. Morari: *Hybrid Control of Voltage Collapse in Power Systems*. Technical Report AUT02-12, Automatic Control Laboratory ETH Zurich, http://control.ee.ethz.ch/, 2002.

[GLM03] Geyer, T., M. Larsson and M. Morari: *Hybrid Emergency Voltage Control in Power Systems*. In *Proceedings of the European Control Conference*, Cambridge, UK, September 2003.

Further reading, in particular on temporal Lagrangian decomposition of MPC applied to the power system case study, can be found in

[BGM04] Beccuti, A.G., T. Geyer and M. Morari: *Temporal Lagrangian Decomposition of Model Predictive Control for Hybrid Systems*. In *Proceedings of the 43th IEEE Conference on Decision and Control*, Atlantis, Bahamas, December 2004.

Part I

Background

2

Background

In this chapter, we provide basic concepts and methodologies needed throughout the thesis. We start by recalling polyhedra and related concepts in Section 2.1, and summarize discrete-time hybrid modelling in Section 2.2, in particular Discrete Hybrid Automata (DHA), Mixed Logical Dynamical (MLD) systems and Piecewise Affine (PWA) systems. Moreover, we point out related modelling methodologies and briefly mention the modelling language HYSDEL. After defining Linear Programs (LP) and Mixed-Integer Linear Programs (MILP) in Section 2.3, we present constrained optimal control in Section 2.4. Here, we distinguish between on-line computation to derive for *one* given state the optimal control input, and off-line computation to pre-solve the control problem and to derive for *all* (feasible) states the control law.

2.1 Polyhedra

In this section, we recall the notion of convex sets and functions, and state some basic definitions related to hyperplanes and polyhedra.

Definition 2.1 (Convex Set, [BV04]) *A set $\mathcal{X}$ is convex if the line segment between any two points in $\mathcal{X}$ lies in $\mathcal{X}$, i.e., if for any x_1, $x_2 \in \mathcal{X}$ and any θ with $0 \leq \theta \leq 1$, we have*

$$\theta x_1 + (1-\theta)x_2 \in \mathcal{X}\,. \tag{2.1}$$

Definition 2.2 (Convex Function, [Flo95]) *Let $\mathcal{X}$ be a convex subset of $\mathbb{R}^d$, and $f(x)$ be a real valued function defined on $\mathcal{X}$. The function $f(x)$ is said to be convex if for any x_1, $x_2 \in \mathcal{X}$, and $0 \leq \theta \leq 1$, we have*

$$f((1-\theta)x_1 + \theta x_2) \leq (1-\theta)f(x_1) + \theta f(x_2)\,. \tag{2.2}$$

A function is strictly convex, if the strict inequality in (2.2) holds whenever $x_1 \neq x_2$ and $0 < \theta < 1$.

Definition 2.3 (Hyperplane) *The i-th hyperplane in the d-dimensional Euclidian space $\mathbb{R}^d$ is given by the linear equality $H_i = \{x \in \mathbb{R}^d \mid a_i^T x = b_i\}$ with the normal vector $a_i \in \mathbb{R}^d$ and $b_i \in \mathbb{R}$.*

Definition 2.4 (Halfspace) *The hyperplane H_i induces the two (closed) halfspaces $\{x \in \mathbb{R}^d \mid a_i^T x \leq b_i\}$ and $\{x \in \mathbb{R}^d \mid a_i^T x \geq b_i\}$.*

Definition 2.5 (Polyhedron) *A convex set $\mathcal{P} \subseteq \mathbb{R}^d$ given by $\mathcal{P} = \{x \in \mathbb{R}^d \mid a^T x \leq b\}$ is called a* polyhedron *with $a \in \mathbb{R}^{d \times n}$ and $b \in \mathbb{R}^n$, and the operator $\leq$ denoting an element-wise comparison of two vectors.*

Equivalently, the polyhedron $\mathcal{P}$ can be considered as the intersection of a finite number of half spaces (here n) defined by the hyperplanes H_i. In particular, a and b hold the n hyperplanes, i.e. $a = [a_1, \ldots, a_n]$ and $b = [b_1, \ldots, b_n]^T$.

Definition 2.6 (Facet) *If $\mathcal{P} \cap H_i$ is $(d-1)$-dimensional then $\mathcal{P} \cap H_i$ is called a* facet *of the polyhedron $\mathcal{P}$.*

Definition 2.7 (Redundancy) *The i-th inequality $a_i^T x \leq b_i$ is* redundant *for $\mathcal{P}$ if its removal preserves the polyhedron, i.e. $\mathcal{P} = \{x \in \mathbb{R}^d \mid a^T x \leq b\} = \{x \in \mathbb{R}^d \mid a_j^T x \leq b_j \ \forall j \neq i\}$.*

Definition 2.8 (NMR) *The polyhedron $\mathcal{P} = \{x \in \mathbb{R}^d \mid a^T x \leq b,\ a_i^T a_i = 1 \ \forall i\}$ without redundant inequalities is called a* normalized minimal representation *(NMR) polyhedron.*

Definition 2.9 (Polyhedral Partition) *A collection of polyhedra $\mathcal{P}_i \subseteq \mathcal{R}$, $i \in \mathcal{I} \subset \mathbb{N}$, is a* polyhedral partition *of the polyhedron $\mathcal{R}$, iff*

$$(i) \quad \bigcup_{i \in \mathcal{I}} \mathcal{P}_i = \mathcal{R}\,, \tag{2.3a}$$

$$(ii) \quad \mathcal{P}_i \cap \mathcal{P}_j \text{ is lower dimensional } \forall i, j \in \mathcal{I},\ i \neq j \tag{2.3b}$$

2.2 Discrete-Time Hybrid Modelling

Hybrid systems are heterogenous systems incorporating both continuous-valued components, which are governed by differential or difference equations, and discrete-valued components, such as finite state machines, if-then-else rules, on/off switches, digital circuits and software code. In particular, besides continuous-valued states and inputs, such systems have discrete-valued states and/or discrete-valued inputs. In general, hybrid systems switch between different operating modes, where each mode is governed by a dynamical law. Mode transitions are triggered by variables crossing specific thresholds, by the elapse of certain time periods, or by external inputs.

Hybrid systems are predominant in applications. In this thesis, we will focus on applications in the fields of power electronics and power systems, which are intrinsically hybrid. In power electronics, semiconductor switches constitute the (binary) manipulated variables. Specifically, in DC-DC converters, switches are used to regulate the converter's output voltage despite changes in the input voltage and the load, and in DC-AC converters driving an induction machine, the switch positions are manipulated such that, among others, the desired torque and/or speed is achieved. In power systems, many manipulated variables (like load shedding and capacitor switching) are discrete-valued, too, and secondary controllers (like under load tap changers) incorporate thresholds, logic and finite state machines. Moreover, saturations are present, which are enforced for example by Automatic Voltage Regulators to protect generators from overheating. Apart from these hybrid features, most power electronics and power systems are nonlinear. Other examples for hybrid systems include digital controllers that regulate a plant featuring only continuous-valued states and inputs [BBB+00], circuits integrating relays or diodes, biomolecular networks [ABI+01], TCP/IP networks [HBOL01], and closed-loop linear system with constraints on the manipulated variable.

In this thesis, we restrict ourselves to the discrete-time domain, and we confine our models to (piecewise) affine dynamics rather than allowing general nonlinear dynamics. This not only avoids a number of mathematical problems (like Zeno behavior), but allows us to derive models for which we can pose analysis and optimal control problems that are computationally tractable. To model such discrete-time linear hybrid systems, we present in the following three frameworks, which will be used throughout the thesis: Discrete Hybrid Automata (DHA), Mixed Logical Dynamical (MLD) systems and Piecewise Affine (PWA) systems.

2.2.1 Discrete Hybrid Automata

As shown in Fig. 2.1, Discrete Hybrid Automata (DHA) [TB04] result from the interconnection of a *Finite State Machine* (FSM), which provides the discrete part of the hybrid system, with a *Switched Affine System* (SAS) providing the continuous part of the system. The interaction between the two is based on two connecting elements: The *Event Generator* (EG) and the *Mode Selector* (MS). The EG extracts binary signals from the continuous part. Those binary events and other exogenous binary inputs trigger switches of the FSM states. The MS combines all binary variables (states, inputs, and events) to choose the mode and thus the corresponding continuous dynamic of the SAS. Next, we define each of the four components.

Switched Affine System (SAS). A Switched Affine System is a collection of affine

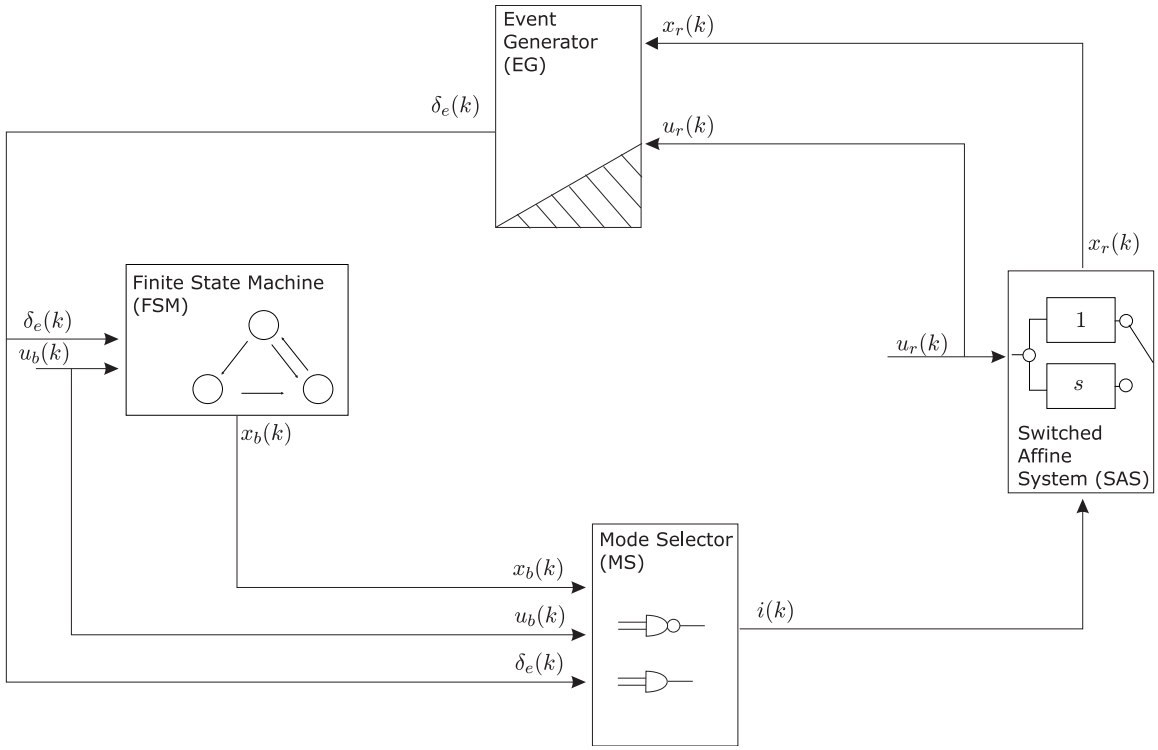


Figure 2.1: A Discrete Hybrid Automaton (DHA) is the connection of a Finite State Machine (FSM) and a Switched Affine System (SAS) through a Mode Selector (MS) and an Event Generator (EG). The output signals are omitted for clarity

systems

$$x_r(k+1) = A_{i(k)}x_r(k) + B_{i(k)}u_r(k) + f_{i(k)} \tag{2.4a}$$

$$y_r(k) = C_{i(k)}x_r(k) + D_{i(k)}u_r(k) + g_{i(k)}\,, \tag{2.4b}$$

where $k \in \mathbb{N}_0$ is the discrete time-instant, $x_r \in \mathcal{X}_r \subseteq \mathbb{R}^{n_r}$ is the real state, $u_r \in \mathcal{U}_r \subseteq \mathbb{R}^{m_r}$ is the exogenous real input, $y_r \in \mathcal{Y}_r \subseteq \mathbb{R}^{p_r}$ is the real output, $\{A_i, B_i, f_i, C_i, D_i, g_i\}_{i\in\mathcal{I}}$ is a collection of matrices of appropriate dimensions, and the mode $i \in \mathcal{I} \subset \mathbb{N}$ is an input signal selecting the affine state-update and output function.

Event Generator (EG). An Event Generator generates the binary event signal δ_e according to the fulfillment of affine constraints or thresholds

$$\delta_e(k) = f_{\mathrm{H}}(x_r(k), u_r(k))\,, \tag{2.5}$$

where $f_{\mathrm{H}} : \mathbb{R}^{n_r} \times \mathbb{R}^{m_r} \to \mathcal{D} \subseteq \{0,1\}^{n_e}$ is a vector of descriptive functions of a set of affine constraints. In particular, *threshold events* are modelled as $[\delta_e^i(k) = 1] \leftrightarrow [f_H^i(x_r(k), u_r(k)) \leq 0]$, and *time events* are modelled by adding a clock variable t in the switched affine system with dynamic $t(k+1) = t(k) + T_s$ and setting $[\delta_e^i(k) = 1] \leftrightarrow [t(k) \geq t_0]$, where the superscript i denotes the i-th component of a vector and T_s is the sampling interval.

Finite State Machine (FSM). A Finite State Machine (or automaton) is a discrete dynamic process that evolves according to a binary state-update function [PB97]

$$x_b(k+1) = f_{\mathrm{B}}(x_b(k), u_b(k), \delta_e(k)) \tag{2.6a}$$
$$y_b(k) = g_{\mathrm{B}}(x_b(k), u_b(k), \delta_e(k)), \tag{2.6b}$$

where $x_b \in \mathcal{X}_b \subseteq \{0,1\}^{n_b}$ is the binary state, $u_b \in \mathcal{U}_b \subseteq \{0,1\}^{m_b}$ the exogenous binary input, $y_b \in \mathcal{Y}_b \subseteq \{0,1\}^{p_b}$ the binary output, δ_e the event, and $f_{\mathrm{B}} : \mathcal{X}_b \times \mathcal{U}_b \times \mathcal{D} \to \mathcal{X}_b$, $g_{\mathrm{B}} : \mathcal{X}_b \times \mathcal{U}_b \times \mathcal{D} \to \mathcal{Y}_b$ are deterministic binary functions.

Mode Selector (MS). The binary state x_b, the binary input u_b and the event δ_e select the mode i of the SAS through a deterministic binary function $f_{\mathrm{M}} : \mathcal{X}_b \times \mathcal{U}_b \times \mathcal{D} \to \mathcal{I}$, which is therefore called *Mode Selector*. The output of this function

$$i(k) = f_{\mathrm{M}}(x_b(k), u_b(k), \delta_e(k)) \tag{2.7}$$

is called *active mode*. We say that a *mode switch* occurs at time-instant k if $i(k-1) \neq i(k)$. Note that one may associate with a state x_b of the FSM more than one mode i according to the event δ_e.

To shorten the notation, we will use the definitions $x \triangleq \left[\begin{smallmatrix} x_r \\ x_b \end{smallmatrix}\right]$, $u \triangleq \left[\begin{smallmatrix} u_r \\ u_b \end{smallmatrix}\right]$, $y \triangleq \left[\begin{smallmatrix} y_r \\ y_b \end{smallmatrix}\right]$, $\mathcal{X} \triangleq \mathcal{X}_r \times \mathcal{X}_b$, $\mathcal{U} \triangleq \mathcal{U}_r \times \mathcal{U}_b$ and $\mathcal{Y} \triangleq \mathcal{Y}_r \times \mathcal{Y}_b$ throughout the thesis.

Definition 2.10 *A DHA is* well-posed *on $\mathcal{X}$, $\mathcal{U}$, $\mathcal{Y}$, if for all initial conditions $x(0) \in \mathcal{X}$ and for all inputs $u(k) \in \mathcal{U}$ the state trajectory $x(k) \in \mathcal{X}$ and the output trajectory $y(k) \in \mathcal{Y}$ are uniquely defined for all $k \in \mathbb{N}_0$.*

2.2.2 Mixed Logical Dynamical Systems

The general MLD form of a hybrid system introduced in [BM99a] is

$$x(k+1) = Ax(k) + B_1 u(k) + B_2 \delta(k) + B_3 z(k) \tag{2.8a}$$
$$y(k) = Cx(k) + D_1 u(k) + D_2 \delta(k) + D_3 z(k) \tag{2.8b}$$
$$E_2 \delta(k) + E_3 z(k) \leq E_4 x(k) + E_1 u(k) + E_5, \tag{2.8c}$$

where $k \in \mathbb{N}_0$ is the discrete time-instant, and $x \in \mathcal{X}$ denotes the states, $u \in \mathcal{U}$ the inputs and $y \in \mathcal{Y}$ the outputs, with both real and binary components as defined in the previous section. Furthermore, $\delta \in \{0,1\}^{n_\delta}$ and $z \in \mathbb{R}^{n_z}$ represent binary and auxiliary continuous variables, respectively. These variables are introduced when translating propositional logic or PWA functions into linear inequalities.

All constraints on states, inputs, outputs and auxiliary variables are summarized in the mixed-integer linear inequality constraint (2.8c). Note that (2.8a) and (2.8b) are linear; the nonlinearity is hidden in the integrality constraints on the binary variables. A

combination of a binary state x_b, binary input u_b and binary variable δ is called *mode*. If for a given triple x_b, u_b and δ there exists a triple $x_r \in \mathcal{X}_r$, $u_r \in \mathcal{U}_r$ and $z \in \mathbb{R}^{n_z}$ such that the inequality (2.8c) holds, the mode is called *feasible*; else it is *infeasible*.

Well-posedness of MLD models is defined according to Definition 2.10. The following lemma follows directly.

Lemma 2.1 *If for every given pair $x(k) \in \mathcal{X}$ and $u(k) \in \mathcal{U}$, the values of $\delta(k)$ and $z(k)$ are uniquely defined by the inequality* (2.8c)*, the MLD model is well-posed.*

Hereafter, we consider only MLD models that are well-posed. This assumption is not restrictive and is always satisfied when real plants are described in the MLD form [BM99a]. Note that the MLD framework allows for describing automata, propositional logic, *if ... then ... else* statements and PWA functions. General nonlinear functions, however, cannot be directly incorporated, and have to be approximated by PWA functions.

2.2.3 Piecewise Affine Systems

Polyhedral piecewise affine (PWA) systems [Son81,HSB01] are defined by partitioning the state-space into polyhedra and associating with each polyhedron an affine state-update and output function

$$x(k+1) = A_{j(k)}x(k) + B_{j(k)}u(k) + f_{j(k)} \tag{2.9a}$$

$$y(k) = C_{j(k)}x(k) + D_{j(k)}u(k) + g_{j(k)} \tag{2.9b}$$

$$\text{with } j(k) \text{ such that } \begin{bmatrix} x(k) \\ u(k) \end{bmatrix} \in \mathcal{P}_{j(k)}, \tag{2.9c}$$

where $x \in \mathcal{X}$, $u \in \mathcal{U}$, $y \in \mathcal{Y}$ denote at time-instant $k \in \mathbb{N}_0$ the real and binary states, inputs and outputs, respectively, the polyhedra $\mathcal{P}_{j(k)}$ define a set of polyhedra $\{\mathcal{P}_j\}_{j\in\mathcal{J}}$ on $\mathcal{X} \times \mathcal{U}$, and the real matrices $A_{j(k)}$, $B_{j(k)}$, $C_{j(k)}$, $D_{j(k)}$ and real vectors $f_{j(k)}$, $g_{j(k)}$ with $j(k) \in \mathcal{J}$, $\mathcal{J}$ finite, are constant and have suitable dimensions. We refer to $j(k)$ as the *mode* of the system and to $\#\mathcal{J}$ as the number of modes.

Instead of (2.9), we often use the equivalent simplified notation

$$x(k+1) = f_{\text{PWA}}(x(k), u(k)) \tag{2.10a}$$

$$y(k) = g_{\text{PWA}}(x(k), u(k))\,. \tag{2.10b}$$

For PWA models, we define well-posedness as in Definition 2.10. The following lemma follows directly.

Lemma 2.2 *Let Σ_{PWA} be a PWA model as in* (2.9)*. If $\{\mathcal{P}_j\}_{j\in\mathcal{J}}$ is a polyhedral partition of $\mathcal{X} \times \mathcal{U}$, then Σ_{PWA} is well-posed.*

Note however, that the converse statement does not hold in general as a well-posed PWA system may be defined on an overlapping set of polyhedra.

2.2.4 HYSDEL

Modelling complex hybrid systems directly for example in MLD or PWA form is, in general, a tedious and non-trivial task. To facilitate the modelling, the HYbrid Systems DEscription Language (HYSDEL) [TB04] has been developed, which allows the designer to describe a hybrid system on a textual basis. For a detailed description of the HYSDEL modelling language and its capabilities, we defer to [TB04]. Note that a hybrid system described in HYSDEL can be regarded as a composition of DHAs [TB04]. We will exploit this fact in the next chapter when proposing the Mode Enumeration Algorithm. In this thesis, all case studies and applications have been modelled in HYSDEL, the code of which can be found in Appendix A.

2.2.5 Summary

Before proceeding, we sum up the main features of the above introduced hybrid modelling frameworks. Even though we have avoided to introduce general nonlinear hybrid models, binary states and binary inputs can be directly included, the state-update mapping of the hybrid system may be discontinuous, events that trigger mode transitions may be both external (set by switching in a binary input), or internal (induced by a state variable hitting a threshold). Moreover, PWA constraints can be imposed on states and inputs. In particular, these models can approximate nonlinear systems arbitrarily accurate.

Moreover, these hybrid models feature thresholds or guardlines defined on states, inputs and internal variables. Binary signals are associated with them that are either true or false according to the fulfillment of these thresholds. Additionally, as mentioned above, these models may encompass binary states that are part of a finite state machine or an automaton. A (feasible) combination of binary signals and binary states is called a *mode*. Since we have restricted the thresholds to be linear, the set of states and inputs corresponding to the same mode form a (convex) polyhedron whose facets are a subset of the thresholds. For each mode, the hybrid model features an associated dynamic. Here, we have restricted ourselves to discrete-time and PWA dynamics. Furthermore, as the modes are defined such that the binary signals are constant for a given mode, these signals can be directly included in the affine expressions of the PWA dynamics.

Indeed, the aforementioned modelling frameworks are equivalent as shown in [Bem02], and several tools are available to transform them into each other. More specifically, DHA models can be considered as building blocks of HYSDEL code, which can be compiled to MLD models or transformed into PWA form using the Mode Enumeration Algorithm presented in Chapter 3. Alternatively, MLD models can be translated into PWA form using an approach based on multi-parametric programming and Mixed-Integer Linear Programming [Bem02]. Describing PWA models as DHAs or MLD models is trivial.

Yet, these modelling frameworks carry different benefits. DHAs can be easily described in HYSDEL, MLD models are very suitable for (on-line) optimal control, whereas PWA models are the starting point to derive off-line the optimal control law as a state-feedback control law [Bor03], to design moving horizon observers for hybrid systems [FMM02], or to perform analysis tasks.

Besides DHAs, MLD and PWA models, *Linear Complementarity* (LC) systems [SS98], *Extended Linear Complementarity* (ELC) systems [HSB01], and *Max-Min-Plus-Scaling* (MMPS) systems [Dv01, HSB01] have been proposed. However, as shown first in [Son96] for PWA and a class of hybrid systems, and then, with different arguments, in [HSB01, TB04] all those modelling frameworks are equivalent, and it is possible to represent the same system using any of these frameworks.

Beyond the above, a number of additional modelling frameworks exists, in particular for the continuous-time domain. We refer to the in-depth report [Kam01] and its references for an overview of related frameworks to model hybrid systems both in the continuous and the discrete-time domain. This report also provides a comparison of such schemes including the MLD framework.

2.3 Optimization Problems

Before introducing constrained optimal control, we briefly recall basic terminology for mathematical programming in general, and introduce Linear Programming (LP) and Mixed-Integer Linear Programming (MILP) in particular. For details, the reader is referred to one of the numerous textbooks (e.g. [BV04, Flo95]).

We start by introducing basic terminology for optimization problems according to [BV04]. Consider

$$\begin{aligned} \min_{\zeta} \quad & f_0(\zeta) \\ \text{subj. to} \quad & f_i(\zeta) \leq 0, \qquad i = 1, \ldots, n_f \end{aligned} \tag{2.11}$$

with the optimization variable $\zeta \triangleq \left[\begin{smallmatrix} \zeta_r \\ \zeta_b \end{smallmatrix}\right]$ that we introduce here in a general way containing both a real-valued part $\zeta_r \in \mathbb{R}^{d_r}$ and a binary part $\zeta_b \in \{0,1\}^{d_b}$ with $d = d_r + d_b$. The problem amounts to finding an ζ that minimizes the objective (or cost) function $f_0 : \mathbb{R}^d \to \mathbb{R}$ such that the inequality constraints $f_i(\zeta) \leq 0$, $i = 1, \ldots, n_f$ hold, where the corresponding function $f_i : \mathbb{R}^d \to \mathbb{R}$ is called the inequality constraint function.

A point ζ is said to be *feasible* if it satisfies all constraints $f_i(\zeta)$. The problem (2.11) is *feasible* if there exists at least one feasible point, else it is *infeasible*. The set of all feasible points is called the *feasible set*. The *optimal value* J^* of the problem (2.11) is defined as

$$J^* = \inf\{f_0(\zeta) \,|\, f_i(\zeta) \leq 0, i = 1, \ldots, n_f\}, . \tag{2.12}$$

If the problem is infeasible, we set $J^* = \infty$, and if the problem is unbounded below, we have $J^* = -\infty$. The solution ζ^* to the minimization problem (2.11) is referred to as the *optimizer*, if ζ^* is feasible and $f_0(\zeta^*) = J^*$.

We say a feasible point ζ is *locally optimal* if it minimizes f_0 (only) in a subset of the feasible set, whereas ζ is *globally optimal* if it minimizes f_0 for the whole feasible set.

A very important class of optimization problems (2.11) are *convex optimization* problems, where the objective function and the inequality constraint functions are convex, and no binary variables are present in the optimization variable ($d_b = 0$). This implies that the feasible set is convex, and, most importantly, any locally optimal point is also globally optimal.

Throughout this thesis, we will use another class of optimization problems, namely the class of Mixed-Integer Linear Programs (MILP)

$$\begin{aligned} \min_{\zeta} \quad & c^T \zeta \\ \text{subj. to} \quad & G\zeta \leq g \\ & \zeta_b \in \{0,1\}^{d_b}, \end{aligned} \tag{2.13}$$

where G is a matrix of appropriate dimension and c and g are row vectors. Even though the formulation (2.13) has a linear objective function and linear constraint functions in ζ, the optimization problem is not convex, due to the binary variables ζ_b present in the MILP. This implies that locally optimal points are not necessarily globally optimal. In fact, MILPs are in general NP-hard, meaning that using the algorithms available, the solution time grows in the worst case exponentially with the number of binary variables [RG91]. Yet, several algorithms have been proposed and applied successfully to applications of large size. These include branch and bound, cutting plane, decomposition and logic-based methods. Details about and references of these algorithms are beyond the scope of the thesis, but may be found for example in [Flo95].

If, however, the optimization vector ζ contains only real components ($d_b = 0$), problem (2.13) reduces to a Linear Program (LP), which is convex and can be solved in polynomial time.

2.4 Constrained Optimal Control

In the following, we briefly introduce Model Predictive Control (MPC) for discrete-time linear hybrid systems and summarize its basic features. In MPC, the current control input is obtained by solving at each sampling instant a constrained optimal control problem using the predictions provided by an internal model of the controlled process. The optimal control problem is formulated over a finite or infinite horizon using the current state

of the plant as initial state. The underlying optimization procedure yields an optimal control sequence that minimizes a given objective function. A receding horizon policy is employed, which refers to only applying the first control input of this sequence, and to recomputing the control sequence at the next sampling instant over a shifted horizon, thus providing feedback and closing the control loop. Hence, MPC combines (open-loop) constrained optimal control with a receding horizon policy. The significant advantages of MPC, including its ability to systematically cope with hard constraints on manipulated variables, states and outputs, and to easily address systems with multiple inputs and outputs, have led to its success and widespread use, which started in the process industry more than two decades ago. Most importantly, the MLD framework can be straightforwardly embedded in MPC allowing one to use hybrid models given in the MLD form as prediction models for MPC. For details on MPC, the reader is referred to the survey papers [GPM89,BM99b,MRRS00,Raw00,May01,QB03] and the monograph [Mac02]. Details about the setup of the MPC formulation in connection with MLD models can be found in [BM99a] and [BBM00].

In the following two sections, we give an overview of the classic on-line computation of the control input, and the novel off-line computation of the corresponding state-feedback control law. The sequel is merely intended to provide a rough introduction; details and technicalities are omitted. These can be found in the relevant literature, which will be pointed out.

2.4.1 On-Line Computation of Control Input

Consider the objective function

$$\begin{aligned} J(x(k),U(k)) = &\sum_{\ell=0}^{N-1} \|Q_1(x(k+\ell|k)-x_{ref})\|_{\{1,\infty\}} \\ &+ \|Q_2(u(k+\ell|k)-u_{ref})\|_{\{1,\infty\}} + \|Q_3(y(k+\ell|k)-y_{ref})\|_{\{1,\infty\}}\,, \end{aligned} \tag{2.14}$$

which penalizes the predicted evolution of the state, control input and output, respectively, over the finite horizon N using a linear norm, namely the 1- or the ∞-norm. We assume that Q_1, Q_2 and Q_3 are full column rank matrices.

Next, we instantiate the MLD model (2.8) at every time-step within the prediction horizon thus repeating it N times. Moreover, we build an optimization vector that holds the sequence of states, inputs, outputs, binary δ variables and real auxiliary z variables, in particular it contains the sequence of control inputs $U(k) = [(u(k))^T, \ldots, (u(k+N-1))^T]^T$. For details on the formulation, see [BBM00]. This leads to the constrained finite time

optimal control problem (CFTOC)

$$\begin{aligned} U^*(k) = & \arg\min_{U(k)} J(x(k), U(k)) \\ & \text{subj. to MLD model (2.8) at } \ell = k, \ldots, k+N-1\,. \end{aligned} \tag{2.15}$$

The control input at time-instant k is obtained by solving the CFTOC (2.15), i.e. by minimizing the objective function (2.14) subject to the evolution of the MLD model (2.8) and its mixed-integer linear inequality constraints. Furthermore, additional integrality constraints are present on the binary δ variables and on binary inputs[1]. This yields the sequence of optimal control inputs $U^*(k) = [(u^*(k))^T, \ldots, (u^*(k+N-1))^T]^T$. As we are using a linear norm in all cost expressions, the CFTOC problem amounts to solving a *Mixed-Integer Linear Program* (MILP) for which efficient solvers exist, like [ILO02].

2.4.2 Off-Line Computation of State-Feedback Control Law

Often, the computation times needed for solving the optimal control problem on-line are well beyond the sampling interval, and the controller cannot be directly implemented. In order to overcome this obstacle, the calculation of the explicit state-feedback control law is necessary by pre-computing off-line the solution to the optimal control problem for the whole state-space using multi-parametric programming, where the state vector is treated as a parameter. The resulting control law is a PWA state-feedback control law defined over a polyhedral partition of the state-space which can be stored in a look-up table. Computing the optimal control law on-line is thus reduced to a simple evaluation of a look-up table. For hybrid systems, such a method has been recently introduced, which is based on a PWA description of the controlled system.

In the sequel, we outline multi-parametric programming, summarize two algorithms for pre-computing off-line the optimal control problem, characterize the solution to the optimal control problem, and discuss issues concerning the practical implementation. For this, we partly follow the line of thought of the summary paper [MBB03].

Multi-Parametric Programming

Reconsider problem (2.11) that now additionally depends on a parameter appearing in the cost function and the constraints

$$\begin{aligned} J^*(x) = \min_{\zeta} \quad & f_0(\zeta, x) \\ \text{subj. to} \quad & f_i(\zeta, x) \leq 0, \qquad i = 1, \ldots, n_f\,, \end{aligned} \tag{2.16}$$

[1] It is not necessary to impose integrality constraints on binary states, since the state-update dynamics preserve integrality of the states for all future steps.

with the optimization variable $\zeta \in \mathbb{R}^{d_r} \times \{0,1\}^{d_b}$ as before, the parameter $x \in \mathbb{R}^{n_r}$, the cost function $f_0 : \mathbb{R}^d \times \mathbb{R}^{n_r} \to \mathbb{R}$, and the inequality constraint functions $f_i : \mathbb{R}^d \times \mathbb{R}^{n_r} \to \mathbb{R}$, $i = 1, \ldots, n_f$. Deriving the optimizer $\zeta^*(x)$ as a function of the parameter x is referred to as multi-parametric programming. The real-valued function $J^*(x)$, which expresses the dependence of the minimum value of the objective function in terms of x, is called the (optimal) value function.

In the following, we define two special classes of multi-parametric programming that will be used in this thesis.

Definition 2.11 (mp-LP) *If $f_0(\cdot,\cdot)$ and $f_i(\cdot,\cdot)$ are linear, $\zeta \in \mathbb{R}^{d_r}$ (with $d_b = 0$) and $x \in \mathbb{R}^{n_r}$ the problem* (2.16) *is called a multi-parametric Linear Program (mp-LP).*

Definition 2.12 (mp-MILP) *If $f_0(\cdot,\cdot)$ and $f_i(\cdot,\cdot)$ are linear, $\zeta \in \mathbb{R}^{d_r} \times \{0,1\}^{d_b}$ and $x \in \mathbb{R}^{n_r}$ the problem* (2.16) *is called a multi-parametric Mixed-Integer Linear Program (mp-MILP).*

Moreover, multi-parametric Quadratic Programming (mp-QP) is well-established, too, where $f_0(\cdot,\cdot)$ is a quadratic function. Multi-parametric Mixed-Integer Quadratic Programming (mp-MIQP), however, leads to a number of complications such as non-polyhedral sets [BBBM05].

Algorithms

As shown next, the optimal control problem (2.14)–(2.15) can be reformulated as the multi-parametric program (2.16), where ζ is the input sequence to be optimized and the parameter x is the current state of the plant. For MILPs, two approaches exist to solve the mp-MILP: mp-LP with MILP, and mp-LP with Dynamic Programming (DP).

The *mp-LP with MILP* scheme directly tackles the problem formulation (2.14)–(2.15) with the MLD model, by decomposing the mp-MILP problem into an mp-LP and an MILP subproblem and iterating between them. Specifically, in a first step, a solution to the MILP is obtained by treating the state as a free parameter. This yields a feasible integer vector that is fixed in the mp-MILP, and reduces the mp-MILP to an mp-LP. Then, for the fixed integer vector, the mp-LP is solved and the PWA value function is derived on the corresponding polyhedra, the so called critical regions. For each critical region, solving an new MILP subproblem provides a new integer vector, using the current value function as an upper bound. The parametric solutions corresponding to two different integer solutions are then compared and inferior parts are removed. The algorithm terminates in a critical region if the MILP subproblem is infeasible, which implies that the current upper bound is the final optimal solution. For details, the reader is referred to [DP00].

As proposed in [Bor03, BBBM05], an alternative approach is to construct the state-feedback control law by combining *mp-LP with DP*, i.e. by moving backwards in time

using mp-LP. For this scheme, the CFTOC needs to be reformulated. By replacing the MLD model by the equivalent PWA model, we obtain the reformulated CFTOC

$$\begin{aligned} U^*(k) = \arg\min_{U(k)} & J(x(k), U(k)) \\ & \text{subj. to PWA model (2.9) at } \ell = k, \ldots, k+N-1\,. \end{aligned} \tag{2.17}$$

A comparison of the two approaches can be found in [BCM03b]. It is clear, that the mp-LP with MILP approach is more general, since it directly tackles MILPs, which in our case are obtained by setting up the CFTOC with the MLD model. The mp-LP with DP approach, however, is based on PWA models, which contain structural information that is in general hidden in its MLD counterpart. In particular, the PWA model contains only feasible modes, while most of the integer combinations in an MLD model refer to infeasible modes. This leads to computational advantages of the DP approach with respect to its MILP counterpart. In particular, the computation times are in general smaller, unnecessary slicing of the state-space is handled in a better way thus leading to less polyhedra in the control law, and the DP approach is numerically more reliable. Moreover, it is possible to detect if the solution for a specific prediction horizon is identical to the infinite horizon solution [BCM03b]. Yet, the DP approach suffers from the enumeration of all feasible transitions between polyhedra (at one time-step) and the enumeration of all binary state-input combinations. Nevertheless, in this thesis, we will adopt the DP approach included in the multi-parametric toolbox (MPT) [KGBM04], which is freely available from http://control.ee.ethz.ch/~mpt/.

More details about multi-parametric programming can be found in [BBM03] for LPs, in [BMDP02, TJB01] for QPs, in [DP00] for MILPs, and in [Bor03, DBP02] for MIQPs. Furthermore, [BBBM05] provides an in depth analysis and description of multi-parametric programming for MILPs and MIQPs.

Properties

Next, we restate the main result about the solution to the CFTOC problem proven in [Bor03].

Theorem 2.1 *The solution to the CFTOC* (2.14) *and* (2.17) *is a state-feedback control law* $u^*(k)$ *that is a PWA function of the state* $x(k)$ *defined on a polyhedral partition of the feasible state-space.*

More specifically, the feasible state-space is partitioned into polyhedral sets, and for each of these sets the optimal control law is given as an affine function of the state. Moreover, the value function $J^*(x(k)) = J(x(k), U^*(k))$ is also PWA in the state.

Implementation

As a result, such a state-feedback controller can be implemented on-line, since computing the control input amounts to the following two steps. First, the polyhedron needs to be determined in which the measured state lies. The brute force approach is to go through (in the worst case) the whole set of polyhedra and to check the corresponding inequalities of the polyhedra. An alternative approach is to build a binary search tree [TJB03] that reduces the on-line computational burden but increases the memory requirements. In a last step, once the proper polyhedron has been found, one only needs to identify and evaluate the corresponding affine control law.

In many cases, polyhedra with the same control law form a convex union and can thus be merged and replaced by their union. This leads to an equivalent PWA control law with fewer polyhedra and thus reduced complexity. Such a representation is highly preferable as it allows one to relax the memory requirements and to reduce the computational burden for the controller hardware. Indeed, it is possible to derive an equivalent PWA control law that is *minimal* in the number of polyhedra by merging polyhedra associated with the same control law in an optimal way (see Chapter 4).

2.4.3 Receding Horizon Control

The CFTOC problem presented above yields at time-step k an *open-loop* optimal sequence $U^*(k)$ of control inputs. To provide feedback, only the first input $u^*(k)$ from the sequence is applied to the plant. At the next sampling interval, k is set to $k+1$, a new state measurement (or estimate) is obtained, and the CFTOC problem is solved again over the shifted horizon. This policy is referred to as Receding Horizon Control[2], which has become standard practise in modern control applications. For details, see for example [MRRS00, QB03, Mac02].

[2] As shown above, the CFTOC problem might be pre-solved off-line and replaced by the corresponding state-feedback controller. This does not affect the concept of Receding Horizon Control.

Part II

Mode Enumeration and Optimal Complexity Reduction

3

Mode Enumeration of Compositional Hybrid Systems

3.1 Introduction

Hybrid systems can be composed to form *compositional hybrid systems* [AH97, Joh00, LSV01, RL99]. In general, the resulting system is very complex, as the number of different operational modes depends exponentially on the number of component systems. The explosion of the size of the logic state leads to computational difficulties as the time and space complexity of many algorithms depends on the number of operational modes.

In some cases, the composition induces a structure that can be exploited, like in hierarchical hybrid systems [ADE+01]. This allows one to break the problem down into pieces and to apply the assume-guarantee approach [AH97, HMP01]. Recognizing that a system can be modelled as a hierarchical hybrid system is part of the "art" of model building. In many cases it may not be possible at all, because the cross interactions among the components are too tight. On the other hand, tight interactions often render many modes infeasible and the complexity of the system can be reduced by explicitly computing and taking into account only the feasible modes. This chapter presents an efficient technique to enumerate the feasible modes of a compositional hybrid system.

We will focus on compositions of *discrete hybrid automata* (DHA) [TB04]. DHAs, which are formulated in the discrete-time domain, are as mentioned in the previous chapter a mathematical abstraction of the features provided by other computation oriented and domain specific hybrid system frameworks. Thus DHAs generalize many discrete-time modelling frameworks for hybrid systems and represent a universal starting point for solving complex analysis and synthesis problems. In particular, the enumeration of feasible modes is easily solvable for DHA systems by using algorithms that compute the cells of a hyperplane arrangement [FFL01]. This is a classical problem in computational geometry [Buc43] and admits optimal [Ede87] and efficient [AF96, FF02, FFL01] algorithms.

The impact of those techniques on applications is threefold. First, at the modelling stage, the enumeration of modes allows the designer to understand the real complexity of the compound model.

Second, after the modelling stage, the model can be efficiently translated into a PWA representation. This operation is trivial once all modes have been enumerated. In this respect, this chapter solves a problem similar to [Bem02], where the author computes the PWA model equivalent to an MLD model. The main difference is that the approach in [Bem02] is based on multi-parametric programming and mixed integer linear programming and deals directly with the MLD model, while the approach presented here relies on the computation of the cells of the hyperplane arrangement and is applicable to DHAs. Note that, as shown in [TB04], DHAs can be automatically translated into MLD models using the tool HYSDEL and most of the MLD models that have been presented in the literature were derived from DHA descriptions using HYSDEL [BGKH02, BBFH01, ED02]. Most importantly, using the structural information only available in the HYSDEL code (and not in the corresponding MLD model) allowed us to design a conversion tool that is by at least an order of magnitude faster compared with its counterpart [Bem02]. Furthermore, the PWA representation of the compound DHA model allows one to determine if the model is well-posed. This option becomes notably interesting if loops are present in the compound model, as a composition of well-posed DHAs is not necessarily well-posed as a whole.

Third, during the computational stage (i.e. analysis and control), the explicit computation of the set of feasible modes of the compound system allows one to prune unnecessary modes from the resulting system and to reduce the combinatorial explosion of related algorithms. This is of particular importance for *model predictive control* (MPC) of hybrid models [BM99a], where the aim is to compute the next N inputs to the system that optimize a performance index defined on the variables of a hybrid *prediction model*. The prediction model is the series connection of N identical single-step prediction models where each model uses the state predicted by the previous model. The mode enumeration allows one to introduce cuts on the modes of the complete prediction model.

This chapter is organized as follows: Section 3.2 presents the problem of cell enumeration in hyperplane arrangements. In Section 3.3, we show the equivalence of DHAs and PWA systems. Based on this, Section 3.4 details how these results can be applied in the hybrid domain. In particular, we present an algorithm that enumerates the modes of a composition of DHAs and transforms it into an equivalent PWA representation. Section 3.5 contains simulation results, where we enumerate the modes of a hybrid model built using the HYSDEL [TB04] modelling language. The information collected during this mode enumeration allows one to either build efficiently an equivalent PWA model or to speed up the computation of the hybrid MPC feedback law by automatically adding cuts to the underlying optimization problem as shown in Section 3.6. Section 3.7 summarizes

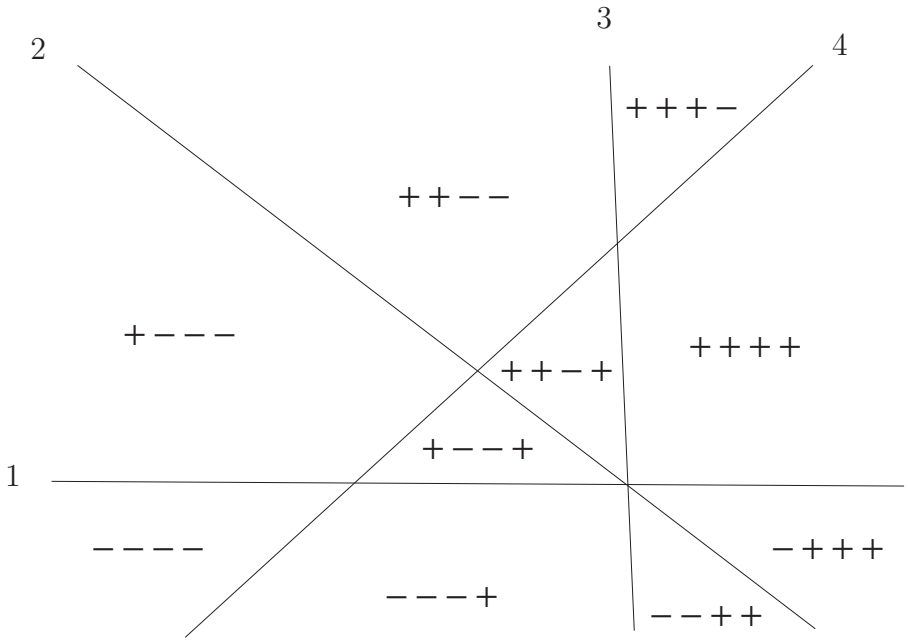


Figure 3.1: Arrangement of four hyperplanes (lines) in $\mathcal{R} = \mathbb{R}^2$ with markings $m \in M(\mathcal{R})$

the chapter.

The presented algorithm is implemented in MATLAB and assumes that the composition of DHAs is given as HYSDEL code. The latest version of the algorithm can be downloaded from http://control.ethz.ch/~hybrid/hysdel.

3.2 Cell Enumeration in Hyperplane Arrangements

Let $\mathcal{A}$ 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 $H_i = \{x \in \mathbb{R}^d \mid a_i^T x = b_i\}$. We say that the hyperplanes of $\mathcal{A}$ 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 $\mathrm{SV} : \mathbb{R}^d \to \{-,+\}^n$ be the simplified sign vector[1] defined as

$$\mathrm{SV}_i(x) = \begin{cases} - & \text{if } a_i^T x \leq b_i, \\ + & \text{if } a_i^T x > b_i \end{cases} \quad \text{for } i \in \{1, 2, \dots, n\}. \tag{3.1}$$

Consider the set $\mathcal{P}_m = \{x \in \mathbb{R}^d \mid \mathrm{SV}(x) = m\}$ for a given sign vector m. This set is called a *cell* of the arrangement and is according to Definition 2.5 a polyhedron as it is defined by linear inequalities. We will refer to m as the *marking* of the polyhedron (or cell) $\mathcal{P}_m$

[1]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 and not meaningful in the context of PWA systems.

in the *hyperplane arrangement* $\mathcal{A}$ (see Fig. 3.1). Let $M(\mathcal{R})$ be the image of the function $\mathrm{SV}(x)$ 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. 3.1 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 [Buc43] defines the upper bound

$$\#M \leq \sum_{i=0}^{d} \binom{n}{i} = O(n^d), \tag{3.2}$$

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 $O(n^d)$ [Ede87]. An alternative approach based on reverse search was presented in [AF96], improved in [FFL01] and implemented in [FF02]. 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.

Proposition 3.1 *[FFL01, Theorem 4.1] There exists a reverse search algorithm for enumerating hyperplane arrangements that runs in $O(n\,\mathrm{lp}(n,d)\,\#M)$ time and $O(n,d)$ space, where $\mathrm{lp}(n,d)$ denotes the complexity of solving a Linear Program (LP) with n constraints and d variables.*

Note that in many cases of interest, the hyperplanes are not in general position and $\#M$ is considerably smaller than the theoretical upper bound. Moreover, reverse search is a standard search algorithm for which efficient parallel implementations exist [BMFN99].

The following proposition follows directly from the definition of $\mathcal{P}_m$ and (3.1).

Proposition 3.2 *The collection of polyhedral sets $\{\mathcal{P}_m\}_{m \in M(\mathcal{R})}$ satisfies:*

$$(i)\ \ \bigcup_{m \in M(\mathcal{R})} \mathcal{P}_m = \mathcal{R}, \qquad (ii)\ \ \mathcal{P}_i \cap \mathcal{P}_j = \emptyset,\ \forall i, j \in M(\mathcal{R}),\ i \neq j$$

A collection of polyhedral sets that satisfies points *(i)* and *(ii)* in Proposition 3.2 is a *polyhedral partition* of the polyhedral set $\mathcal{R}$.

3.3 Equivalence of DHAs and PWA Systems

This section states that any well-posed DHA can be transformed into an equivalent PWA representation. For the definition of DHAs and PWA models, the reader is deferred to Sections 2.2.1 and 2.2.3, respectively. The constructive proof serves as a basis for the mode enumeration algorithm proposed in the next section.

Definition 3.1 *Let Σ_1, Σ_2 be well-posed hybrid models with states x_1, $x_2 \in \mathcal{X}$, inputs u_1, $u_2 \in \mathcal{U}$ and outputs y_1, $y_2 \in \mathcal{Y}$. The hybrid models Σ_1 and Σ_2 are* equivalent *on $\mathcal{X}$, $\mathcal{U}$, $\mathcal{Y}$, if for all initial conditions $x_1(0) = x_2(0) \in \mathcal{X}$ and for all $u_1(k) = u_2(k) \in \mathcal{U}$ the state and output trajectories coincide, i.e. $x_1(k) = x_2(k)$ and $y_1(k) = y_2(k)$ for all steps $k \in \mathbb{N}_0$.*

Lemma 3.1 *[TB04, Lemma 1] Let Σ_{PWA} be a well-posed PWA model with states $x \in \mathcal{X}$, inputs $u \in \mathcal{U}$ and outputs $y \in \mathcal{Y}$. Then there exists a well-posed DHA Σ_{DHA} equivalent to Σ_{PWA} on $\mathcal{X}$, $\mathcal{U}$, $\mathcal{Y}$.*

The equivalence of the previous lemma allows one to refer to the set $\mathcal{J}$ as the modes of the PWA system (2.9).

Lemma 3.2 *Let Σ_{DHA} be a well-posed DHA with states $x \in \mathcal{X}$, inputs $u \in \mathcal{U}$ and outputs $y \in \mathcal{Y}$. Then there exists a well-posed PWA model Σ_{PWA} equivalent to Σ_{DHA} on $\mathcal{X}$, $\mathcal{U}$, $\mathcal{Y}$.*

Proof. Consider the affine thresholds of the event generator (2.5) that define a hyperplane arrangement, which forms by Proposition 3.2 a polyhedral partition. Let $\mathcal{Q}_m$ be a polyhedron of this partition. By construction, $\bar{\delta}_e(m) = f_{\mathrm{H}}(x_r, u_r)$ for any point $[x_r^T, u_r^T]^T \in \mathcal{Q}_m$, namely all points in $\mathcal{Q}_m$ trigger the same event $\bar{\delta}_e(m)$. Given a marking m, the associated event $\bar{\delta}_e(m)$, a binary state $\bar{x}_b \in \mathcal{X}_b$ and a binary input $\bar{u}_b \in \mathcal{U}_b$, the mode selector determines the mode $\bar{\imath} = f_{\mathrm{M}}(\bar{x}_b, \bar{u}_b, \bar{\delta}_e(m))$ using the logic function (2.7). The $\bar{\imath}$-th dynamic in the switched affine system given by (2.4a) and (2.4b) is the corresponding affine dynamic. The finite state machine yields the binary state-update as well as the binary output according to (2.6a) and (2.6b). Therefore, for each $m \in M$, $\bar{x}_b \in \mathcal{X}_b$ and $\bar{u}_b \in \mathcal{U}_b$, the system

$$x_r(k+1) = A_{\bar{\imath}} x_r(k) + B_{\bar{\imath}} u_r(k) + f_{\bar{\imath}}, \tag{3.3a}$$

$$x_b(k+1) = f_{\mathrm{B}}(\bar{x}_b, \bar{u}_b, \bar{\delta}_{e(m)}), \tag{3.3b}$$

$$y_r(k) = C_{\bar{\imath}} x_r(k) + D_{\bar{\imath}} u_r(k) + g_{\bar{\imath}}, \tag{3.3c}$$

$$y_b(k) = g_{\mathrm{B}}(\bar{x}_b, \bar{u}_b, \bar{\delta}_{e(m)}), \tag{3.3d}$$

$$\text{if } [x_r^T(k), u_r^T(k)]^T \in \mathcal{Q}_m, \quad x_b(k) = \bar{x}_b, \quad u_b(k) = \bar{u}_b, \tag{3.3e}$$

defines a PWA system. In fact, by collecting $x(k) = \begin{bmatrix} x_r(k) \\ x_b(k) \end{bmatrix}$, $u(k) = \begin{bmatrix} u_r(k) \\ u_b(k) \end{bmatrix}$ and $y(k) = \begin{bmatrix} y_r(k) \\ y_b(k) \end{bmatrix}$, by performing the substitutions $A_{j(k)} = \left[\begin{smallmatrix} A_{\bar{i}} & 0 \\ 0 & 0 \end{smallmatrix}\right]$, $B_{j(k)} = \left[\begin{smallmatrix} B_{\bar{i}} & 0 \\ 0 & 0 \end{smallmatrix}\right]$, $f_{j(k)} = \begin{bmatrix} f_{\bar{i}} \\ f_{\mathrm{B}(\cdot)} \end{bmatrix}$ and similarly for $C_{j(k)}$, $D_{j(k)}$ and $g_{j(k)}$, and by defining $\mathcal{P}_{j(k)} \triangleq \mathcal{Q}_m \times \bar{x}_b \times \bar{u}_b \in \mathcal{X} \times \mathcal{U}$, (3.3a)–(3.3d) are formally equivalent to (2.9a)–(2.9b) and (3.3e) is formally equivalent to (2.9c). The well-posedness of the PWA model follows from Proposition 3.2 and Lemma 2.2. □

3.4 Mode Enumeration Algorithm

Based on the cell enumeration in hyperplane arrangements summarized in Section 3.2 and the equivalence of DHAs and PWA models shown in Section 3.3, we present in this section an algorithm that enumerates efficiently the feasible modes of a composition of DHAs and derives an equivalent PWA model.

3.4.1 Single DHA

Consider the DHA Σ as in Section 2.2.1 and let $\mathcal{X} \times \mathcal{U}$ denote the state-input space of the DHA, for which we want to solve the following problem. Given a binary state $x_b(k) \in \mathcal{X}_b$ and a binary input $u_b(k) \in \mathcal{U}_b$ find the set of feasible modes $\mathcal{J} \subseteq \mathcal{I}$[2], the set of polyhedra $\{\mathcal{P}_j\}_{j\in\mathcal{J}}$ and the corresponding PWA dynamics $\{\mathcal{S}_j\}_{j\in\mathcal{J}}$, where $\mathcal{S}_j = \{A_j, B_j, f_j, C_j, D_j, g_j\}$. As this is the same problem as in Lemma 3.2, we derive an algorithm from the constructive proof of the lemma. Note that $\mathcal{I}$ is the image of the Mode Selector and can be computed once the set $M(\mathcal{X}_r \times \mathcal{U}_r)$ has been enumerated.

Algorithm 3.1

function [$\{\mathcal{P}_j\}_{j\in\mathcal{J}}$, $\{\mathcal{S}_j\}_{j\in\mathcal{J}}$] = SingleDHA (Σ, $\mathcal{R}$, $x_b(k)$, $u_b(k)$)
  $j = 0$, $\mathcal{J} = \emptyset$
  for $m \in M(\mathcal{R})$
    $j = j + 1$, $\mathcal{J} = \mathcal{J} \cup \{j\}$
    $\mathcal{P}_j = \mathcal{P}_m = \{x \in \mathcal{R} : \mathrm{SV}(x) = m\}$
    get $\delta_e(k)$ based on m
    $i(k) = f_{\mathrm{M}}(x_b(k), u_b(k), \delta_e(k))$
    $\mathcal{S}_j = \Big\{ \left[\begin{smallmatrix} A_{i(k)} & 0 \\ 0 & 0 \end{smallmatrix}\right], \left[\begin{smallmatrix} B_{i(k)} & 0 \\ 0 & 0 \end{smallmatrix}\right], \begin{bmatrix} f_{i(k)} \\ f_{\mathrm{B}}(x_b(k),u_b(k),\delta_e(k)) \end{bmatrix}, \left[\begin{smallmatrix} C_{i(k)} & 0 \\ 0 & 0 \end{smallmatrix}\right], \left[\begin{smallmatrix} D_{i(k)} & 0 \\ 0 & 0 \end{smallmatrix}\right], \begin{bmatrix} g_{i(k)} \\ g_{\mathrm{B}}(x_b(k),u_b(k),\delta_e(k)) \end{bmatrix} \Big\}$
  return [$\{\mathcal{P}_j\}_{j\in\mathcal{J}}$, $\{\mathcal{S}_j\}_{j\in\mathcal{J}}$]

Algorithm 3.1 enumerates the feasible modes for a given $x_b(k) \in \mathcal{X}_b$ and $u_b(k) \in \mathcal{U}_b$. Re-

[2] $\mathcal{J} = \mathcal{I}$ holds, if all modes $\mathcal{I}$ of the Switched Affine System are feasible.

peated calls of Algorithm 3.1 lead to a set of PWA models defined on $\mathcal{X}_r \times \mathcal{U}_r$, where each model is associated with a feasible combination of binary states and inputs $x_b(k) \in \mathcal{X}_b$, $u_b(k) \in \mathcal{U}_b$. This representation is advantageous if determining the state-update and the outputs for a given state and input is the main purpose, as choosing the respective PWA model can be done by binary search. However, the model can be transformed easily into a PWA model defined on $\mathcal{X} \times \mathcal{U}$ as shown in the proof of Lemma 3.2.

Remark 3.1 If the DHA Σ is well-posed, the resulting PWA model is well-posed, too, as shown in Lemma 3.2. Furthermore, by Proposition 3.2, the set of polyhedra $\{\mathcal{P}_j\}_{j\in\mathcal{J}}$ forms a polyhedral partition.

3.4.2 Composition of DHAs

The algorithm proposed above can be extended in a natural way to deal with a composition of DHAs. Consider s DHAs denoted as Σ_i, $i \in \{1, 2, \ldots, s\}$ with states $x_i \in \mathcal{X}_i$, inputs $u_i \in \mathcal{U}_i$ and outputs $y_i \in \mathcal{Y}_i$[3]. Let $\mathcal{I}_i$ be the set of feasible modes of the DHA Σ_i. The composition has the exogenous input $u \in \mathcal{U}$ and the exogenous output $y \in \mathcal{Y}$. We define the real and binary state spaces of the composition $\mathcal{X}_r \triangleq \mathcal{X}_r^1 \times \ldots \times \mathcal{X}_r^s$ and $\mathcal{X}_b \triangleq \mathcal{X}_b^1 \times \ldots \times \mathcal{X}_b^s$ and let the compound vectors $x_r \triangleq [(x_r^1)^T, \ldots, (x_r^s)^T]^T \in \mathcal{X}_r$ and $x_b \triangleq [(x_b^1)^T, \ldots, (x_b^s)^T]^T \in \mathcal{X}_b$ be the sorted aggregation of the real and binary states of the s DHAs, respectively. Summing up, the compound system has the compound state vector $x = \left[\begin{smallmatrix} x_r \\ x_b \end{smallmatrix}\right] \in \mathcal{X}_r \times \mathcal{X}_b$ and the exogenous input u and output y. The time indices k have been omitted for the sake of readability.

Before describing the algorithm, we recall some definitions and results from graph theory [Deo74] to describe the topology of the composition.

A *directed graph* or *digraph* $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ consists of a set of *vertices* $\mathscr{V}$, a set of *edges* $\mathscr{E}$, and a mapping that maps every edge onto some ordered pair of vertices. A *directed closed walk* is an alternating sequence of vertices and edges, beginning and ending with the same vertex, such that each edge is oriented from the vertex preceding it to the vertex following it. If additionally, no vertices except the initial and terminal one appear more than once, the directed closed walk is called a *directed circuit*. If a digraph has no directed circuits, it is called *acyclic*, otherwise it is *cyclic*.

The definitions above can be applied directly to the composition of DHAs by defining the DHAs as vertices and the connections from outputs to inputs as directed edges. In general one edge can represent several connections between two DHAs.

Note that directed circuits in $\mathscr{G}$ correspond to feedback loops in the composition. Conversely, an acyclic directed graph implies the lack of loops. Besides that, the state-updates

[3] As in the last section x_i, u_i and y_i encompass both real and binary components. Furthermore, $\mathcal{X}_i \triangleq \mathcal{X}_r^i \times \mathcal{X}_b^i$ and accordingly for $\mathcal{U}_i$ and $\mathcal{Y}_i$.

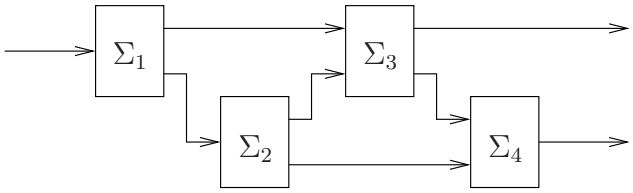


Figure 3.2: Composition of DHAs of Example 3.1 after reordering

$x(k+1)$ can be considered as outputs of the composition similarly to the outputs $y(k)$. In particular, state-update functions cannot be part of loops and they have no influence on the set of polyhedra of the resulting PWA model.

We define the topology of the connections among the DHAs by an adjacency matrix, which can be easily determined based on the connections.

Definition 3.2 *Let $\mathscr{G}$ be a digraph with s vertices containing at most one edge per pair of vertices. Then the adjacency matrix $W = [w_{ij}]$ of the digraph $\mathscr{G}$ is a $s \times s$ $(0,1)$-matrix with $w_{ij} = 1$ if there is an edge directed from the i-th vertex to the j-th vertex and $w_{ij} = 0$ otherwise. The sequence of indices of W is given by $\{1, 2, \ldots, s\}$.*

Theorem 3.1 *[Deo74, Theorem 9.17] The digraph $\mathscr{G}$ is* acyclic *if and only if* $\det(I - W) \neq 0$*, where I is the identity matrix.*

Theorem 3.2 *[Deo74, Theorem 9.16] If the digraph $\mathscr{G}$ is acyclic, then its vertices can be ordered such that the adjacency matrix of the reordered graph is an upper (or lower) triangular matrix.*

As defined in [GTM03a], the sequence of indices of the reordered adjacency matrix implies a computational order along which the algorithm will proceed.

Example 3.1 Figure 3.2 depicts four DHAs and the connections among them after reordering the corresponding graph. DHA 1 has an exogenous input, DHA 3 and 4 have exogenous outputs. The computational order is given by $\{1, 2, 3, 4\}$.

Compositions Without Loops

In a first step, we assume that the connections do not form loops. This can be easily determined by Theorem 3.1. From Theorem 3.2 follows, that the adjacency matrix can be transformed into an upper triangular matrix employing for example topological sorting [Deo74] or matrix permutation. Furthermore, we assume that the indices of the DHAs are such that the adjacency matrix is upper triangular. This implies that the computational order is given by $\{1, 2, \ldots, s\}$. Consequently, Σ_i depends only on exogenous inputs and on outputs of Σ_j, $j < i$ with $i, j \in \{1, \ldots, s\}$.

As in the single DHA case, our aim is to determine for the composition of DHAs $\{\Sigma_i\}_{i=1,\ldots,s}$ defined on $\mathcal{X} \times \mathcal{U}$ the set of feasible modes $\mathcal{J} \subseteq \mathcal{I}_1 \times \ldots \times \mathcal{I}_s$, the set of polyhedra $\{\mathcal{P}_j\}_{j\in\mathcal{J}}$ and the corresponding PWA dynamics $\{\mathcal{S}_j\}_{j\in\mathcal{J}}$. Note that $\mathcal{X} \times \mathcal{U}$ is now generalized to the *compound* state *exogenous* input space. For a given binary compound state $x_b(k) \in \mathcal{X}_b$ and a given exogenous binary input $u_b(k) \in \mathcal{U}_b$, Algorithm 3.2 partitions the real state-input space $\mathcal{X}_r \times \mathcal{U}_r$ recursively as described in the following.

Consider the first DHA Σ_l, $l = 1$. As mentioned before, the input of this DHA is a subset of the exogenous input and thus given. Therefore, Algorithm 3.1 can be used to determine the modes $\mathcal{J}_l$, the polyhedra $\{\mathcal{P}_j\}_{j\in\mathcal{J}_l}$ and the corresponding PWA dynamics $\{\mathcal{S}_j\}_{j\in\mathcal{J}_l}$ of Σ_l.

Each mode $j \in \mathcal{J}_l$ with its PWA dynamic $\mathcal{S}_j$ defines together with the connections inputs of DHAs with higher computational order given by Σ_i, $i \in \{l+1, \ldots, s\}$. Before proceeding, we have to replace their inputs by $\mathcal{X}_r$, $\mathcal{U}_r$, $x_b(k)$ and $u_b(k)$ using the function Subst(). Given the set of DHAs $\{\Sigma_i\}_{i=l+1,\ldots,s}$, the PWA dynamic $\mathcal{S}_j$ corresponding to mode $j \in \mathcal{J}_l$, the function Subst() cycles through all DHAs of the given set. If the adjacency matrix indicates a connection from an output of Σ_l to an input of one of the DHAs of the set $\{l+1, \ldots, s\}$, the respective input is replaced by $C_j \begin{bmatrix} x_r(k) \\ x_b(k) \end{bmatrix} + D_j \begin{bmatrix} u_r(k) \\ u_b(k) \end{bmatrix} + g_j$, where C_j, D_j, g_j are elements of $\mathcal{S}_j$ and $x_r(k) \in \mathcal{X}_r$, $u_r(k) \in \mathcal{U}_r$, and $x_b(k)$, $u_b(k)$ are given. This operation assures, that after the replacement Σ_{l+1} solely depends on compound states and exogenous inputs.

Next, for a given mode $j \in \mathcal{J}_l$, l is increased by one and the algorithm is called again to partition the polyhedron $\mathcal{P}_j$ into a set of polyhedra using the hyperplanes of the DHAs with computational order greater than l. This is repeated for all the remaining $j \in \mathcal{J}_l$. If l reaches its maximum s, the current branch terminates and the set of polyhedra of Σ_s, which are part of the overall set of polyhedra $\{\mathcal{P}_j\}_{j\in\mathcal{J}}$ of the compound DHA system, are added to it. Stepping sequentially through the composition of DHAs according to their computational order leads to the set of polyhedra $\{\mathcal{P}_j\}_{j\in\mathcal{J}}$ and the corresponding PWA dynamic $\{\mathcal{S}_j\}_{j\in\mathcal{J}}$.

Remark 3.2 In every step of the algorithm, the set of polyhedra is always defined on the complete real state-input space – or more precisely, on the real compound state exogenous input space – $\mathcal{X}_r \times \mathcal{U}_r$. As the algorithm proceeds, additional hyperplanes are added cutting the existing polyhedra into smaller ones.

Remark 3.3 As described above, the algorithm is currently implemented as a depth-first algorithm. Alternatively, it could be easily restructured to work breadth-first. In any case, the chosen strategy has no influence on the complexity.

Recapitulating the above and given $x_b(k) \in \mathcal{X}_b$ and $u_b(k) \in \mathcal{U}_b$, the Algorithm 3.2 is

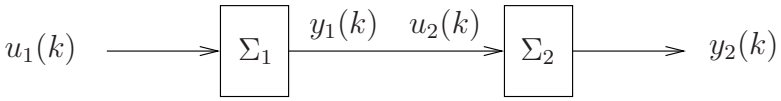


Figure 3.3: Composition of DHAs of Example 3.2

summarized as follows:

Algorithm 3.2

reorder $\{\Sigma_i\}_{i=1,\dots,s}$ such that W is upper triangular
$[\ \{\mathcal{P}_j\}_{j\in\mathcal{J}},\ \{\mathcal{S}_j\}_{j\in\mathcal{J}}\ ] = \mathsf{Comp}(\ \{\Sigma_i\}_{i=1,\dots,s},\ \mathcal{X}_r \times \mathcal{U}_r,\ x_b(k),\ u_b(k),\ 1\ )$

function $[\ \{\mathcal{P}_j\}_{j\in\mathcal{J}},\ \{\mathcal{S}_j\}_{j\in\mathcal{J}}\ ] = \mathsf{Comp}\ (\ \{\Sigma_i\}_{i=1,\dots,s},\ \mathcal{R},\ x_b(k),\ u_b(k),\ l\ )$
  $[\ \{\mathcal{P}_j\}_{j\in\mathcal{J}_l},\ \{\mathcal{S}_j\}_{j\in\mathcal{J}_l}\ ] = \mathsf{SingleDHA}\ (\ \Sigma_l,\ \mathcal{R},\ x_b(k),\ u_b(k)\ )$
  if $l < s$ **then**
    $P = \emptyset, \quad S = \emptyset$
    for $j \in \mathcal{J}_l$
      $[\ P_{new},\ S_{new}\ ] = \mathsf{Comp}\ (\ \mathsf{Subst}\ (\ \{\Sigma_i\}_{i=l+1,\dots,s},\ \mathcal{S}_j,\ l\ ),\ \mathcal{P}_j,\ x_b(k),\ u_b(k),\ l+1\ )$
      $P = \{\ P,\ P_{new}\ \}, \quad S = \{\ S,\ S_{new}\ \}$
    return $[\ P,\ S\ ]$
  else
    return $[\ \{\mathcal{P}_j\}_{j\in\mathcal{J}_l},\ \{\mathcal{S}_j\}_{j\in\mathcal{J}_l}\ ]$

function $\{\Sigma_i\}_{i=l+1,\dots,s} = \mathsf{Subst}\ (\ \{\Sigma_i\}_{i=l+1,\dots,s},\ \mathcal{S}_j,\ l\ )$
  for $i \in \{l+1,\dots,s\}$
    if there is a connection $y_l = u_i$ **then**
      substitute u_i in Σ_i by $C_j x(k) + D_j u(k) + g_j$
  return $\{\Sigma_i\}_{i=l+1,\dots,s}$

As for the single DHA case, the algorithm yields a PWA model defined on $\mathcal{X}_r \times \mathcal{U}_r$ for every feasible binary state-input combination. The following corollary extends Remark 3.1. It follows in a constructive way from Algorithm 3.2, Proposition 3.2 and Lemma 3.2.

Corollary 3.1 *Given a composition of DHAs $\{\Sigma_i\}_{i=1,\dots,s}$ without loops, where each Σ_i is well-posed, the resulting PWA model is well-posed, too, and its set of polyhedra forms a polyhedral partition.*

Example 3.2 Consider now the composition of DHAs shown in Fig. 3.3 with the state $x_1(k) \in \mathcal{X} = [0, 10]$, the exogenous input $u_1(k) \in \mathcal{U} = [0, 7]$, the exogenous output

$y_2(k) \in \mathbb{R}$, and the connection $u_2(k) = y_1(k)$. The DHA Σ_1 is given by[4]

$$\text{EG}_1: \quad \begin{cases} \delta_1(k) = [x_1(k) \geq 4], \\ \delta_2(k) = [x_1(k) \geq 8] \end{cases}$$

$$\text{MS}_1: \quad i_1(k) = \begin{cases} 1 & \text{if } \bar{\delta}_1(k) \ \wedge \ \bar{\delta}_2(k), \\ 2 & \text{if } \delta_1(k) \ \wedge \ \bar{\delta}_2(k), \\ 3 & \text{if } \delta_1(k) \ \wedge \ \delta_2(k) \end{cases}$$

$$\text{SAS}_1: \quad y_1(k) = \begin{cases} 2u_1(k) - 6 & \text{if } i_1(k) = 1, \\ x_1(k) + u_1(k) - 7 & \text{if } i_1(k) = 2, \\ u_1(k) + 1 & \text{if } i_1(k) = 3 \end{cases}$$

and Σ_2 yields as output the 1-norm of its input which amounts to

$$\text{EG}_2: \quad \delta_3(k) = [u_2(k) \geq 0]$$

$$\text{MS}_2: \quad i_2(k) = \begin{cases} 1 & \text{if } \bar{\delta}_3(k), \\ 2 & \text{if } \delta_3(k) \end{cases}$$

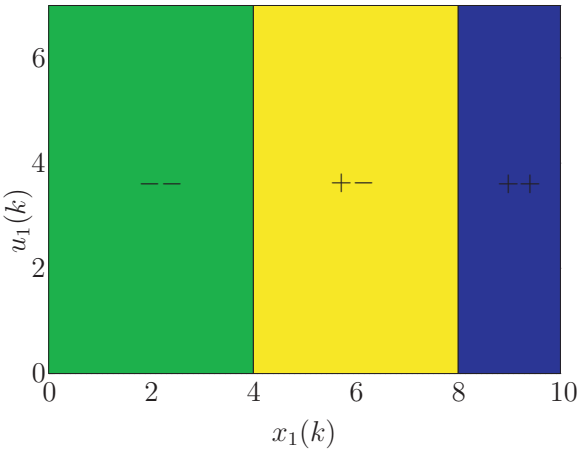
$$\text{SAS}_2: \quad y_2(k) = \begin{cases} -u_2(k) & \text{if } i_2(k) = 1, \\ u_2(k) & \text{if } i_2(k) = 2 \end{cases}$$

Clearly, the state-input space is given by $\mathcal{X} \times \mathcal{U}$, the corresponding digraph is acyclic and the indices of the DHAs are already ordered such that the adjacency matrix is upper triangular.

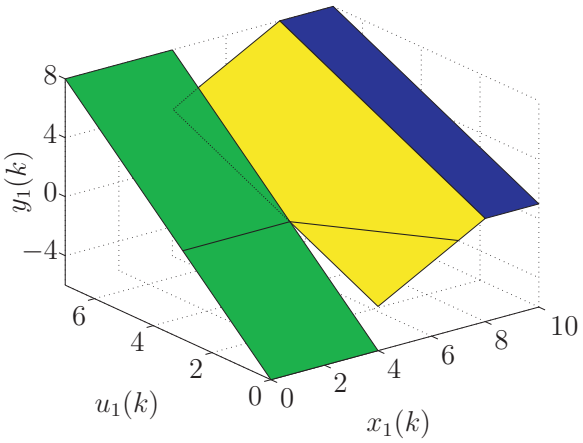
In a first step, the algorithm determines the hyperplane arrangement of Σ_1 which contains the hyperplanes of the EG_1, namely $\{\left[\begin{smallmatrix} x_1 \\ u_1 \end{smallmatrix}\right] \in \mathcal{X} \times \mathcal{U} \,|\, x_1 = 4\}$ and $\{\left[\begin{smallmatrix} x_1 \\ u_1 \end{smallmatrix}\right] \in \mathcal{X} \times \mathcal{U} \,|\, x_1 = 8\}$. This leads to the markings and the polyhedral partition $\{\mathcal{P}_j\}_{j \in \mathcal{J}_1}$, $\mathcal{J}_1 = \{1, 2, 3\}$, as shown in Fig. 3.4(a). The corresponding PWA output functions are depicted in Fig. 3.4(b).

In a second step, $\{\mathcal{P}_j\}_{j \in \mathcal{J}_1}$ is further partitioned by the hyperplane defined in the EG_2. Starting with mode $j = 1 \in \mathcal{J}_1$ this is done in the following way. The function Subst() replaces the expressions for $u_2(k)$ in the EG_2 and in the SAS_2 by $2u_1(k) - 6$. Thus, for this particular mode, the hyperplane arrangement of Σ_2 is defined within the polyhedron $\{\left[\begin{smallmatrix} x_1 \\ u_1 \end{smallmatrix}\right] \in \mathcal{X} \times \mathcal{U} \,|\, 0 \leq x_1 \leq 4\}$ and holds the single hyperplane $\{\left[\begin{smallmatrix} x_1 \\ u_1 \end{smallmatrix}\right] \in \mathcal{X} \times \mathcal{U} \,|\, u_1 = 3\}$. The corresponding markings are shown in Fig. 3.4(c). As both modes of the EG_2 are feasible, $\mathcal{J}_2$ contains two modes and the polyhedron $\{\left[\begin{smallmatrix} x_1 \\ u_1 \end{smallmatrix}\right] \in \mathcal{X} \times \mathcal{U} \,|\, 0 \leq x_1 \leq 4\}$ is partitioned into two. Accordingly, the mode $j = 2 \in \mathcal{J}_1$ leads to the hyperplane

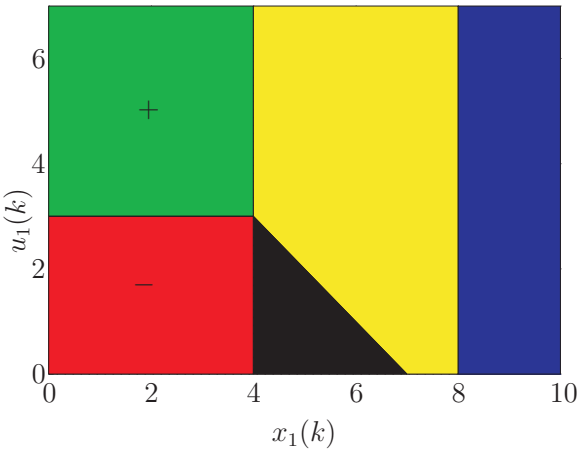
[4] As mentioned before, the state-update functions have no influence on the set of polyhedra and are thus omitted for brevity.



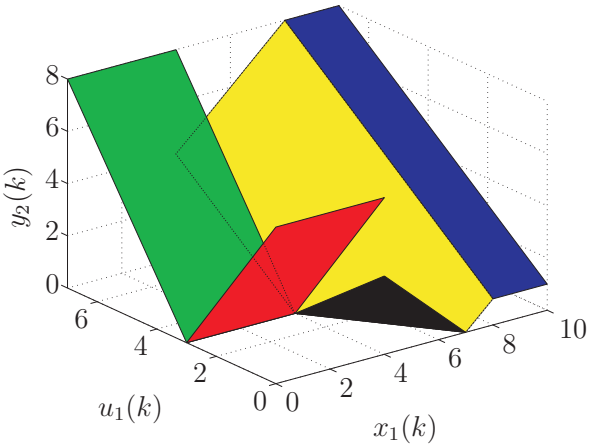
(a) Polyhedral partition $\{\mathcal{P}_j\}_{j\in\mathcal{J}_1}$



(b) PWA output functions given in $\{\mathcal{S}_j\}_{j\in\mathcal{J}_1}$. Bold lines indicate the intersections of the output functions with $y_1(k) = 0$



(c) Polyhedral partition $\{\mathcal{P}_j\}_{j\in\mathcal{J}_2}$



(d) PWA output functions given in $\{\mathcal{S}_j\}_{j\in\mathcal{J}_2}$

Figure 3.4: Polyhedral partitions and PWA output functions of the composition of DHAs of Example 3.2

arrangement $\{\left[\begin{smallmatrix} x_1 \\ u_1 \end{smallmatrix}\right] \in \mathcal{X} \times \mathcal{U} \,|\, x_1 + u_1 = 7\}$ and also to two modes, whereas the hyperplane arrangement corresponding to $j = 3 \in \mathcal{J}_1$ is empty and thus no additional mode is added.

The final polyhedral partition $\{\mathcal{P}_j\}_{j\in\mathcal{J}} = \{\mathcal{P}_j\}_{j\in\mathcal{J}_2}$ and the PWA dynamics $\{\mathcal{S}_j\}_{j\in\mathcal{J}} = \{\mathcal{S}_j\}_{j\in\mathcal{J}_2}$ are shown in Fig. 3.4(c) and 3.4(d), respectively.

Compositions With Loops

The algorithm is now generalized to compositions of DHAs containing loops. Having determined the adjacency matrix W and verified that the digraph is cyclic, we have to identify the connections whose removal breaks all loops and renders the corresponding digraph acyclic. These connections correspond to the feedback arc set.

Definition 3.3 *Let $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ be a digraph. A set $\mathscr{F} \subseteq \mathscr{E}$ is a* feedback arc set *(FAS) for $\mathscr{G}$, if $\mathscr{G}' = (\mathscr{V}, \mathscr{E} - \mathscr{F})$ is acyclic. The set $\mathscr{F}$ is a* minimum *FAS if the number of edges in $\mathscr{F}$ is minimum.*

Finding the minimum FAS is $\mathcal{NP}$-hard. However, our algorithm does not require the FAS to be minimal, and for a given digraph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ fast and effective heuristics exist [ELS93] with time complexity $O(\#\mathscr{E})$ which yield an FAS $\mathscr{F}$ with upper bounded cardinality $\#\mathscr{F} \leq \#\mathscr{E}/2 - \#\mathscr{V}/6$.

Removing the loops in the composition of DHAs is equivalent to replacing the connections corresponding to feedback arcs by newly created auxiliary inputs. In general, a feedback arc f corresponds to more than one connection between two DHAs and encompasses therefore real as well as binary variables. Thus, we add a real auxiliary input for every connection from a real output to a real input and accordingly a binary auxiliary input for every connection corresponding to binary variables. Repeating this for all $f \in \mathscr{F}$ yields a composition of DHAs without loops defined on the augmented exogenous input space $\mathcal{U}_r \times \mathcal{V}_r \times \mathcal{U}_b \times \mathcal{V}_b$, where $\mathcal{V}_r$ and $\mathcal{V}_b$ denote the auxiliary real and binary input spaces, respectively. We denote the vector of auxiliary real and binary inputs by $v_r(k) \in \mathcal{V}_r$ and $v_b(k) \in \mathcal{V}_b$, respectively. As connections between DHAs are equivalent to equality constraints of the respective inputs and outputs, the removed connections are kept as equality constraints $\mathcal{C}$.

Now assume again, that the indices of the DHAs are ordered such that the adjacency matrix is upper triangular. Given a binary state $x_b(k) \in \mathcal{X}_b$ and an augmented binary input $\left[\begin{smallmatrix} u_b(k) \\ v_b(k) \end{smallmatrix}\right] \in \mathcal{U}_b \times \mathcal{V}_b$, this assumption allows us to use Algorithm 3.2 to derive the set of feasible modes $\mathcal{J}'$, the set of polyhedra $\{\mathcal{P}'_j\}_{j \in \mathcal{J}'}$ and the corresponding PWA dynamics $\{\mathcal{S}'_j\}_{j \in \mathcal{J}'}$ defined on the augmented real state-input space $\mathcal{X}_r \times \mathcal{U}_r \times \mathcal{V}_r$. As the algorithm proceeds, the outputs of the DHAs are replaced step by step by affine combinations of states and exogenous inputs. Therefore, the expressions for the constraints $\mathcal{C}$ must be updated at the same time replacing outputs by affine combinations of states and exogenous inputs according to the respective PWA output function. This yields the set of constraints $\{\mathcal{C}'_j\}_{j \in \mathcal{J}'}$, where $\mathcal{C}'_j$ denotes the updated constraints corresponding to the mode $j \in \mathcal{J}'$.

In a last step, we cycle through all modes $\mathcal{J}'$ and impose the updated constraints in order to remove the auxiliary inputs. This yields the set of modes $\mathcal{J}$ of the original composition containing loops and the set of polyhedra $\{\mathcal{P}_j\}_{j \in \mathcal{J}}$ with the corresponding PWA dynamics $\{\mathcal{S}_j\}_{j \in \mathcal{J}}$ defined on the original space $\mathcal{X}_r \times \mathcal{U}_r$. In general, some of the modes in the set $\mathcal{J}'$ will prove to be infeasible and thus $\mathcal{J} \subseteq \mathcal{J}'$. Consider now the mode $j \in \mathcal{J}'$ with the associated polyhedron $\mathcal{P}'_j$ and the constraint $\mathcal{C}'_j$, which is of the form $H_x x_r(k) + [H_u \; H_v] \left[\begin{smallmatrix} u_r(k) \\ v_r(k) \end{smallmatrix}\right] = K$, where H_x, H_u and H_v are matrices with $\#\mathscr{F}$ rows and an appropriate number of columns, and K is a column vector with $\#\mathscr{F}$ components. The following three cases may occur:

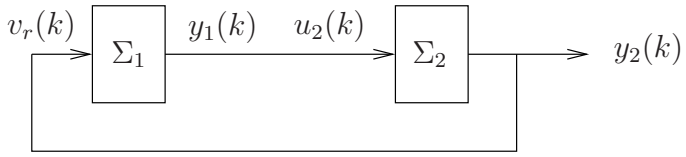


Figure 3.5: Composition of DHAs of Example 3.3 containing a feedback loop

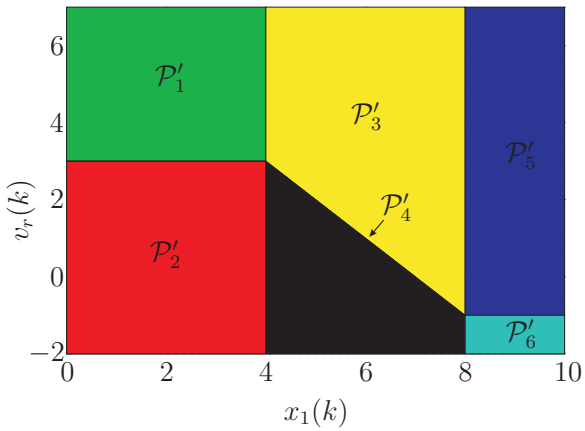
(i) If $\det(H_v) \neq 0$, we express the auxiliary input as a function of the real state and the real exogenous input, and we substitute $v_r(k)$ in $\mathcal{P}'_j$. This is the same as intersecting $\mathcal{P}'_j$ with the constraint $\mathcal{C}'_j$ and projecting the result on the original state-input space $\mathcal{X}_r \times \mathcal{U}_r$. The associated PWA dynamic $\mathcal{S}_j$ is derived by substituting $v_r(k)$ in $\mathcal{S}'_j$. If the polyhedron $\mathcal{P}_j$ is non-empty, the auxiliary input has been removed successfully and $\mathcal{P}_j$ is now solely defined on $\mathcal{X}_r \times \mathcal{U}_r$. Therefore, we add the mode j to $\mathcal{J}$, the polyhedron $\mathcal{P}_j$ to $\{\mathcal{P}_j\}_{j\in\mathcal{J}}$ and the PWA dynamic $\mathcal{S}_j$ to $\{\mathcal{S}_j\}_{j\in\mathcal{J}}$. If $\mathcal{P}_j$ is empty, the corresponding mode j is infeasible and thus discarded.

(ii) However, if $\det(H_v) = 0$, there are either zero or an infinite number of solutions for $v_r(k)$. The corresponding mode and polyhedron are infeasible and are removed.

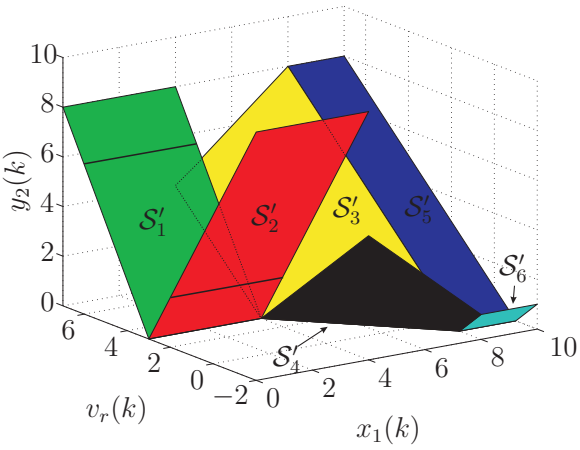
(iii) The third case results from the fact, that the constraints $\{\mathcal{C}'_j\}_{j\in\mathcal{J}'}$ are defined only on real states, real inputs and auxiliary real inputs. As we have seen, however, replacing the feedback arcs may lead to auxiliary binary inputs $v_b(k)$, too. These inputs are given together with the binary states and binary inputs when calling the algorithm. In general, some of these combinations will prove infeasible and the algorithm will discard the associated modes and polyhedra.

The first two cases, in which loops have either zero, one or an infinite number of solutions are well-known from linear systems theory with the only difference, that $\det(H_v)$ is in our case a local property that holds only for a given polyhedron and not for the whole state-input space. However, when dealing with loops in hybrid systems, additional difficulties may arise. As the next example will show, even if for all modes $\det(H_v) \neq 0$, the resulting polyhedra do not necessarily form a polyhedral partition of $\mathcal{X}_r \times \mathcal{U}_r$ and the composition of DHAs is thus not well-posed in general.

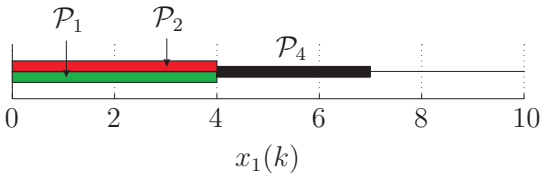
Example 3.3 Reconsider the composition of DHAs in Example 3.2 to which we add now a feedback loop as shown in Fig. 3.5. Thus, the input $u_1(k)$ is removed and the state-input space is reduced to $\mathcal{X} = [0, 10]$. The corresponding digraph is cyclic and contains the feedback arc $f = (\Sigma_2, \Sigma_1)$. Introducing the auxiliary real variable $v_r(k) \in \mathcal{V} = \mathbb{R}$ and adding $v_r(k) = y_2(k)$ to the constraint $\mathcal{C}$ allows us to break the loop. The composition is now defined on the augmented state-input space $\mathcal{X} \times \mathcal{V}$. Algorithm 3.2 leads to the set of



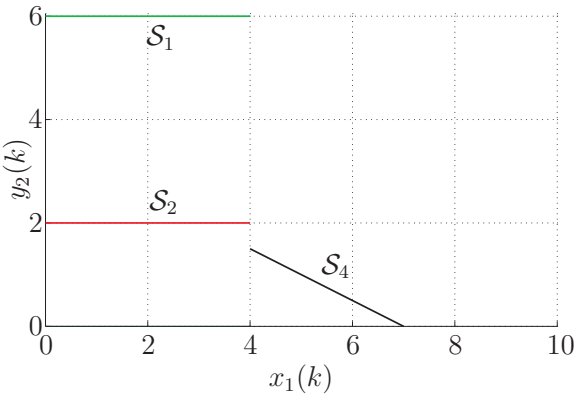
(a) Set of polyhedra $\{\mathcal{P}'_j\}_{j\in\mathcal{J}'}$ defined on the augmented state-input space $\mathcal{X}\times\mathcal{V}$



(b) PWA output functions given in $\{\mathcal{S}'_j\}_{j\in\mathcal{J}'}$ and defined on the augmented state-input space $\mathcal{X}\times\mathcal{V}$. Bold lines indicate the intersections of the output functions with $y_2(k)=v_r(k)$



(c) Set of polyhedra $\{\mathcal{P}_j\}_{j\in\mathcal{J}}$ defined on the original state-input space $\mathcal{X}$



(d) PWA output functions given in $\{\mathcal{S}_j\}_{j\in\mathcal{J}}$ and defined on the original state-input space $\mathcal{X}$

Figure 3.6: Set of polyhedra and PWA output functions of the composition of DHAs of Example 3.3

polyhedra $\{\mathcal{P}'_j\}_{j\in\mathcal{J}'}$ and the PWA dynamics $\{\mathcal{S}'_j\}_{j\in\mathcal{J}'}$ shown in Fig. 3.6(a) and Fig. 3.6(b), respectively[5].

Consider the mode $j=1\in\mathcal{J}'$ with the polyhedron $\mathcal{P}'_1=\{\left[\begin{smallmatrix}x_1\\v_r\end{smallmatrix}\right]\in\mathcal{X}\times\mathcal{V}\,|\,\left[\begin{smallmatrix}1&0\\0&-1\end{smallmatrix}\right]\left[\begin{smallmatrix}x_1\\v_r\end{smallmatrix}\right]\leq\left[\begin{smallmatrix}4\\-3\end{smallmatrix}\right]\}$, the output function $y_2(k)=2v_r(k)-6$ and the updated constraint $v_r(k)=6$. The fact, that $\det(H_v)$ is different from zero allows us to derive $\mathcal{P}_1=\{x_1\in\mathcal{X}\,|\,x_1\leq 4\}$. As $\mathcal{P}_1$ is not empty, the mode $j=1$ is feasible. The corresponding output function is given by $y_2(k)=6$. The modes $j=2$ and $j=4$ are handled in a similar way. For the modes $j=3$ and $j=5$, the updated constraints are $x_1(k)=7$ and $0=1$, respectively. In both cases, $\det(H_v)$ is zero. For $j=3$, this leads to an infinity number of solutions and a lower

[5]Note that the v_r-axis has been artificially restricted to $-2\leq v_r(k)\leq 7$ in order to facilitate the plotting.

dimensional polyhedron containing only the single point $x_1(k) = 7$, whereas for $j = 5$ no solution exists. Both modes together with the associated polyhedra and dynamics are therefore removed.

The resulting set of polyhedra $\{\mathcal{P}_j\}_{j\in\mathcal{J}}$, $\mathcal{J} = \{1, 2, 4\}$, and the PWA output functions $\{\mathcal{S}_j\}_{j\in\mathcal{J}}$, which are both defined on the one-dimensional state-input space $\mathcal{X}$, are shown in Fig. 3.6(c) and 3.6(d), respectively. Note that $\mathcal{P}_1 = \mathcal{P}_2$, but that two different output functions are associated to them. Because of that and as a part of the state-input space, namely $\{x_1 \in \mathcal{X} \,|\, x_1 \geq 7\}$, is not covered, $\{\mathcal{P}_j\}_{j\in\mathcal{J}}$ does not form a polyhedral partition. Therefore, the resulting PWA system and consequently also the corresponding composition of DHAs is not well-posed.

The example demonstrates that if loops are present in a composition of DHAs the resulting polyhedra do not necessarily form a polyhedral partition. The reason for this is twofold. First, modes might be infeasible either because the intersection of the associated polyhedron with the updated constraint is empty or because $\det(H_v)$ of the constraint is zero. In general, infeasible modes result in gaps in the state-input space. Second, polyhedra may overlap.

Summing up, using well-posed DHAs to form a composition of DHAs does not guaranty well-posedness of the overall composition in the presence of loops. However, as well-posedness of the composition of DHAs relates directly to well-posedness of the corresponding PWA model, we can conclude, that the composition of DHAs is well-posed if and only if the corresponding PWA system is well-posed. Based on the polyhedra and the dynamics, we can easily evaluate well-posedness of the PWA model. If the set of polyhedra $\{\mathcal{P}_j\}_{j\in\mathcal{J}}$ of the PWA model forms a polyhedral partition, well-posedness is assured by Lemma 2.2. On the other hand, if the union of $\{\mathcal{P}_j\}_{j\in\mathcal{J}}$ covers the state-input space completely, and if all pairs of overlapping polyhedra are associated to the same PWA dynamic, the PWA model is well-posed, too.

3.5 Examples and Applications

This final section presents two examples showing how the mode enumeration algorithm can be used to efficiently derive the PWA representation of a given hybrid system.

3.5.1 Car Example

In [TB01], the authors proposed a hybrid model of a car with a robotized gear shift. This example was adopted in [Bem02] where the author computes the MLD model using HYSDEL and the PWA system equivalent to the MLD model using multi-parametric and mixed integer linear programming. As the model is given in HYSDEL, the algorithm in

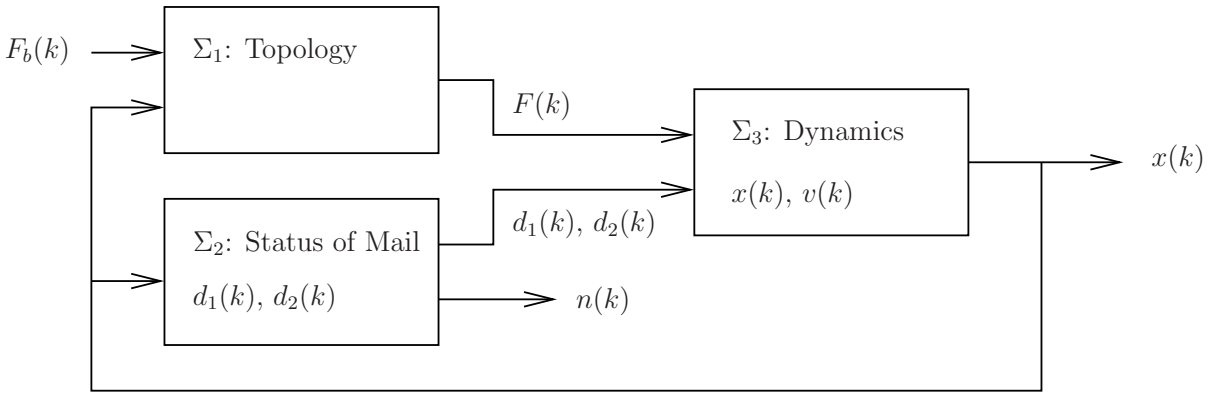


Figure 3.7: Paperboy example consisting of three DHAs with the respective states

Section 3.4 starts from this description to translate the car example into a PWA model. The resulting PWA model encompasses 30 polyhedra and six modes and is computed in 1.9 s using MATLAB 5.3 on a Pentium III 650 MHz machine. This is 40 times faster than the algorithm reported in [Bem02] on a similar machine.

The reason for this is twofold. First, the algorithm presented here exploits the structure of the DHA models, while the algorithm presented in [Bem02] deals with MLD models concealing that structural information. Second, the approach in [Bem02] needs to remove redundant inequalities at each iteration of the exploration algorithm. This operation may dominate the total computation time in [Bem02].

Apart from this, [Bem02] lacks the compositional capability and can only handle the transformation from a single MLD to a single PWA model. In particular, models with loops cannot be tackled.

3.5.2 Paperboy Example

A paperboy delivers by bike two heavy and bulky mail items to two different houses within a neighborhood consisting of four properties and one road. The properties and the road have different slopes and different friction coefficients.

The input of the system at time-instant k is given by the force $F_b(k) \in \mathcal{U} \subset \mathbb{R}^2$, $\mathcal{U} = [-F_{\max}, F_{\max}]^2$, $F_{\max} = 162\,\mathrm{N}$ that the paperboy applies to his bike in order to accelerate and brake. Driven by F_b, the paperboy cycles in the two-dimensional neighborhood $\mathcal{X}_1 = [-s_n, s_n]^2$ with $s_n = 1000\,\mathrm{m}$. His position is given by $x(k) = [x_1(k), x_2(k)]^T \in \mathcal{X}_1 \subset \mathbb{R}^2$ and his speed $v(k) \in \mathcal{X}_2 \subset \mathbb{R}^2$ is limited by $\mathcal{X}_2 = [-v_{\max}, v_{\max}]^2$, where $v_{\max} = 15\,\mathrm{m/s}$. Two binary states $d_1(k), d_2(k) \in \{0, 1\}$ denote the status of the mail delivery. The outputs of the model are the position $x(k)$ and the number of delivered mail items $n(k) \in \{0, 1, 2\}$.

As depicted in Fig. 3.7, the paperboy problem can be decomposed into three DHAs. Each DHA is described in the following.

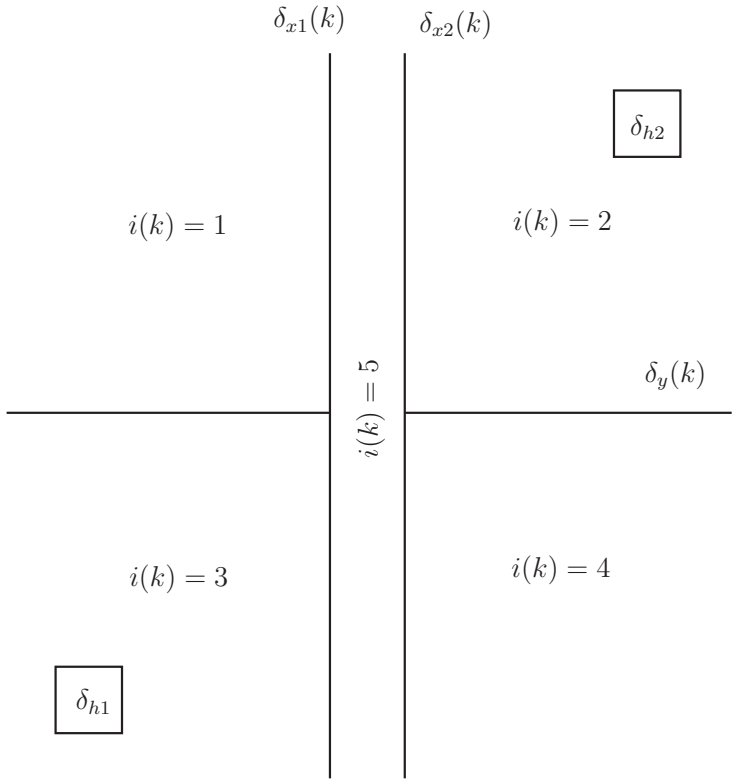


Figure 3.8: Topology of the neighborhood in the paperboy example with the thresholds and modes

Topology of Neighborhood. A road of width $w_r = 4\,\mathrm{m}$ divides the neighborhood into two parts, which are further partitioned into two properties yielding a total of four properties and one road (see Fig. 3.8). These five regions are each characterized by different slopes and friction coefficients. Given the force $F_b(k) \in \mathcal{U} \subset \mathbb{R}^2$ that the paperboy applies, the effective force acting on the bike

$$F(k) = F_b(k) - \mu_{i(k)} - \nu_{i(k)} v(k) \in \mathbb{R}^2$$

depends on the partition $i(k) \in \{1, \ldots, 5\}$, the grade resistance $\mu_{i(k)}$ corresponding to the slope, and the friction coefficient $\nu_{i(k)}$. The parameters are given by $\mu_1 = \left[\begin{smallmatrix} 2 \\ 0 \end{smallmatrix}\right]$, $\mu_2 = \left[\begin{smallmatrix} 1.5 \\ 0 \end{smallmatrix}\right]$, $\mu_3 = \left[\begin{smallmatrix} -0.5 \\ 0 \end{smallmatrix}\right]$, $\mu_4 = \left[\begin{smallmatrix} -1 \\ 0 \end{smallmatrix}\right]$, $\mu_5 = \left[\begin{smallmatrix} 0 \\ 0 \end{smallmatrix}\right]$ and $\nu_1 = \left[\begin{smallmatrix} 2 & 0 \\ 0 & 2 \end{smallmatrix}\right]$, $\nu_2 = \left[\begin{smallmatrix} 0.5 & 0 \\ 0 & 0.5 \end{smallmatrix}\right]$, $\nu_3 = \left[\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right]$, $\nu_4 = \left[\begin{smallmatrix} 1.5 & 0 \\ 0 & 1.5 \end{smallmatrix}\right]$, $\nu_5 = \left[\begin{smallmatrix} 0.05 & 0 \\ 0 & 0.05 \end{smallmatrix}\right]$.

Therefore, the first DHA is static and has the real inputs $x(k)$, $v(k)$ and $F_b(k)$. The real output is $F(k)$.

$$\mathrm{EG}_1\colon \begin{cases} \delta_{x1}(k) &= [x_1(k) \leq -0.5 w_r], \\ \delta_{x2}(k) &= [x_1(k) \geq 0.5 w_r], \\ \delta_y(k) &= [x_2(k) \geq 0] \end{cases}$$

$$\text{MS}_1: \quad i(k) = \begin{cases} 1 & \text{if } \delta_{x1}(k) \ \wedge \ \delta_y(k), \\ 2 & \text{if } \delta_{x2}(k) \ \wedge \ \delta_y(k), \\ 3 & \text{if } \delta_{x1}(k) \ \wedge \ \bar{\delta}_y(k), \\ 4 & \text{if } \delta_{x2}(k) \ \wedge \ \bar{\delta}_y(k), \\ 5 & \text{if } \bar{\delta}_{x1}(k) \ \wedge \ \bar{\delta}_{x2}(k) \end{cases}$$

$$\text{SAS}_1: \quad F(k) = F_b(k) - \mu_{i(k)} - \nu_{i(k)} v(k), \quad i(k) = 1, \ldots, 5$$

Status of Mail Delivery. The houses are squares of size $s_h = 10\,\text{m}$ centered at $x_{h1} = [-p_h, -p_h]$ and $x_{h2} = [p_h, p_h]$, $p_h = 40\,\text{m}$ as depicted in Fig. 3.8. The four walls of a house can be modelled by four hyperplanes with corresponding binary variables equal to 1, if the paperboy is "within" the respective wall. The respective flag $\delta_{hi}(k)$, $i = 1, 2$ denoting that the paperboy has reached House i and delivered the mail, is the logic *and* of these four binary variables. Finally, a FSM stores the mail delivery status using the binary states $d_1(k)$, $d_2(k)$.

This leads to the following DHA with the real input $x(k)$, the binary states $d_1(k)$ and $d_2(k)$, which are also outputs, and a third output $n(k) = d_1(k) + d_2(k)$ denoting the number of delivered mail items.

$$\text{EG}_2: \quad \begin{cases} \delta_{h1}(k) = ||x(k) - x_{h1}||_\infty \leq 0.5 s_h, \\ \delta_{h2}(k) = ||x(k) - x_{h2}||_\infty \leq 0.5 s_h \end{cases}$$

$$\text{FSM}_2: \quad \begin{cases} d_1(k+1) = d_1(k) \ \vee \ \delta_{h1}(k), \\ d_2(k+1) = d_2(k) \ \vee \ \delta_{h2}(k) \end{cases}$$

Dynamics of Paperboy. The total weight $m(k)$ is the weight of the paperboy and his bike ($M_b = 90\,\text{kg}$) plus the weight of the undelivered mail items (each mail item weighs $M_m = 10\,\text{kg}$). Therefore, the total weight is time-dependent and decreasing as the paperboy delivers the mail. By Newton's law, the effective force $F(k)$ divided by the total weight is the acceleration and by integrating this, the velocity and the position of the paperboy are obtained. The integral is approximated by two discrete-time dynamical systems with sampling time $T_s = 1\,\text{s}$.

The third DHA has the real input $F(k) \in \mathbb{R}^2$, the two binary inputs $d_1(k)$, $d_2(k)$ denoting the status of the mail delivery and the real states $x(k)$ and $v(k)$ characterizing the position and the velocity, respectively. The outputs are the position $x(k)$ and the velocity $v(k)$.

$$\text{MS}_3: \quad i(k) = \begin{cases} 1 & \text{if } \bar{d}_1(k) \wedge \bar{d}_2(k), \\ 2 & \text{if } (d_1(k) \wedge \bar{d}_2(k)) \vee (\bar{d}_1(k) \wedge d_2(k)), \\ 3 & \text{if } d_1(k) \wedge d_2(k) \end{cases}$$

$$\text{SAS}_3\text{:} \quad \begin{bmatrix} v(k+1) \\ x(k+1) \end{bmatrix} = \begin{bmatrix} v(k) \\ x(k) \end{bmatrix} + \begin{bmatrix} F(k)/m(k) \\ v(k) \end{bmatrix} T_s,$$

where

$$m(k) = \begin{cases} M_b + 2M_m & \text{if } i(k) = 1, \\ M_b + M_m & \text{if } i(k) = 2, \\ M_b & \text{if } i(k) = 3 \end{cases}$$

Summing up, the paperboy example is defined on the eight-dimensional state-input space $\mathcal{X}_1 \times \mathcal{X}_2 \times \{0,1\}^2 \times \mathcal{U}$. Furthermore, it contains no loops. Algorithm 3.2 yields the equivalent PWA model within 6.5 s on a 2.8 GHz Pentium IV PC. It encompasses 168 feasible modes and polyhedra, what is by far below the upper bound of 7099 given by Buck's formula (3.2).

The paperboy starts the mail delivery at a random position $x(0)$ with speed $v(0) = 0$. His objective is to first deliver one mail item to House 1 centered around x_{h1} and then to move on to House 2 at position x_{h2} to deliver the second mail item. Using the force that the paperboy applies as manipulated variable, namely $u(k) = F_b(k)$, this control objective can be expressed by the cost function

$$J(x(k), v(k), U(k)) = \sum_{\ell=0}^{N-1} \|x(k+\ell|k) - x_{ref}(k+\ell|k)\|_1 + \epsilon \|u(k+\ell|k)\|_1 \,, \tag{3.4}$$

which penalizes the predicted deviation of the position from its reference over the horizon N using the 1-norm for the sequence of manipulated variables $U(k) = [(u(k))^T, \ldots, (u(k+N-1))^T]^T$. The reference x_{ref} is switched from x_{h1} to x_{h2} when the paperboy reaches House 1. Additionally, a very small penalty term $\epsilon = 10^{-6}$ is imposed on the manipulated variable.

In the next section, we will use the paperboy example to evaluate the potential of the mode enumeration algorithm to reduce the computation time of MPC. The MPC control problem amounts to minimizing the cost function (3.4) subject to the evolution of the paperboy model over the prediction horizon and subject to constraints on F_b, x and v as given above. The solution of this optimization problem, which is a *Mixed-Integer Linear Program* (MILP), yields the force F_b.

3.6 Cuts for Model Predictive Control

MPC of hybrid systems [BM99a] uses an internal hybrid model, which is usually in MLD form (2.8). In the great majority of cases, the MLD model is derived starting from a composition of DHAs described textually in HYSDEL. When translating a composition of DHAs into an MLD model, information about the structure of the hybrid model is lost. However, the explicit computation of the set of feasible modes of the composition

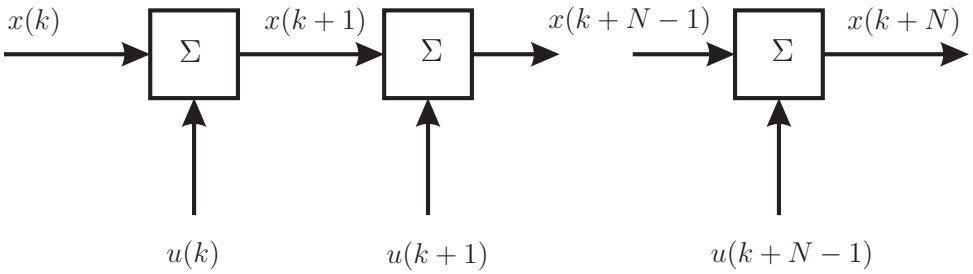


Figure 3.9: Conceptual scheme of the N-step prediction model

of DHAs allows one to add this structural information to the MLD model in the form of cuts. These cuts are inequality constraints on the binary inputs, binary states and binary variables δ in (2.8), allowing one to prune infeasible combinations of these binary variables, or equivalently modes, from the MLD model.

Given the current state and the input, the internal (MLD) model computes the state at the next time-instant and the output. One might refer to such a model as a single-step prediction model defined on the state-input space $\mathcal{X} \times \mathcal{U}$. When building the optimization problem for MPC with horizon N, this model is repeated N times. More specifically, the series connection of N identical single-step models is built as shown in Fig. 3.9, where each model uses the state predicted by the previous one as initial state. We might consider these N models as one single model defined on the state-input space $\mathcal{X} \times \mathcal{U}^N$. Given the initial state $x(k) \in \mathcal{X}$ and the sequence of inputs $U(k) = [(u(k))^T, \ldots, (u(k+N-1))^T]^T \in \mathcal{U}^N$, this models provides the state evolution over N. We thus refer to it as the N-step prediction model. The mode enumeration allows us to introduce cuts not only on the modes of the single-step, but also on the N-step prediction model. As a result, additional cuts on the $\mathcal{X} \times \mathcal{U}^N$ space can be added taking into account the interaction between the single-step models.

Cuts can be formulated in terms of additional logic constraints. According to [Mig02], two methods can be used to transform logic constraints into mixed integer inequalities which can be added to the MLD model. The *Symbolical Method* converts the constraints into a canonical normal form, which is then translated into integer inequalities, whereas the *Geometrical Method* computes the convex hull of the integer points for which the constraints are fulfilled. In general, the second method is superior to the first one because the convex hull is the smallest set containing all the integer feasible points, and because it introduces less additional inequalities.

Example 3.4 As an example of adding cuts to the single-step prediction model, reconsider the paperboy example of the previous section. We used the compiler HYSDEL to transform the paperboy composition of DHAs into an MLD model. The mode enumeration algorithm in Section 3.4 computed the set of feasible modes, which were added

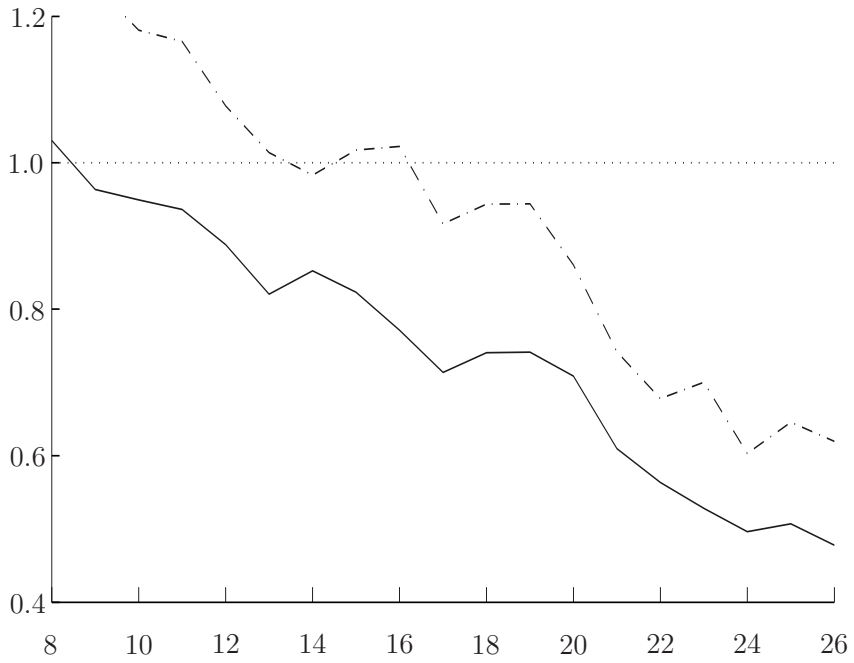


Figure 3.10: Normalized average computation time versus prediction horizon N, $N = 8, \ldots, 26$, when using single-step prediction models with cuts generated by either the symbolical method (dash-dotted line) or the geometrical method (solid line)

as additional constraints to the MLD model using the symbolical method as well as the geometrical method. We solved MPC with the cost function (3.4) and various prediction horizons. Using CPLEX [ILO02] as MILP solver, Fig. 3.10 reports the average computation times for MPC on the two improved models normalized to the plain model produced by the HYSDEL compiler.

Note that both methods add non-trivial cuts to the model reducing the computation time of CPLEX up to a factor of two. This improvement is more evident when using less advanced solvers like [BM00], where for a prediction horizon of three for example, the additional information reduces the computation time by a factor of 210. Figure 3.10 shows clearly, that the cuts introduced by the geometrical method are more effective than the ones of the symbolical method. This is mainly due to the fact that the symbolical method needs much more constraints and consequently memory space to define the feasible modes thus delaying the calling of CPLEX. For the paperboy example, the symbolical method introduces 239 additional constraints, whereas the geometrical method only adds 42. The third conclusion is, that both methods become more effective as the prediction horizon is increased, as the benefit of additional cuts grows with the number of binary variables.

Adding cuts to the N-step prediction model has not been investigated numerically. In

general, we expect that additional cuts can be generated that remove combinations of modes of the different single-step models. Obviously, the reduction of computation time when solving the optimization problem will be problem specific and hard to generalize. Nevertheless, in particular for long prediction horizons, an additional improvement is to be expected for most examples.

Alternatively, one might first transform the HYSDEL model into PWA form, and describe the PWA model in MLD form by introducing for each mode of the PWA model a binary variable that is one if the mode is active. An *exclusive or* (XOR) constraint assures that exactly one binary variable is one. For many MILP solvers like CPLEX, the structural information of the XOR constraint can be passed directly to the solver in the form of a *special ordered set* (SOS) thus leading to a performance improvement that is expected to be significant.

3.7 Conclusions and Future Research

We have presented an effective method to enumerate the set of feasible modes for a given composition of DHAs. The same procedure transforms the compound model into a PWA model. The algorithm is capable of handling feedback loops in such a composition – in contrast to HYSDEL that is not able to transform compositions containing loops into equivalent MLD models. As a byproduct the algorithm can also determine whether a composition is well-posed or not. Improving the data handling and storing already computed hyperplane arrangements led to an reduction of the computation time by a factor of up to 30 compared with the original code presented in [GTM03a].

In general, some neighboring polyhedra will have the same PWA dynamics and should be joined in order to reduce the model complexity. This topic is treated in the next chapter, where we propose optimal complexity reduction algorithms to derive – based on the hyperplane arrangement – an equivalent PWA model that is minimal in the number of polyhedra.

With respect to optimal control, the mode enumeration can be exploited to reduce the computation time of MPC by adding cuts. Here, we have added cuts only to the single-step prediction model. Extending this idea to the N-step prediction model should further increase the benefits in terms of reduction of the on-line computation time. Another promising alternative is to transform the N-step prediction model into an equivalent PWA model, to derive a minimal representation using the optimal complexity reduction of Chapter 4, and to translate it then into a very compact MLD model to be used for MPC.

4

Optimal Complexity Reduction of Piecewise Affine Systems

4.1 Introduction

In the preceding chapter, we have formulated an algorithm that efficiently enumerates the feasible modes of a composition of DHAs and transforms the compound model into an equivalent PWA model. As every mode relates to a polyhedron with an associated affine dynamic, the PWA model is given by a set of polyhedra with associated affine dynamics. Often, different modes correspond to the same dynamic. If some or all of the associated polyhedra form a convex union, those polyhedra should be merged in order to reduce the complexity of the model.

The motivation for deriving a PWA model minimal in the number of polyhedra is twofold. When pre-computing the optimal control law off-line to derive the state-feedback controller as shown in Section 2.4.2, an internal PWA model is required. Due to the combinatorial nature of the problem, both the computation time and controller complexity are in the worst case exponential in the number of polyhedra of the PWA model [Bor03]. Hence it is of upmost importance to derive a model with as few polyhedra as possible.

On the other hand, 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. When using a binary search tree as proposed in [TJB03], the computational burden can be reduced on the expense of enlarging the memory requirement. More precisely, the computation time becomes sublinear in the number of polyhedra, while the memory requirement gets superlinear.

Therefore, this chapter focuses on 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.

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) [BFT01] 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.

The mode enumeration algorithm not only derives the polyhedral partition, but also the corresponding set of markings of the associated hyperplane arrangement. Using the markings enables us to determine *a priori* – i.e. without solving any LP – if a given combination of polyhedra is convex. Exploiting this fact, we propose in this chapter two algorithms that yield the minimal number of polyhedra without solving any LP. The first algorithm executes a branch and bound on the markings yielding a set of non-overlapping (disjoint) merged polyhedra, using additional heuristics on the branching strategy to reduce the computation time. The second approach relies on the fact that the optimal complexity reduction problem can be reformulated as logic minimization problem by replacing the markings by Boolean variables and minterms. Logic minimization is a fundamental problem in digital circuit, and efficient tools have been developed to successfully tackle these problems, which often encounter hundreds or thousands of variables. The resulting polyhedra, however are in general not disjoint and thus overlapping. For most algorithms subsequently used, this has no negative effects.

As both algorithms refrain from solving additional LPs, they are not only optimal but also computational feasible. The applicability of the algorithms can be extended to general PWA systems lacking the hyperplane arrangement (like PWA state-feedback control laws) by first computing the hyperplane arrangement. Examples illustrate the algorithms and show their computational effectiveness.

This chapter is organized as follows. Section 4.2 recalls the PWA systems we are considering, namely PWA models and PWA state-feedback control laws, states formally the (disjoint and non-disjoint) optimal complexity reduction problems, and derives a key lemma to evaluate convexity of polyhedra using only their markings. Algorithms for optimal complexity reduction based on branch and bound, and logic minimization, are proposed in Sections 4.3 and 4.4, respectively. Section 4.5 extends the applicability of the algorithms by deriving the (global) hyperplane arrangement and elaborates on the notion of optimality. In Section 4.6, we propose two techniques to reduce the computational complexity of large problems, specifically the simplification of the hyperplane arrangement and *Divide and Conquer* strategies. The effectiveness of the approaches is demonstrated in Section 4.7 through three examples. Section 4.8 summarizes this chapter and proposes an efficient technique to implement PWA control laws as future research topic.

The optimal complexity reduction algorithms have been implemented in MATLAB and are included in the multi-parametric toolbox (MPT) [KGBM04], which is freely available from http://control.ee.ethz.ch/~mpt/.

Throughout the rest of the chapter, we will often abbreviate *optimal complexity reduction* with OCR.

4.2 Problem Statement and Properties

4.2.1 Target Systems

For the OCR, we consider two forms of polyhedral PWA systems, namely *PWA models* and *PWA state-feedback control laws*. Both are assumed to be polyhedral PWA systems and well-posed according to Section 2.2.3. Yet, the polyhedra are not required to form a polyhedral partition.

PWA Models

PWA models are of the form (2.9), which we repeat here for completeness:

$$x(k+1) = A_{j(k)}x(k) + B_{j(k)}u(k) + f_{j(k)} \tag{4.1a}$$

$$y(k) = C_{j(k)}x(k) + D_{j(k)}u(k) + g_{j(k)} \tag{4.1b}$$

$$\text{with } j(k) \text{ such that } \begin{bmatrix} x(k) \\ u(k) \end{bmatrix} \in \mathcal{P}_{j(k)}\,, \tag{4.1c}$$

where $x \in \mathcal{X}$, $u \in \mathcal{U}$, $y \in \mathcal{Y}$ denote at time k the (real and binary) states, inputs and outputs, respectively, and the polyhedra $\mathcal{P}_{j(k)}$ define a set of polyhedra $\{\mathcal{P}_j\}_{j\in\mathcal{J}}$ on the state-input space $\mathcal{X} \times \mathcal{U}$.

In most cases, PWA models have been modelled in HYSDEL as a composition of DHAs, and the mode enumeration algorithm has been used to transform the composition into PWA form. The algorithm also yields the hyperplane arrangement and the set of markings $M(\mathcal{R})$, which are both defined on $\mathcal{R}$, with $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{U}$. These markings form the basis for the OCR in this chapter. Next, we define global and local hyperplane arrangements.

If all the thresholds of the DHAs are defined only on states and inputs, the hyperplane arrangement and the markings are defined obviously on the state-input space, too, and consequently $\mathcal{R} = \mathcal{X} \times \mathcal{U}$. We refer to such a hyperplane arrangement as a *global* (or globally valid) hyperplane arrangement. Furthermore, because of the equivalence of DHAs and PWA models stated in Section 3.3, the modes $j \in \mathcal{J}$ are formally equivalent to the markings $m \in M(\mathcal{R})$.

If, however, some of the thresholds also depend on auxiliary variables that on their part depend on other thresholds, the mode enumeration algorithm yields a *collection*

of hyperplane arrangements that are sequentially defined within each other. Hence, $\mathcal{R}$ is not the whole state-input space but rather a polytopic subset of it. We say that the hyperplane arrangement is *local* (or locally valid). This is generally the case for compositions of DHAs that sequentially depend on each other, where each DHA defines a hyperplane arrangement within a cell of the hyperplane arrangement of the preceding DHA. Consequently, the (global) modes of the PWA model do not correspond to the markings of the (local) hyperplane arrangements.

PWA State-feedback Control Laws

PWA state-feedback control laws are of the form

$$u(k) = F_{j(k)}x(k) + g_{j(k)} \tag{4.2a}$$

$$\text{with } j(k) \text{ such that } x(k) \in \mathcal{P}_{j(k)}, \tag{4.2b}$$

where $x \in \mathcal{X}$ and $u \in \mathcal{U}$ denote at time k the (real and binary) states and inputs, respectively, and the polyhedra $\mathcal{P}_{j(k)}$ define a set of polyhedra $\{\mathcal{P}_j\}_{j\in\mathcal{J}}$ on the state-space $\mathcal{X}$.

In our context, PWA state-feedback control laws are obtained by pre-solving an optimal control law off-line including the computation of the feedback law of MPC controllers for linear or hybrid systems. Apart from the case of hybrid systems with quadratic performance indices in the cost function, the resulting feedback laws are again PWA expressions defined on polyhedral partitions as recalled in Section 2.4.2. Yet, the polyhedra are not defined in an hyperplane arrangement and markings are not available. Given a set of polyhedra, we will present in Section 4.5 an algorithm that derives a hyperplane arrangement and the corresponding markings, thus allowing us to extend the OCR algorithms to PWA state-feedback control laws. For completeness, we set $\mathcal{R} = \mathcal{X}$.

4.2.2 Problem Statement

In the following, we assume that besides the PWA representation a corresponding global hyperplane arrangement $\mathcal{A}$ is available together with the markings $M(\mathcal{R})$. Specifically, we assume the following[1].

Assumption 4.1 *The polyhedra of the given PWA system are cells in a (global) hyperplane arrangement, of which the markings are available.*

For a given PWA representation the aim of the OCR algorithms is to derive an equivalent representation that is minimal in the number of polyhedra by replacing polyhedra with the

[1] In Section 4.3, we will relax this assumption and extend the OCR algorithms to general PWA system not defined in hyperplane arrangements.

same affine dynamics (or feedback law) by new sets of polyhedra of minimal cardinality. For clarity of exposition, we associate with each affine dynamic (or feedback law) a different color, and we collect the polyhedra with the same color. Then, for a given color, we pose the following three problems, where we distinguish between results formed by disjoint and non-disjoint polyhedra.

Problem 4.1 (Disjoint Optimal Complexity Reduction (DOCR)) Given an initial set of polyhedra $\{\mathcal{P}_i\}_{i=1,\dots,p}$ with the same color satisfying Assumption 4.1, the *disjoint* optimal complexity reduction problem amounts to derive a new set of polyhedra $\{\mathcal{Q}_i\}_{i=1,\dots,q}$ 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 $\{\mathcal{Q}_i\}_{i=1,\dots,q}$ with a smaller number of polyhedra, (iii) the new polyhedra are mutually disjoint, i.e. $\mathcal{Q}_i \neq \mathcal{Q}_j$ for all $i, j \in \{1, \dots, q\}$, $i \neq j$, and (iv) the new polyhedra are formed as unions of the old ones, i.e. for each $\mathcal{Q}_j, j \in \{1, \dots, q\}$, there exists an index set $\mathcal{I} \subseteq \{1, \dots, p\}$, 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 the Requirements (iii) and (iv) thus allowing for overlaps in the resulting polyhedra.

Problem 4.2 (Non-Disjoint Optimal Complexity Reduction (NOCR)) Given an initial set of polyhedra $\{\mathcal{P}_i\}_{i=1,\dots,p}$ with the same color satisfying Assumption 4.1, the *non-disjoint* optimal complexity reduction problem amounts to derive a new set of polyhedra $\{\mathcal{Q}_i\}_{i=1,\dots,q}$ with the Properties (i) and (ii) as in Problem 4.1.

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

Next, we drop the assumption that the original polyhedra are cells of an hyperplane arrangement and that markings are available, 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 (polyhedral) PWA systems.

Problem 4.3 (General Non-Disjoint Optimal Complexity Reduction (GNOCR)) Given an initial set of polyhedra $\{\mathcal{P}_i\}_{i=1,\dots,p}$ with the same color, where Assumption 4.1 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 the Properties (i) and (ii) as in Problem 4.1, and (iii) the number of facets for each $\mathcal{Q}_i$ being minimal.

All three tasks are non-trivial, as the union of polyhedra with the same color 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 [Cha84] and references therein). As a direct consequence, fast algorithms are unlikely to exist leaving us either with rather long computation times or suboptimal solutions. Thus, the challenge is to design algorithms that are applicable to problems of meaningful size and nevertheless yield the global optimum.

4.2.3 Convexity of Unions of Polyhedra

Definition 4.1 (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 $\mathcal{P}_2$.*

The proof of the following lemma follows directly from the definition of the markings.

Lemma 4.1 (Separating Hyperplane) *Given the hyperplane arrangement $\{H_i\}_{i=1,\ldots,n}$ 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, H_j is a separating hyperplane for $\mathcal{P}_1$ and $\mathcal{P}_2$.*

Definition 4.2 (Envelope, [BFT01] p. 144) *Given two polyhedra $\mathcal{P}_1$ and $\mathcal{P}_2$, the* envelope $\text{env}(\mathcal{P}_1, \mathcal{P}_2)$ *of the two polyhedra is defined as the intersection of half spaces that contain both polyhedra, where the half spaces are given by the facets of the two polyhedra.*

Lemma 4.2 (Envelope) *Given the hyperplane arrangement $\{H_i\}_{i=1,\ldots,n}$ 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 $\mathcal{I}' = \{1, \ldots, n\} \setminus \mathcal{I}$, 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 4.1, the proof follows from the definition of the envelope. □

The proof can be easily generalized to envelopes of more than two polyhedra.

Theorem 4.1 (Convexity, [BFT01] Theorem 3) *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 $\mathcal{P}_1 \cup \mathcal{P}_2 = env(\mathcal{P}_1, \mathcal{P}_2)$.*

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.

Lemma 4.3 (Convexity) *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 4.1 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 4.2. 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 4.3 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.

4.2.4 Connectivity of Polyhedra

Definition 4.3 (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. The 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.

4.3 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.

Algorithm based on Branch and Bound

Let the envelope of the white polyhedra with markings M_w be denoted by $\mathcal{P}_m$. It is given by the marking m, which is constructed as in Lemma 4.2. Slightly abusing the notation we will write $m = \text{env}(M_w)$. As all the 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 $\mathcal{I}$ denote the index set of hyperplanes in $\mathcal{A}$ that are separating hyperplanes for polyhedra in the envelope $\mathcal{P}_m$. According to Lemma 4.1, $\mathcal{I}$ 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. $M_{w|m(i)=-} = \{m \in M_w \mid m(i) =$'$-$'$\}$. $M_{w|m(i)=+}$ 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 variables z and $\bar{z}$. z denotes the current number of merged white polyhedra and $\bar{z}$ is the local upper bound on z, where initially $z = 0$ and $\bar{z} = \infty$. 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, where $\#M$ denotes the number of elements in the set M.

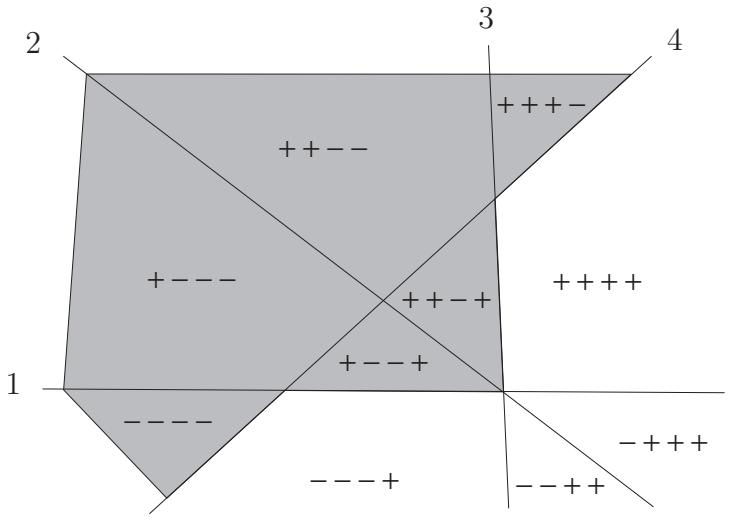


Figure 4.1: Example with four hyperplanes in $\mathcal{R} = \mathbb{R}^2$ and the corresponding markings. The polyhedra corresponding to M_w are white and the polyhedra corresponding to M_b' are grey shaded, respectively

Algorithm 4.1

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 4.1 As an example with four hyperplanes in a two-dimensional space consider Fig. 4.1. The envelope of the white polyhedra is given by the positive half space of H_4 and the marking $m = ***+$. 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 $M_m = \{-**+, ++++\}$. 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 4.4 *Algorithm 4.1 solves the Disjoint Optimal Complexity Reduction Problem 4.1.*

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.

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 branching order:

1. Hyperplanes that separate two non-connected groups of white polyhedra thus allowing us to divide the problem into two equivalent subproblems. Connectivity can be easily determined as described in Section 4.2.4. 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[2]. If so, we choose the hyperplane yielding the maximal number of white polyhedra.

[2]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.

4.4 Non-Disjoint Optimal Complexity Reduction

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. [Kat94]).

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 '$^-$' (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.

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 number of gates in a circuit 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 [Vei52] and Karnaugh [Kar53]. They introduced the K-map to manually minimize simple two-level[3] Boolean functions with up to six variables. A few years later, Quine [Qui55] and McCluskey [McC56] developed more sophisticated and systematic minimization techniques to obtain a two-level implementation of a given Boolean function with the minimum

[3]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.

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 [HCO74] and PRESTO [Bro81].

Inspired by MINI and PRESTO, ESPRESSO-II [BHMS84] 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 [DoE82], and it has been employed for the examples presented in the remainder of this chapter.

Problem Formulation with Boolean Logic

For a hyperplane arrangement with n hyperplanes $H_i = \{x \in \mathbb{R}^d \mid a_i^T x = b_i\}$, $i \in \{1, \ldots, n\}$, in the d-dimensional Euclidian space $\mathbb{R}^d$ we had defined in Section 3.2 in (3.1) 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 vector of Boolean variables

$$\mathrm{SV}'_i(x) = \begin{cases} 0 & \text{if } a_i^T x \leq b_i, \\ 1 & \text{if } a_i^T x > b_i \end{cases} \quad \text{for } i \in \{1, 2, \ldots, n\}, \tag{4.3}$$

using the dash to distinguish it from the original sign vector (3.1). 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 $\mathcal{P}_m$, is translated into Boolean variables that are removed from the Boolean vector δ. Thus δ has in general a variable number of components. Obviously, the definitions and lemmas of Section 4.2.3 can be directly used for Boolean variables with $\mathrm{SV}'(x)$ and δ, too.

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 4.5, 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.

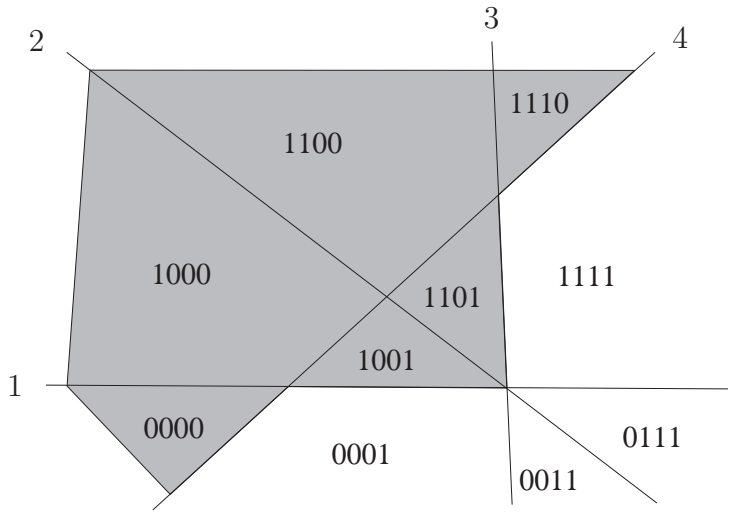
Alternatively, one may represent f_{W} in form of a truth table[4]. Such a truth table is the preferred input of ESPRESSO-II, which we use to perform the logic minimization. With respect to a Boolean function, a truth table carries the main advantage that it allows one to provide the logic minimization tool with additional structural information, namely empty polyhedra can be specified with an 'X'[5]. During the minimization process, the tool assigns 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 $\{\mathcal{Q}_i\}_{i=1,\ldots,q}$.

We refer to the logic minimization as Algorithm 4.2. Summing up, for a given color, the truth table with the Boolean function f_{W} is built, a logic minimization tool (here

[4] 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. The rows list all possible combinations of inputs together with the corresponding outputs.

[5] The number of rows in the truth table is exponential in n (the length of the Boolean vector δ and the number of hyperplanes in the arrangement). Yet according to Buck's formula (3.2), 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 $f_{\mathrm{W}} = 0$ and $f_{\mathrm{W}} = 1$, where ESPRESSO-II complements the rows with $f_{\mathrm{W}} = X$ internally. This technique allows for greatly reducing the memory requirement when passing the OCR problem to ESPRESSO-II.



(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

Figure 4.2: Revisited Example 4.1 with the hyperplane arrangement, the corresponding Boolean variables and the truth table

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 4.2 Reconsider Example 4.1 with the hyperplanes and markings as in Fig. 4.1, 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. 4.2(a). The Boolean function for the white polyhedra follows immediately to

$$f_{\mathrm{W}} = \bar{\delta}_1\bar{\delta}_2\bar{\delta}_3\delta_4 + \bar{\delta}_1\bar{\delta}_2\delta_3\delta_4 + \bar{\delta}_1\delta_2\delta_3\delta_4 + \delta_1\delta_2\delta_3\delta_4\,. \tag{4.4}$$

Thus, a given $x \in \mathcal{R}$ determines δ via (4.3), and f_{W} answers the question, whether x belongs to a white or black polyhedron. Simplifying this functions leads to $f_{\mathrm{W}} = \bar{\delta}_1\bar{\delta}_2\delta_4 + \delta_2\delta_3\delta_4$.

Alternatively, we may translate Fig. 4.2(a) into the truth table for white polyhedra shown in Table 4.2(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 4.1, where the disjoint OCR Algorithm based on the markings yielded $M_m = \{-{*}{*}+, {+}{+}{+}{+}\}$, the solution here is reduced by

two facets. As we will see in Section 4.5, 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 4.5 *Algorithm 4.2 solves the Non-Disjoint Optimal Complexity Reduction Problem 4.2.*

Proof. Assume that the set of resulting white polyhedra $\{\mathcal{Q}_i\}_{i=1,\dots,q}$ is an optimal solution to Problem 4.2. Then, the proof contains two steps. Firstly, we need to prove that the introduction of 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, 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. □

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.

4.5 Local and Global Optimality

4.5.1 Derivation of Global Hyperplane Arrangement

The Algorithms 4.1 and 4.2 in the proposed form are only applicable to problems with a globally valid hyperplane arrangement. In this section, we remove Assumption 4.1 and

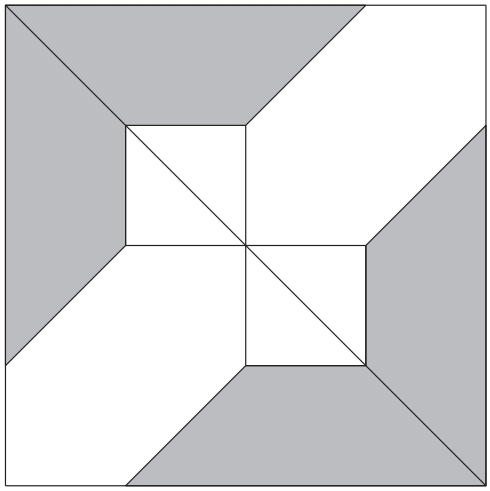
propose two extensions that will allow us to also employ the algorithms for problems with local hyperplane arrangements, or even more general, for problems that altogether lack a hyperplane arrangement.

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 4.1 or 4.2 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.

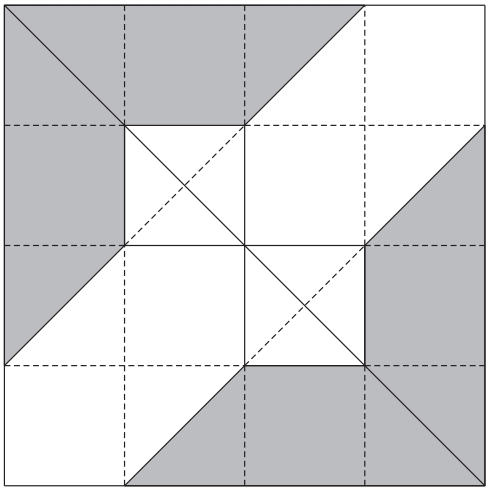
If the aim is to derive the *optimal* solution, we need to compute the global hyperplane arrangement by extending the facets of the polyhedra. Here, we give only a brief outline of such an algorithm, which consists of three major steps. First, we collect the facets of all polyhedra. By removing duplicates, 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 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 4.3.

Example 4.3 Consider the sets of white and black polyhedra in Fig. 4.3(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. 4.3(b). As a result, the number of white polyhedra is blown up from 6 to 16. OCR restricted to disjoint polyhedra (Algorithm 4.1) yields three white polyhedra depicted in Fig. 4.3(c), whereas Algorithm 4.2 yields only two white polyhedra that are overlapping as indicated by the dashed lines in Fig. 4.3(d).

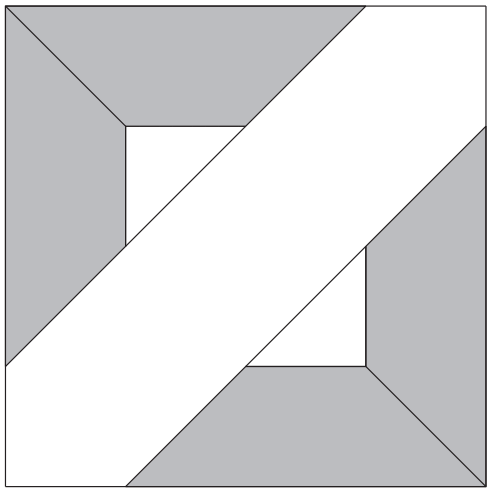
It is particularly interesting to observe that merging the original white polyhedra in Fig. 4.3(a) in a optimal way without using a global hyperplane arrangement would lead



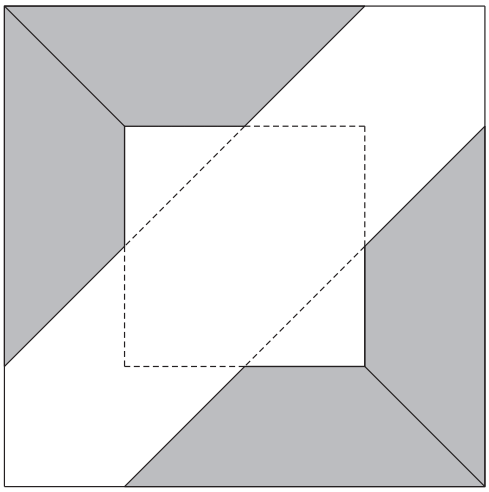
(a) Original white and black polyhedra



(b) Modified polyhedra in a (global) hyperplane arrangement



(c) Polyhedra resulting from disjoint OCR

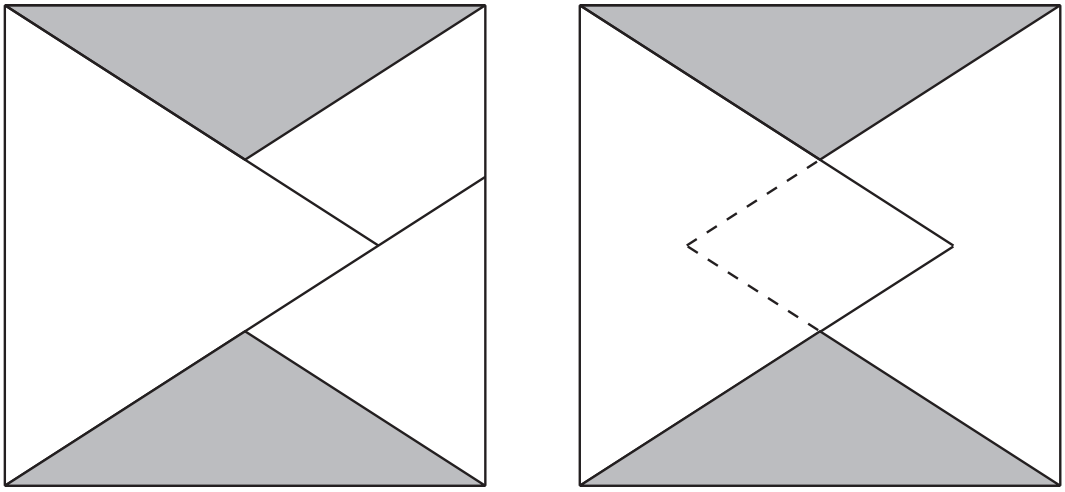


(d) Polyhedra resulting from non-disjoint OCR

Figure 4.3: Derivation of cells defined in a global hyperplane arrangement and OCR in Example 4.3

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 [BFT01], 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



(a) Polyhedra resulting from disjoint OCR

(b) Polyhedra resulting from non-disjoint OCR

Figure 4.4: OCR in Example 4.4 visualizing the consequence of restricting the hyperplane arrangement to hyperplanes given by facets of the original white polyhedra

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.

4.5.2 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 4.1 and 4.2). Yet, as the problems differ regarding 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 4.1, the resulting polyhedra are required to be disjoint and unions of the original polyhedra. Thus, Problem 4.1 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 4.4 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 4.1 that yields the three white polyhedra shown

in Fig. 4.4(a). This solution is optimal with respect to Problem 4.1. 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 4.2 leads to the two white polyhedra depicted in Fig. 4.4(b), where the dashed lines indicate the overlaps. Adding additional hyperplanes to the arrangement before running Algorithm 4.2 would not improve on the solution. This holds in general thanks to Lemma 4.6 presented at the end of this section.

We conclude that even though Algorithm 4.1 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 4.2, 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 4.2 is less restrictive than Problem 4.1, we expect Algorithm 4.2 to yield in general a lower number of polyhedra and facets than Algorithm 4.1. This is confirmed by the Examples 4.3 and 4.4. In particular, as already mentioned above, adding additional hyperplanes does not improve on the solution. This leads to the following key lemma.

Lemma 4.6 *Algorithm 4.3 followed by Algorithm 4.2 solves the General Non-Disjoint Optimal Complexity Reduction Problem 4.3.*

Proof. The proof follows directly from Problem 4.2, Lemma 4.5 and the fact that Algorithm 4.2 minimizes (with second priority) the number of facets. □

4.6 Extensions

In this section, we briefly present two techniques to extend the applicability of the algorithms to problems of larger size, namely to problems with polyhedra defined in higher dimension or to problems with several hundred or even thousands of polyhedra. As the complexity of the proposed algorithms depends exponentially on the number of hyperplanes in the arrangement, both techniques presented in the following aim at reducing the number of hyperplanes. The first technique simplifies the hyperplane arrangement, whereas the second approach divides the problem into easier to solve subproblems with a reduced number of hyperplanes, solves them, and recombines them in a final step, which is commonly known as *Divide and Conquer* technique.

4.6.1 Simplified Hyperplane Arrangement

In many cases, the hyperplane arrangement contains numerous hyperplanes that are almost identical, at least very similar. Given two hyperplanes $H_i = \{x \in \mathbb{R}^d \mid a_i^T x = b_i\}$ and $H_j = \{x \in \mathbb{R}^d \mid a_j^T x = b_j\}$, we use as a measure for similarity the norm $\mu = ||[a_i^T \ b_i]^T - [a_j^T \ b_j]^T||_1$, and say that the hyperplanes H_i and H_j are similar, if μ is below a given threshold. Here, we assume that all hyperplanes are normed, i.e. $a_i^T a_i = 1$.

Here, we suggest an algorithm that simplifies the hyperplane arrangement by replacing clusters of similar hyperplanes by their weighted average. In general, the result will have (small) color errors. We define the color errors as follows: The radius of the largest Chebycheff ball that can be inscribed in the intersection of two polyhedra $\mathcal{P}$ and $\mathcal{Q}$, where $\mathcal{P}$ is in the set of original polyhedra, $\mathcal{Q}$ is in the set of resulting polyhedra, and both polyhedra are associated with different colors. Often, small errors in the color of the resulting polyhedra can be tolerated, particularly in the presence of model uncertainties (in case of PWA models) or noise on the state (in case of PWA state-feedback control laws).

Next, we outline the algorithm for deriving a simplified hyperplane arrangement based on the set of initial polyhedra $\{\mathcal{P}_i\}_{i=1,\dots,p}$.

1. *Hyperplane Arrangement:* Collect the facets of all polyhedra and remove duplicates. This yields the hyperplane arrangement to be simplified, where $\mathcal{I}$ denotes the index set of hyperplanes.

2. *Clusters:* For a given μ, identify clusters of similar hyperplanes by initially choosing a random hyperplane and iteratively adding hyperplanes to the cluster that are similar to the weighted average of the cluster. Remove all these hyperplanes from $\mathcal{I}$. If no more hyperplanes in $\mathcal{I}$ can be added to the cluster, initiate a new cluster by picking a hyperplane in $\mathcal{I}$ and proceed as above until $\mathcal{I}$ is empty. Consequently, similar hyperplanes are clustered. Replace the clusters by one hyperplane given by the weighted average of the cluster thus leading to the simplified hyperplane arrangement.

3. *Markings and Colors:* Compute the markings of the simplified hyperplane arrangement. Identify the color of each marking by building the corresponding cell and intersecting it with all given initial polyhedra $\mathcal{P}_i$. If all non-empty intersections are with $\mathcal{P}_i$ of the same color, assign this color to the cell. Otherwise, either use the color of the largest intersection, or assign a don't care to the cell.

4. *Optimal Complexity Reduction:* Run Algorithm 4.1 or 4.2 to obtain $\{\mathcal{Q}_i\}_{i=1,\dots,q}$, which denotes the minimal representation of $\{\mathcal{P}_i\}_{i=1,\dots,p}$.

5. *Error:* A posteriori, intersect each $\mathcal{Q}_i$ with all $\mathcal{P}_i$ to determine the distribution of the color error as defined above. If the maximal color error is in the range of the level of uncertainty or noise, the result should be generally acceptable; else repeat the above procedure with a smaller μ.

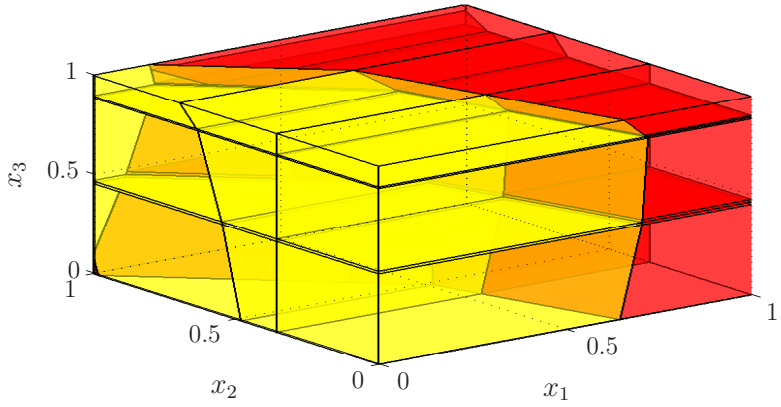
Example 4.5 Consider in the three-dimensional space the sets of yellow (front part of figure) and red (back part) polyhedra in Fig. 4.5(a), where we aim at minimizing both the 15 yellow and the 28 red polyhedra. Thus, after running Algorithm 4.3 to obtain the (global) hyperplane arrangement, we execute Algorithm 4.2 twice. When refraining from simplifying the hyperplane arrangement, the one yellow and six red polyhedra result as shown in Fig. 4.5(b). Alternatively, simplifying the hyperplane arrangement with $\mu = 0.04$ leads to one yellow and five red polyhedra and a color error below 0.004. Increasing μ to 0.1 and 0.35 reduces the red polyhedra to four and three, and increases the maximal color error to 0.01 and 0.037, respectively. As the polyhedra are scaled to $[0, 1]$, even the absolute error of 0.037 corresponds to an inaccuracy of only 3.7 percent.

Summing up this example, the non-disjoint OCR algorithm with $\mu = 0$ reduced the number of polyhedra by 84 percent. Setting $\mu = 0.35$ additionally reduced the complexity by 43 percent while introducing a color error of 3.7 percent.

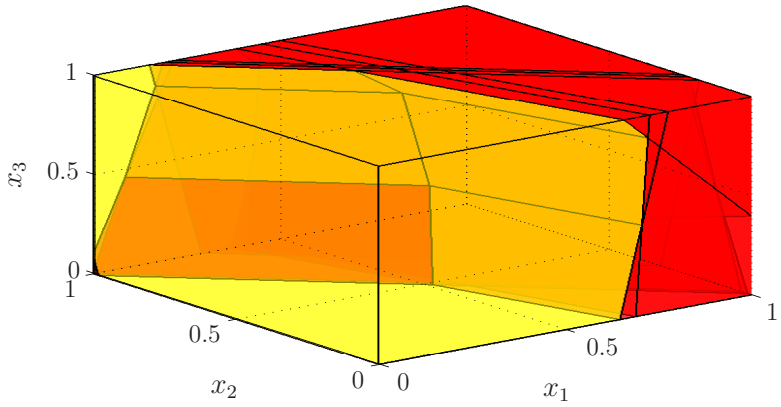
4.6.2 Divide and Conquer

Another approach to tackle complex problems is to divide the original large problem sequentially into pairs of smaller subproblems, which can be solved efficiently, and to subsequently recombine the solutions of the subproblems. Although each subproblem is solved to optimality, the overall solution would be in general suboptimal. To obtain optimality, we subsequently run the OCR algorithm on the unions of the solutions of the subproblems. In many cases, this is computationally feasible as the subproblems have been greatly simplified in the first step. Such a scheme is commonly known as *Divide and Conquer*. It is particularly useful in our context here, since the computational burden and memory requirement is basically exponential in the number of hyperplanes.

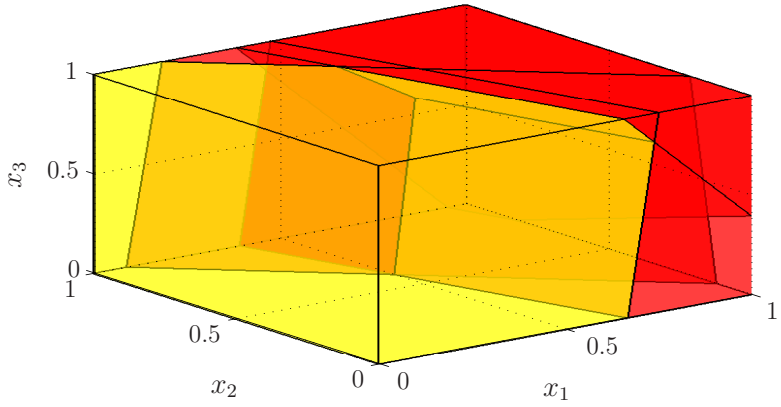
Various variations are possible. For example, the subproblems can be recombined pairwise and iteratively. Yet, more important is the way, a problem with the hyperplane arrangement $\{H_i\}_{i=1,\dots,n}$ is split into two subproblems. As the computational burden mostly depends on n, it is beneficial to divide the problem such that the number of hyperplanes in the subproblems n_1 and n_2 are both (close to being) minimal and balanced ($n_1 \approx n_2$).



(a) Original sets of yellow and red polyhedra

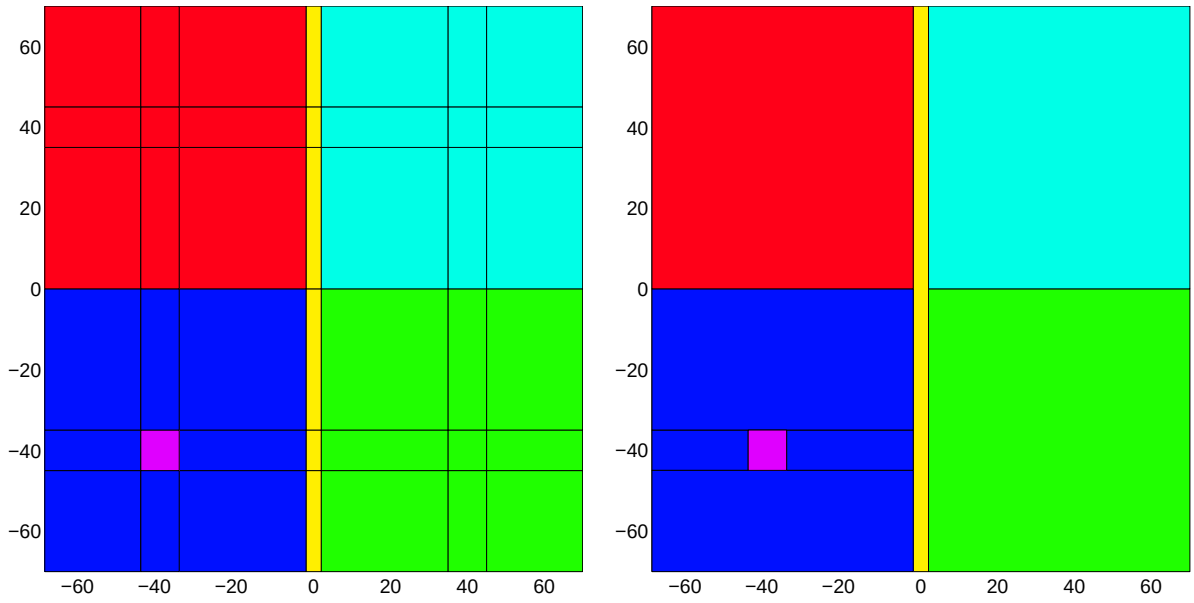


(b) Polyhedra resulting from non-disjoint OCR, where the hyperplane arrangement has not been simplified ($\mu = 0$)



(c) Polyhedra resulting from non-disjoint OCR, where the hyperplane arrangement has been simplified ($\mu = 0.1$)

Figure 4.5: OCR and simplification of the hyperplane arrangement in Example 4.5



(a) The original polyhedra

(b) The merged polyhedra

Figure 4.6: Polyhedral partitions of the PWA model of the paperboy problem plotted on the real state-space for a given binary state combination, where each color relates to a different pair of state-update and output functions

4.7 Examples

In this final section we present three examples showing how the OCR algorithms can be applied to PWA models as well as to PWA state-feedback control laws in order to efficiently derive an equivalent minimal representation.

4.7.1 Paperboy Problem

In Section 3.5.2, we have described the model of a paperboy delivering by bike mail items to households within a neighborhood consisting of four properties and one road, where the properties and the road have different slopes and different friction coefficients. The paperboy problem encompasses two real inputs, four real states and two binary states.

The mode enumeration algorithm described in the previous chapter yields the equivalent PWA model together with the corresponding hyperplane arrangement encompassing 11 hyperplanes and the set of markings. The PWA model encompasses 168 polyhedra, what is by far below the upper bound 7099 defined by Buck's formula (3.2). Although the paperboy example is given by a composition of three DHAs, all hyperplanes are defined solely on the state-space and not sequentially on outputs of DHAs, as can be seen from the two event generators of the paperboy model. Thus, the hyperplane arrangement is globally valid. Even though hyperplane arrangement and the PWA model are defined on

the eight-dimensional state-input space, the hyperplanes only depend on two real and the two binary states. This allows us to visualize the polyhedral partition for a given binary state combination as shown in Fig. 4.6(a), where each color relates to a different pair of state-update and output functions.

The disjoint OCR Algorithm 4.1 yields the polyhedra depicted in Fig. 4.6(b). Repeating this for all four binary state combinations allows us to reduce the number of polyhedra from 168 down to 36 within 0.22 s running MATLAB 6.5 on a Pentium IV 2.8 GHz machine. This is a reduction of the complexity by roughly 80 percent. In particular, both PWA models are equivalent, meaning that for every given state and input they yield the same state-updates and outputs.

This example clearly shows that even though merging polyhedra with the same dynamic in an optimal way is $\mathcal{NP}$-hard the concept of the markings in a hyperplane arrangement makes the algorithms applicable to problems of meaningful size, as those markings contain all the information needed to decide on the convexity of a set of polyhedra.

4.7.2 Constrained Infinite Time Optimal Control Law

Next, we perform OCR to a PWA state-feedback control law. As an example, consider the following simple PWA model from [BM99a]

$$x(k+1) = 0.8 \begin{bmatrix} \cos\alpha(k) & -\sin\alpha(k) \\ \sin\alpha(k) & \cos\alpha(k) \end{bmatrix} x(k) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(k), \tag{4.5a}$$

$$\alpha(k) = \begin{cases} \frac{\pi}{3} & \text{if} \quad [1\ 0]\, x(k) \geq 0, \\ -\frac{\pi}{3} & \text{if} \quad [1\ 0]\, x(k) < 0, \end{cases} \tag{4.5b}$$

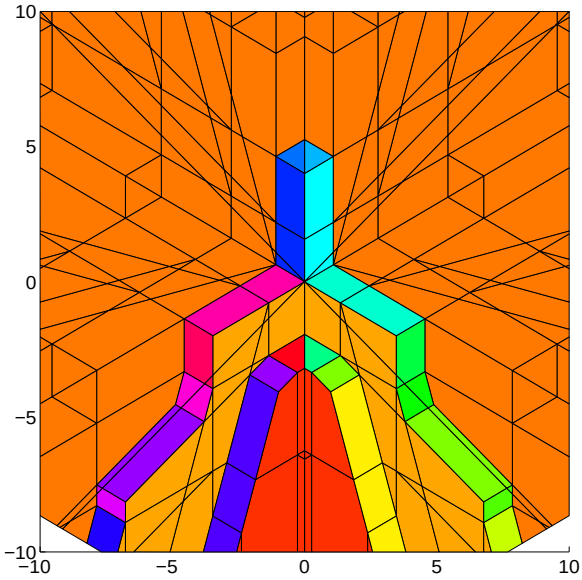
$$x(k) \in [-10, 10] \times [-10, 10], \qquad u(k) \in [-1, 1]\,, \tag{4.5c}$$

which features two real states and two modes. For this model, the authors in [BCM03a] have formulated and solved a constrained infinite time optimal control problem of the form

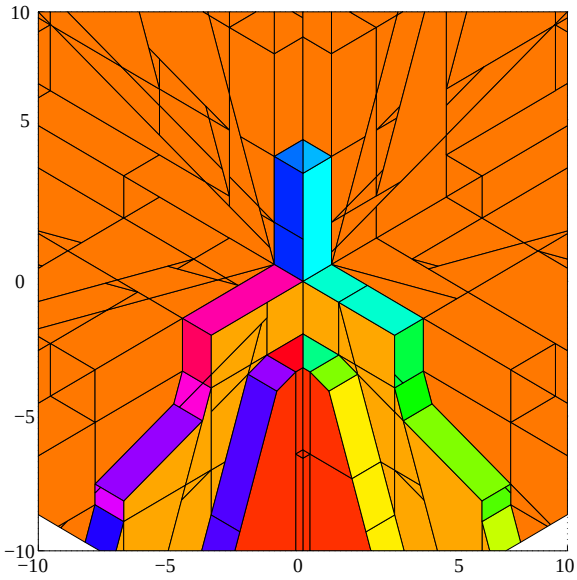
$$\begin{aligned} J^*(x(k)) = \min_{U_\infty(k)} \sum_{\ell=0}^{\infty} \|x(k+\ell|k)\|_\infty + \|u(k+\ell)\|_\infty, \\ \text{subj. to the PWA model (4.5)}\,, \end{aligned} \tag{4.6}$$

where $U_\infty(k) = [(u(k))^T, (u(k+1))^T, \ldots]^T$ is the sequence of manipulated variables u. The resulting polyhedral partition of the state-space is shown in Fig. 4.7(a), where each color relates to a different affine control law. Note that there exist 19 different control laws and 252 polyhedra.

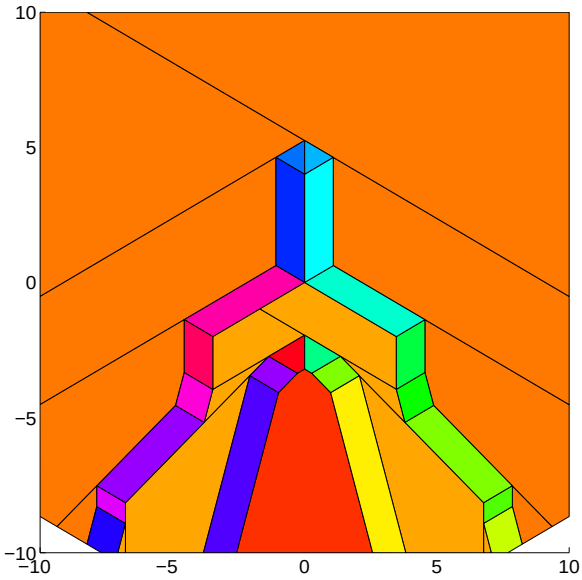
For this example, we compare three algorithms with each other. Namely the disjoint and non-disjoint OCR Algorithms 4.1 and 4.2 preceded by Algorithm 4.3, which derives



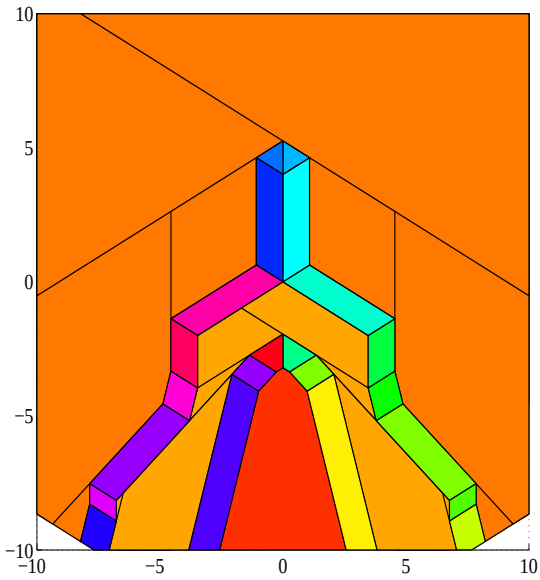
(a) Original set of polyhedra



(b) Set of polyhedra resulting from greedy merging



(c) Set of polyhedra resulting from Algorithms 4.3 and 4.1



(d) Set of polyhedra resulting from Algorithms 4.3 and 4.2

Figure 4.7: Polyhedral partitions of the PWA state-feedback control law, where each color relates to a different affine feedback law, using different complexity reduction schemes

the hyperplane arrangement, and a third algorithm, to which we refer as *greedy merging*. To speed up the computations, the greedy merging algorithm builds in a first step a sparse matrix indicating whether two polyhedra are neighbors according to Definition 4.3. Based on this list, it determines by solving LPs if a pair of neighboring polyhedra forms a convex

union. If so, the pair is replaced by its union. Here, the merging procedure is done in a greedy way.

Greedy merging fails to adequately reduce the complexity of the problem. It leads to the result shown in Fig. 4.7(b) with 189 polyhedra. As will be shown in the next paragraph, this is five times worse than the result obtained by the two OCR algorithms. Furthermore, the computation time is with 17 s comparatively large.

Algorithm 4.3 derives a hyperplane arrangement with 135 hyperplanes containing 5200 polyhedra within 34 s on the same machine as above. The disjoint OCR Approach 4.1 leads to 39 polyhedra, which are shown in Fig. 4.7(c). Compared to the initial 252 polyhedra, this is a reduction of 84 percent. The computation time amounts to 3 min summing up to a total of 3.5 min. The non-disjoint OCR Algorithm 4.2 also leads to 39 polyhedra. These are overlapping and differ from the previous solution as shown in Fig. 4.7(d). Yet the computation time is with 5 s very small yielding the total computation time of 39 s.

Based on our experience, we conclude the following. The disjoint OCR algorithm based on branch and bound is rather slow limiting its applicability mostly to problems with a few thousands of polyhedra defined in a two- or three-dimensional space. On the other hand, the non-disjoint OCR algorithm based on logic minimization is generally not only by two orders of magnitude faster, it also scales better as the problem size increases. Problems with hyperplane arrangements comprising hundreds of hyperplanes with some 100′000 cells have been tackled successfully within a few minutes. For larger problems, the memory requirement for storing the truth table becomes an issue. The bottleneck, however, is the computation of the cells in the hyperplane arrangement, namely Algorithm 4.3. The main problem is the major increase in the number of polyhedra when deriving the hyperplane arrangement (in the above example from 252 to 5200 polyhedra). This can be overcome by applying the techniques presented in Section 4.6, namely simplifying the hyperplane arrangement or using a divide and conquer approach as will be shown in the next example.

4.7.3 Optimal Direct Torque Control Law

So far, we have considered only examples with polyhedra defined in two dimensions. In the last example, we apply the non-disjoint OCR algorithm to a real application, namely to the field of power electronics. Specifically, we consider a three-phase two-level DC-AC inverter driving an induction motor as shown in Fig. 5.2(a).

For this setup, we formulate an optimal direct torque control problem in Section 7.2, which we solve off-line in Section 7.2.5 according to the procedure outlined in Section 7.2.4.

The eight different combinations of switch positions of the inverter constitute the manipulated variable u. As we use a δu formulation in the cost function, where $\delta u(k) = u(k) - u(k-1)$, we obtain for each $u(k-1)$ one PWA state-feedback con-

$u(k-1)$	n_{org}	n_{red}	reduction [%]	t_{red} [h]
$[-1\ -1\ -1]^T$	5246	709	86.5	9.1
$[-1\ -1\ +1]^T$	5325	753	87.6	10.3
$[-1\ +1\ -1]^T$	4737	486	89.7	10.9
$[-1\ +1\ +1]^T$	5292	625	88.2	9.8
$[+1\ -1\ -1]^T$	7019	930	86.8	8.9
$[+1\ -1\ +1]^T$	8512	880	90.0	13.3
$[+1\ +1\ -1]^T$	5425	631	88.4	8.7
$[+1\ +1\ +1]^T$	6295	617	90.2	7.3

Table 4.1: Overview of the PWA state-feedback control law for the DTC drive with a two-level inverter, to which we apply the non-disjoint OCR. For each last control input $u(k-1)$, n_{org} is the number of polyhedra of the original controller, n_{red} is the number of polyhedra of the equivalent controller of reduced complexity, and t_{red} is the computation time for the complexity reduction in hours run on a 2.8 GHz Pentium IV with Linux and MATLAB 6.5.

trol law defined on the real three-dimensional state-space. The result is summarized in Table 4.1, which is basically a replicate of Table 7.4. The control law is overly complex and comprises for a given $u(k-1)$ up to 8500 polyhedra.

Hence, we apply OCR to each of the eight control laws. To make our schemes applicable, we employ the Divide and Conquer strategy outlined in Section 4.6. If the number of hyperplanes exceeds 100, we divide the problem into two subproblems, such that both subproblems have roughly the same number of hyperplanes and as few as possible. We continue doing so until a subproblem has less than 100 hyperplanes. For each subproblem, we first translate the control law into a problem with cells defined in a hyperplane arrangement using Algorithm 4.3. Subsequently, we apply the non-disjoint OCR Algorithm 4.2. The solutions to each subproblem pair are combined and if the number of hyperplanes is below 130, Algorithms 4.3 and 4.2 are applied to it.

The result is shown in Table 4.1. The complexity of the control laws is reduced by roughly 90 percent. Strictly speaking this result is not optimal, since some of the recombined problems have more than 130 hyperplanes. The computation times are definitely large and sum up to several hours. Yet we want to recall that the complexity reduction needs to be performed only once off-line. For the particular application the reduction of the number of polyhedra by an order of magnitude may be decisive for a successful hardware implementation, since the sampling interval is as short as 25 μs.

4.8 Conclusions and Future Research

Conclusions

Exploiting the markings of the hyperplane arrangement allowed us to build an equivalent PWA system minimal in the number of polyhedra by using either branch and bound techniques to derive a disjoint set of polyhedra, or logic minimization to obtain polyhedra that are in general overlapping. If the markings and the hyperplane arrangement were already given (e.g. from a preceding run of the mode enumeration algorithm), it was not necessary to solve additional LPs. This guaranteed that the algorithm is not only optimal but also computationally tractable. By computing the (global) hyperplane arrangement, the applicability of the algorithm could be extended to derive minimal PWA representations of general PWA models and control laws lacking a hyperplane arrangement. To reduce the computational complexity of large problems, two techniques were proposed, namely the simplification of the hyperplane arrangement and a *Divide and Conquer* strategy. The effectiveness of the OCR approaches was demonstrated through various examples.

Future Research

When implementing the feedback law for on-line use, one faces the problem of finding the control law for the given state. The standard solution is to determine the polyhedron the state lies in by cycling through (in the worst case) all polyhedra and checking if the corresponding inequalities hold. Even though these operations only involve matrix multiplications, for small sampling times and/or large numbers of polyhedra this may become a challenge, as we need to guarantee that the computation time is always below the sampling interval.

The OCR algorithms not only allow for deriving a representation of the control law that is minimal in the number of polyhedra, but also provide a simple and efficient way to implement it. After the OCR step, we propose to compute again the hyperplane arrangement and the markings for the reduced problem. Based on the markings, the controller can be implemented either as a collection of Boolean functions or as a binary search tree.

Boolean Functions. Each marking or Boolean vector is associated with a certain color, where each color represents a feedback law. Similar to Section 4.2, we build for each color a Boolean function with the Boolean vector as argument. This yields a collection of Boolean functions. Thus on-line, for a given state, one only needs to determine the Boolean vector based on the modified sign vector, evaluate which Boolean function is true, which directly relates to the feedback law. Hence, only the sign vector together with the Boolean functions needs to be implemented. In particular, polyhedra do not need to be

stored and evaluated thus reducing the memory requirement and the on-line computation time. Such an approach is particular suitable for a hardware implementation, since the Boolean functions can be easily implemented as two-level disjunctive normal form using AND, OR and NOT gates.

Binary Search Tree. Alternatively, we can build a binary search tree similar to [TJB03], which allows one to determine for a given state the polyhedron and the associate control law efficiently. Given a state, such a tree is evaluated by traversing from the root node to the leafs. Each node is associated to a hyperplane, and depending in which half space of the hyperplane the state lies, one branches accordingly. Each leaf is associated with a control law (and a polyhedron).

A non-trivial task is to build a search tree that is of minimal depth. The depth directly corresponds to the maximal number of hyperplanes that need to be on-line evaluated and thus to the worst-case computational burden. The authors in [TJB03] use heuristics to derive a tree of little depth. Using the markings, however, enables us to solve the problem to global optimality by setting up a branch and bound algorithm similar to Algorithm 4.1 that derives a binary search tree of minimal depth.

Part III

Direct Torque Control

5

Problem Description

5.1 Introduction

During the last decades, the rapid development of power semiconductor devices has led to the increased use of adjustable speed induction motor drives in a variety of applications. In these systems, DC-AC inverters are used to drive induction motors as variable frequency three-phase voltage or current sources. One of the various methods that are used for controlling the induction motor's torque and speed in such systems is Direct Torque Control (DTC), which was first introduced in 1985 by Takahashi and Noguchi [TN86] and is nowadays a well established industrial standard for induction motor drives [PTL94, TO89].

The basic principle of DTC is to exploit the motor's fast stator flux dynamics and to directly manipulate the stator flux vector such that the desired torque is produced. This is achieved by choosing an inverter switch combination that drives the stator flux vector to the desired position by directly applying the appropriate voltages to the motor windings. This choice is made usually with a sampling time $T_s = 25\,\mu$s using a pre-designed switching table that is derived in a heuristic way and, depending on the particularities of the application, addresses a number of different control objectives. These primarily concern the behavior of the induction motor - more specifically, the stator flux and the electromagnetic torque need to be kept within pre-specified bounds around their references. In high power applications, where three-level inverters with Gate Turn-Off (GTO) thyristors are used, the control objectives are extended to the inverter. In such a case, they also include the minimization of the average switching frequency and the balancing of the inverter's neutral point potential around zero.

DTC features a number of benefits with respect to performance and implementation. The achieved dynamic torque responses are rapid and accurate throughout the whole operating range of the machine. Additionally, the integrated approach to the control problem of both the inverter and the motor, and the absence of a modulation unit lead to

a simple controller implementation. On the other hand, DTC carries some considerable disadvantages, such as the presence of a high current and torque ripple, the fact that the average switching frequency is not directly controllable, and the difficulty of controlling torque and flux at low frequencies [CPST02]. Different approaches have been reported in the literature [CST00, PA01] that cope with these drawbacks by mainly aiming at improving the design of the switching table. Given the trade-off between the switching frequency and the torque and flux ripple, which generally exists in DTC, such an improvement can be translated either into a reduction of the average inverter switching frequency for the same torque and flux ripple, or vice-versa into a reduction of the ripple for the same switching frequency.

Although the above mentioned methods for improving the switching table are well suited for two-level inverters, their extension to more complex problems featuring a higher degree of freedom for the manipulated variables remains a challenging task [MRMC02]. The main difficulty arising during the controller design is, in general, the fact that a DTC drive constitutes a hybrid system, i.e. a system incorporating both continuous and discrete dynamics - in particular discrete-valued manipulated variables. Additionally, constraints on states, inputs and outputs are present imposing further complications on the controller design, since the underlying mathematical problems are intrinsically complex and hard to solve.

Motivated by the lack of a systematic design procedure for the switching table, novel modelling and control approaches to the DTC problem of three-phase induction motors are presented in Chapter 6 and Chapter 7, respectively, aiming at making the design process independent of the drive's characteristics and specifications. To do so, Chapter 5 serves as a starting point. Specifically, using the concept of the dq0 reference frame summarized in Section 5.2, Section 5.3 presents the nonlinear continuous-time models of two- and three-level inverters and induction machines. Subsequently, the direct torque control problem is stated in Section 5.4, and Section 5.5 presents the traditional control approach for DTC that is commonly used in industry nowadays. Sections 5.6 and 5.7, respectively, summarize the state of research and the contribution of this thesis concerning DTC. Summing up, this chapter is a summary of the state of the art, whereas the following two chapters present new contributions for modelling and controlling DTC drives.

Throughout the following three chapters, we use the normalized time scale t with one time unit corresponding to $1/\omega_b$ seconds, where ω_b is the base angular velocity used to calculate the inductive reactances of the motor. Additionally, we use $x(t)$, $t \in \mathbb{R}$, to denote continuous-time variables, and $x(k)$, $k \in \mathbb{N}_0$, to denote discrete-time variables with the sampling time $T_s = 25\,\mu$s. Furthermore, the state estimation of the motor is considered to be ideal.

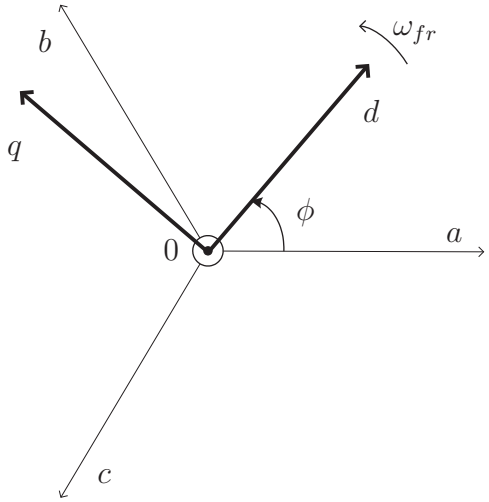


Figure 5.1: Rotating dq0 reference frame

5.2 The dq0 Reference Frame

To simplify the modelling of the DTC drive, it is common practice to transform all variables from the three-phase system (abc) to an orthogonal dq0 reference frame with a direct (d), a quadrature (q) and a zero (0) axis, that can be either stationary or rotating. Details regarding reference frame theory can be found in the relevant literature [Kra86]. For the needs of this paper, the transformation of a vector $\xi_{abc} = [\xi_a \ \xi_b \ \xi_c]^T$ from the three-phase system to the vector $\xi_{dq0} = [\xi_d \ \xi_q \ \xi_0]^T$ in the dq0 frame is carried out through

$$\xi_{dq0} = P(\varphi)\xi_{abc}\,, \tag{5.1}$$

where φ is the angle between the a-axis of the three-phase system and the d-axis of the reference frame, and[1]

$$P(\varphi) = \frac{2}{3}\begin{bmatrix} \cos\varphi & \cos(\varphi - \frac{2\pi}{3}) & \cos(\varphi + \frac{2\pi}{3}) \\ -\sin\varphi & -\sin(\varphi - \frac{2\pi}{3}) & -\sin(\varphi + \frac{2\pi}{3}) \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix}. \tag{5.2}$$

If the frame is rotating with the angular speed ω_{fr} as shown in Fig. 5.1, then $\varphi = \omega_{fr}t + \varphi_0$, otherwise, if the frame is stationary, φ is time-invariant. In particular, for $\varphi = 0$ when the reference frame is fixed and the d-axis is aligned with the a-axis, the transformation

[1] Note that the transformation matrix is not orthonormal, i.e. $P(\varphi)P^T(\varphi) \neq I$. This definition is in contrast to [PGM04b] but in accordance with [PGM04c, GP05, GPM04b] to ensure a unified presentation and the same scaling of the machine variables.

matrix

$$P = \frac{2}{3}\begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \tag{5.3}$$

results.

Often, when transforming ξ_{abc} vectors using $P(\varphi)$, we are only interested in the d and q component, but not in the zero component of ξ_{dq0}. For this, we introduce the transformation matrix $\tilde{P}(\varphi)$ that features only the upper two rows of $P(\varphi)$ and yields

$$\xi_{dq} = [\xi_d \ \xi_q]^T = \tilde{P}(\varphi)\xi_{abc}\,. \tag{5.4}$$

$\tilde{P}$ is defined accordingly.

5.3 Nonlinear Continuous-Time Modelling

5.3.1 Two-Level Inverter

An equivalent representation of a three-phase two-level inverter driving an induction motor is shown in Fig. 5.2(a). At each phase, the inverter can produce two different voltages $-\frac{V_{dc}}{2}, \frac{V_{dc}}{2}$, where V_{dc} denotes the voltage of the dc-link. The switch positions of the inverter can therefore be fully described using the three integer variables u_a, u_b, $u_c \in \{-1, 1\}$, where each variable corresponds to one phase of the inverter, and the values $-1, 1$ correspond to the phase potentials $-\frac{V_{dc}}{2}, \frac{V_{dc}}{2}$, respectively.

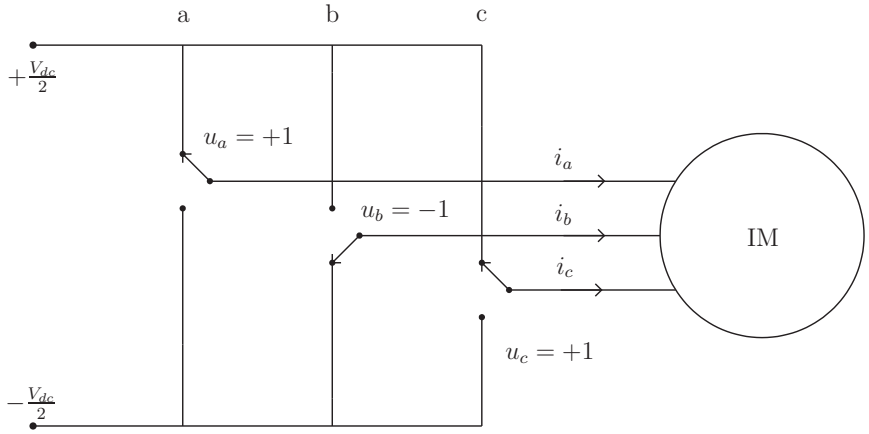
There are $2^3 = 8$ different vectors of the form $u_{abc} = [u_a \ u_b \ u_c]^T$. Using (5.1) these vectors can be transformed into the dq0 frame resulting in vectors of the form $u_{dq0} = [u_d \ u_q \ u_0]^T$. The latter are shown in Fig. 5.2(b), where they are mapped into the (two-dimensional) dq plane. Even though they are commonly referred to as voltage vectors, this term describes the switch positions rather than the actual voltages applied to the machine terminals. These voltages are calculated from

$$v_{dq0} = [v_d \ v_q \ v_0]^T = \frac{V_{dc}}{2} u_{dq0}. \tag{5.5}$$

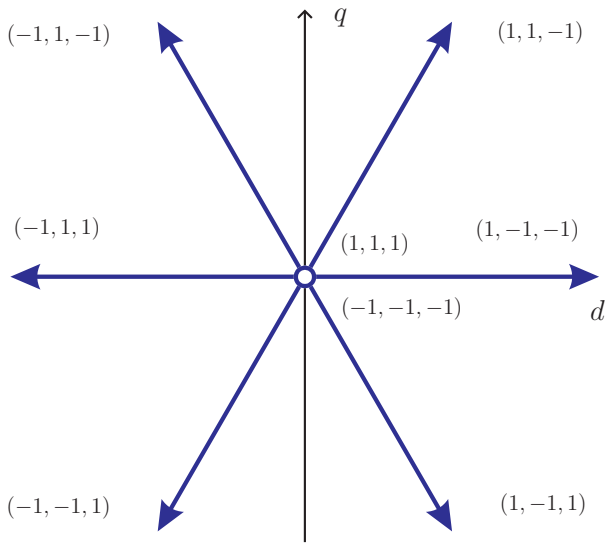
The voltage vectors can be divided in two groups: six long vectors forming the outer hexagon and two zero vectors. The zero vectors correspond to the switch combinations $(+1, +1, +1)$ and $(-1, -1, -1)$, and short-circuit the machine terminals.

5.3.2 Three-Level Inverter

The equivalent representation of a three-phase three-level inverter driving an induction motor is shown in Fig. 5.3(a). The additional feature of the three-level inverter is that



(a) Equivalent representation

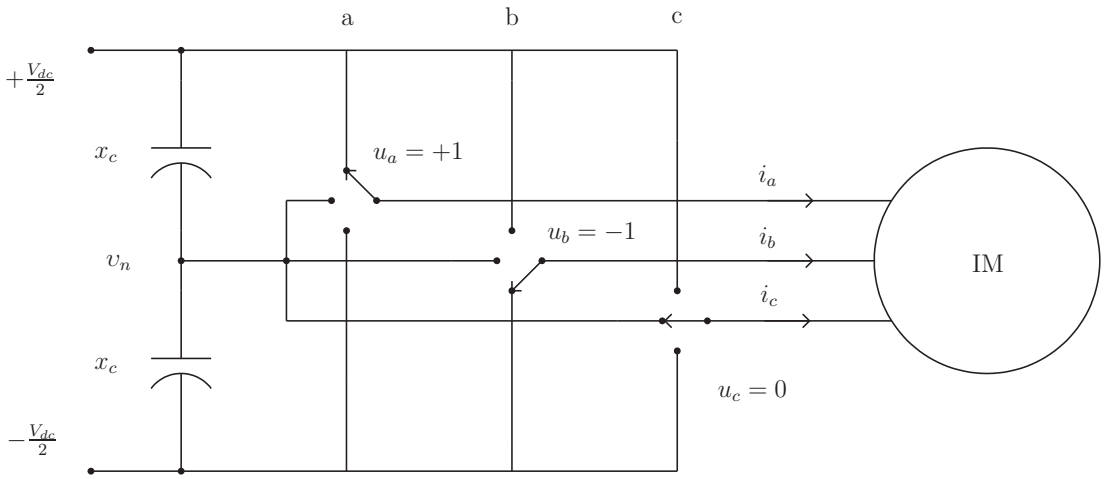


(b) Voltage vectors on the dq plane and the corresponding values of the integer variables (switch positions)

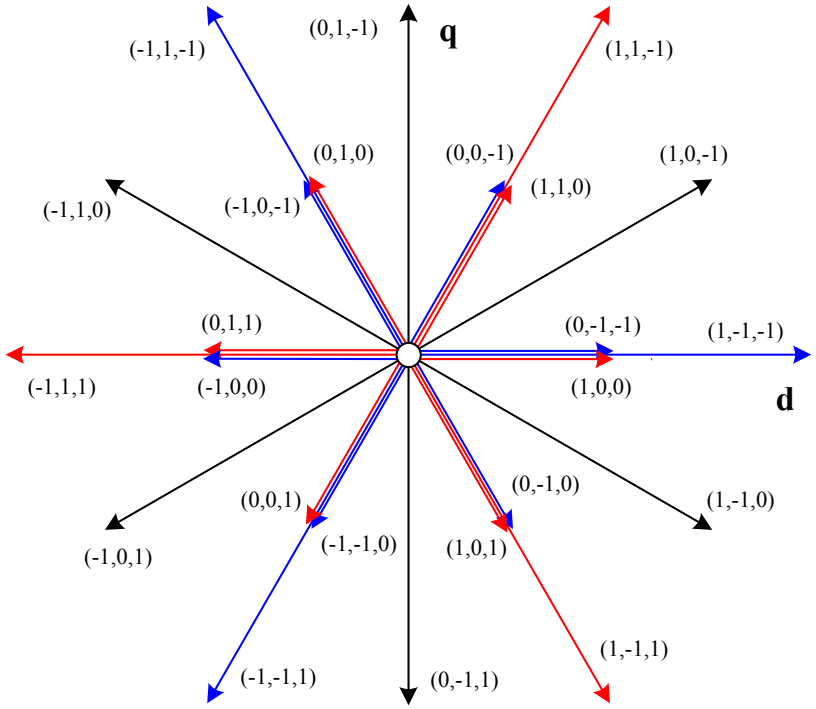
Figure 5.2: Three-phase two-level inverter driving an induction motor

it can also produce a zero phase voltage resulting in a total of three different possible voltages $-\frac{V_{dc}}{2}, 0, \frac{V_{dc}}{2}$ at each phase. The switch positions of the three-level inverter are now described using the integer variables u_a, u_b, $u_c \in \{-1, 0, 1\}$. As with the two-level inverter, each variable corresponds to one phase of the inverter, and the values $-1, 0, 1$ correspond to the phase potentials $-\frac{V_{dc}}{2}, 0, \frac{V_{dc}}{2}$, respectively.

Similarly, there exist $3^3 = 27$ different vectors of the form $u_{abc} = [u_a\ u_b\ u_c]^T$, that can be transformed accordingly into the dq0 frame using (5.1) resulting in vectors of the form $u_{dq0} = [u_d\ u_q\ u_0]^T$ as shown in Fig. 5.3(b), where they are mapped into the dq plane. The voltage vectors of the three-level inverter can be divided in four groups: six long vectors forming the outer hexagon, six vectors of medium length, twelve short vectors spanning



(a) Equivalent representation



(b) Voltage vectors on the dq plane and the corresponding values of the integer variables (switch positions), where the three zero vectors $(-1,-1,-1)$, $(0,0,0)$ and $(+1,+1,+1)$ in the origin of the dq plane have been omitted due to lack of space

Figure 5.3: Three-phase three-level inverter driving an induction motor

the inner hexagon and three zero vectors. The twelve short vectors form six pairs on the dq plane, where each pair comprises vectors with the same d- and q- but opposite 0-components. The zero vectors correspond to the switch combinations $(+1,+1,+1)$,

$(0,0,0)$ and $(-1,-1,-1)$, and short-circuit the machine terminals.

When operating a three-level inverter, the potential υ_n of the neutral point (see Fig. 5.3(a)) and the smooth distribution of the switching effort between the upper and the lower half of the inverter deserve particular attention. As can be seen in Fig. 5.3(a), the neutral point potential depends on the state of charge of the two dc-link capacitors and is only affected when current is drawn directly from it, i.e. if one of the voltage vector components is zero. Introducing υ_n as a real state, the neutral point potential is described in continuous-time by

$$\frac{\mathrm{d}\upsilon_n}{\mathrm{d}t} = -\frac{1}{2x_c}[(1-|u_a|)i_a + (1-|u_b|)i_b + (1-|u_c|)i_c]\,, \tag{5.6}$$

with i_a, i_b, i_c being the phase stator currents and x_c one of the two symmetric capacitors of the dc-link. Taking into account that

$$i_a + i_b + i_c = 0, \tag{5.7}$$

it is straightforward to derive

$$\frac{\mathrm{d}\upsilon_n}{\mathrm{d}t} = \frac{1}{2x_c}|u_{abc}|^T\, i_{abc}\,, \tag{5.8}$$

where $|u_{abc}| = [|u_a|\ |u_b|\ |u_c|]^T$ is the componentwise absolute value of the inverter switch positions. For more details about the nature of the neutral point potential and methods employed to tackle the related balancing problem the reader is referred to [dTM02] and [CB00].

Distributing the switching losses evenly between the semiconductor devices is essential for the inverter's operation, since it helps to prevent that some devices are overloaded while others remain under-utilized. Such phenomena are especially likely at low frequencies (below 50 % of the nominal), and can have obvious negative effects on the inverter's life time. Here, we approximate the problem by considering the distribution of the switching effort between the upper and the lower half of the inverter using the integer state λ. In discrete-time, the distribution of the switching effort is given by

$$\lambda(k+1) = \lambda(k) + u_a(k) + u_b(k) + u_c(k)\,. \tag{5.9}$$

It is straightforward to see that if λ becomes too positive (negative), the upper (lower) inverter half is used excessively.

In practise, additional complications arise due to the design of the inverter. Often, only one snubber circuit per stack is used (instead of three) to minimize the construction cost and size of the inverter. In this case, restrictions on the possible inverter switch transitions need to be imposed as safety constraints to avoid the inverter's destruction. Fig. 5.4 depicts the allowed switch transitions when imposing these restrictions.

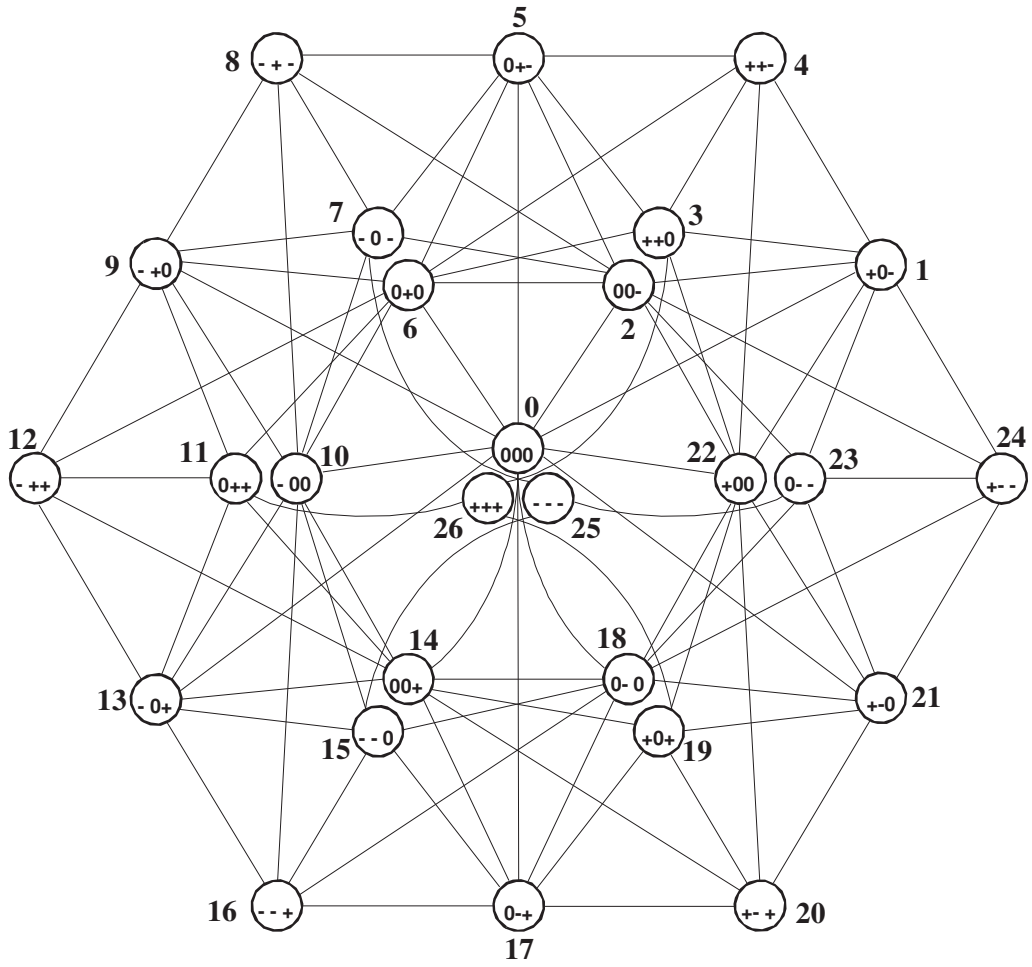


Figure 5.4: Voltage vectors on the the dq plane, the corresponding values of the integer variables (switch positions), an arbitrary numbering from 0 to 26 and the feasible switch transitions (courtesy of ABB ATDD, Switzerland)

5.3.3 Induction Motor

The dynamics of the squirrel-cage rotor induction motor are modelled in the dq0 reference frame that is rotating with the angular speed ω_{fr}. The d- and q-components of the stator and rotor flux linkages per second ψ_{ds} , ψ_{qs} , ψ_{dr} and ψ_{qr}, respectively, and the rotor's rotational speed ω_r are used as state variables. The 0-axis components are neglected, since they do not contribute to the electromagnetic torque and are decoupled from the dynamics in the d- and q-axis. The input voltages v_d and v_q are the transformation of the voltages applied to the stator into the dq0 frame. The model parameters are the base angular velocity ω_b, the stator and rotor resistances r_s and r_r, the stator, rotor and mutual inductive reactances x_{ls}, x_{lr} and x_m, respectively, the inertia constant H expressed in seconds, and the mechanical load torque T_ℓ. Note that throughout the chapters on DTC,

if not otherwise stated, we are using normalized quantities, and the rotor quantities are referred to the stator circuit. The state equations are [Kra86]

$$\frac{\mathrm{d}\psi_{ds}}{\mathrm{d}t} = -r_s\frac{x_{rr}}{D}\psi_{ds} + \omega_{fr}\psi_{qs} + r_s\frac{x_m}{D}\psi_{dr} + v_d \tag{5.10a}$$

$$\frac{\mathrm{d}\psi_{qs}}{\mathrm{d}t} = -\omega_{fr}\psi_{ds} - r_s\frac{x_{rr}}{D}\psi_{qs} + r_s\frac{x_m}{D}\psi_{qr} + v_q \tag{5.10b}$$

$$\frac{\mathrm{d}\psi_{dr}}{\mathrm{d}t} = r_r\frac{x_m}{D}\psi_{ds} - r_r\frac{x_{ss}}{D}\psi_{dr} + (\omega_{fr} - \omega_r)\psi_{qr} \tag{5.10c}$$

$$\frac{\mathrm{d}\psi_{qr}}{\mathrm{d}t} = r_r\frac{x_m}{D}\psi_{qs} - (\omega_{fr} - \omega_r)\psi_{dr} - r_r\frac{x_{ss}}{D}\psi_{qr} \tag{5.10d}$$

$$\frac{\mathrm{d}\omega_r}{\mathrm{d}t} = \frac{1}{2H\omega_b}\Big(\frac{x_m}{D}(\psi_{qs}\psi_{dr} - \psi_{qr}\psi_{ds}) - T_\ell\Big)\,, \tag{5.10e}$$

where

$$x_{ss} = x_{ls} + x_m \tag{5.11a}$$

$$x_{rr} = x_{lr} + x_m \tag{5.11b}$$

$$D = x_{ss}x_{rr} - x_m^2\,. \tag{5.11c}$$

Let ψ_s and ψ_r denote the stator and rotor flux vectors, respectively, i.e.

$$\psi_s = \begin{bmatrix}\psi_{ds} & \psi_{qs}\end{bmatrix}^T\,, \tag{5.12a}$$

$$\psi_r = \begin{bmatrix}\psi_{dr} & \psi_{qr}\end{bmatrix}^T. \tag{5.12b}$$

The electromagnetic torque

$$T_e = \frac{x_m}{D}\psi_s \times \psi_r = \frac{x_m}{D}(\psi_{qs}\psi_{dr} - \psi_{qr}\psi_{ds}) \tag{5.13}$$

and the length of the stator flux vector

$$\Psi_s = \sqrt{\psi_{ds}^2 + \psi_{qs}^2} \tag{5.14}$$

are nonlinear functions of the stator and rotor flux vectors.

The equations (5.10)–(5.14) represent the standard dynamical model of the induction motor, where the saturation of the machine's magnetic material, the changes of the rotor resistance due to the skin effect, and the temperature changes of the stator resistance are ignored. The induction motor dynamics can be classified into three groups with regard to their time constants. Depending on the load conditions, the mechanical part of the motor, in particular the rotational speed, features time constants in the range of seconds. The rotor fluxes have a typical rotor time constant of 100–200 ms. Finally, the stator flux presents the fastest dynamics, which can be manipulated by the applied stator voltage within a few μs.

5.4 Control Problem

In the following, we state the control objectives in DTC. Recalling that one of the main features of DTC is to address the control problems of the motor and of the inverter in a combined approach, the control objectives naturally include objectives regarding both the motor and the inverter.

The most significant control objective concerning the induction motor is to keep the electromechanical torque close to its reference, which is set either directly by the user or by an external speed control loop. In order to avoid the saturation or demagnetization of the motor, the amplitude of the stator flux has to be kept between certain pre-specified bounds around the reference, which are in general time-invariant. The main control objective concerning the inverter is to minimize the average switching frequency

$$\lim_{N\to\infty} \frac{1}{NT_s} \sum_{\ell=0}^{N} ||u(\ell) - u(\ell-1)|| \,, \tag{5.15}$$

where $||\cdot||$ denotes the 1-norm and T_s the sampling interval. In the case of the three-level inverter, the control objectives are extended to keeping the neutral point potential of the inverter within certain limits around zero, and to evenly distributing the switching effort between the upper and the lower half of the inverter.

The physical setup, i.e. the inverter driving the induction motor with discrete voltage vectors, makes it impossible to regulate and keep the torque and the stator flux arbitrarily close to their references with a finite switching frequency. Even at steady state, a ripple on the torque and the stator flux is unavoidable, and the inverter will constantly switch. Reducing the torque ripple can only be achieved by increasing the switching frequency and vice versa. This results in a fundamental trade-off between the amplitude of the torque ripple and the switching frequency. As the switch transitions lead to heat losses in the inverter, the maximal switching frequency is limited by the technology of the inverter and the cooling system being used. Thus, instead of trying to regulate the torque to its reference, this control objective is relaxed in DTC, and the controller rather aims at keeping the torque and the stator flux within certain bounds around their references.

Summing up, the manipulated variables are the switch positions of the inverter u, and the controlled variables are for a two-level inverter the electromechanical torque T_e and the length of the stator flux Ψ_s, while a three-level inverter adds the neutral point potential υ_n and the distribution of the switching effort λ to the vector of controlled variables. The control objectives are to keep the torque T_e, the length of the stator flux Ψ_s and the neutral point potential υ_n within their respective upper and lower bounds $T_{e,max}$ and $T_{e,min}$, $\Psi_{s,max}$ and $\Psi_{s,min}$, and $\upsilon_{n,max}$ and $\upsilon_{n,min}$, to keep the distribution of the switching effort close the zero, and to minimize the average switching frequency (5.15).

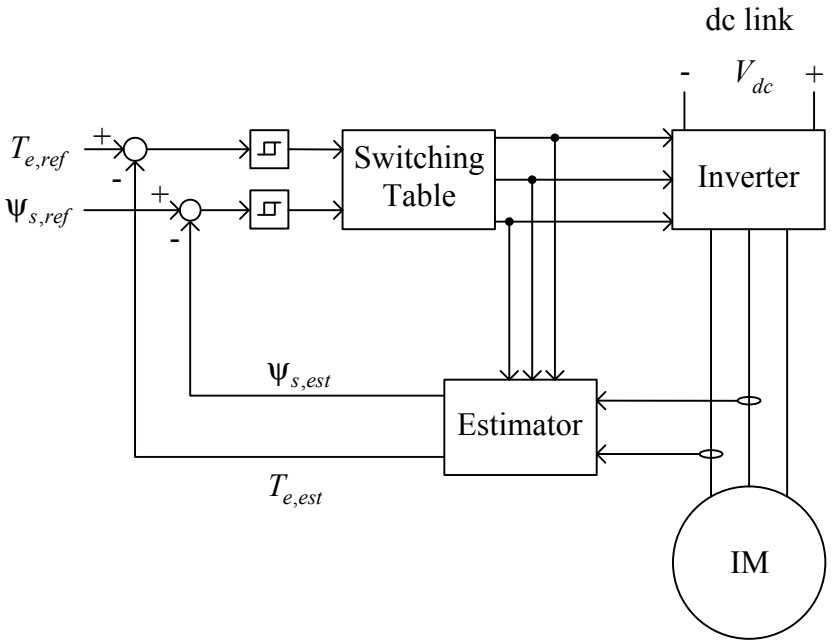


Figure 5.5: The basic scheme of classic DTC

5.5 Classic DTC

The basic configuration of classic DTC is depicted in Fig. 5.5. The only measurements acquired are the stator currents and the inverter voltages. In particular, there is no measurement at the rotor shaft of the motor. From these measurements using a model of the motor, an estimator provides the stator flux and the electromagnetic torque which are compared every 25 μs with their respective reference. The core of the DTC drive is the hysteresis control unit and the look-up table that contains the following switching logic. If the estimated torque and/or flux are outside the hysteresis bounds, a new voltage vector is selected that rapidly drives the stator flux vector to a position such that the torque and the stator flux both respect the corresponding hysteresis bounds (see Fig. 5.6). In high-power applications, where three-level GTO inverters operating at low switching frequencies (in the range of 250 Hz) are used, an extra hysteresis control unit is added in order to also keep the potential v_n of the inverter's neutral point within pre-specified bounds around zero.

As the look-up table directly sets the switch positions of the semiconductor devices in the inverter, a Pulse Width or a Space Vector Modulator is obsolete. This simplifies the control circuit, but also leads to the problem of a variable switching frequency, which is directly connected with the width of the hysteresis zones being used: Tighter bounds correspond to higher switching frequencies and vice versa. Given the assumption that

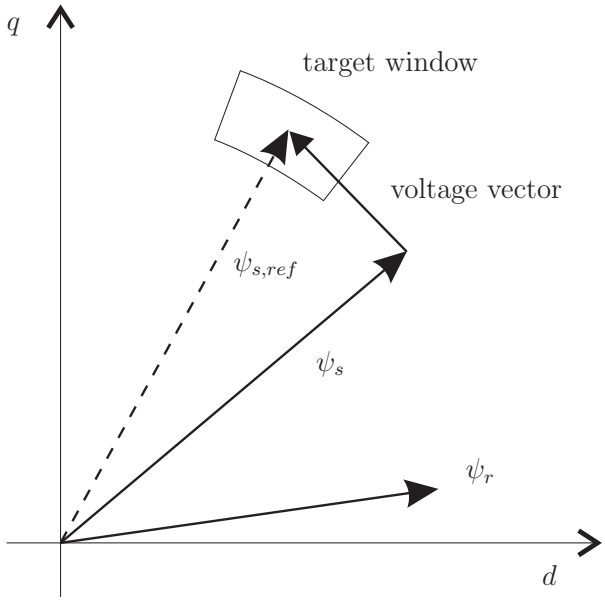


Figure 5.6: The principle of DTC. A voltage vector is chosen that positions the stator flux vector in the target window. The center of the target window is given by the torque and flux references, where the torque reference determines the angle between the stator and the rotor flux vector. The width of the window corresponds to the hysteresis bounds, where the angular width corresponds to the torque bounds and the radial width to the stator flux bounds

the average switching frequency represents a rough measure of the switching losses of the inverter, the basic trade-off between the torque and stator flux ripple and the switching frequency emerges. To avoid on the one hand overheating of the inverter due to a too high switching frequency, and on the other hand unnecessary high ripples on torque and flux due to a too low switching frequency, an outer control loop is often needed that keeps the switching frequency constant over the whole range of operation by setting the hysteresis bounds accordingly.

5.6 Review of Academic State of the Art

Applying model-based and predictive control strategies to the control problem of motor drives is not an unknown notion in the power electronics research community. As shown in [KL00], a number of already existing strategies might be – loosely speaking – classified by their nature as predictive control strategies, the most prominent among them being DTC itself, where the voltage vectors are classified into vectors that increase, decrease or keep the torque or flux (approximately) constant. However, DTC lacks a model for prediction and the notion of a cost function. Such elements are present in more recent approaches,

which, to the best of our knowledge, are limited to the ones presented in [KLL01], [RPS+04] and [RLSLM04]. These schemes differ in several significant aspects from the MPC schemes proposed in this thesis.

- The prediction horizon is set to one[2]. There is no mentioning of a possible use of a larger (control) horizon combined with a receding horizon policy.
- The DTC problem is formulated as a reference tracking problem, namely the formulated control problem tries to minimize the deviation of the controlled variables from their reference. There are no hysteresis bounds on the controlled variables.
- The cost function does not emulate the switching frequency of the inverter. In combination with reference tracking and the limited number of voltage vectors, an extremely high switching frequency is to be expected, which is only upper bounded by the inverse of the sampling interval[3]. For a high power application, which we are considering in this thesis, this is not acceptable.
- Only two-level inverters are considered.
- Linear (or locally linearized) models are used as prediction models. In particular for three level inverters including the nonlinearities of the neutral point potential, linear models are rather inaccurate.

Moreover, the work presented in [KLL01] focuses not on DTC, but on field oriented control, which is a different control method for induction motors. In the latter, the manipulated variables of the control problem are continuous and not discrete valued. This simplifies the underlying optimization problem but necessitates the use of a Pulse Width or a Space Vector Modulator for the implementation of the method. The employed Generalized Predictive Control (GPC) [CMT87] framework seems to us to be of little appeal as the scheme focuses on unconstrained linear systems in an input output representation, making it – in contrast to MPC – hard to extend it to PWA or nonlinear models, to incorporate constraints and to straightforwardly tackle systems with multiple in- and outputs.

The authors of [RLSLM04] suggest a geometric control approach that selects the voltage vector closest to the direction into which the stator flux is required to be driven. Furthermore, the fraction of time the chosen voltage vector is to be applied for is determined a priori leading to an open-loop rather than a receding horizon concept. To limit the impact of the nonlinearities on the predictions, this fraction of time is upper bounded. Apart

[2] In [KLL01], the control horizon is not explicitly provided. Yet, the paper indicates that it is equal to one, while the prediction horizon is larger than one.

[3] Strictly speaking by $\frac{2}{T_s}$.

from that, a synchronous machine is considered, and the DC-DC converter manipulating the field voltage is included in the control problem thus allowing for an additional degree of freedom.

The approach proposed in [RPS+04] is rather formal and apart from the aforementioned issues closest to our MPC schemes.

5.7 Summary of Research Contributions

The goals of our research project on DTC are threefold.

- Replace the classic design procedure of the switching table by a systematic controller design that is easily applicable to drives with different inverter topologies (i.e. with different number of levels) and arbitrary machine ratings thus reducing the commissioning time of drives and making time-consuming tuning of parameters obsolete.

- Improve the performance of DTC drives, namely reduce the switching frequency while keeping the ripples on the controlled variables constant, or vice versa. A reduction in the switching frequency translates into a reduction of the losses, and in particular for high power applications, into energy (and money) savings. On the other hand, smaller ripples are obviously beneficial for a large number of applications. Furthermore, they also lead to a reduction of the total harmonic distortion of the stator current of the machine hence reducing the losses of the machine.

- Derive control schemes that feature a computational complexity that allows for an implementation today or within the near future.

To achieve these goals, we propose controllers that are based on the principles of MPC. Here, the current control input is obtained by solving at each sampling instant an open-loop optimal control problem over a finite horizon using the current state of the plant as the initial state. The underlying optimization procedure yields an optimal control sequence that minimizes a given objective function. By only applying the first control input in this sequence and by recomputing the control sequence at the next sampling instant, a receding horizon policy is achieved. A major advantage of MPC is its ability to cope with hard constraints on manipulated variables, states and outputs. Furthermore, as introduced in [BM99a], the MLD framework can be straightforwardly embedded in MPC allowing one to use hybrid models given in the MLD form as prediction models for MPC. Moreover, there is a rich theory to pre-solve the optimization problem off-line by computing the (explicit) state-feedback control law using multi-parametric and dynamic programming [Bor03].

These MPC schemes share the following features that distinguishes them from the control approaches summarized in the previous section.

- A model-based predictive control problem is set up and solved over a prediction horizon larger than one.
- A receding horizon strategy is used.
- The classic DTC objectives are considered, namely to keep the controlled variables within given hysteresis bounds while minimizing the average switching frequency.
- A cost function emulating the switching frequency is used.
- All controlled variables are considered when deciding on the next control input. Specifically, the control input is evaluated not only in terms of the controlled variable that necessitated the switching (as currently done in classic DTC), but by taking into account the predicted behavior of all controlled variables simultaneously.
- Drives with three-level inverters are tackled. In particular, we consider the balancing of the neutral point potential.
- The DTC drive is modelled as a hybrid system. Specifically, we use integer variables to represent the inverter switch positions, and derive PWA or even general nonlinear models, rather than models linearized around an operating point.
- The control methods are directly extendable to other inverter topologies and additional control objectives.

Chapter 7 aims at deriving MPC schemes that are conceptually and computationally simple yet yield a significant performance improvement with respect to the state of the art. More specifically, the term *conceptually simple* refers to controllers allowing for straightforward tuning of the controller parameters or even a lack of such parameters, and easy adaptation to different physical setups and drives, whereas *computationally simple* implies that the control scheme does not require excessive computational power to allow the implementation on DTC hardware that is currently available or at least will be so within a few years. Nevertheless, as the control schemes are model-based, the computations are expected to be significantly more demanding than the simple evaluation of the currently used look-up table.

To make our MPC schemes applicable, several techniques and methods had to be developed aiming at tailoring the principles of MPC to the DTC problem in order to derive conceptually and computationally simple control algorithms. These techniques include:

- Priority levels in the cost function.

- Time-varying penalties (leading to the so called *Late Switching Strategy*).
- Models with different sampling intervals within the prediction horizon (*Multiple-Rate Prediction Model Approach*).
- Restrictions on the degrees of freedom for switching.
- Open-loop computation of the number of steps a voltage vector keeps the controlled variables within their bounds.
- An expression in the cost function approximating the average switching frequency by dividing the number of switch transitions within the horizon by either the length of the prediction horizon or the number of steps a voltage vector keeps the controlled variables within their bounds.
- Linear and quadratic extrapolation of the trajectories of the controlled variables to allow for very long prediction horizons while keeping the computational burden low.
- The notion of the feasible switching path in the presence of constraints on the allowed switch transitions.

As for any model-based control scheme, modelling is a fundamental task before solving the control problem. This is done in Chapter 6, where we exploit a number of physical properties of DTC drives to derive discrete-time models of DTC drives with two- or three-level inverters tailored to our needs, more specifically, models that are of low complexity yet of sufficient accuracy to serve as prediction model for the model-based control schemes. These properties are the slow rotor flux and speed dynamics, the symmetry of the voltage vectors, and the invariance of the motor outputs under flux rotation. The low-complexity models are derived by assuming constant speed within the prediction horizon, mapping the states (the fluxes) into a 60 degree sector, and aligning the rotor flux vector with the d-axis of a reference frame rotating with the rotational speed of the rotor. The benefits of doing this are a reduction of the number of states from five to three, and a highly reduced domain on which the nonlinear functions need to be approximated by PWA functions. The resulting model can be easily described in HYSDEL. Subsequently, the HYSDEL compiler derives the MLD model and the mode enumeration algorithm (see Section 3) yields an equivalent PWA model of the DTC drive.

6

Discrete-Time Modelling of DTC Drives

The purpose of this chapter is to derive discrete-time hybrid models of the DTC drive that are suitable to serve as prediction models for the MPC schemes presented in Chapter 7. In particular, these models need to predict the evolution of the electromagnetic torque, the stator flux, and – in the case of a three-level inverter – also of the inverter neutral point potential and the distribution of the switching losses over several control cycles in an open-loop fashion with sufficient accuracy. However, the models should be of low complexity to reduce the computational burden of the underlying optimization problems.

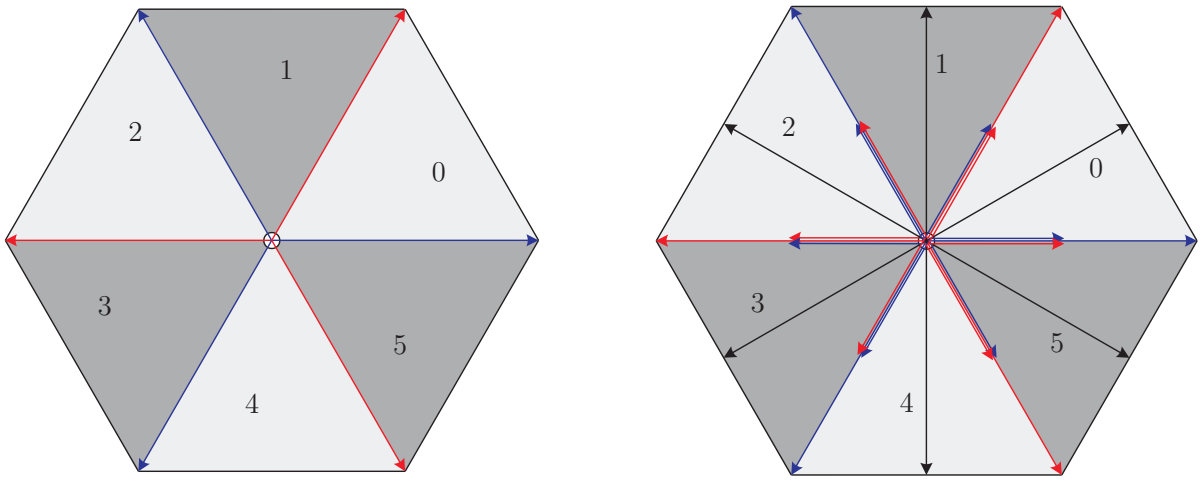
6.1 Physical Properties of DTC Drives

DTC drives feature the following three important properties that will allow us in a subsequent step to obtain low-complexity models of the drives.

Slow Rotor Flux and Speed Dynamics As mentioned in Section 5.3.3, a basic characteristic of induction machines is that the stator flux dynamics are significantly faster than the dynamics of the rotor flux and the rotational speed. Thus, the application of a certain voltage vector to the machine terminals has an immediate effect only on the stator flux, turning it rapidly to the position required by the torque demand, while the rotor speed ω_r and the length of the rotor flux vector remain constant during several control cycles.

Symmetry of Voltage Vectors The voltage vectors that can be produced by both a two- and a three-level inverter exhibit strong symmetrical properties in the dq plane[1]. As

[1] Inverters with more than three levels have similar symmetrical properties making the hereafter shown modelling approach extendable to such inverters.



(a) Two-level inverter (b) Three-level inverter

Figure 6.1: Symmetrical properties of the voltage vectors produced by a two- and a three-level inverter

shown in Fig. 6.1, a certain pattern is repeated with an angle spread of $\frac{\pi}{3}$. Defining such a pattern as a sector leads to the formation of six sectors. In the dq plane, rotating any sector by $\frac{\pi}{3}$ yields the voltage vectors of the neighboring sector, while the zero components of vectors of neighboring sectors have opposite sign. More formally, the voltage vectors of the μ sector can be used to produce the vectors of the ν sector through

$$u_{dq0}^{(\nu)} = \Pi^{\mu-\nu}\, u_{dq0}^{(\mu)} \quad \forall \mu, \nu \in \mathbb{N}_0 \tag{6.1}$$

using the matrix

$$\Pi = \begin{bmatrix} \cos\frac{\pi}{3} & \sin\frac{\pi}{3} & 0 \\ -\sin\frac{\pi}{3} & \cos\frac{\pi}{3} & 0 \\ 0 & 0 & -1 \end{bmatrix}. \tag{6.2}$$

This relationship can be also applied to the switch positions u_{abc} given in the three-phase system (abc) by transforming the quantities to the dq0 reference frame, rotating them there from the μ to the ν sector, and transforming them subsequently back.

$$u_{abc}^{(\nu)} = P^{-1}(\varphi)\ \Pi^{\mu-\nu}\ P(\varphi)\ u_{abc}^{(\mu)} \quad \forall \varphi \in \mathbb{R} \tag{6.3}$$

For the componentwise absolute voltage vectors, however, the relationship is slightly more complicated and non-trivial. It can be shown that a rotation of a voltage vector by an even number of sectors preserves its componentwise absolute values, while a rotation by an odd number, alternates the sign. Taking this into account, the following relation results

$$|u_{abc}^{(\nu)}| = (-1)^{\mu-\nu}\ P^{-1}(\varphi)\ \Pi^{\mu-\nu}\ P(\varphi)\ |u_{abc}^{(\mu)}| \quad \forall \varphi \in \mathbb{R}\,. \tag{6.4}$$

Invariance of Motor Outputs under Flux Rotation The third characteristic is that the electromagnetic torque depends only on the relative (and not the absolute) position of the stator and rotor flux vectors, as the torque is the external product of these two vectors. Trivially, the same also holds for the length of the stator flux. Hence, the output variables of the motor, namely the electromagnetic torque and the length of the stator flux vector are invariant under flux rotations.

6.2 Nonlinear Model of DTC Drive with Three-Level Inverter

In this section, we derive a discrete-time nonlinear model of the DTC drive with a three-level inverter that will be used as prediction model for the MPC scheme based on extrapolation presented in Section 7.4. In particular, this model needs to predict the evolution of the electromagnetic torque, stator flux and inverter neutral point potential over several sampling intervals in an open-loop fashion, while the distribution of the switching effort is not considered.

Exploiting the fact that the time-constant of the rotational speed dynamic exceeds the length of the prediction interval by several orders of magnitude, the speed dynamic can be neglected and the speed is assumed to remain constant within the prediction horizon. This allows us to treat the speed as a model parameter rather than as a state thus removing (5.10e) from the motor model. Using a stationary dq0 reference frame with $\omega_{fr} = 0$ and the transformation matrix (5.3) leads to the following motor model

$$\frac{\mathrm{d}\psi_{ds}}{\mathrm{d}t} = -r_s\frac{x_{rr}}{D}\psi_{ds} + r_s\frac{x_m}{D}\psi_{dr} + v_d \tag{6.5a}$$

$$\frac{\mathrm{d}\psi_{qs}}{\mathrm{d}t} = -r_s\frac{x_{rr}}{D}\psi_{qs} + r_s\frac{x_m}{D}\psi_{qr} + v_q \tag{6.5b}$$

$$\frac{\mathrm{d}\psi_{dr}}{\mathrm{d}t} = r_r\frac{x_m}{D}\psi_{ds} - r_r\frac{x_{ss}}{D}\psi_{dr} - \omega_r\psi_{qr} \tag{6.5c}$$

$$\frac{\mathrm{d}\psi_{qr}}{\mathrm{d}t} = r_r\frac{x_m}{D}\psi_{qs} + \omega_r\psi_{dr} - r_r\frac{x_{ss}}{D}\psi_{qr} \tag{6.5d}$$

The model of the inverter has one state, namely the neutral point potential whose dynamic is described by (5.8) as a function of $|u_{abc}|$ and i_{dq0}. The d- and q-components of the stator current i_{dq0} are linear combinations of the d- and q-components of the stator and rotor flux vectors (see e.g. [Kra86] for details), and the 0-component is always zero[2].

$$i_{dq0} = \begin{bmatrix} \frac{x_{rr}}{D}\psi_s^T - \frac{x_m}{D}\psi_r^T & 0 \end{bmatrix}^T. \tag{6.6}$$

[2]This follows from (5.3), taking into account that $i_a + i_b + i_c = 0$.

Next, we define the overall state vector of the DTC drive as

$$x = \begin{bmatrix} \psi_{ds} & \psi_{qs} & \psi_{dr} & \psi_{qr} & \upsilon_n \end{bmatrix}^T , \tag{6.7}$$

the integer variables u_a, u_b and u_c as the input vector

$$u = \begin{bmatrix} u_a & u_b & u_c \end{bmatrix}^T \in \{-1, 0, 1\}^3 , \tag{6.8}$$

and the electromagnetic torque (5.13), the length of the stator flux (5.14) and the neutral point potential as the output vector

$$y = \begin{bmatrix} T_e & \Psi_s & \upsilon_n \end{bmatrix}^T . \tag{6.9}$$

Combining the motor model (6.5), (5.13) and (5.14) with the model of the inverter (5.8) and (6.6), and setting $u = u_{abc}$, the continuous-time model of the DTC drive is

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = \begin{bmatrix} A & 0 \\ 0 & 0 \end{bmatrix} x(t) + \begin{bmatrix} B_1 \\ 0 \end{bmatrix} u(t) + \begin{bmatrix} 0 \\ B_2(x(t)) \end{bmatrix} |u(t)| \tag{6.10a}$$

$$y(t) = g(x(t)) , \tag{6.10b}$$

with

$$A = \begin{bmatrix} -r_s \frac{x_{rr}}{D} & 0 & r_s \frac{x_m}{D} & 0 \\ 0 & -r_s \frac{x_{rr}}{D} & 0 & r_s \frac{x_m}{D} \\ r_r \frac{x_m}{D} & 0 & -r_r \frac{x_{ss}}{D} & -\omega_r \\ 0 & r_r \frac{x_m}{D} & \omega_r & -r_r \frac{x_{ss}}{D} \end{bmatrix} , \tag{6.11}$$

$$B_1 = \frac{V_{dc}}{2} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} P , \qquad B_2(x(t)) = x^T(t) \frac{1}{2x_c} \begin{bmatrix} \frac{x_{rr}}{D} & 0 & 0 \\ 0 & \frac{x_{rr}}{D} & 0 \\ -\frac{x_m}{D} & 0 & 0 \\ 0 & -\frac{x_m}{D} & 0 \\ 0 & 0 & 0 \end{bmatrix} P^{-T} , \tag{6.12}$$

and

$$g(x(t)) = \begin{bmatrix} \frac{x_m}{D}(x_2(t)x_3(t) - x_4(t)x_1(t)) \\ \sqrt{x_1^2(t) + x_2^2(t)} \\ x_5(t) \end{bmatrix} , \tag{6.13}$$

where x_i denotes the i-th component of the vector x. Note that the model has no feed-through, i.e. $y(t)$ is independent of $u(t)$, the zeros in (6.10a) are vectors and matrices of appropriate dimension, and the nonlinear expression $B_2(x(t))|u(t)|$ captures the dynamic of the neutral point potential.

In a last step, the continuous-time model (6.10) is replaced by its discrete-time representation using forward Euler with a sampling time of $T_s = 25\,\mu$s. For the sake of completeness, this leads to

$$x(k+1) = (I + \begin{bmatrix} A & 0 \\ 0 & 0 \end{bmatrix} T_s)x(k) + \begin{bmatrix} B_1 \\ 0 \end{bmatrix} T_s u(k) + \begin{bmatrix} 0 \\ B_2(x(k)) \end{bmatrix} T_s|u(k)| \quad (6.14a)$$

$$y(k) = g(x(k))\,. \quad (6.14b)$$

This model is very accurate, in particular, piecewise affine (PWA) approximations of nonlinearities are avoided. Furthermore, the model parameters can be easily adapted and updated on-line. However, both the state-update and the output function are nonlinear.

6.3 Low Complexity Modelling

In this section, we derive discrete-time nonlinear models of the induction motor and the three-level inverter which are of low complexity and will serve as key modules to derive the MLD and PWA models in the following sections. For this, we will take advantage of all physical properties of the DTC drive summarized in Section 6.1.

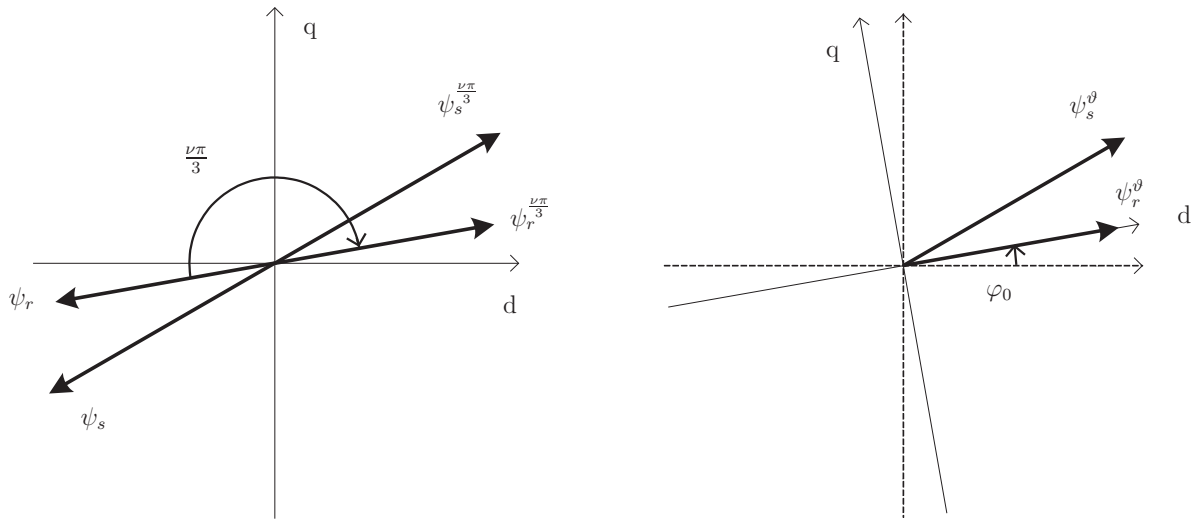
6.3.1 Stator Flux Dynamics

As the low-complexity model of the induction motor focuses on the fast stator flux dynamics we will refer to it as the *Stator Flux Dynamics Model*.

Rotation and State Reduction

Exploiting the symmetrical properties of the voltage vectors and the invariance of the motor outputs under flux rotation, allows us to map the fluxes into the zero sector, to solve the control problem in this sector, and to subsequently rotate the result back into the original sector yielding the voltage vector to be actually applied to the motor terminals. By restricting the fluxes to the zero sector one reduces the domain on which the nonlinear functions (like the stator flux or the neutral point potential) are defined. Later, when approximating these nonlinearities by piecewise affine (PWA) functions, a reduced domain allows for less complex approximations with less polyhedra.

Furthermore, by using an additional rotation that aligns the rotor flux with the d-axis of a rotating reference frame and by exploiting the slow dynamics of the rotor flux, the number of states can be reduced. To maintain the alignment for several sampling intervals, we choose a reference frame that is rotating synchronously with the rotor flux. Strictly speaking, such a reference frame would be rotating with the speed of the magnetic field



(a) First stage: A clockwise rotation with an angle equal to an integer multiple of $\frac{\pi}{3}$ maps the fluxes into the zero sector

(b) Second stage: An anti-clockwise rotation of the dq0 frame by a continuous angle φ_0 aligns the rotor flux vector with the d-axis of the frame

Figure 6.2: Rotation of the fluxes

ω_e. Another possible choice would be the speed of the rotor ω_r, where $\omega_e > \omega_r$ in motor operation. The normalized difference of the two speeds is referred to as the slip given by $(\omega_e - \omega_r)/\omega_e$. Here, any of the two speeds can be chosen without leading to noticeable differences. This is due to the following. Firstly, due to the short prediction horizon, we require the model to be accurate only for a small number of sampling intervals, say two or three. During this short time, the difference between the two speeds leads only to a negligible difference in the angles of the two frames. Furthermore, as we are considering in this thesis mainly the case of large power induction machines, the slip amounts only to a few per cent. We conclude that depending on the available measurements or estimates, one can choose any of the two speeds without introducing hardly any error. In the following, we will use ω_r.

The mapping and the subsequent alignment are shown in Fig. 6.2 as a procedure carried out in two stages. The first stage maps the problem into the zero sector by rotating the flux vectors clockwise by an integer multiple of $\frac{\pi}{3}$, whereas the second stage is an anti-clockwise rotation of the reference frame by an angle $\varphi_0 \in [0, \frac{\pi}{3}]$, that aligns the rotor flux vector with the d-axis of the reference frame. Assume that the rotor flux initially lies in the ν sector. Rotating the fluxes by the angle

$$\vartheta = \frac{\nu\pi}{3} + \varphi_0 \tag{6.15}$$

using the rotation matrix

$$R(\vartheta) = \begin{bmatrix} \cos\vartheta & \sin\vartheta \\ -\sin\vartheta & \cos\vartheta \end{bmatrix} \tag{6.16}$$

yields the rotated flux vectors

$$\psi_s^\vartheta = R(\vartheta)\psi_s, \tag{6.17a}$$
$$\psi_r^\vartheta = R(\vartheta)\psi_r, \tag{6.17b}$$

which are defined according to (5.12)

$$\psi_s^\vartheta = \begin{bmatrix} \psi_{ds}^\vartheta & \psi_{qs}^\vartheta \end{bmatrix}^T, \tag{6.18a}$$
$$\psi_r^\vartheta = \begin{bmatrix} \psi_{dr}^\vartheta & \psi_{qr}^\vartheta \end{bmatrix}^T. \tag{6.18b}$$

Since the rotational speed dynamic is several orders of magnitude slower than the stator flux dynamics, we remove the speed dynamic from the motor model, assume that ω_r is constant within several sampling intervals and consider ω_r as a parameter. Next, we align the rotor flux with the d-axis of the rotating reference frame, which rotates with the speed ω_r as discussed above. Recalling the slow dynamic of the rotor flux vector, we may then assume that ψ_{dr}^ϑ remains approximately constant and ψ_{qr}^ϑ is equal to zero during several sampling intervals.

State-Space Representation

This allows us to model the fast stator flux dynamics using only the first two state equations of the induction motor model, namely (5.10a) and (5.10b), and to regard ψ_{dr}^ϑ and ω_r as parameters. These two equations are repeated here for ease of readability

$$\frac{\mathrm{d}\psi_{ds}^\vartheta}{\mathrm{d}t} = -r_s\frac{x_{rr}}{D}\psi_{ds}^\vartheta + \omega_r\psi_{qs}^\vartheta + r_s\frac{x_m}{D}\psi_{dr}^\vartheta + v_d^{(0)} \tag{6.19a}$$
$$\frac{\mathrm{d}\psi_{qs}^\vartheta}{\mathrm{d}t} = -\omega_r\psi_{ds}^\vartheta - r_s\frac{x_{rr}}{D}\psi_{qs}^\vartheta + r_s\frac{x_m}{D}\psi_{qr}^\vartheta + v_q^{(0)}\,, \tag{6.19b}$$

where the superscript (0) on the d- and q-components of the voltage vector denotes the fact that it refers to the zero sector, as the problem has been mapped from the ν to the zero sector.

Defining the matrices

$$F_m = \begin{bmatrix} -r_s\frac{x_{rr}}{D} & 0 \\ 0 & -r_s\frac{x_{rr}}{D} \end{bmatrix}, \tag{6.20a}$$
$$F_r = \begin{bmatrix} 0 & \omega_r \\ -\omega_r & 0 \end{bmatrix}, \tag{6.20b}$$

and recalling (5.4), the set of affine state equations (6.19) is described in vector form as

$$\frac{\mathrm{d}\psi_s^\vartheta(t)}{\mathrm{d}t} = (F_m + F_r)\psi_s^\vartheta(t) + r_s\frac{x_m}{D}\psi_r^\vartheta + \frac{V_{dc}}{2}\tilde{P}(\varphi(t))u_{abc}^{(0)}(t)\,. \tag{6.21}$$

Note that the matrix $\tilde{P}(\varphi(t))$ performing the transformation of the inverter voltages into the rotating dq0 frame is time-varying. In particular, it depends on the angle $\varphi(t)$ that captures the evolution of the rotating reference frame with

$$\frac{\mathrm{d}\varphi(t)}{\mathrm{d}t} = \omega_r T_s \tag{6.22}$$

starting from the initial value

$$\varphi(0) = \varphi_0 \tag{6.23}$$

given by the second stage of the rotation.

The discrete-time linear model of the induction machine's stator flux dynamics is obtained by integrating (6.21) from $t = kT_s$ to $t = (k+1)T_s$

$$\begin{aligned}\psi_s^{\vartheta}((k+1)T_s) = e^{(F_m+F_r)T_s}\,\psi_s^{\vartheta}(kT_s) + r_s\frac{x_m}{D}\int_0^{T_s} e^{(F_m+F_r)(T_s-\tau)}\,\mathrm{d}\tau\,\psi_r^{\vartheta} \\ + \frac{V_{dc}}{2}\int_0^{T_s} e^{(F_m+F_r)(T_s-\tau)}\,\tilde{P}(\varphi(kT_s+\tau))\,u_{abc}^{(0)}(kT_s+\tau)\,\mathrm{d}\tau\,.\end{aligned} \tag{6.24}$$

In the following, we focus on the third expression on the right-hand side of (6.24), which can be simplified using the following three relations. Firstly, as the componentwise product between F_m and F_r is zero, we have

$$e^{(F_m+F_r)(T_s-\tau)} = e^{F_m(T_s-\tau)}e^{F_r(T_s-\tau)}\,. \tag{6.25}$$

Secondly, since $\varphi(kT_s+\tau) = \varphi(k) + \omega_r\tau$, the relation

$$e^{-F_r\tau}\,\tilde{P}(\varphi(kT_s+\tau)) = \tilde{P}(\varphi(kT_s)) \tag{6.26}$$

can be shown to hold. Moreover, it is straightforward to derive

$$e^{F_rT_s}\,\tilde{P}(\varphi(kT_s)) = \tilde{P}(\varphi(kT_s)+\omega_rT_s)\,. \tag{6.27}$$

Using (6.25), (6.26) and (6.27), and recalling that the voltage vector applied to the motor terminals remains constant within one sampling interval we can rewrite the third expression in (6.24) and obtain the simplified discrete-time linear model of the stator flux dynamics

$$\begin{aligned}\psi_s^{\vartheta}(k+1) = e^{(F_m+F_r)T_s}\,\psi_s^{\vartheta}(k) + r_s\frac{x_m}{D}\int_0^{T_s} e^{(F_m+F_r)(T_s-\tau)}\,\mathrm{d}\tau\,\psi_r^{\vartheta} \\ + \frac{V_{dc}}{2}\int_0^{T_s} e^{F_m(T_s-\tau)}\,\mathrm{d}\tau\,\tilde{P}(\varphi(k)+\omega_rT_s)\,u_{abc}^{(0)}(k)\,,\end{aligned} \tag{6.28}$$

where we have used k rather than kT_s to denote the sampling instants. Last, the discrete-time representation of (6.22) is simply given by

$$\varphi(k+1) = \varphi(k) + \omega_rT_s \tag{6.29}$$

with the initial value as in (6.23)[3].

Rather than introducing $\varphi(k)$ as a state, we choose $\cos(\varphi(k))$ as a third state, as this proves to be beneficial in terms of the model complexity as detailed at the end of this section. Rewriting (6.29), the corresponding state equation is

$$\alpha(k+1) = \cos(\omega_r T_s)\,\alpha(k) - \sin(\omega_r T_s)\,\beta(k)\,, \tag{6.30}$$

where

$$\alpha(k) = \cos(\varphi(k)), \tag{6.31a}$$
$$\beta(k) = \sin(\varphi(k))\,. \tag{6.31b}$$

Hence, the three-dimensional state vector of the induction motor amounts to

$$x_m(k) = \begin{bmatrix} \psi_{ds}^{\vartheta}(k) & \psi_{qs}^{\vartheta}(k) & \alpha(k) \end{bmatrix}^T . \tag{6.32}$$

The two outputs of the model are the electromagnetic torque

$$T_e(k) = \frac{x_m}{D}\,\psi_{dr}^{\vartheta}\,x_{m2}(k)\,, \tag{6.33}$$

which is a linear expression of the second state, and the length of the stator flux vector

$$\Psi_s(k) = \sqrt{x_{m1}^2(k) + x_{m2}^2(k)}\,, \tag{6.34}$$

which is nonlinear in the first two states.

Summing up, the following steps have been performed to reduce the complexity of the motor model and to derive the Stator Flux Dynamics Model.

- Regarding the rotor speed as a parameter reduced the number of states in the motor model from five to four.
- Mapping the fluxes into the zero sector reduced the domain on which the nonlinear functions (like the stator flux or the neutral point potential) are defined. Later, when approximating these nonlinearities by PWA functions, a reduced domain allows for less complex approximations (with less polyhedra) while maintaining the accuracy of the approximation.
- Aligning the rotor flux vector with the d-axis of the rotating reference frame and regarding the length of the rotor flux as a parameter allowed us to remove the two states of the rotor flux dynamics. Yet, one additional state was needed to account for the rotation of the reference frame. Hence in total, one state was spared. Furthermore, this turned the nonlinear torque expression into a linear one.

[3]In then actual implementation of the model, we replace $\tilde{P}(\varphi(k) + \omega_r T_s)$ in (6.28) by $\tilde{P}(\varphi(k))$ and $\varphi(0) = \varphi_0$ in (6.23) by $\varphi(0) = \varphi_0 + \omega_r T_s$.

6.3.2 Three-Level Inverter

In contrast to the two-level inverter that lacks the neutral point potential and the issue of properly distributing the switching losses, those aspects must be addressed by the controller when using a three-level inverter and thus they need to be modelled appropriately. In particular, the rotation of the fluxes has to be taken into account.

Neutral Point Potential

In the first part of this section, our aim is to derive a discrete-time representation of the neutral point potential's dynamic that is a function solely of the variables in the zero sector, like the (componentwise absolute values of the) inverter switch positions in the zero sector and the rotated fluxes. For ease of readability, the continuous-time state equation of the neutral point potential (5.8) is repeated here, which holds for the original ν sector.

$$\frac{\mathrm{d}\upsilon_n(t)}{\mathrm{d}t} = \frac{1}{2x_c}|u_{abc}^{(\nu)}(t)|^T\, i_{abc}^{(\nu)}(t) \tag{6.35}$$

Firstly, expanding this expression with $P^{-1}(\vartheta(t))\,P(\vartheta(t))$ and introducing $i_{dq0}^{\vartheta}(t)$ as the rotated stator current, the above equation is rewritten as

$$\frac{\mathrm{d}\upsilon_n(t)}{\mathrm{d}t} = \frac{1}{2x_c}\,|u_{abc}^{(\nu)}(t)|^T\,P^{-1}(\vartheta(t))\,i_{dq0}^{\vartheta}(t)\,, \tag{6.36}$$

where we have used $i_{dq0}^{\vartheta}(t) = P(\vartheta(t))\,i_{abc}^{(\nu)}(t)$, which holds by definition. Secondly, setting $\mu = 0$ in (6.4) and employing $(\Pi^{-\nu})^T = \Pi^{\nu}$ yields the relation

$$|u_{abc}^{(\nu)}(t)|^T = (-1)^{\nu}\;|u_{abc}^{(0)}(t)|^T\;P^T(\varphi(t))\;\Pi^{\nu}\;(P^{-1}(\varphi(t)))^T\,. \tag{6.37}$$

Thirdly, using trigonometric operations, one can show that

$$P^T(\varphi(t))\,\Pi^{\nu}\,(P^{-1}(\varphi(t)))^T\,P^{-1}(\vartheta(t)) = P^{-1}(\varphi(t))\,, \tag{6.38}$$

where we have used $\vartheta(t) = \frac{\nu\pi}{3} + \varphi(t)$, which generalizes (6.15). Finally, combining (6.36), (6.37) and (6.38) leads to the simple expression

$$\frac{\mathrm{d}\upsilon_n(t)}{\mathrm{d}t} = (-1)^{\nu}\frac{1}{2x_c}\,|u_{abc}^{(0)}(t)|^T\,P^{-1}(\varphi(t))\,i_{dq0}^{\vartheta}(t)\,. \tag{6.39}$$

Now, it is straightforward to obtain the following discrete-time representation of the neutral point potential

$$\upsilon_n(k+1) = \upsilon_n(k) + (-1)^{\nu}\frac{T_s}{2x_c}\,|u_{abc}^{(0)}(k)|^T\,P^{-1}(\varphi(k))\,i_{dq0}^{\vartheta}(k)\,. \tag{6.40}$$

Recall that the stator currents in the dq0 frame are linear functions of the d- and q-components of the flux vectors. Rewriting (6.6) using the rotated flux vectors, we obtain

$$i^{\vartheta}_{dq0}(k) = \left[\frac{x_{rr}}{D}(\psi^{\vartheta}_s(k))^T - \frac{x_m}{D}(\psi^{\vartheta}_r(k))^T \quad 0\right]^T . \tag{6.41}$$

As a result, the neutral point potential only depends on the rotated flux vectors and the componentwise absolute values of the inverter switch positions in the zero sector, in which the control problem will be solved.

Distribution of Switching Losses

In the second part of this section, we derive a discrete-time representation of the distribution of the switching losses that is – similarly to the neutral point potential – only a function of the inverter switch positions in the zero sector. We start by recalling the discrete-time state equation of the switching losses (5.9) given in the ν sector

$$\lambda(k+1) = \lambda(k) + \mathbb{1}^T u^{(\nu)}_{abc}(k)\,, \tag{6.42}$$

where $\mathbb{1}$ denotes a column vector of ones of appropriate length. Setting $\mu = 0$ in (6.3), basic trigonometric manipulations can be used to show that the relation

$$\mathbb{1}^T u^{(\nu)}_{abc} = (-1)^{\nu}\mathbb{1}^T u^{(0)}_{abc} \tag{6.43}$$

holds that directly leads to the rewritten dynamic of the switching losses

$$\lambda(k+1) = \lambda(k) + (-1)^{\nu}\mathbb{1}^T u^{(0)}_{abc}\,. \tag{6.44}$$

6.4 PWA Approximation of Nonlinearities

6.4.1 Bounds on State Space

The Stator Flux Dynamics Model (and the three-level inverter) contain a number of nonlinearities, which must be approximated in a subsequent step by PWA functions to enable us to cast the model into the MLD framework. As all these nonlinearities are only functions of the motor state vector x_m (and not of the input vector), it is sufficient to derive the set of states $\mathcal{X}^0 \in \mathbb{R}^3$ that includes all motor states during the whole range of operation, while it is tight (not conservative).

The set $\mathcal{X}^0$ can be easily determined by translating the output hysteresis bounds imposed by the control objectives into constraints on the motor state-space. Recalling that the torque is a linear expression of the second motor state, the lower and upper bounds on

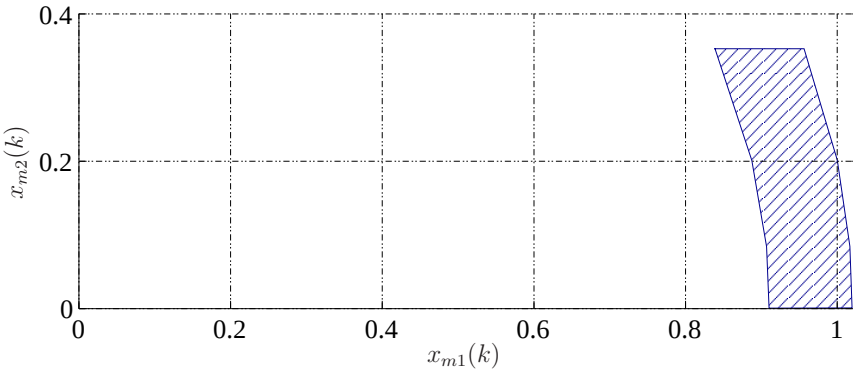


Figure 6.3: Output constraints on the torque and flux reformulated as state constraints in the $x_{m1}x_{m2}$ plane for the parameters as in Case Study I (see Table 7.13) and $\psi_{dr}^{\vartheta} = 0.89$

the electromagnetic torque $T_{e,min}$ and $T_{e,max}$ can be directly translated into linear bounds on $x_{m2}(k)$

$$\frac{D}{x_m \psi_{dr}^{\vartheta}} T_{e,min} \leq \; x_{m2}(k) \; \leq \frac{D}{x_m \psi_{dr}^{\vartheta}} T_{e,max} \,. \tag{6.45}$$

Similarly for the stator flux, its lower and upper bounds $\Psi_{s,min}$ and $\Psi_{s,max}$ turn into the quadratic state constraint

$$\Psi_{s,min}^2 \leq \; x_{m1}^2(k) + x_{m2}^2(k) \; \leq \Psi_{s,max}^2. \tag{6.46}$$

The bounds on the third motor state are derived from the angle $\varphi(k)$. To ensure that the model remains feasible for at least N time-steps when starting with a $\varphi(k)$ close to $\frac{\pi}{3}$ (i.e. the upper limit of the zero sector), the bounds on $\varphi(k)$ are modified to

$$0 \leq \varphi(k) \leq \frac{\pi}{3} + N\omega_r T_s \,, \tag{6.47}$$

which translates into the following bounds on $x_{m3}(k)$

$$\cos(\frac{\pi}{3} + N\omega_r T_s) \leq x_{m3}(k) \leq 1 \,. \tag{6.48}$$

Hence, the constraints (6.45), (6.46) and (6.48) define the set of states $\mathcal{X}^0$. To account for measurement noise and small disturbances causing the torque or the stator flux to slightly violate the imposed bounds, we relax (6.45) and (6.46) by 20 % of the corresponding bound width.

Setting the lower and upper flux bounds to the typical values $\Psi_{s,min}^2 = 0.82\,\text{p.u.}$ and $\Psi_{s,max}^2 = 1.04\,\text{p.u.}$, Fig. 6.3 shows the resulting quadratic constraints on the $x_{m1}x_{m2}$ plane which are approximated here by PWA constraints. The torque bounds would add two horizontal constraints on the shaded set of states. Here, we have only conservatively imposed $0 \leq T_e \leq 1.2\,\text{p.u.}$. Note that $\mathcal{X}^0$ is a very small subset of the zero sector.

Nonlinear function	Argument	Domain	Number of regions	1-norm of error	∞-norm of error
x_{m1}^2	x_{m1}	$[0.8, 1.05]$	2	0.001	0.003
x_{m2}^2	x_{m2}	$[0, 0.4]$	3	0.001	0.002
β	x_{m3}	$[0.4, 1]$	4	0.001	0.006

Table 6.1: PWA approximations of the nonlinearities in the Stator Flux Dynamics Model

Therefore, translating the torque and flux constraints into constraints on x_{1m} and x_{2m} allowed us to derive a set $\mathcal{X}^0$ that is tight and significantly less conservative than a set resulting from the bounds of the zero sector would be.

6.4.2 Stator Flux Dynamics

The nonlinearities of the Stator Flux Dynamics Model are the length of the stator flux vector (6.34) and the matrix $\tilde{P}(\varphi)$ (5.2) with the components $\sin(\varphi)$ and $\cos(\varphi)$ that result from trigonometric expansion and are nonlinear in φ. As the MLD framework does not allow for modelling general nonlinear functions, they need to be approximated by PWA functions. As $u_{abc}^{(0)}$ only takes integers, the multiplication between $\tilde{P}(\varphi)$ and $u_{abc}^{(0)}$ in (6.28) can be handled using *if ... then ... else* statements.

To account for the evolution of the rotating reference frame, it would be straightforward to introduce φ as a state and to use (6.29) to describe its evolution. This, however, would necessitate the approximation of $\sin(\varphi)$ and $\cos(\varphi)$ as a function of φ. Instead, we have chosen above $\alpha = \cos(\varphi)$ as a state. Therefore, we need to only approximate β as a function of α thus avoiding the approximation of one nonlinearity.

Additionally, from now on, we use the squared length of the stator flux vector rather than its length, thus turning the nonlinearity (6.34) with two arguments into two nonlinearities where each one has only one argument. This greatly simplifies the approximation task reducing the approximation error and the complexity. Table 6.1 summarizes the PWA approximations together with the function domains and the number of regions used. The domains slightly over-approximate the set of states $\mathcal{X}^0$, while the number of regions results from a trade-off between the required model accuracy and the increase in the model complexity. For the problem considered here, the maximum approximation error was chosen to be smaller than 0.5 %, as this is in the range of the accuracy of the estimation scheme used in Chapter 7. The resulting 1-norm of the error is with 0.001 p.u. very small.

Nonlinear function	Arguments	Domain	Number of regions	1-norm of error	∞-norm of error
$\alpha\, i_d^\vartheta$	α, i_d^ϑ	$[0.4, 1] \times [0, 0.6]$	8	0.005	0.015
$\alpha\, i_q^\vartheta$	α, i_q^ϑ	$[0.4, 1] \times [0, 1.2]$	8	0.007	0.022
$\beta\, i_d^\vartheta$	β, i_d^ϑ	$[0, 0.91] \times [0, 0.6]$	8	0.010	0.029
$\beta\, i_q^\vartheta$	β, i_q^ϑ	$[0, 0.91] \times [0, 1.2]$	8	0.014	0.044

Table 6.2: PWA approximations of the nonlinearities in the three-level inverter model

6.4.3 Three-Level Inverter

The PWA approximation of the dynamic of the neutral point potential is slightly more involved. Similarly to above, the multiplication with $|u_{abc}^{(0)}|$ in (6.40) can be handled using *if ... then ... else* statements leaving only the term $P^{-1}(\varphi)\, i_{dq0}^\vartheta$ to be approximated. As the inverse of the transformation matrix is given by

$$P^{-1}(\varphi) = \begin{bmatrix} \cos\varphi & -\sin\varphi & 1 \\ \cos(\varphi - \frac{2\pi}{3}) & -\sin(\varphi - \frac{2\pi}{3}) & 1 \\ \cos(\varphi + \frac{2\pi}{3}) & -\sin(\varphi + \frac{2\pi}{3}) & 1 \end{bmatrix}, \tag{6.49}$$

and recalling (6.31), the four terms $\alpha\, i_d^\vartheta$, $\alpha\, i_q^\vartheta$, $\beta\, i_d^\vartheta$ and $\beta\, i_q^\vartheta$ need to be approximated, where we have used $i_{dq0}^\vartheta = [i_d^\vartheta\ i_q^\vartheta\ i_0^\vartheta]^T$.

Again, the function domains are chosen such that the whole range of operation is covered. As the neutral point potential does not need to be controlled with a high precision, a relatively large approximation error is tolerable. Nevertheless, the approximation errors as shown in Table 6.2 are very small. In particular for the multiplications involving α, approximations with four regions might be sufficient, too.

6.5 MLD Model

Having derived the low-complexity models of the induction motor and the two- and three-level inverters, and having approximated the contained nonlinearities by PWA functions, we are now ready to cast the model of the DTC drive in MLD form. As MLD forms have been recapitulated, we do not repeat the general MLD representation here, but refer the reader to Section 2.2.2. In the following, we only summarize the MLD models of a DTC drive with a two- and a three-level inverter. Indeed, the derivation of the MLD models is performed by the compiler HYSDEL generating the matrices of the MLD system starting from a high-level description of the system. The corresponding HYSDEL codes

can be found in Appendices A.2 and A.3, respectively. Note that the MLD models use the rotational speed and the length of the rotor flux as parameters.

6.5.1 DTC Drive with Two-Level Inverter

The overall MLD model of the DTC drive includes the two submodels of the induction motor (the Stator Flux Dynamics Model) and the two-level inverter. As the latter lacks states, the overall state vector is given by the motor state vector

$$x(k) = x_m(k) = \begin{bmatrix} \psi_{ds}^{\vartheta}(k) & \psi_{qs}^{\vartheta}(k) & \alpha(k) \end{bmatrix}^T . \tag{6.50}$$

The model outputs are the electromagnetic torque T_e and the squared length of the stator flux vector Ψ_s^2 yielding the output vector

$$y(k) = \begin{bmatrix} T_e(k) & \Psi_s^2(k) \end{bmatrix}^T . \tag{6.51}$$

The model inputs are the integer variables u_a, u_b and u_c

$$u(k) = u_{abc}^{(0)}(k) = \begin{bmatrix} u_a^{(0)}(k) & u_b^{(0)}(k) & u_c^{(0)}(k) \end{bmatrix}^T \in \{-1, 1\}^3 . \tag{6.52}$$

Note that because of the mapping operation, the models are defined in the zero sector. Yet in the following, we will drop the superscript (0) from the manipulated variables to simplify the notation.

Restricting the state-space to $\mathcal{X}^0$ derived in Section 6.4.1 and using the PWA approximations proposed in Section 6.4.2, an MLD model with three real states, three binary inputs, two outputs, 49 z-variables, 9 δ-variables and 165 inequality constraints results, where those figures include the soft constraints and the slack variables of the cost function.

6.5.2 DTC Drive with Three-Level Inverter

In the case of a DTC drive with a three-level inverter, the overall MLD model is augmented by the states describing the inverter's neutral point potential and the distribution of the switching effort between the upper and the lower half of the inverter. Therefore, the three-level inverter adds the two state equations (6.40) and (6.44) leading to the overall state vector

$$x(k) = \begin{bmatrix} \psi_{ds}^{\vartheta}(k) & \psi_{qs}^{\vartheta}(k) & \alpha(k) & \upsilon_n(k) & \lambda(k) \end{bmatrix}^T . \tag{6.53}$$

The motor outputs are augmented by the neutral point potential and the distribution of the switching effort resulting in

$$y(k) = \begin{bmatrix} T_e(k) & \Psi_s^2(k) & \upsilon_n(k) & \lambda(k) \end{bmatrix}^T , \tag{6.54}$$

and the model inputs are the integer variables u_a, u_b and u_c

$$u(k) = u_{abc}^{(0)}(k) = \left[u_a^{(0)}(k) \quad u_b^{(0)}(k) \quad u_c^{(0)}(k)\right]^T \in \{-1, 0, 1\}^3 . \tag{6.55}$$

As with the two-level inverter, we restrict the state-space to $\mathcal{X}^0$, use the PWA approximations proposed in Section 6.4.2 for the Stator Flux Dynamics model, and the PWA approximations of Section 6.4.3 for the three-level inverter to derive an MLD model with five real states, six binary inputs, two outputs, 100 z-variables, 68 δ-variables and 527 inequality constraints, including the soft constraints and the slack variables of the cost function.

6.6 PWA Model

In Sections 7.2.4 and 7.3.2, when computing state-feedback control laws, a PWA model of the DTC drive needs to be available. As shown in Section 3.4, the mode enumeration algorithm can be directly used to efficiently translate the HYSDEL code, which is used to describe the model on a textual basis, into an equivalent PWA form. Equivalence implies, that for all feasible initial states and for all feasible input trajectories, both models yield the same state and output trajectories. Concerning the notation, we will use the same as in (2.10) to denote PWA systems. As in this thesis we are computing state-feedback control laws only for drives with two-level inverters, we focus in the next section only on this case.

6.6.1 DTC Drive with Two-Level Inverter

As the MLD model features the rotational speed and the length of the rotor flux as parameters, we need to set these first to fixed values. Then, it is straightforward to derive the equivalent PWA form, where we restrict the state-space to $\mathcal{X}^0$ to remove unnecessary regions from the model.

Example 6.1 As an example, consider the MLD model in Section 6.5.1 with the motor and inverter parameters shown in Table 7.13, the rotational speed $\omega_r = 0.8\,\text{p.u.}$ and the length of the rotor flux vector $\psi_{dr}^{\vartheta} = 0.89\,\text{p.u.}$. We restrict the state-space to $\mathcal{X}^0$ given by the following three constraints: $0 \leq T_e \leq 1.2$, $0.776 \leq \Psi_s^2 \leq 1.084$ and $0.4 \leq \alpha \leq 1$. Within 6.5 s on a 2.8 GHz Pentium PC, the mode enumeration algorithm derives the equivalent PWA model defined on the six-dimensional state-input space. However, as the state-input-space is only partitioned into polyhedra due to the PWA approximations of nonlinear functions whose arguments are only states (and not inputs), we conclude that the polyhedra are obviously independent from the input vector. Hence, the polyhedral

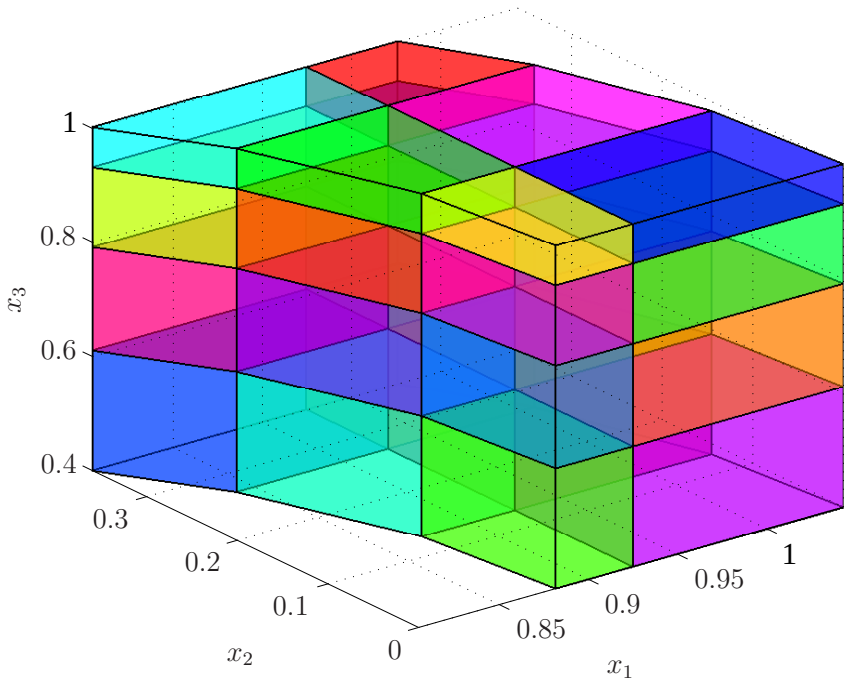


Figure 6.4: The polyhedral partition of the three-dimensional state-space for the equivalent PWA model of the DTC drive with a two-level inverter

partition is only defined on the state-space as shown in Fig. 6.4. One can observe the facets introduced by the approximation of the stator flux components and β as a function of α. Additionally, when restricting the torque around its operating point, it is easy to see that the 24 polyhedra can be easily reduced by one or two thirds.

7

Model Predictive Direct Torque Control

7.1 Introduction

Based on the hybrid models of the DTC drive derived in the preceding chapter, we propose in this chapter three novel control approaches to tackle the DTC problem, namely *MPC based on Priority Levels* (MPC-PL), *MPC based on Feasibility and Move Blocking* (MPC-FMB) and *MPC based on Extrapolation* (MPC-E). All control schemes share the following common features: (i) An internal discrete-time model of the drive to predict the future evolution of the controlled output variables for a sequence of control inputs over a prediction horizon, (ii) a cost function which is minimized to obtain the optimal control input sequence, and (iii) a receding horizon policy, meaning that only the first control input of the sequence is implemented, the horizon is shifted by one time-step and the above procedure is repeated. The main differences of the control approaches are summarized in Tables 7.1 and 7.2 for the two-level and three-level inverter, respectively, where the performance improvement refers to the reduction of the average switching frequency while keeping the same bounds. Note that the indicated on-line computation times and the memory requirements are rough estimates, merely intended to hint to the reader that the computation times and the memory requirement starting from MPC-PL over MPC-FMB to MPC-E steadily decrease. In particular the last control scheme, MPC-E, is expected to be implementable on the currently available DTC hardware. At the same time, the performance is steadily improved.

In the following three sections, if not otherwise stated, the control schemes are described for the case of a three-level inverter as this naturally comprehends also the two-level inverter, meaning that if the case of a three-level inverter can be tackled successfully, the scheme can be directly simplified to also handle the two-level inverter.

At the end of each section, we compare the performance of the proposed MPC scheme with the industrial state of the art via simulations. The comparisons are done in terms

	MPC-PL	MPC-FMB	MPC-E
Solution obtained	off-line	off-line	on-line
Computation scheme	Dynamic Progr.	forward in time	MINLP
Prediction model	PWA	PWA	nonlinear
Operating point	fixed	fixed	any
On-line computation time	medium	short	short
Memory requirement	medium	low	low
Performance improvement	no reference	5 %	not investigated

Table 7.1: Summary of the differences between the three proposed MPC schemes for the DTC problem with a two-level inverter, where the performance improvement of MPC-FMB is achieved with respect to MPC-PL

	MPC-PL	MPC-FMB	MPC-E
Solution obtained	on-line	on-line	on-line
Computation scheme	MILP	MILP	MINLP
Prediction model	MLD	PWA	nonlinear
Operating point	any	any	any
On-line computation time	very long	medium	short
Memory requirement	high	medium	medium
Performance improvement	20 %	not investigated	up to 50 %

Table 7.2: Summary of the differences between the three proposed MPC schemes for the DTC problem with a three-level inverter, where the performance improvement is achieved with respect to ABB's ACS6000

of the average switching frequency and the mean squared violation[1] of the torque and stator flux violation of the imposed bounds. These simulations were carried out using ABB's Matlab/Simulink model of the ACS6000 drive [ABB], where a look-up table stores ABB's DTC strategy. For the MPC schemes the same simulation environment was used except for the look-up table, which was replaced by a function solving at each time-step the model-based control problem. The bounds for the torque and the stator flux depend on the operating point and are imposed by an outer control loop in the Matlab/Simulink

[1]We are using the mean squared error rather than the *root* mean squared (rms) error.

Case study	Rated machine power	Rated machine voltage
I	1.6 MW	3.3 kV
II	6.6 MW	3.1 kV
III	15 kW	400 V

Table 7.3: Overview of the machine ratings of the three case studies

model, whereas the bounds on the neutral point potential, which do not explicitly exist in ABB's ACS6000 scheme, are time-invariant. For both control schemes, the same bounds were used for the torque and flux. For the neutral point potential they were chosen to reflect the behavior of ABB's control scheme, thus ensuring that the comparison is meaningful.

Note that as the ACS6000 drive has been designed for three-level inverters, we only have an industrial reference for DTC drives with three-level inverters to compare our control schemes with. For two-level inverters, however, we are left with comparing the model-based control schemes only with each other, but not to an industrial reference. Nevertheless, the focus in this thesis lies on DTC drives with three-level inverters as they are significantly more challenging to control with respect to drives with two-level inverters.

For the comparisons, three case studies were considered with DTC drives of medium and low power. Table 7.3 provides a rough overview of the case studies, whereas the respective ratings and parameters are detailed in Tables 7.13, 7.14 and 7.15 at the end of this chapter.

7.2 MPC based on Priority Levels

The first of the three proposed MPC schemes is rather conventional, in the sense that the particular problem structure of DTC is not exploited to speed up the computation of the control input or to derive a low-complexity controller. Here, we formulate and solve a standard CFTOC problem, where integer variables are introduced for the inverter switch positions, which represent the manipulated variables, and the motor and inverter control objectives are reformulated such that they can be incorporated into an objective function. In particular, soft constraints are used to model the hysteresis bounds on the torque, stator flux and neutral point potential, and the average switching frequency (over an infinite horizon) is approximated by the number of switch transitions over a short horizon. To make this approximation meaningful and to avoid excessive switching, the *Late Switching Strategy* has to be added, which favors the postponement of switch transitions as explained later on.

The proposed approach is applied to DTC drives featuring a two- and a three-level inverter. In both cases, the design procedure remains essentially the same, and the extension to other inverter topologies with more degrees of freedom (like the 5-level inverter) is straightforward. Concerning the potential performance improvement, we focus on the case of the three-level inverter considering ABB's ACS6000 DTC drive [ABB] containing a squirrel-cage rotor induction motor with a rated apparent power of 2 MVA and a 4.3 kV three-level dc-link inverter. The performance of the control scheme is evaluated through simulations and compared with ABB's well-established DTC strategy showing a performance improvement in terms of a reduction of the switching frequency in the range of 20 % while simultaneously reducing the torque and flux ripples. Unfortunately, it seems that the complexity of the control law is prohibitive thus not allowing for an implementation using standard control hardware.

In a first step, we formulate the CFTOC problem using the MLD model and solve the corresponding MILP on-line to obtain the control input. This can be done easily for the two- and the three-level inverter. In a second step, we use an equivalent PWA model of the drive and solve the MILP off-line using multi-parametric programming and Dynamic Programming to obtain a state-feedback control law. At the time of writing, the latter is meaningful to be tried out, with the computational power at our disposal, only for a two-level inverter.

7.2.1 Priority Levels

The controller objectives can be classified in three priority levels. The main objective is to keep the torque and the length of the stator flux within the pre-specified bounds, and to also retain the neutral point potential within bounds that are typically symmetric around zero. As these bounds shall not be (significantly) violated, we assign to them the highest priority, and express them in the objective function using soft constraints.

The control objective with secondary priority is to minimize the average switching frequency over an infinite horizon. This is approximated by minimizing the number of switch *transitions* within a finite prediction interval. Due to the limited length of the prediction interval, one needs to enforce that switch transitions are only performed if absolutely necessary, i.e. when refraining from switching would lead to a violation of the bounds on the controlled variables within one time-step. This leads to what we call the *Late Switching Strategy*, where the controller postpones any scheduled switch transition until absolutely necessary. This strategy is implemented by imposing a time-decaying penalty on the switch transitions, where switch transitions within the first time-step of the prediction interval result in larger penalties then those that are far in the future.

In particular for short prediction intervals, for a given state, two or more control inputs

may have the same associated costs according to the two penalty levels introduced above[2]. In the presence of such ambiguities, the control input that moves some of the controlled variables closest to their references is preferable, in particular the stator flux and the neutral point potential. We account for that by adding a third, low priority penalty level on the deviation of the stator flux and the neutral point potential from their respective references. For the torque, however, it is preferable to take full advantage of the window width. Thus we refrain from adding such a penalty term to the torque. Another objective of low priority is to distribute the switching effort evenly between the upper and lower half of the inverter. We account for that by adding a small penalty on the distribution of the switching effort.

Summing up, we introduce three priority levels in the objective function. Starting with the highest priority, these are the following. (i) keep the torque, flux and neutral point potential within their bounds, (ii) minimize the number of switch transitions, and (iii) regulate the controlled variables except the torque to their references and evenly distribute the switching effort.

7.2.2 Objective Function

Based on the controller objectives, we establish next the mathematical expression of the objective function, which is composed of a number of cost expressions. The soft constraints on the upper and lower torque bounds $T_{e,max}$ and $T_{e,min}$, respectively, lead for the electromagnetic torque to the cost expression

$$\varepsilon_T(k) = \begin{cases} q_T(T_e(k) - T_{e,max}) & \text{if } T_e(k) \geq T_{e,max} \\ q_T(T_{e,min} - T_e(k)) & \text{if } T_e(k) \leq T_{e,min} \\ 0 & \text{else}\,, \end{cases} \tag{7.1}$$

where $q_T \gg 0$ is the weight on the soft constraints. The cost expression for the length of the stator flux vector is defined similarly using the upper and lower flux bounds $\Psi_{s,max}$ and $\Psi_{s,min}$, respectively, with an additional term penalizing the deviation from the reference $\Psi_{s,ref}$

$$\varepsilon_\Psi(k) = \begin{cases} q_F(\Psi_s(k) - \Psi_{s,max}) & \text{if } \Psi_s(k) \geq \Psi_{s,max} \\ q_F(\Psi_{s,min} - \Psi_s(k)) & \text{if } \Psi_s(k) \leq \Psi_{s,min} \\ q_f|\Psi_s(k) - \Psi_{s,ref}| & \text{else}\,, \end{cases} \tag{7.2}$$

with the weights q_F and q_f, $q_F \gg q_f > 0$, on the soft constraints and on the deviation from the reference, respectively. The switch transitions are penalized using a time-varying

[2] As an example consider the voltage vectors of medium length which are always given as pairs. For a given pair, both voltage vectors lead to the same torque and flux response, yet they have an opposite impact on the neutral point potential.

weight $q_u(k) > 0$ and the 1-norm

$$\varepsilon_u(k) = q_u(k)\|u(k) - u(k-1)\|_1 \,. \tag{7.3}$$

The above stated cost expressions are the same for the two- and the three-level inverter. For the three-level inverter, the following two additional cost expressions need to be defined.

For the neutral point potential, the cost $\varepsilon_\upsilon(k)$ is defined according to (7.2) with the respective bounds $\upsilon_{n,max}$ and $\upsilon_{n,min}$, the reference 0, and the weights q_N and q_n. The distribution of the switching effort has the simple cost

$$\varepsilon_\lambda(k) = q_\lambda|\lambda(k) - 0| \tag{7.4}$$

with $q_\lambda > 0$.

Finally, we define

$$\varepsilon_1 = \varepsilon_u \qquad \varepsilon_2 = \begin{bmatrix}\varepsilon_T & \varepsilon_\Psi\end{bmatrix}^T \tag{7.5}$$

for the case of the two-level inverter and

$$\varepsilon_1 = \begin{bmatrix}\varepsilon_\lambda & \varepsilon_u\end{bmatrix}^T \qquad \varepsilon_2 = \begin{bmatrix}\varepsilon_T & \varepsilon_\Psi & \varepsilon_\upsilon\end{bmatrix}^T \tag{7.6}$$

for the three-level inverter, and consider the objective function

$$J(x(k), u(k-1), U(k)) = \sum_{\ell=0}^{N-1} \|\varepsilon_1(k+\ell|k)\|_1 + \sum_{\ell=1}^{N} \|\varepsilon_2(k+\ell|k)\|_1 \,, \tag{7.7}$$

which penalizes the predicted evolutions of ε_1 and ε_2 over the finite horizon N using the 1-norm, while taking into account that the model has no direct feed-through, ε_1 is a function of the input vector and ε_2 only depends on the state vector.

7.2.3 On-Line Computation of Control Input

The control input at time-instant k is then obtained by minimizing the objective function (7.7) over the finite sequence of control inputs $U(k) = [(u(k))^T, \ldots, (u(k+N-1))^T]^T$ subject to the evolution of the MLD model and its mixed-integer linear inequality constraints, the integrality constraints on $U(k)$ and the cost expressions (7.1)–(7.4). This amounts to the constrained finite time optimal control problem (CFTOC)

$$\begin{aligned} U^*(k) = &\arg\min_{U(k)} J(x(k), u(k-1), U(k)) \\ &\text{subj. to MLD model, (6.8), (7.1)–(7.4)} \end{aligned} \tag{7.8}$$

yielding the sequence of optimal control inputs $U^*(k) = [(u^*(k))^T, \ldots, (u^*(k+N-1))^T]^T$, of which only the first input $u^*(k)$ is applied to the inverter. At the next sampling interval,

k is set to $k+1$, a new state measurement (or estimate) is obtained, and the CFTOC problem is solved again over the shifted horizon according to the receding horizon policy. As we are using the 1-norm in all cost expressions, the CFTOC problem amounts to solving an *Mixed-Integer Linear Program* (MILP) for which efficient solvers exist, like [ILO02].

The optimal control problem posed above is intended to capture the average switching frequency. Therefore, a long prediction interval is beneficial. However, the computational complexity explodes. To account for that, we propose to use a rather long prediction *interval*, but a short prediction *horizon* N. This is achieved by finely sampling the prediction model with $25\,\mu$s only for the first steps, but more coarsely with a multiple of $25\,\mu$s for steps far in the future. Although this approach is similar to utilizing the technique of blocking control moves [QB03], using a coarsely sampled model instead proves to be advantageous in the hybrid domain reducing the complexity and thus the computation times. We call this approach the *Multiple-Rate Prediction Model Approach* that leads to a time-varying prediction model with two different sampling rates. As the simulation results will show, this allows us to greatly increase the length of the prediction interval thus enhancing the performance of MPC while keeping the computation times low.

Recapitulating this section, the MPC objective function is based on the following main ideas. Three penalty levels with corresponding penalties of different orders of magnitude provide clear controller priorities and make the fine-tuning of the objective function obsolete. The Late Switching Strategy assures that unnecessary switch transitions are avoided, and the Multiple-Rate Prediction Model Approach allows us to extend the prediction interval without increasing the computational burden.

7.2.4 Off-Line Computation of State-Feedback Control Law

As the computation times needed for solving the optimal control problem on-line are well beyond the $25\,\mu$s sampling time of DTC, the proposed controller cannot be directly implemented. In order to overcome this obstacle, the calculation of the explicit state-feedback control law is necessary by pre-computing off-line the solution to the optimal control problem for the whole state-space. For hybrid systems, such a method has been recently introduced, which is based on a PWA description of the controlled system. Hence, we rewrite the CFTOC (7.8) by replacing the MLD model by the equivalent PWA model

$$\begin{aligned} U^*(k) = \arg\min_{U(k)} & J(x(k), u(k-1), U(k)) \\ \text{subj. to PWA model, } & (6.8), (7.1)\text{–}(7.4)\,. \end{aligned} \tag{7.9}$$

Note that the CFTOC problem is not only a function of the state vector $x(k)$, but also of the last control input $u(k-1)$, as we are penalizing the switch transitions in the objective function.

Then, as proposed in [BCM03b, BBBM05], Dynamic Programming and multi-parametric programming, where the state vector is treated as a parameter, can be used to derive the PWA state-feedback control law. The resulting control law is a PWA state-feedback control law defined over a polyhedral partition of the state-space, which can be stored in a look-up table. Computing the optimal control law on-line is thus reduced to a simple evaluation of a look-up table. For a summary of multi-parametric programming, the algorithms currently available, the properties of the resulting control law and issues concerning the implementation of the controller, the reader is referred to Section 2.4.2.

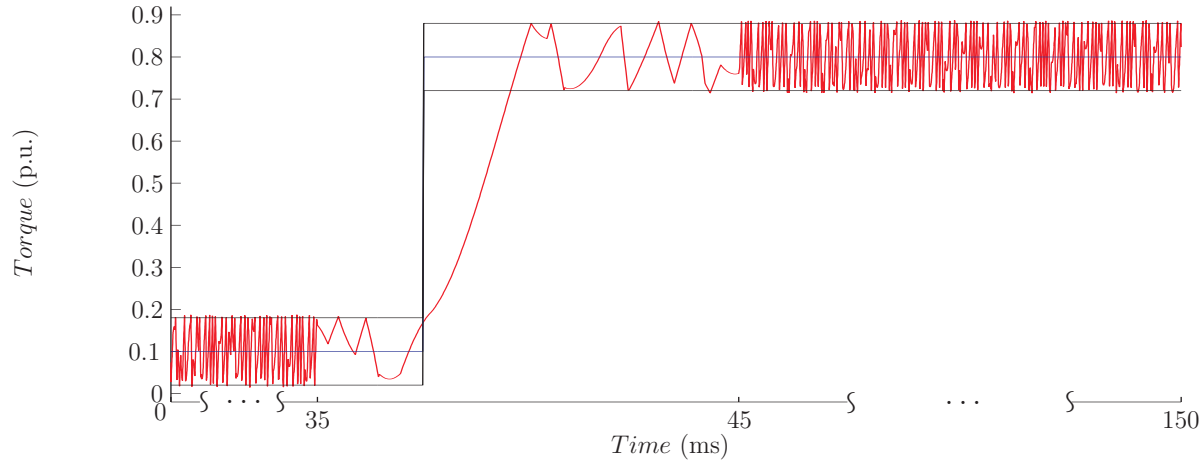
7.2.5 Performance Evaluation

The simulation results presented in this section illustrate the performance of the MPC-PL scheme for the two- and the three-level inverter, respectively. This parallelism is used to demonstrate that the tuning of the objective function is done in a straightforward and systematic way, regardless of the problem complexity. Using Case Study I for the performance evaluation, the parameter values used are given in Table 7.13 at the end of this chapter. In all graphs, the units are normalized and the time scaling is in ms.

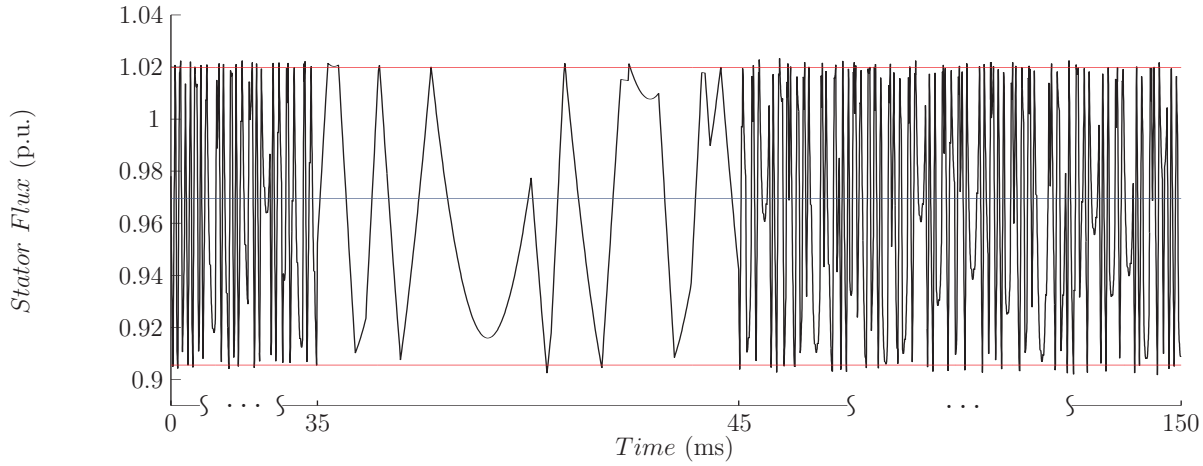
DTC Drive with Two-Level Inverter

For a DTC drive featuring a two-level inverter, the optimal control problem was solved for the objective function (7.7) using a prediction horizon of $N = 2$. Employing a single model approach, all steps are set equal to the usual DTC sampling time of $25\,\mu$s. The penalties on the soft constraints are chosen to be $q_T = 3000$ for the torque and $q_F = 4500$ for the stator flux. The switch transitions are penalized with $q_u(0) = 14$, exponentially decaying within the prediction horizon. The deviation of the stator flux from its reference is penalized with $q_f = 0.01$.

On-Line Computation of Control Input Initially, the motor is running with a speed of $\omega_r = 0.8\,$p.u. under a load torque of $T_\ell = 0.1\,$p.u., when a step in the torque reference $T_{e,ref}$ is applied from 0.1 p.u. to 0.8 p.u.. As the simulation results in Fig. 7.1 show, the torque response under MPC-PL is rapid, while the length of the stator flux remains within the specified bounds. Note that for the benefit of visualization, two different time scales are used, showing the step response between 35 ms and 45 ms in greater detail. The average switching frequency of the inverter was 515 Hz. The computation times required for the solution of the optimal control problem on-line at each time-step are in the range of 50 ms running CPLEX [ILO02] on a 2.8 GHz Pentium PC.



(a) Electromagnetic torque



(b) Stator flux

Figure 7.1: Closed-loop simulation of the MPC-FMB scheme during a step change in the torque reference for a DTC drive with a two-level inverter

Off-Line Computation of State-Feedback Control Law To simplify the derivation of the explicit state-feedback control law, we only consider the case where the drive is operating at steady state at a fixed operating point. For this, we choose the operating point shown in Fig. 7.1 from 40 ms on, which is given by the speed $\omega_r = 0.8\,\text{p.u.}$, the length of the rotor flux $\psi_{dr}^{\vartheta} = 0.89\,\text{p.u.}$, the load torque $T_\ell = 0.8\,\text{p.u.}$, and the torque bounds $T_{e,min} = 0.72\,\text{p.u.}$ and $T_{e,max} = 0.88\,\text{p.u.}$. The reference of the stator flux is $\Psi_{s,ref} = 0.970\,\text{p.u.}$ and the corresponding bounds are $\Psi_{s,min} = 0.905\,\text{p.u.}$ and $\Psi_{s,max} = 1.020\,\text{p.u.}$, or equivalently, $\Psi^2_{s,min} = 0.82\,\text{p.u.}$ and $\Psi^2_{s,max} = 1.04\,\text{p.u.}$, respectively. The control problem formulation is the same as for the on-line optimization set-up.

Using the PWA model of the DTC drive with a two-level inverter in Example 6.1, which is defined on a polyhedral partition with 24 polyhedra in the three-dimensional state-space, the procedure described in Section 7.2.4 yields for each of the eight last

$u(k-1)$	n_{org}	n_{red}	Reduction [%]	t_{expl} [h]	t_{red} [h]
$[-1\ -1\ -1]^T$	5246	709	86.5	8.6	9.1
$[-1\ -1\ +1]^T$	5325	753	87.6	8.9	10.3
$[-1\ +1\ -1]^T$	4737	486	89.7	11.1	10.9
$[-1\ +1\ +1]^T$	5292	625	88.2	8.5	9.8
$[+1\ -1\ -1]^T$	7019	930	86.8	11.2	8.9
$[+1\ -1\ +1]^T$	8512	880	90.0	11.2	13.3
$[+1\ +1\ -1]^T$	5425	631	88.4	11.2	8.7
$[+1\ +1\ +1]^T$	6295	617	90.2	6.1	7.3

Table 7.4: Overview of the state-feedback control law for the DTC drive with a two-level inverter

control inputs (discrete states) a PWA state-feedback control law defined on the three-dimensional real state-space. Running subsequently the non-disjoint optimal complexity reduction algorithm (see Chapter 4) reduces the complexity of the control law by a factor of nine, as the overview in Table 7.4 shows. For each of the last control inputs $u(k-1)$, this table depicts the number of polyhedra of the original controller n_{org}, the number of polyhedra of the equivalent controller of reduced complexity n_{red}, the percentage reduction in the number of polyhedra, the computation time for the derivation of the explicit control law t_{expl} in hours, and the computation time for the complexity reduction t_{red} in hours. All computations were run on a 2.8 GHz Pentium with Linux using Matlab 6.5. More details about the complexity reduction applied to the DTC state-feedback control law can be found in Section 4.7.3.

Fig. 7.2 shows a two-dimensional cut through the polyhedral partition of the control law. Recalling that the states x_1 and x_2 refer to the d- and q-components of the stator flux vector, and noting that the voltage vectors in Fig. 5.2(b) directly manipulate the stator flux, the control law can be easily interpreted and justified. The yellow polyhedra in the center refer to the control policy, where switching is avoided by applying the last voltage vector again, i.e. $u(k) = u(k-1) = [+1\ -1\ -1]^T$. This is obviously the best choice, as the torque and flux lie well inside their bounds. The fact that this control policy is also used for stator fluxes with small d-components and/or large q-components is reasonable, too, as the selected voltage vector increases the d-component, while leaving the q-component unchanged. Note that, as the reference frame is rotating counter-clockwise, stator fluxes with large q-components are quickly 'caught' by the rotating frame. Particularly interesting are the partitions for large d- and small q-components. Here, for the magenta, red

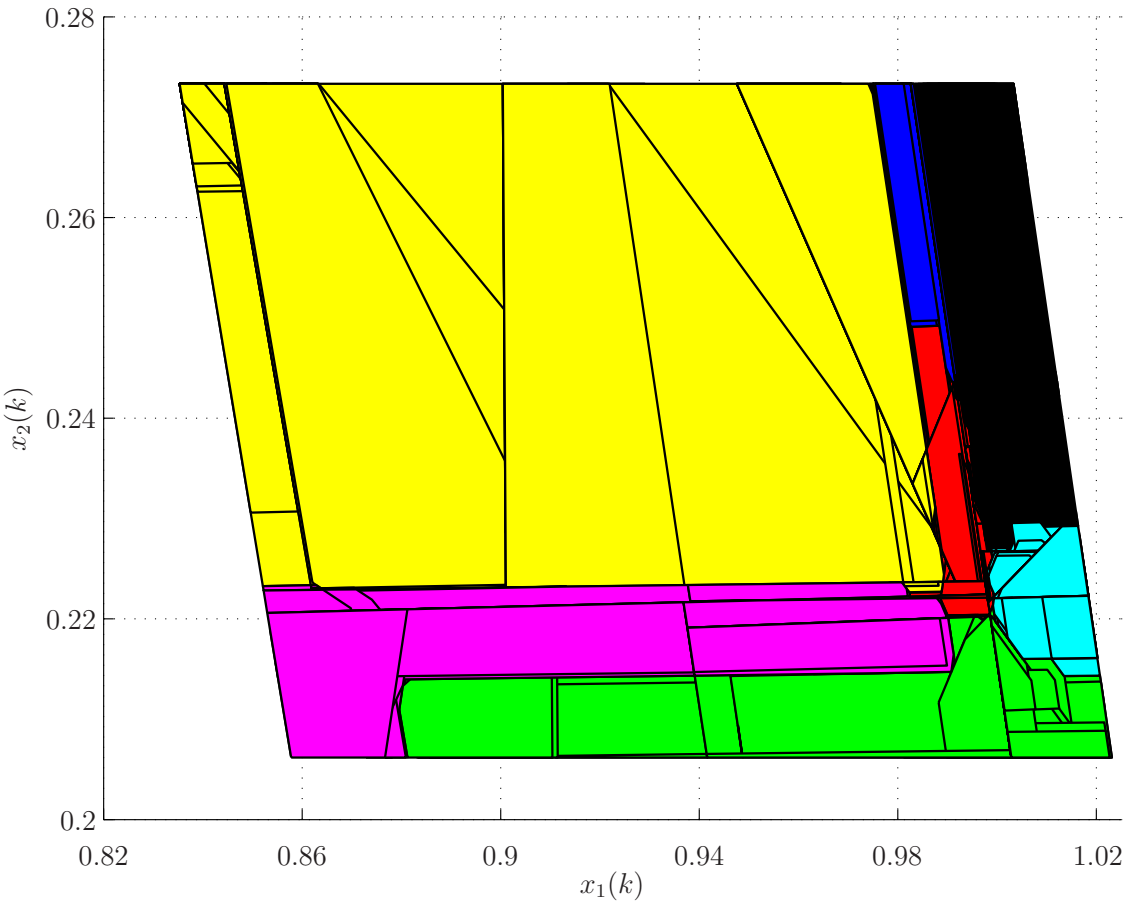


Figure 7.2: The explicit state-feedback control law for the DTC drive with a two-level inverter for $u(k-1) = [+1 \ -1 \ -1]^T$ intersected with $x_3(k) = 0.95$, where the colors correspond to the control inputs $u(k)$ ordered as in Table 7.4: red, black, green, cyan, yellow, blue, magenta and red

and blue polyhedra neighboring the yellow center polyhedra, where the torque and flux are close to their bounds, it is optimal with respect to the defined cost function to perform only in one stack a switch transition, i.e. $||u(k) - u(k-1)||_1 = 2$, where $u \in \{-1, 1\}^3$. Yet voltage vectors that slowly adjust the stator flux are chosen in these polyhedra. Alternatively, they can be interpreted as intermediate steps between the voltage vector in the center and the voltage vectors applied in polyhedra with very large d- and/or very small q-components. In the latter polyhedra, namely the green, cyan and black polyhedra, voltage vectors are chosen that drive the stator flux rapidly back into the center accepting switch transitions in two or three stacks simultaneously, i.e. $||u(k) - u(k-1)||_1 \in \{4, 6\}$.

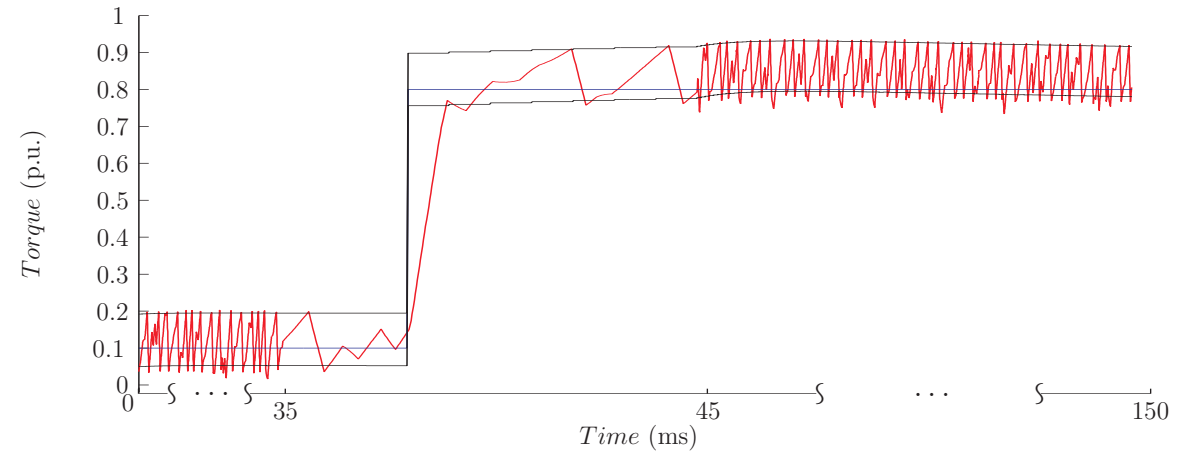
For completeness, we mention that choosing the same initial state at time 40 ms, both the on-line computation and the evaluation of the state-feedback control law yield the same closed-loop simulation results.

DTC Drive with Three-Level Inverter

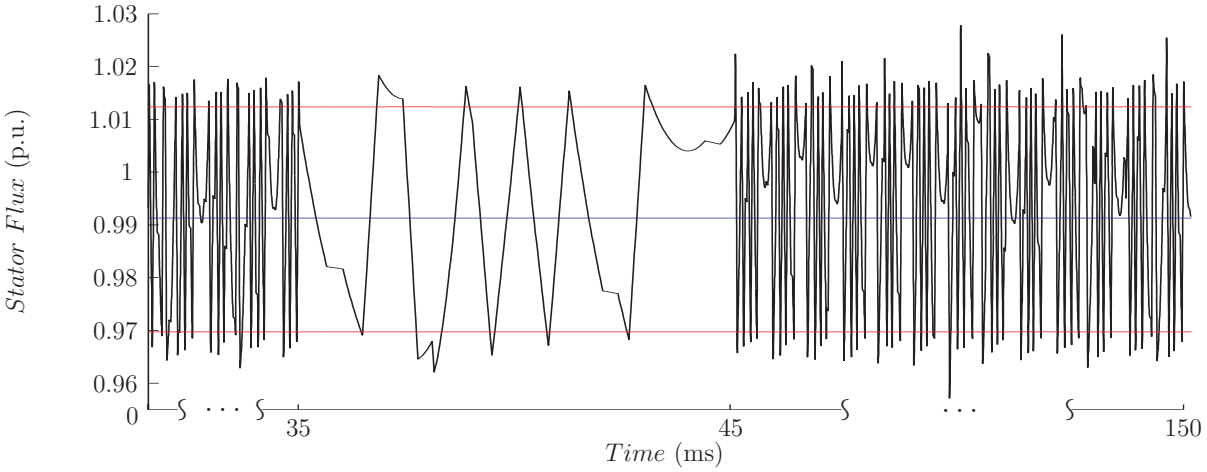
For a DTC drive featuring a three-level inverter, the simulations were carried out using ABB's Matlab/Simulink model of the ACS6000 drive [ABB], where the look-up table with ABB's DTC strategy was replaced by a function solving at each time-step the optimal control problem on-line. For both control schemes, the same bounds were used for the torque, flux and neutral point potential, thus ensuring that the comparison is meaningful. In particular, the bounds on the torque depend on the operating point and are thus time-varying. Apart from that, the considered inverter has restrictions on the allowed switch transitions that stem from technicalities regarding the construction of the inverter as explained in Section 5.3.2. By introducing additional constraints on the integer manipulated variables, these restrictions are easily taken into account in the MPC-PL scheme.

The optimal control problem is solved for the objective function (7.7) using a prediction horizon of $N = 3$. Employing the Multiple-Rate Prediction Model Approach, the first two steps are set equal to the sampling time of $25\,\mu$s and the remaining step is equal to $100\,\mu$s. The models are time-discretized accordingly. To allow for a comparison of ABB's DTC with the proposed MPC-PL scheme, the penalties on the soft constraints are chosen such that the resulting ripples for the torque, flux and neutral point potential are roughly the same. This leads to $q_T = 800$ for the torque, $q_F = 1000$ for the stator flux and $q_N = 4500$ for the neutral point potential. The switch transitions are penalized with $q_u(0) = 16$, exponentially decaying within the prediction horizon. The deviations from the references are penalized with $q_f = q_n = 0.04$ for the stator flux and the neutral point potential, respectively. The distribution of the switching effort is enforced with $q_\lambda = 0.2$. The computation times required for the solution of the optimal control problem at each time-step are in the range of 100 ms running CPLEX [ILO02] on a 2.8 GHz Pentium PC.

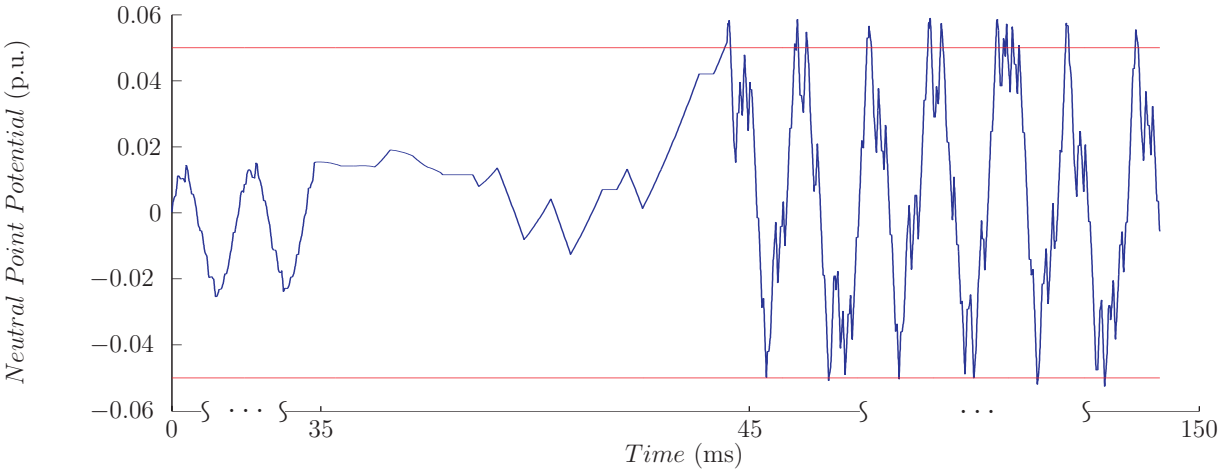
Initially, the motor is running with a speed of $\omega_r = 0.4$ p.u. under a load torque of $T_\ell = 0.1$ p.u., when a step in the torque reference $T_{e,ref}$ is applied from 0.1 p.u. to 0.8 p.u.. Fig. 7.4 depicts the closed-loop behavior of the torque, the stator flux and the neutral point potential under MPC-PL, whereas Fig. 7.3 shows as a comparison the corresponding trajectories resulting from ABB's DTC strategy. As can be seen, the MPC-PL scheme preserves the rapid dynamic responses achieved by the classic DTC approach, while the bounds imposed on the torque, stator flux and neutral point potential are slightly better respected. This degree of the violation of the bounds is a design parameter adjustable by the penalties on the soft constraints. Most important, the average switching frequency for MPC-PL is only 196 Hz compared with ABB's 256 Hz. This improvement amounts to a reduction of the average switching frequency in the range of 20 %, which translates into an equivalent reduction of the switching losses. Alternatively, the ripples on the controlled variables can be reduced by tightening the corresponding bounds until the switching frequency is brought back to 256 Hz.



(a) Electromagnetic torque

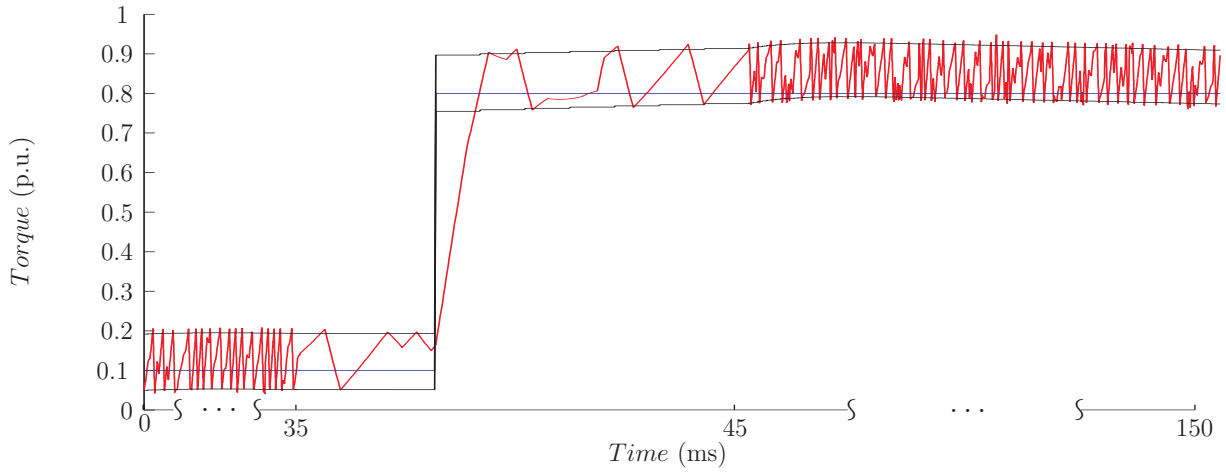


(b) Stator flux

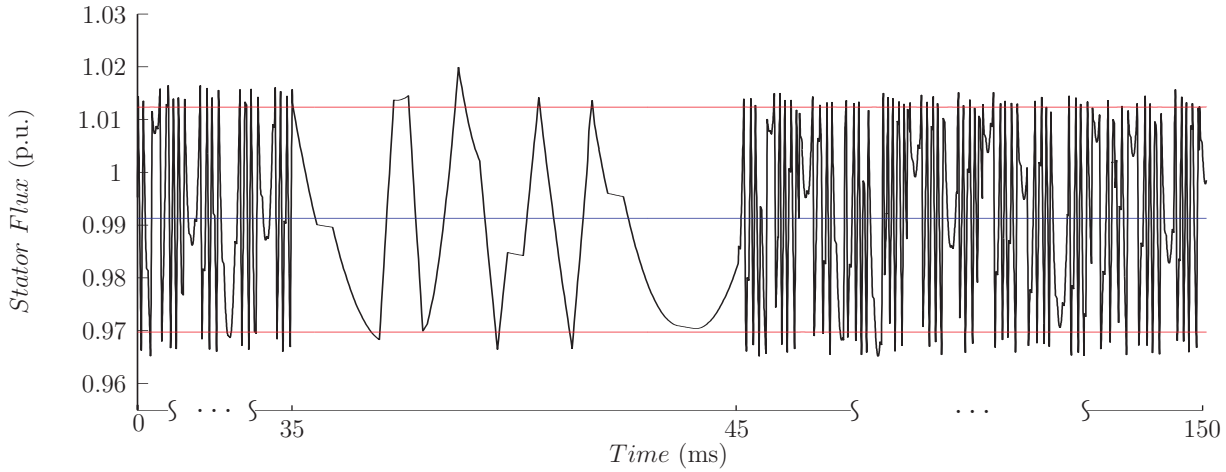


(c) Neutral point potential

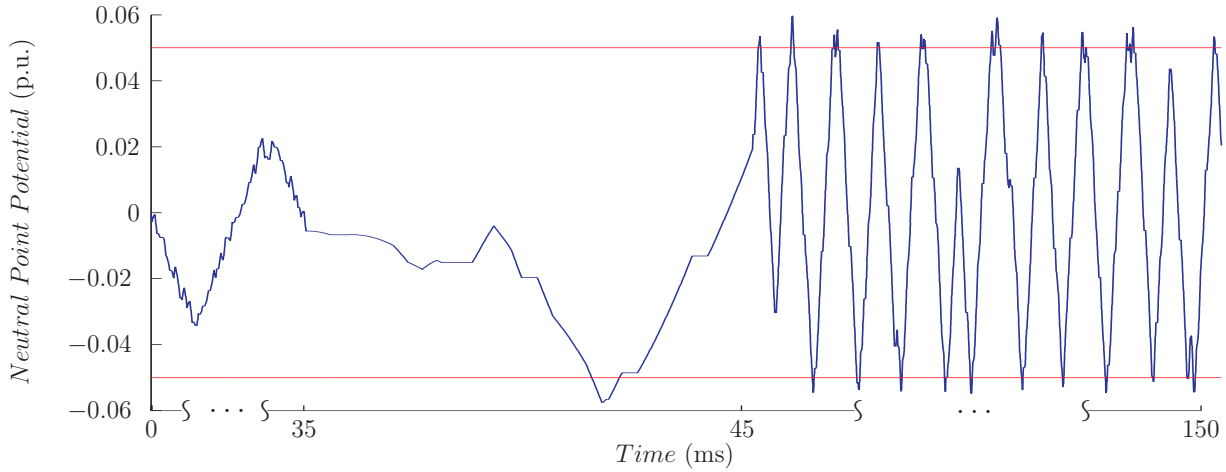
Figure 7.3: Closed-loop simulation of the classic DTC scheme of ABB during a step change in the torque reference for a DTC drive with a three-level inverter



(a) Electromagnetic torque



(b) Stator flux



(c) Neutral point potential

Figure 7.4: Closed-loop simulation of the MPC-PL scheme during a step change in the torque reference for a DTC drive with a three-level inverter

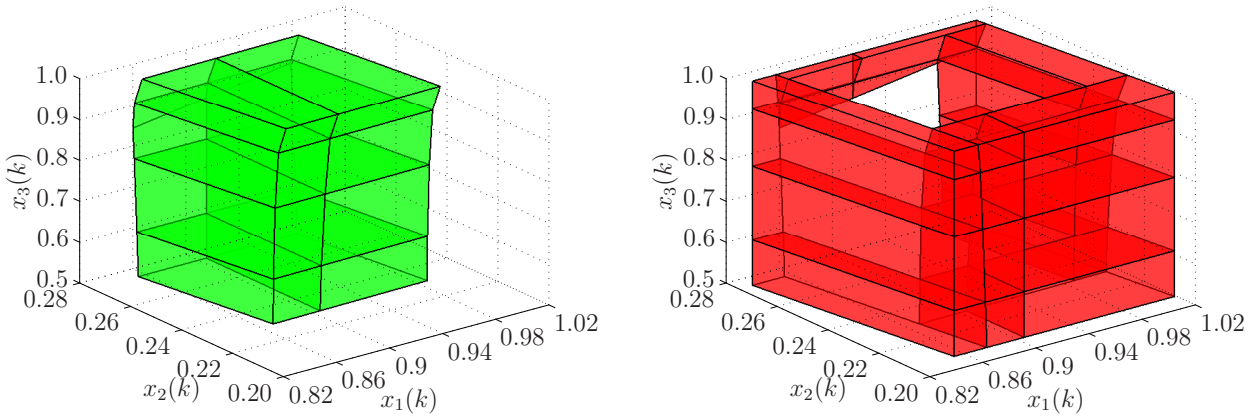
7.3 MPC based on Feasibility and Move Blocking

Even though the MPC approach presented in the last section is conceptually rather simple, systematic and thus appealing, it contains a number of tuning parameters such as the weights on the soft constraints and the late switching penalty. More unfortunate, however, is the enormous complexity of the state-feedback controller that basically prohibits the implementation on the currently employed controller hardware in industry – aside from the issue of the on-line control input computation, which is computationally by order of magnitudes more demanding than the simple evaluation of the nowadays used look-up table. On the other hand, two observations suggest the existence of a low complexity controller resulting from a systematic design procedure. Firstly, albeit their very simple controller structure, the existing DTC schemes have proven to yield a satisfactory control performance. Secondly, the post analysis of the derived state-feedback control law reveals a simple and robust pattern in the solution to the optimal control problem (see Section 7.2.5).

These observations motivate the control scheme presented in the sequel, which is based on the following fundamental property of DTC. The control objectives only weakly relate to optimality but rather to feasibility, in the sense that the main objective is to find a control input that keeps the controlled variables within their bounds, i.e. a control input that is feasible. The second, weaker objective is to select among the set of feasible control inputs the one that minimizes the average switching frequency. The latter can be approximated by the number of switch transitions over the (short) horizon.

We therefore propose an MPC scheme based on feasibility with a prediction horizon N and an internal model of the DTC drive for the predictions. We propose to switch only at the current time-step and to disregard switching within the prediction horizon, which is equivalent to a move blocking strategy. This greatly reduces the total number of control input sequences from 8^N to 8 and allows us to evaluate a small number of input sequences by moving forward in time. For each input sequence, we determine the number of steps the controlled variables are kept within their bounds, i.e. remain feasible. Next we define the number of switch transitions divided by the number of predicted time-steps an input remains feasible as a cost function emulating the switching frequency. In a last step, the control input that minimizes the cost function is chosen. We refer to this concept as *Model Predictive Control based on Feasibility and Move Blocking* (MPC-FMB). The simplicity of the control methodology translates into a state-feedback control law with a complexity that is of an order of magnitude lower than the one of its counterpart obtained through solving the optimal control problem of MPC-PL in the previous section. Moreover, the MPC-FMB scheme has only one design parameter, namely the length of the horizon N.

To simplify the expositions, we focus in this section on a DTC drive with a two-level



(a) The core (b) The ring

Figure 7.5: Core and ring for $u(k-1) = [1 \ -1 \ -1]^T$

inverter. Based on the low-complexity PWA model of the drive, we solve the control problem on-line in Section 7.3.1 and derive off-line the corresponding state-feedback control law in Section 7.3.2. Simulation results are shown in Section 7.3.3, while Section 7.3.4 discusses the extendability of the control approach to DTC drives featuring a three-level inverter.

7.3.1 On-Line Computation of Control Input

Assume a PWA model of the DTC drive is available, which is defined on the set of states $\mathcal{X}^0$. For the model, we use the notation (2.10), namely f_{PWA} for the state-update and g_{PWA} for the output function. For details concerning the derivation of the PWA model, the reader is referred to Section 6.6.

Let $u(k-1)$ denote the last voltage vector. If $u(k-1)$ is also feasible at time-instant k, i.e. all controlled variables are predicted to lie within their bounds at time-instant $k+1$, a reasonable choice is to apply it again, i.e. $u(k) = u(k-1)$. If not, however, the controller must choose another voltage vector. For each of the remaining seven voltage vectors, one can easily compute through open-loop predictions the number of time-steps this voltage vector would keep the controlled variables within their bounds. This step reduces the optimal control problem to a feasibility problem. The voltage vector is chosen that minimizes the average switching frequency over the prediction interval, i.e. the number of switch transitions over the number of time-steps, thus re-introducing the notion of optimality. This control concept is summarized in Algorithm 7.1. An output vector $y(k)$ is said to be feasible, if the corresponding bounds are met, and $U = \{-1, 1\}^3$ denotes the set of available voltage vectors.

Algorithm 7.1

function $u(k) = \text{MPC-FMB\_online}\ (\ x(k),\ u(k-1)\ )$
  $x(k+1) = f_{\text{PWA}}(x(k), u(k-1))$
  if $y(k+1) = g_{\text{PWA}}(x(k+1))$ feasible
    $u(k) = u(k-1)$
  else
    for all $u(k) \in U \setminus u(k-1)$
      $n_u = -1$
      repeat
        $n_u = n_u + 1$
        $x(k+n_u+1) = f_{\text{PWA}}(x(k+n_u), u(k))$
      until $\big(\, y(k+n_u+1) = g_{\text{PWA}}(x(k+n_u+1))$ infeasible $\big)$ **or** $\big(\, n_u = N \,\big)$
    endfor
    $u(k) = \arg\min_{u(k)} \frac{||u(k)-u(k-1)||}{n_u}$
  endif

Compared to MPC-PL, this control policy is by definition significantly simpler, as only eight control sequences (or control strategies) need to be compared with each other. Unlike MPC-PL, switch transitions within the prediction interval are not considered and can only be performed at the current time-instant k. Furthermore, the length of the prediction horizon is time-varying, ranging from one step to ten or even twenty steps. As the next section will show, an explicit form of the proposed controller can be computed easily. Even more important, the explicit form has a low complexity but maintains or even improves the control performance with respect to MPC-PL.

7.3.2 Off-Line Computation of State-Feedback Control Law

After fixing the operating point, setting the bounds and deriving the PWA model as above, we restrict the computation of the explicit state-feedback control law to the set of states $\mathcal{X}^0$. Rewriting (6.45) and (6.46), let $\mathcal{C}$ denote the set of states whose corresponding outputs are feasible

$$\mathcal{C} = \{x \in \mathcal{X}^0 \mid \begin{bmatrix} T_{e,min} \\ \Psi^2_{s,min} \end{bmatrix} \leq g_{\text{PWA}}(x) \leq \begin{bmatrix} T_{e,max} \\ \Psi^2_{s,max} \end{bmatrix}, \tag{7.10}$$

where we have replaced the quadratic expression in (6.46) by the PWA approximation for the stator flux using the output function of the PWA model.

Before presenting the computation of the state-feedback control law in three stages, we introduce the following notation. Let n denote the time-step within the prediction

horizon N, $\mathcal{X}^n_{\text{feas}}$ the set of states at time-step $k+n$ corresponding to feasible outputs $y(k+\ell)$ for all $\ell \in \{1,\ldots,n\}$, $\mathcal{X}^n_{\text{infs}}$ the set of states at time-step $k+n$ with feasible outputs $y(k+\ell)$ for all $\ell \in \{1,\ldots,n-1\}$, but infeasible outputs $y(k+n)$, and $\mathcal{Q}^n_u$ the set of states at time-step k that keep the outputs for n time-steps feasible when applying the voltage vector u. Note that the feasibility of the outputs at the current time-step k is not considered.

Stage I First, we determine the set of states $x(k) \in \mathcal{X}^0$ for which the controlled variables are feasible at time-step $k+1$ when applying $u(k) = u(k-1)$. We denote this set of states as the *core*

$$\mathcal{Q}^c_u = \{x \in \mathcal{X}^0 \mid f_{\text{PWA}}(x,u) \in \mathcal{C}\}\,, \tag{7.11}$$

and its complement in $\mathcal{X}^0$ as the *ring*

$$\mathcal{Q}^r_u = \mathcal{X}^0 \setminus \mathcal{Q}^c_u\,. \tag{7.12}$$

Example 7.1 To visualize the algorithm, consider as an example a two-level inverter driving an induction machine with the rated voltage 3.3 kV and the rated real power 1.587 MW (as in Example 6.1). The remaining parameters can be found in Table 7.13. The operating point is given by the rotor speed $\omega_r = 0.8$ p.u., the length of the rotor flux $\psi^{\vartheta}_{dr} = 0.89$ p.u. and the load torque $T_\ell = 0.8$ p.u.. The lower and upper bounds on the torque and flux, respectively, are set to $T_{e,min} = 0.72$ p.u., $T_{e,max} = 0.88$ p.u., $\Psi^2_{s,min} = 0.82$ p.u. and $\Psi^2_{s,max} = 1.04$ p.u.. After deriving the PWA model on $\mathcal{X}^0$ (enlarged by 20 % as in Section 6.4.1), and determining the set $\mathcal{C}$, the core and the ring can be easily computed as shown in Fig. 7.5 for the voltage vector $u(k-1) = [1\ \ -1\ \ -1]^T$. On a Pentium IV with 2.8 GHz, this operation takes roughly 1 s.

Stage II For each new voltage vector $u(k) \in U \setminus u(k-1)$, the following procedure is performed for the initial set[3] $\mathcal{X}^0$. Initially, we set $n = 0$. Next, we map the polyhedra $\mathcal{X}^n$ from time-step $k+n$ to $k+n+1$ yielding $\mathcal{X}^{n+1}$. The states corresponding to infeasible outputs form the set $\mathcal{X}^{n+1}_{\text{infs}}$. Consequently, we map $\mathcal{X}^{n+1}_{\text{infs}}$ back to the time-step k and associate with them the number of time-steps n. We denote these polyhedra by $\mathcal{Q}^n_u$, where u corresponds to the chosen voltage vector $u(k)$, and n denotes the number of time-steps this voltage vector $u(k)$ can be applied to the set of states before any of the outputs

[3] Conceptually, this stage of the algorithm should be initialized with the ring $\mathcal{Q}^r_u$ rather than $\mathcal{X}^0$. Let us note though that since the facets of the initial set are mapped forward and backward in time, in the worst case, the complexity of the algorithm both in terms of the computation time and the number of resulting polyhedra is exponential in the number of facets of the initial set. Therefore, as $\mathcal{X}^0$ is by definition a very simple polytopic set with only a few facets, whereas the ring is a non-convex set with possibly many facets, we initialize Algorithm 7.2 with $\mathcal{X}^0$ rather than the ring.

violates a bound. If there remain any feasible states, we move one time-step forward in the future by increasing n by one and repeat the above procedure.

This yields for each new voltage vector a polyhedral partition of the ring $\{\mathcal{Q}_u^n\}_{n=0}^N$, where each polyhedron is associated with a unique number indicating for how many time-steps the respective voltage vector can be applied before any of the controlled variables violates a bound.

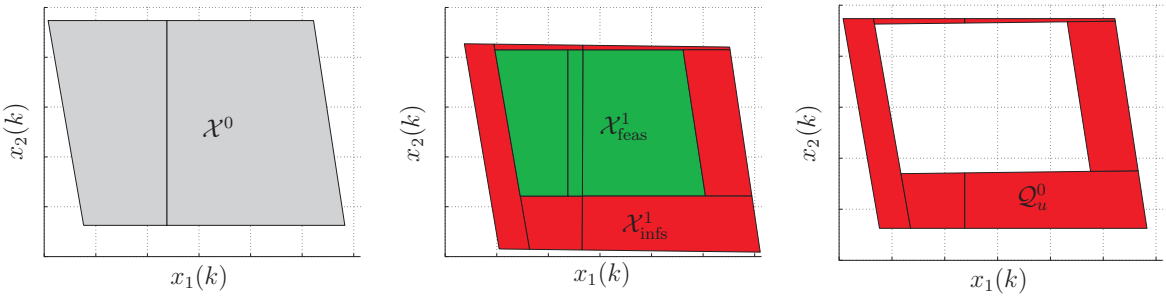
Next, the algorithm is summarized, where the two subfunctions `mapForw` and `mapBack` are affine transformations of polyhedra using the PWA model (2.9) for a fixed voltage vector $u(k)$. Specifically, `mapForw` yields $\mathcal{X}^{n+1} = \{f(x,u) \mid x \in \mathcal{X}^n, u = u(k)\}$, and `mapBack` yields $\mathcal{Q}_u^n = \{x \mid (f_u \circ \ldots \circ f_u)(x) \in \mathcal{X}_{\text{infs}}^{n+1}\}$, where we have set $f_u(x) = f(x,u)$ and concatenated f_u n times. Note that `mapForw` maps a set of states by one time-step forward in time, whereas `mapBack` maps a set of states by n time-steps backwards. The subscript feas (infs) refers to sets of states corresponding to feasible (infeasible) outputs.

Algorithm 7.2

function $\{\mathcal{Q}_u^n\}_{n=0}^N$ = MPC-FMB_explicit ($\mathcal{C}$, $\mathcal{X}^0$, u, N)
  $n = 0$
  while $\mathcal{X}^n \neq \emptyset$ **and** $n < N$
    $\mathcal{X}^{n+1}$ = mapForw ($\mathcal{X}^n$, u)
    $\mathcal{X}_{\text{feas}}^{n+1} = \mathcal{X}^{n+1} \cap \mathcal{C}$
    $\mathcal{X}_{\text{infs}}^{n+1} = \mathcal{X}^{n+1} \setminus \mathcal{X}_{\text{feas}}^{n+1}$
    $\mathcal{Q}_u^n$ = mapBack ($\mathcal{X}_{\text{infs}}^{n+1}$, u)
    $\mathcal{X}^{n+1} = \mathcal{X}_{\text{feas}}^{n+1}$
    $n = n + 1$
  endwhile
  $\mathcal{Q}_u^n$ = mapBack ($\mathcal{X}^n$, u)

Example 7.2 Setting $N = 4$, we proceed with Example 7.1. Fig. 7.6 visualizes the first step ($n = 0$) of Algorithm 7.2 in the x_1x_2 plane, where the same scaling is used for all three figures. Starting with the initial set of states $\mathcal{X}^0$ in Fig. 7.6(a), the voltage vector $u(k) = [1\ -1\ -1]^T$ maps $\mathcal{X}^0$ from time-step k to $k+1$ as shown in Fig. 7.6(b). The set $\mathcal{X}^1$ comprises two parts. $\mathcal{X}_{\text{feas}}^1$ ($\mathcal{X}_{\text{infs}}^1$) contains the states corresponding to feasible (infeasible) outputs. This set $\mathcal{X}_{\text{infs}}^1$ is consequently mapped back from time-step $k+1$ to k resulting in $\mathcal{Q}_u^0$ and indicating that this set is zero-step feasible for the chosen $u(k)$. Furthermore, we set $\mathcal{X}^1 = \mathcal{X}_{\text{feas}}^1$.

The second step ($n = 1$) is shown in Fig. 7.7 starting from the set $\mathcal{X}^1$ at time-step $k+1$ in Fig. 7.7(a). Applying $u(k) = [1\ -1\ -1]^T$ to this set maps it from time-step $k+1$ to $k+2$ as shown in Fig. 7.7(b). Again, $\mathcal{X}_{\text{feas}}^2$ ($\mathcal{X}_{\text{infs}}^2$) contains the states corresponding

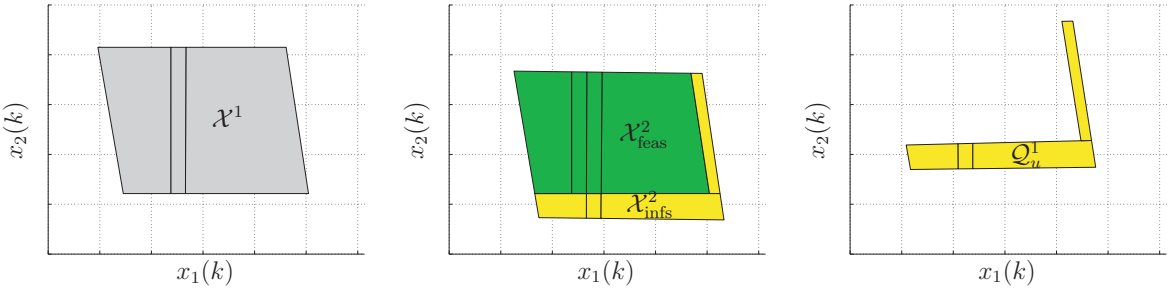


(a) Initial set $\mathcal{X}^0$ at time-step k

(b) Set $\mathcal{X}^1$ at time-step $k+1$

(c) Set $\mathcal{Q}^0_u$ at time-step k that yields infeasible outputs at $k+1$

Figure 7.6: First step of Algorithm 7.2 in the x_1x_2 plane for $u(k) = [1 \ -1 \ -1]^T$



(a) Set $\mathcal{X}^1$ at time-step $k+1$

(b) Set $\mathcal{X}^2$ at time-step $k+2$

(c) Set $\mathcal{Q}^1_u$ at time-step k that yields feasible outputs at time-step $k+1$, but infeasible outputs at $k+2$

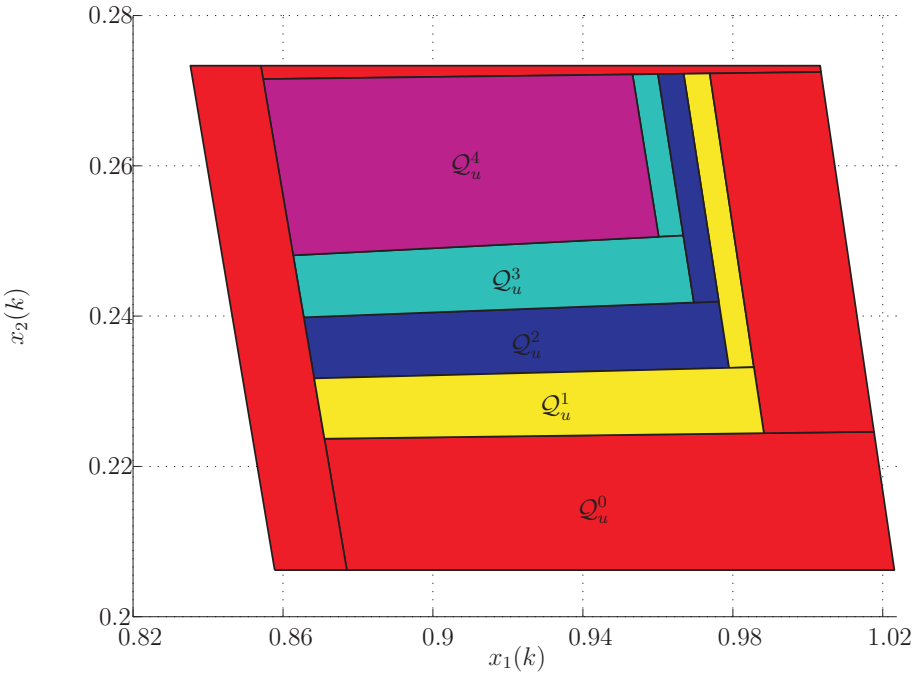
Figure 7.7: Second step of Algorithm 7.2 in the x_1x_2 plane for $u(k) = [1 \ -1 \ -1]^T$

to feasible (infeasible) outputs. The states in $\mathcal{X}^2_{\text{infs}}$ are mapped back for two steps from $k+2$ to k yielding $\mathcal{Q}^1_u$ which is shown in Fig. 7.7(c) and refers to states which are one-step feasible for $u(k)$.

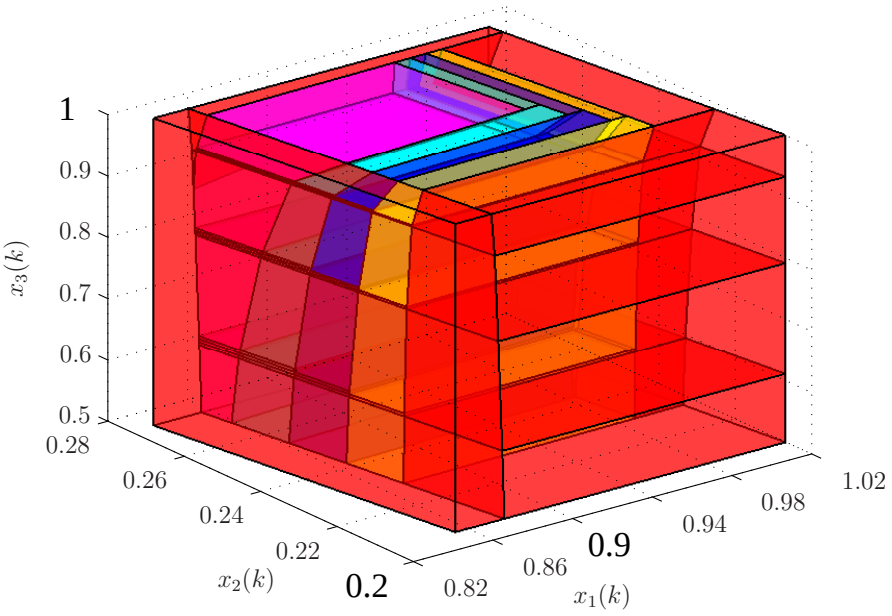
Repeating the above procedure for $n = 2, 3, 4$ and collecting the sets $\mathcal{Q}^n_u$ yields the polyhedral partition $\{\mathcal{Q}^n_u\}^4_{n=0}$ shown in Fig. 7.8. The outer polyhedra correspond to outputs that are feasible for zero time-steps when applying $u(k) = [1 \ -1 \ -1]^T$, while the inner polyhedra are feasible for one, two, three and four time-steps as $x_2(k)$ is increasing. Note that $\{\mathcal{Q}^n_u\}^4_{n=0}$ is by construction a polyhedral partition of the set $\mathcal{X}^0$.

The computation time for the second stage for the given example is approximately 2 min on a Pentium IV 2.8 GHz machine.

Summing up, Stages I and II yield a control law that is evaluated according to Algorithm 7.1, with the main difference that the number of steps n_u is not calculated by mapping operations but rather by set membership tests evaluating if the given state lies in the respective polyhedron. Specifically, if for the given $u(k-1)$, the state $x(k)$ lies in the



(a) Polyhedral partition in the x_1x_2 plane for $x_3 = 0.95$



(b) Polyhedral partition in the three-dimensional state space

Figure 7.8: The resulting polyhedral partition $\{\mathcal{Q}_u^n\}_{n=0}^N$ of Algorithm 7.2 for $u(k) = [1 \ -1 \ -1]^T$ and $N = 4$, where colors correspond to the number of steps n

core, re-apply the last voltage vector again. Else determine for each new voltage vector the polyhedron in $\{\mathcal{Q}_u^n\}_{n=0}^N$ containing $x(k)$, evaluate the associated number of time-steps n_u, and find the voltage vector $u(k)$ with the lowest cost as defined in Algorithm 7.1. This is formalized in Algorithm 7.3.

Algorithm 7.3

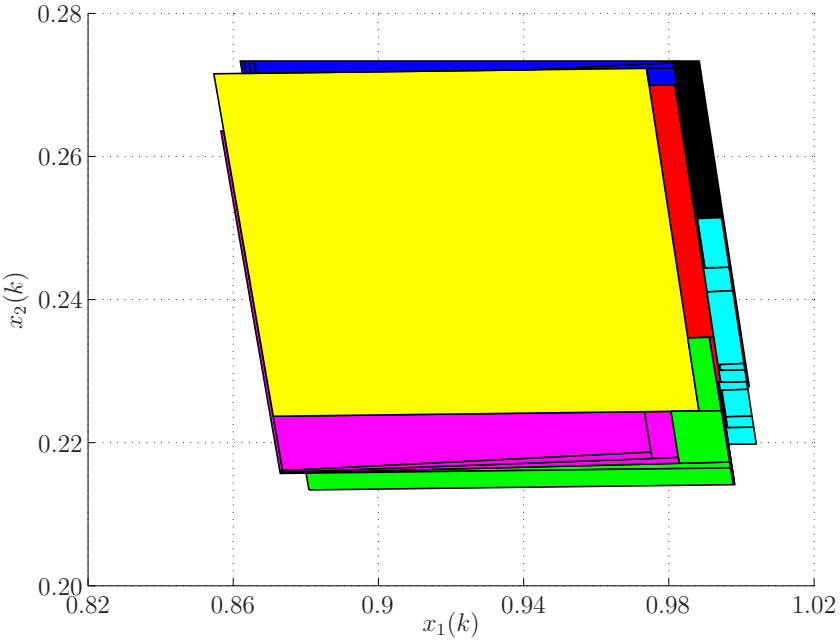
function $u(k) =$ MPC-FMB_evaluateLaw ($x(k)$, $u(k-1)$)
  if $x(k) \in \mathcal{Q}_u^c$
    $u(k) = u(k-1)$
  else
    for all $u(k) \in U \setminus u(k-1)$
      determine n_u such that $x(k) \in \mathcal{Q}_u^{n_u}$
    endfor
    $u(k) = \arg\min_{u(k)} \frac{||u(k)-u(k-1)||}{n_u}$
  endif

Note that the computation of the control input not only involves set membership tests, but also a minimization of a cost function. Therefore, we refer to this state-feedback controller as a *semi-explicit* control law.

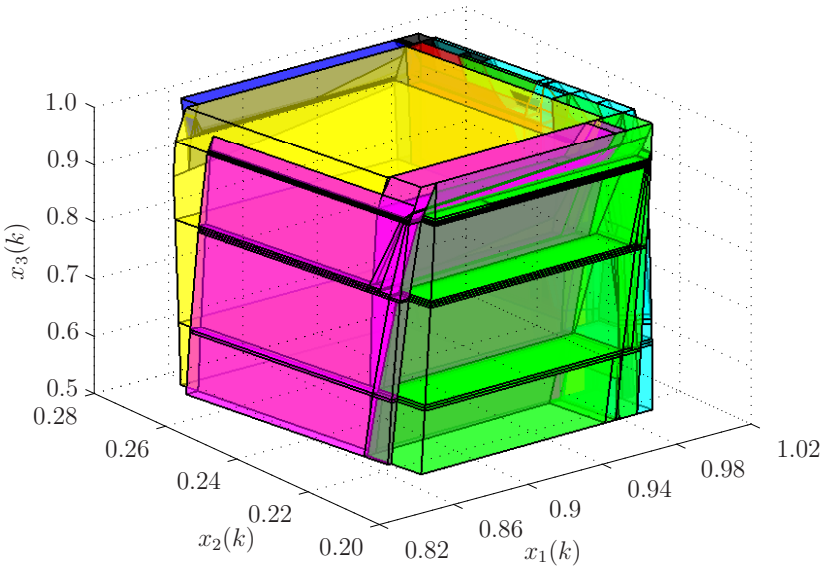
Regarding the computational burden for the on-line computation of the control input, in the worst case, one core and the seven polyhedral partitions of $U \setminus u(k-1)$, which feature in general a low number of polyhedra, need to be evaluated.

Stage III In the third stage we pre-compute Algorithm 7.3 and derive the (fully) explicit control law as a function of the last voltage vector $u(k-1)$ and the current state $x(k)$. For $u(k-1) \in U$, we evaluate for each polyhedron in $\{\mathcal{Q}_u^n\}_{n=0}^N$ the cost and associate with it the voltage vector $u(k)$. Next, the core $\mathcal{Q}_u^c$ is added with zero cost and the voltage vector $u(k) = u(k-1)$. Finally, we compare the cost expressions and iteratively remove (parts of) polyhedra with inferior costs[4]. A detailed exposition and analysis of this algorithm can be found in [BT03]. This yields one polyhedral partition, where each polyhedron refers to a voltage vector $u(k)$ (and not to a number of time-steps). This procedure is

[4]As the cost expressions used in Algorithms 7.1 and 7.3 are rational, where the nominator (in the case of a two-level inverter) is restricted to the integers two, four and six, and the denominator to $0, \ldots, N$, the costs take only a few different values. This increases the possibility that at a given time two or more voltage vectors have the same associated cost leading to ambiguities in the choice of the next voltage vector. In such cases, we suggest to remove the ambiguity by imposing an additional heuristic selection criterion. Examples for such rules are to select the vector that keeps the controlled variables feasible for the maximal number of steps, or to favor zero vectors. Obviously, these ambiguities occur less frequently when the maximal horizon N is increased.



(a) Polyhedral partition in the x_1x_2 plane for $x_3 = 0.95$



(b) Polyhedral partition in the three-dimensional state-space

Figure 7.9: Polyhedral partitions of the state-feedback control law resulting from Stage III for $u(k-1) = [1\ -1\ -1]^T$, where the colors correspond to the control inputs $u(k)$ ordered as in Table 7.4: red, black, green, cyan, yellow, blue, magenta and red

repeated for all the eight former voltage vectors $u(k-1)$ yielding eight different (fully) explicit state-feedback control laws. As a result, the computational burden of evaluating the control law is reduced, as $u(k-1)$ directly defines the one polyhedral partition that needs to be searched through in order to obtain $u(k)$. However, the memory requirements are higher since the polyhedral partitions of the fully explicit control law are in general more complex than the one of the semi-explicit control law.

Example 7.3 Applying Stage III to Example 7.2 yields for $u(k-1) = [1 \ -1 \ -1]^T$ the explicit control law shown in Fig. 7.9. Each color corresponds to a voltage vector $u(k) \in U$. In particular, the large polyhedron in the center of the three-dimensional state space refers to $u(k) = u(k-1)$. The explicit control law comprises a total of eight control laws similar to Fig. 7.9, where each one corresponds to a formerly applied voltage vector $u(k-1) \in U$. The computation was performed using the function `mpt_removeOverlaps` of the Multi-Parametric Toolbox [KGBM04]. The computation time was 15 min on the same machine as before.

Using the same set-up, we have computed in Section 7.2.5 the explicit control for MPC-PL depicted in Fig. 7.2. As Figs. 7.2 and 7.9 show for the same drive, operating point and discrete state (last voltage vector) the polyhedral partitions of the two control laws for MPC-PL and MPC-FMB, respectively, both controllers can be directly compared to each other. We observe three main points. Firstly, the two control laws are very similar, in the sense that the large center polyhedron refers to the repetition of the last voltage vector, the neighboring polyhedra refer to voltage vectors that involve a switch transition in one inverter stack, and the outer polyhedra refer to voltage vectors with multiple switchings. Yet, secondly, the polyhedral partition of the MPC-FMB control law is significantly simpler and contains by far less polyhedra. As mentioned in the introduction of this section, this reduced complexity is the prime motivation for MPC-FMB, which results from the simple control approach. Thirdly, the polyhedra partition of MPC-FMB is a subset of the partition of MPC-PL. This results from the fact that the polyhedral partition of MPC-FMB contains only states referring to outputs that are either feasible at the current time-step or will become feasible at the next time-step. In MPC-PL, however, the strict feasibility of the output variables is relaxed by the use of soft constraints. Therefore, the polyhedral partition of MPC-PL includes states whose outputs potentially may violate the bounds for more than one time-step.

7.3.3 Performance Evaluation

The simulation results presented in this section were derived for a DTC drive featuring a two-level inverter. The parameters of the drive are given in Table 7.13 and the operating point we consider is as in Example 7.1. As the parameters, the operating point and

N	Switching frequency [Hz]	Total number of polyhedra in semi-explicit control law	Total number of polyhedra in fully explicit control law
2	632	292	1192
3	606	440	1891
4	572	625	2226
5	540	860	2907
6	510	1051	3362
7	495	1256	3737
8	547	1467	4443
9	574	1694	4758

Table 7.5: Performance and complexity of the state-feedback control law for MPC-FMB

the bounds imposed on torque and flux are the same as for MPC-PL in the previous section, we can employ MPC-PL as a benchmark to which we compare the performance of the proposed MPC-FMB scheme in terms of the average inverter switching frequency. Recall that the corresponding state-feedback controller of MPC-PL, which was derived for a prediction horizon of two and features a total of 47'000 polyhedra, yields for the above setup an average switching frequency of 525 Hz. Note that in the MPC-PL scheme, switching is allowed at every time-step within the prediction horizon.

As mentioned before, the only design parameter which influences the calculation of the state-feedback controller and consequently the performance of the drive, is the maximal horizon N over which the feasibility of each voltage vector is considered. The results obtained with the MPC-FMB approach are summarized in Table 7.5 for eight different values of the maximal horizon N. For the horizon used in MPC-PL, i.e. $N = 2$, the switching frequency is significantly increased with respect to the benchmark. This is to be expected, since the move blocking strategy (no switching of the control input within the horizon) reduces the degrees of freedom of the control algorithm. However, setting the maximal horizon to $N = 5$ yields a switching frequency that is comparable to the one obtained with the MPC-PL approach, and the choices of $N = 6$ and $N = 7$ reduce the switching frequency. Most important, this performance improvement is achieved despite the complexity reduction of the state-feedback controller by an order of magnitude with respect to its MPC-PL counterpart.

Focusing on the case of $N = 7$, the relative switching frequency improvement with respect to the benchmark amounts to 5.7 %. Furthermore, we should point out that using MPC-FMB the bounds on the torque and the stator flux are very strictly respected. The

MPC-PL scheme, however, allows for small violations of the bounds; the degree of the violations can be adjusted using the penalties on the soft constraints as design parameters, which affect the switching frequency. As tightening the bounds increases the switching frequency, the performance improvement is even more pronounced.

For completeness, one should note that the switching frequency does not monotonically decrease with N. This phenomenon has also been observed with the MPC-PL scheme and is currently under investigation. In particular, a further increase of the horizon to $N = 8$ or $N = 9$ leads to a performance deterioration with respect to $N = 7$.

7.3.4 Extensions

The controller presented in this section could be extended in the following two ways. Firstly, by considering changes in the operating point. This necessitates the parametrization of the drive's PWA model over the rotational speed ω_r and the torque and flux bounds. Concerning the bounds, only one parameter is needed for the median of the torque bounds. The flux bounds and the width of the torque bounds can be assumed to be in general time-invariant. Obviously, the complexity of the resulting controller would be increased. Yet it is to be expected that the low complexity with respect to an accordingly extended MPC-PL scheme is maintained.

Secondly, this approach can be extended by applying the presented method to a DTC drive with a three-level inverter. As a result, two additional control objectives, namely the regulation of the inverter's neutral point potential and the even distribution of the switching effort between the upper and the lower half of the inverter, arise. A straightforward approach would be to accurately model the nonlinear dynamics of the neutral point potential. To avoid such a substantially more complex PWA model, a favorable approach is to refrain from deriving the fully explicit controller and to rather use the semi-explicit realization in combination with time-varying weights on the voltage vectors. An outer loop should monitor the neutral point potential and set the weights accordingly to favor the selection of voltage vectors that keep the potential within given bounds around zero. The same approach can be also used for the even distribution of the switching effort. Since these control objectives are roughly and heuristically defined, they do not require to be strictly met thus rendering the above approach a sufficient approximation.

7.4 MPC based on Extrapolation

In this section, we present a third MPC scheme for the DTC problem that can be considered as a combination of the two preceding MPC-PL and MPC-FMBconcepts. Specifically, we use a rather short horizon N and compute for the input sequences over the horizon

the evolution of the controlled variables using the prediction model. To emulate a long horizon, the "promising" trajectories are extrapolated and the number of time-steps is determined for which the controlled variables are kept within their bounds, thus yielding the length of the extrapolated trajectory[5]. The cost of each input sequence is then computed by dividing the total number of switch transitions in the sequence by the length of the trajectory. Minimizing this cost yields the optimal input sequence and the next control input to be applied. Hence, we refer to this scheme as *Model Predictive Control based on Extrapolation* (MPC-E).

The major benefit of this approach is its superior performance in terms of switching frequency, which is reduced with respect to ABB's scheme [ABB] over the whole range of operating points by up to 50 %, with an average reduction of 25 %. Furthermore, the controller needs no tuning parameters. As the computation of an explicit solution is avoided, all quantities may be time-varying including model parameters, set points and bounds. Those can be adapted on-line, making the concept applicable to the whole range of operating points. As all computations are performed on-line, the prediction model is not restricted to be PWA, allowing us to use the nonlinear discrete-time model, which is formulated in the stationary dq0 reference frame. This nonlinear model, which was introduced in Section 6.2, uses only one modelling simplification, namely the assumption that the rotational speed is constant within the prediction horizon[6].

The MPC-E scheme is available in two forms with $N > 1$ and $N = 1$, differing mostly in the degree of freedom for the input sequences and the handling of the switching constraints and thus in the computational burden and the performance. The control algorithm and the related computations are investigated in detail, allowing one to conclude that this control approach is basically implementable in its present form on currently available control hardware for DTC drives. Furthermore, the scheme is conceptually very simple and flexible.

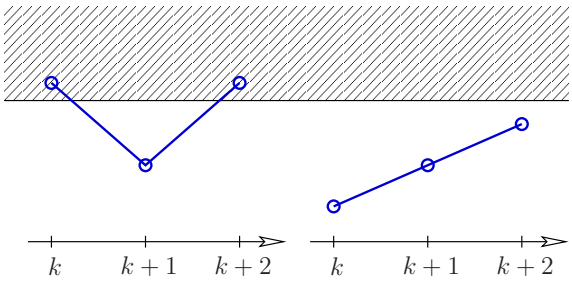
Here, we consider a three-phase three-level inverter driving a squirrel-cage induction motor, but the approach can be also applied to drives with inverters of arbitrary levels (e.g. two- or five-level inverters).

7.4.1 Horizon $N > 1$

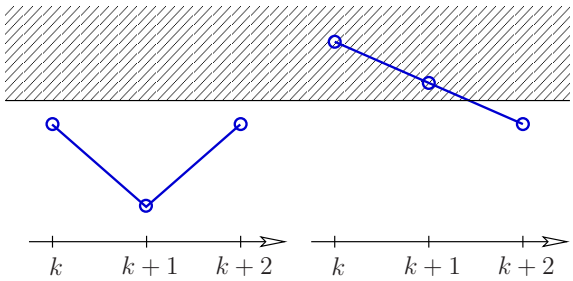
For MPC-E with a horizon larger than one, we consider only input sequences of length N that meet the switching constraints imposed by the physics of the inverter (see Fig. 5.4).

[5] We will later define in detail the notion of "promising" trajectories, to which we will refer as "candidate" trajectories.

[6] The distribution of the switching effort (which roughly resembles the switching losses) is not included in the model as closed-loop simulations with the MPC-E scheme indicate that the switching effort is evenly distributed even without enforcing this objective.



(a) Trajectories that are either feasible (left) or pointing in the proper direction (right)



(b) Trajectories that are neither feasible nor pointing in the proper direction

Figure 7.10: Example output trajectories for $N = 2$. The feasible region above the lower bound is hatched

Control Concept and Algorithm

Given the current state $x(k)$, the last control input $u(k-1)$, the bounds on the controlled variables, and using the discrete-time nonlinear prediction model (6.14) of the DTC drive, the controller computes at time-instant k the next control input $u(k)$ in the following way.

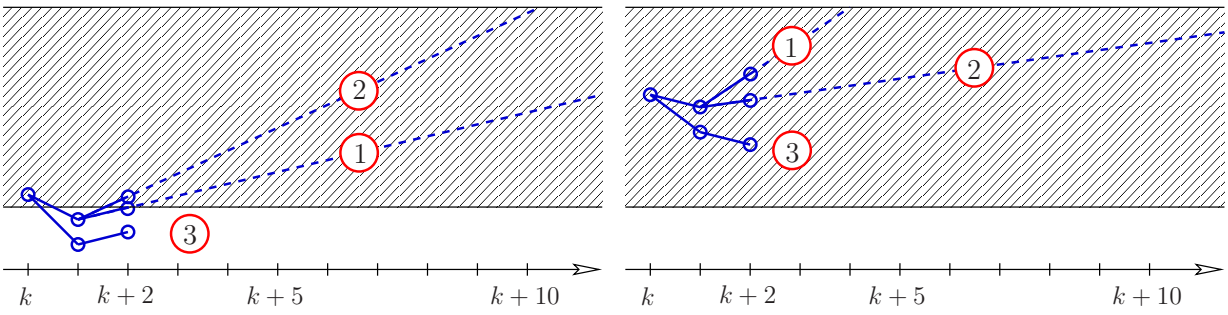
1. Given the last control input $u(k-1)$ and taking into account the constraints on the switch transitions induced by the inverter topology, determine all input sequences $U^i(k) = [u^i(k), \ldots, u^i(k+N-1)]$ over the horizon N, where $i \in \mathcal{I}$.

2. For these input sequences, compute the system response, i.e. compute all open-loop torque, stator flux and neutral point potential trajectories starting from $x(k)$ over the horizon N given by $Y^i(k) = [y^i(k), \ldots, y^i(k+N)]$. As the DTC drive has no feedthrough, the current output $y(k)$ depends only on the measured state $x(k)$ and is included solely for completeness.

3. Determine those input sequences that have output trajectories that are either *feasible* at the end of the horizon or *pointing in the proper direction* for all time-steps within the horizon. Feasibility means that the controlled variable lies within its corresponding bounds at time-step $k+N$; pointing in the proper direction stands for the situation in which a controlled variable is not necessarily feasible, but the degree of the violation is decreasing for all time-steps within the prediction horizon. Fig. 7.10 shows for $N = 2$ example trajectories that visualize these properties. The above condition needs to hold *componentwise*, i.e. for all three controlled variables[7]. We refer to those input sequences as *candidate* sequences $U^i(k)$ with $i \in \mathcal{I}_c \subseteq \mathcal{I}$.

4. For the candidate sequences, extrapolate the output trajectories from time-instant

[7] As an example, consider the following situation. The torque is feasible, the stator flux points in the proper direction, and the neutral point potential is feasible.



(a) Torque trajectories (b) Stator flux trajectories

Figure 7.11: Torque and stator flux trajectories of Example 7.4 starting at time-instant k. The trajectories within the prediction horizon $N = 2$ are solid, their extrapolations are dashed lines. The numbers refer to the indices of the input sequences. The feasible region between the upper and lower bound is hatched

$k + N$ on linearly[8] using the samples at $k + N - 1$ and $k + N$. Derive the number of time-steps when the first of the three output variables leaves the feasible region in between of the corresponding upper and lower bound[9]. This yields the number of time-steps before the next predicted switching n_i, $i \in \mathcal{I}_c$.

5. Compute for each candidate sequence $i \in \mathcal{I}_c$ the cost

$$c_i = \frac{s_i}{n_i}\,, \tag{7.13}$$

where

$$s_i = \sum_{\ell=k}^{k+N-1} ||u_i(\ell) - u_i(\ell-1)||_1 \tag{7.14}$$

is the total number of switch transitions of the input sequence $U^i(k)$. This cost approximates the average switching frequency by the number of switch transitions over the number of time-steps the i-th candidate sequence can be applied before switching again. The number of time-steps can be interpreted as a time-varying horizon. Next, choose the sequence $U^i(k)$ with the minimum cost.

6. Apply the first control input $u^i(k)$ of the sequence.

[8] In particular for high speed operation, it is advantageous to extrapolate the stator flux quadratically using also the flux sample at $k + N + 1$. The latter is computed by applying the control input $u(k + N) = u(k + N - 1)$.

[9] Note that we determine when the first output variable leaves the feasible region rather than when it hits a bound. This is done to account for situations in which an output variable lies outside its bounds but steers towards one of them.

Index i	Steps n_i	Switching transitions s_i	Cost c_i
1	4	1	$\frac{1}{4}$
2	10	2	$\frac{1}{5}$
3	–	–	–

Table 7.6: Characteristics of the three input sequences in Example 7.4

Example 7.4 To visualize the control concept, consider the example shown in Fig. 7.11. Assume there exist only three input sequences $U^i(k)$, $i \in \mathcal{I} = \{1, 2, 3\}$ over the horizon $N = 2$. According to the definition, $U^1(k)$ and $U^2(k)$ are candidate sequences, whereas $U^3(k)$ is not. Extrapolating the torque and the stator flux trajectories and determining when they leave the feasible region leads to the results summarized in Table 7.6. Minimizing the cost yields the sequence $U^2(k)$. Note that without extrapolation, the corresponding cost expressions would have been $\frac{1}{2}$ and 1 for $U^1(k)$ and $U^2(k)$, respectively, and the controller would have selected $U^1(k)$. In the long run, this choice would have been inferior compared with $U^2(k)$ thus motivating the use of the extrapolation. In this example, we have neglected the neutral point potential for the sake of simplicity. It is treated in precisely the same way as the torque and the stator flux.

The computation of the next control input is drastically sped up by first evaluating whether switching can be avoided altogether, i.e. whether the controlled variables are at time-step $k+N$ within their respective bounds when reapplying the last control input for N steps. Only if this simple test fails, the above proposed computations need to be performed. Furthermore, bound techniques can be added to prune suboptimal branches thus avoiding the computation of the whole tree over N steps. In any case, the necessary computations are simple as only a few multiplications and comparisons need to be performed. The pseudo code of the control algorithm and a detailed analysis of the computational effort will follow at the end of this section.

Features

As the control input is computed on-line using the state-space model for the machine and the neutral point potential of the inverter, great flexibility is achieved. In particular, the following changes can be performed on-line. (i) Adaptation of the model parameters (like the stator resistance), (ii) changes in the operating point (e.g. in the rotational speed), and (iii) changes in the bounds on the torque, stator flux or neutral point potential. This flexibility is achieved by refraining from pre-solving the control problem off-line and determining the control input by evaluating the resulting look-up table. Furthermore, using the original nonlinear model and avoiding PWA approximations avoids artificial

restrictions and limitations and keeps the control scheme versatile.

The notion of the candidate sequences with output trajectories, which are elementwise either feasible at the end of the horizon or pointing towards the bounds, leads to the following favorable properties. Firstly, in cases where the bounds are abruptly shifted such that some or all of the controlled variables lie suddenly outside their bounds, an input sequence that moves these controlled variables back inside the bounds within N steps does not always exist. This problem is pronounced in the case of large steps on the torque reference. Considering also sequences that move the controlled variables only towards rather than inside the bounds avoids the problem of infeasibility of the control scheme. Secondly, excessive switching is avoided when the bounds are slightly shifted or when measurement noise or model uncertainties affect the predictions in an adverse way. Note that the bounds on the controlled variables are in general not strictly imposed by the MPC-E scheme, as output trajectories of candidate sequences may violate the bounds. For examples, see Figs. 7.10(a) and 7.11. As a result, one or more outputs might slightly violate their bounds before a new voltage vector is selected.

As was the case for the MPC-PL approach, the proposed MPC-E scheme is intended to capture and minimize the average switching frequency, too. Therefore, a long prediction interval is beneficial. To avoid an explosion of the related computational complexity, we use a short prediction *horizon* N (usually two or three steps), but a long prediction *interval* (up to 80 steps). This is achieved by extrapolating the output trajectories starting at the end of the horizon until the first controlled variable hits one of its bounds. Linear extrapolation is very easy to implement and computationally very cheap. Note that this approach is closely related to blocking control moves and the *Multiple-Rate Prediction Model Approach* that we have introduced for MPC-PL. As the simulation results will show, this allows us to greatly increase the length of the prediction interval thus enhancing the performance of MPC-E while keeping the computation times short.

Nevertheless, as constraints on the allowed switch transitions are present, short prediction horizons N restrict the set of voltage vectors that can be reached within the horizon. More specifically, up to four consecutive steps are necessary to switch from one voltage vector to another (e.g. from $[-1\ -1\ -1]^T$ to $[1\ 1\ 1]^T$ and vice versa, see Fig. 5.4). Thus, a horizon $N = 2$ is too short to ensure that any arbitrary voltage vector can be reached within the prediction horizon. This issue becomes apparent for $N = 2$ only very rarely (every few seconds), because those unreachable voltage vectors involve multiple switch transitions and are thus very expensive in terms of the cost expression. By far more frequent are infeasibilities due to the discrete nature of the voltage vectors and bounds on the controlled variables that are overly tight. Section 7.5 analyzes and visualizes this issue.

Most important, the proposed control scheme can be easily adapted to different drives

stateUpdate	output	candidate	extrapolate	switches
$16+1+8+3$	10	33	19	3

Table 7.7: Number of operations in the subfunctions of the pseudo code for MPC-E with horizon $N = 2$

with different motors and inverter topologies, as only the motor and inverter models need to be updated, and because tuning parameters are missing. Furthermore, possible constraints on the allowed switch transitions for the inverter can be easily taken into account by using a look-up table.

Pseudo Code and Analysis of Computational Effort

In this section, we present the control algorithm for MPC-E with $N > 1$ as pseudo code and analyze the related computational burden in detail. To simplify the exposition, the horizon is fixed to $N = 2$. As with modern Digital Signal Processors (DSP) all basic operations such as additions, multiplications, divisions and comparisons require one cycle, we only count the number of basic operations and do not distinguish between them. Furthermore, we assume that each voltage vector is associated with a unique integer number (e.g. between 0 and 26), and that evaluating a look-up table requires one operation. Possible operations for the loading or storing of variables and the execution of loops are neglected.

The pseudo code uses five subfunctions, which are detailed and analyzed in Section 7.4.3, and whose number of operations are summarized in Table 7.7. The code differs slightly from the control scheme presented in the beginning of this section allowing for a more efficient implementation. To facilitate the readability, the code is divided into the four blocks I to IV.

Block I Let $F(k)$ denote the set of voltage vectors to which the inverter can switch from $u(k-1)$. For all voltage vectors $u(k) \in F(k)$, compute the corresponding output trajectories from k to $k+1$. Store the s_1 results in the data structure `auxPolicy`.

```
F(k) = set of feasible voltage vectors at time-instant k given u(k-1);
s1 = 0; auxPolicy = [];
for u(k) in F(k)
    x(k+1) = stateUpdate(x(k), u(k));
    y(k+1) = output(x(k+1));
    s1 = s1+1;
    store x(k+1), y(k+1) and u(k) in auxPolicy{s1};
end
```

Number of operations: Because the derivation of the set $F(k)$ can be pre-computed and stored in a look-up table, the first line of the code requires only one operation. As $x(k)$ is constant throughout Block I, the Term A in the function `stateUpdate` (see Section 7.4.3) needs to be computed only once yielding $16+(1+8+3)s_1$ operations as shown in Table 7.7. The function `output` leads to $10s_1$ operations.

Block II Branch on the s-th input sequence in `auxPolicy` by adding all feasible voltage vectors at $k+1$ to the input sequence and extend the corresponding output trajectories. Store the s_2 results in the data structure `Policy`.

```
s2 = 0; Policy = [];
for s = 1 to s1
    F(k+1) = set of feasible voltage vectors at time-instant k+1 given
             auxPolicy{s}.u(k);
    for u(k+1) in F(k+1)
        x(k+2) = stateUpdate(auxPolicy{s}.x(k+1), u(k+1));
        y(k+2) = output(x(k+2));
        s2 = s2+1;
        store auxPolicy{s}, x(k+2), y(k+2) and u(k+1) in Policy{s2};
    end
end
```

Number of operations: Similar to Block I, $x(k+1)$ is constant for fixed s leading to $16s_1 + (1+8+3)s_2$ operations for the function `stateUpdate` and $10s_2$ operations for the function `output`. The look-up table for $F(k+1)$ is evaluated s_1 times.

Block III Determine the candidate trajectories, extrapolate them and compute the cost expressions.

```
for s = 1 to s2
    Y(k) = [Policy{s}.y(k) Policy{s}.y(k+1) Policy{s}.y(k+2)];
    if candidate(Y(k))
        U(k) = [Policy{s}.u(k) Policy{s}.u(k+1)];
        cost{s} = switches(u(k-1), U(k)) / extrapolate(Y(k));
    else
        cost{s} = inf;
    end
end
```

Number of operations: Assuming the worst case, namely that all s_2 input sequences are candidate sequences, leads to $(33+3+19+1)s_2 = 56s_2$ operations according to Table 7.7.

Block	I	II	III	IV
Operations	$17 + 22s_1$	$17s_1 + 22s_2$	$56s_2$	$s_2 - 1$

Table 7.8: Number of operations per block for the MPC-E scheme with $N = 2$

Block IV Minimize the cost and get the next voltage vector (control input).

```
s = arg min(cost);
u(k) = Policy{s}.u(k);
```

Number of operations: Minimizing over the s_2-dimensional cost vector requires $s_2 - 1$ operations.

Table 7.8 summarizes the number of operations per block. The total number of operations amounts to $16 + 39s_1 + 79s_2$. For the three-level inverter with the particular constraints on the allowed switch transitions, s_1 is upper bounded by 13 and s_2 is upper bounded by 121. Therefore, the upper bound on the total number of operations at time-step k is given by 10'082. Note that the computationally most expensive parts of the code, namely Block III and IV, which require 70 % of the total computation power, can be easily parallelized.

7.4.2 Horizon $N = 1$

The control concept proposed in the last section with a horizon larger than one is conceptually very simple. Nevertheless, the computational burden might still exceed the capabilities of some of the existing DTC control hardware. Furthermore, given the constraints on the switch transitions, short prediction horizons may impose restrictions on the set of reachable voltage vectors. To eliminate this issue and to further reduce the computation time, we propose in this section a modified scheme that uses a horizon of $N = 1$ and ignores the switching constraints in a first step. The switching constraints are imposed in a last step by building a feasible input sequence whose first element constitutes the next voltage vector to be applied. As a result, the upper bound on the computational effort is reduced by a factor of five with respect to MPC-E with $N = 2$.

Given two voltage vectors μ and ν, $\mu, \nu \in \{-1, 0, 1\}^3$, a *feasible switching sequence* is a sequence of voltage vectors that transforms μ to ν via intermediate voltage vectors while taking into account the restrictions on the allowed switch transitions. From the fact that switching in one component of a voltage vector (one stack of the inverter) by one switch transition at a time-step is always possible, and because the components of the voltage vectors are independent from each other, it follows directly that there always exists a feasible switching sequence with $||\mu - \nu||_1$ switch transitions. Therefore, the constraints

on the allowed switch transitions make it in general impossible to switch within one time-step from μ to ν, yet they do not introduce additional switchings.

The feasible switching sequence is stored in a look-up table, which features only the first voltage vector of the sequence. As the switching sequence is in general not unique, we use the following rules to narrow the choices. (i) Follow the shortest path (in terms of time-steps); (ii) choose the voltage vector that yields the least number of switch transitions in the first step; (iii) choose the voltage vector that gives the most alternatives at time-step $k+1$. This leads to a look-up table of dimension 27×27.

Additionally, one may exploit the $\frac{2\pi}{3}$ symmetry of the voltage vectors in the dq0 plane where three sectors are defined. In a first step, we need to determine the sector i, $i \in \{0, 1, 2\}$, where μ lies in. This can be done either using on-line computations, or by storing the associated sector number for each vector in a look-up table with 27×1 entries. Next, the vectors μ and ν are mapped into the zero sector by shifting their components upwards i times. In the zero sector, we evaluate a look-up table holding the first voltage vector of the feasible switching sequence. As the zero sector comprises three zero, four short and four long voltage vectors, the look-up table has dimension 11×27. In a last step, map the derived voltage vector from the zero sector into the i-sector by shifting its components downwards i times. Summing up, exploiting the $\frac{2\pi}{3}$ symmetry reduces the size of the look-up table and thus the memory requirements, but adds a few (simple) computations.

Control Concept and Algorithm

Setting the horizon to $N = 1$, the control algorithm is the same as described in Section 7.4.1 with the following differences, where the numbering corresponds to Section 7.4.1.

1. Given the last control input $u(k-1)$ and *ignoring* the constraints on the switch transitions induced by the inverter topology, 27 input sequences of length one result. For consistency, set $U(k) = u(k)$.

6. Read out from the look-up table the first control input of the feasible switching sequence from $u(k-1)$ to $u(k)$, and apply it.

Features

The features of MPC-E with $N = 1$ are the same as for MPC-E with $N > 1$ described in Section 7.4.1 except of the following difference. Relaxing the constraints on the switch transitions and using the notion of the feasible switching sequence avoids the restriction on the set of voltage vectors that can be reached within N steps. More precisely, any voltage vector can be chosen, yet only the first voltage vector of the corresponding feasible

`stateUpdate`	`output`	`candidate`	`extrapolate`	`switches`	`feasiblePath`
$16+1+8+3$	10	24	19	1	8

Table 7.9: Number of operations in the subfunctions of the pseudo code for MPC-E with horizon $N = 1$

switching sequence is implemented. This is done in accordance with the receding horizon policy.

Apart from that, bounds on the controlled variables are handled in a different way. In cases where the chosen voltage vector does not conflict with the switching constraints, i.e. the desired voltage vector can be directly used and the feasible switching sequence is of length one, the bounds are strictly respected. However, if the feasible switching sequence comprises more than one element, the bounds are not guaranteed to be strictly respected.

Pseudo Code and Analysis of Computational Effort

In this section, we present the corresponding control algorithm as pseudo code and analyze the related computational burden. As the analysis will show, the upper bound on the computational burden is reduced by a factor of five with respect to MPC-E with $N = 2$. The pseudo code uses the same five subfunctions as with $N > 1$ and additionally the subfunction `feasiblePath` that determines the first voltage vector of a feasible switching path as shown and analyzed in Section 7.4.3. The code is divided into the three blocks I to III.

Block I Let F denote the set of all 27 voltage vectors. For all voltage vectors $u(k) \in F$, compute the corresponding output trajectories from k to $k+1$. Store the s_1 results in the data structure `Policy`.

```
F = set of all voltage vectors;
s1 = 0; Policy=[];
for u(k) in F
    x(k+1) = stateUpdate(x(k), u(k));
    y(k+1) = output(x(k+1));
    s1 = s1+1;
    store x(k+1), y(k+1) and u(k) in Policy{s1};
end
```

Number of operations: As for the algorithm with $N > 1$, the state vector $x(k)$ is constant throughout Block I making it possible to compute the Term A in the function `stateUpdate`

Block	I	II	III
Operations	610	1215	34

Table 7.10: Number of operations per block for the MPC-E scheme with $N = 1$

only once. According to Table 7.9, this yields $16 + (1 + 8 + 3) \cdot 27 = 340$ operations. The function `output` requires $10 \cdot 27 = 270$ operations.

Block II Determine the candidate trajectories, extrapolate them and compute the cost expressions.

```
for s = 1 to 27
    Y(k) = [Policy{s}.y(k) Policy{s}.y(k+1)];
    if candidate(Y(k))
        U(k) = [Policy{s}.u(k)];
        cost{s} = switches(u(k-1), U(k)) / extrapolate(Y(k));
    else
        cost{s} = inf;
    end
end
```

Number of operations: Assuming the worst case, namely that all 27 input sequences are candidate sequences, leads to $(24 + 1 + 19 + 1) \cdot 27 = 1215$ operations.

Block III Minimize the cost and get the preferred (relaxed) next voltage vector. Taking into account the constraints on the switch transitions, determine a feasible next voltage vector (control input) that lies on the feasible switching path.

```
s = arg min(cost);
ur(k) = Policy{s}.u(k);
u(k) = feasiblePath(u(k-1), ur(k));
```

Number of operations: Minimizing over the 27-dimensional cost vector requires 26 operations, and determining the actual voltage vector $u(k)$ using the notion of the feasible switching path requires at most eight operations as shown in Table 7.9.

Table 7.10 summarizes the number of operations per block. The total number of operations is upper bounded by 1859.

7.4.3 Subfunctions of Pseudo Codes

In this section, we present the functions `stateUpdate`, `output`, `candidate`, `extrapolate`, `switches` and `feasiblePath`, which are subfunctions of the two pseudo codes, and analyze the related computational effort.

Subfunction `stateUpdate` Given the state $x(k)$ and the control input $u(k)$, the state-update function computes the successive state $x(k+1)$ using the discrete-time nonlinear model of the drive (6.14a) which is repeated here for the ease of readability.

$$x(k+1) = \underbrace{(I + \begin{bmatrix} A & 0 \\ 0 & 0 \end{bmatrix} T_s)x(k)}_{\text{Term A}} + \underbrace{\begin{bmatrix} B_1 \\ 0 \end{bmatrix} T_s u(k)}_{\text{Term B}} + \underbrace{\begin{bmatrix} 0 \\ B_2(x(k)) \end{bmatrix} T_s |u(k)|}_{\text{Term C}} \tag{7.15}$$

In the sequel, we derive for each of the three terms in (7.15) the number of operations. In a last step, the three terms are summed up.

- Term A: Recalling (6.11) and considering

$$I + \begin{bmatrix} A & 0 \\ 0 & 0 \end{bmatrix} T_s \tag{7.16}$$

  as a 5×5 parameters matrix that is multiplied with the five-dimensional state vector, ten non-trivial multiplications and six additions result yielding a total of sixteen operations.

- Term B: Using (6.12), the computation of the term

$$B_1 T_s u(k) = \frac{V_{dc}}{2} \begin{bmatrix} \frac{2}{3} & -\frac{1}{3} & -\frac{1}{3} \\ 0 & \frac{\sqrt{3}}{3} & -\frac{\sqrt{3}}{3} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} T_s u(k)\,, \tag{7.17}$$

  requires eight operations. These can be avoided by exploiting the fact that the voltage vectors belong to a discrete finite set and that the reference frame is fixed (not rotating). This allows us to pre-compute and store Term B in a look-up table. Encoding the 27 voltage vectors with integers and taking into account that Term B has only two non-trivial rows yields a look-up table of dimension 27×2.

- Term C: Rewriting (6.12) yields

$$B_2(x(k)) T_s |u(k)| = x^T(k) \frac{1}{2x_c} \begin{bmatrix} \frac{x_{rr}}{D} & -\frac{1}{2}\frac{x_{rr}}{D} & -\frac{1}{2}\frac{x_{rr}}{D} \\ 0 & \frac{\sqrt{3}}{2}\frac{x_{rr}}{D} & -\frac{\sqrt{3}}{2}\frac{x_{rr}}{D} \\ -\frac{x_m}{D} & \frac{1}{2}\frac{x_m}{D} & \frac{1}{2}\frac{x_m}{D} \\ 0 & -\frac{\sqrt{3}}{2}\frac{x_m}{D} & \frac{\sqrt{3}}{2}\frac{x_m}{D} \\ 0 & 0 & 0 \end{bmatrix} T_s |u(k)|\,. \tag{7.18}$$

Term A	Term B	Term C	Summation
16	1	8	3

Table 7.11: Number of operations for the subfunction `stateUpdate`

Similar to above, most of the computations can be pre-computed, namely the computation of the componentwise absolute input vector $|u(k)|$ and its multiplication with the 5×3 matrix. As the results is a 5×1 vector, whose fifth component is always zero, it can be stored in a 27×4 look-up table. Only the scalar product between $x(k)$ and the vector stored in the look-up table needs to be computed on-line summing up to seven operations and one evaluation of a look-up table.

- Summation: In a last step, Terms A, B and C are summed up. To simplify the analysis, we rewrite (7.15) as

$$x(k+1) = \begin{bmatrix} * \\ * \\ * \\ * \\ * \end{bmatrix} + \begin{bmatrix} * \\ * \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ * \end{bmatrix}, \tag{7.19}$$

where the three vectors correspond to Terms A, B and C, and the non-zero elements are depicted as stars. Thus, only three additions are needed.

Considering an evaluation of a look-up table as one operation, the number of operations for each term is summarized in Table 7.11.

When adapting model parameters on-line like the stator resistance r_s or the rotational speed ω_r, the expression (7.16) needs to be updated. When the dc-link voltage V_{dc} changes, the look-up table storing Term B needs to be recomputed. All those operations are not time critical as the parameters are changing slowly with respect to the sampling time $T_s = 25\,\mu$s and can be thus performed by an additional outer loop.

Subfunction `output` For the state $x(k)$, this function yields the corresponding output $y(k)$ using the nonlinear output function of the drive (6.14b)

$$y(k) = \begin{bmatrix} \frac{x_m}{D}(x_2(k)x_3(k) - x_4(k)x_1(k)) \\ \sqrt{x_1^2(k) + x_2^2(k)} \\ x_5(k) \end{bmatrix}. \tag{7.20}$$

The total number of operations[10] sums up to ten.

[10]Note that one can avoid the computation of the square root by using the *squared* length of the stator flux as an output rather than its length thus reducing the number of operations to nine. Obviously,

Subfunction `candidate` To simplify the exposition, assume for now a horizon $N = 2$. Then, this function checks whether the input sequence $U(k) = [u(k)\ u(k+1)]$ with the corresponding vector valued output trajectory $Y(k) = [y(k)\ y(k+1)\ y(k+2)]$ is a candidate sequence. This corresponds to Point 3 in Section 7.4.1. Let $Y_j(k) = [y_j(k)\ y_j(k+1)\ y_j(k+2)]$ denote the j-th row of the vector valued trajectory, $j \in \{1, 2, 3\}$, which corresponds to the evolution of one of the three output variables.

For the j-th trajectory, the following operations need to be performed.

- Feasibility at the end of the horizon $N = 2$. Determining whether $y_j(k+2)$ lies in between of its bounds requires three operations.
- Pointing in the proper direction for all time-steps within the horizon. Compute for $y_j(k)$, $y_j(k+1)$ and $y_j(k+2)$ their corresponding distances $d_j(k)$, $d_j(k+1)$ and $d_j(k+2)$ to the reference which lies between the bounds. Evaluate whether $d_j(k+1) \leq d_j(k)$ and $d_j(k+2) \leq d_j(k+1)$. This sums up to six operations.
- Checking if at least one of the two conditions holds is equivalent to one operation.

The above computations need to be done for $j = 1, 2, 3$ and then combined with an `and` operation. This leads to $3(3 + 6 + 1) + 3 = 33$ operations. In the case of MPC-E with $N = 1$, the number of operations reduces to $3(3 + 3 + 1) + 3 = 24$.

Subfunction `extrapolate` The extrapolation function is used to extrapolate either linearly or quadratically the vector valued output trajectory $Y(k) = [y(k)\ \ldots\ y(k+N)]$, and to determine the number of time-steps at which the first output variable leaves the feasible region between the bounds. This corresponds to Point 4 in Section 7.4.1. Let $Y_j(k) = [y_j(k)\ \ldots\ y_j(k+N)]$ denote the j-th row of the vector valued trajectory, with $j \in \{1, 2, 3\}$.

For the j-th trajectory, linear extrapolation involves the following operations. Let $\lambda_j = y_j(k+N) - y_j(k+N-1)$ denote the j-th slope from $k+N-1$ to $k+N$, and $y_{j,min}$ and $y_{j,max}$ the corresponding lower and upper bound, respectively. The number of time-steps at which the extrapolated trajectory starting at $k+N$ leaves the feasible region is given by

$$n_j = \frac{\max\left(y_{j,max} - y(k+N)\,,\ \ y_{j,min} - y(k+N)\right)}{\lambda_j}\,. \tag{7.21}$$

This procedure requires five operations and needs to be performed for $j = 1, 2, 3$. Taking the minimum and adding the length of the horizon yields the number of time-steps $n = \min(n_1, n_2, n_3) + N$ at which the first output variable leaves the feasible region. A total of $3 \cdot 5 + 4 = 19$ operations is necessary for that. Here, we have presented only the linear extrapolation. Quadratic extrapolation requires roughly twice as many operations.

the bounds on the stator flux need to be adapted accordingly.

Subfunction `switches` This function computes for an input sequence $U(k) = [u(k) \dots u(k+N-1)]$ starting from $u(k-1)$ the total number of switch transitions given by (7.14). The computation of $||u(\ell) - u(\ell-1)||_1$ would require eight operations. Alternatively, we suggest to use a (large) look-up table with 27×27 elements leading to $2N-1$ operations.

Subfunction `feasiblePath` This function determines the first component of the feasible switching path from the previously used voltage vector $u(k-1)$ to the one selected by the control algorithm $u_r(k)$ where the switching constraints had been relaxed. As described in Section 7.4.2, the following operations are needed. One look-up evaluation to determine the sector number $u(k-1)$ lies in, at most four operations for mapping the $u(k-1)$ and $u_r(k)$ into the zero sector, another look-up table evaluation to derive $u(k)$ in the zero sector, and at most two shifting operations to map the voltage vector into the initial sector. Summing up, eight operations result in the worst case.

7.4.4 Performance Evaluation

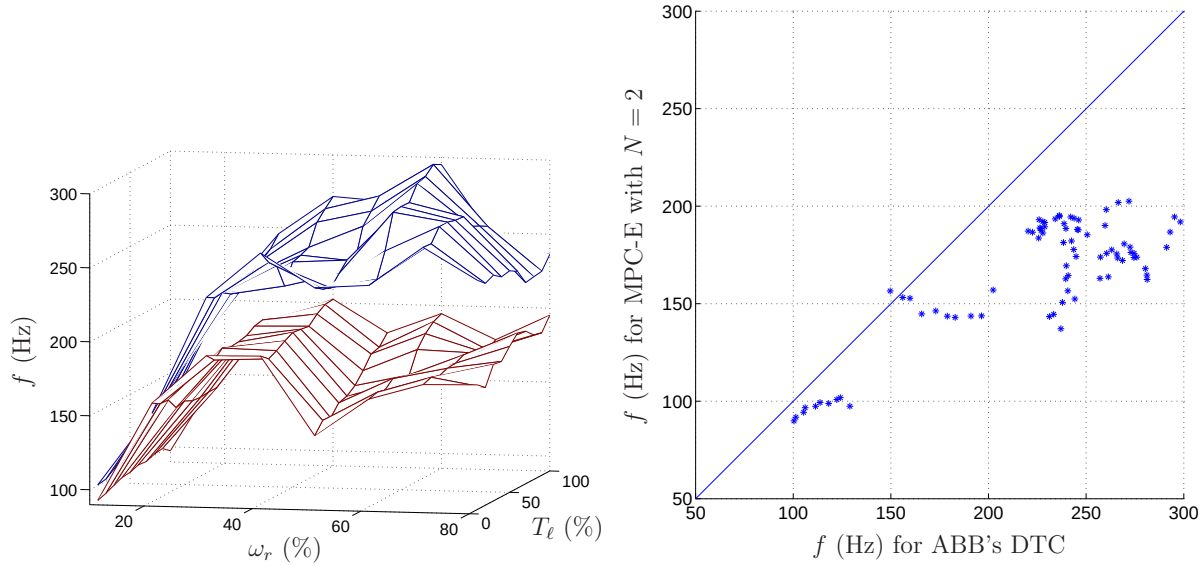
This section compares the performance of the proposed MPC-E scheme with ABB's ACS6000 drive [ABB] via simulations over the whole range of operating points. The comparison is done in terms of the average switching frequency and the mean squared violation[11] of the torque and stator flux violation of the imposed bounds. For both control schemes, the same bounds were used for the torque and flux. For the neutral point potential they were chosen to reflect the behavior of ABB's control scheme, thus ensuring that the comparison is meaningful.

Three case studies were considered with DTC drives of medium and low power. The respective ratings and parameters are summarized in Tables 7.13, 7.14 and 7.15 at the end of this chapter, whereas Table 7.3 provides a rough overview of the case studies. The evaluation was done for the whole range of operating points by gridding the speed ω_r and the load torque T_ℓ at $0.1, 0.2, \dots 1.0$ p.u.. The case of very high speed (0.9 and 1.0 p.u.) was left out for Case Studies I and II as the dc-link voltage is too low to allow for a reasonable operation at high speed. For each operating point, the simulations were run for 2 s.

Horizon $N > 1$

Starting with MPC-E with a horizon larger than one as proposed in Section 7.4.1, we set the horizon to $N = 2$ and restrict the number of input sequences by imposing an upper bound of three on the total number of switch transitions over the horizon. This bound

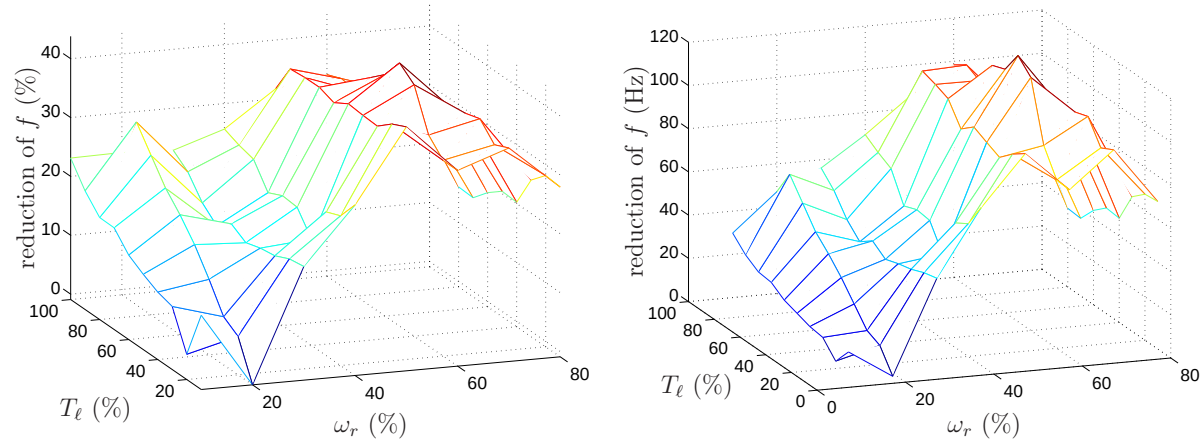
[11] We are using the mean squared error rather than the *root* mean squared (rms) error.



(a) Switching frequency of ABB's DTC (upper surface) compared with MPC-E (lower surface)

(b) Switching frequency of ABB's DTC versus MPC-E

Figure 7.12: Case Study I: Comparison of switching frequency f of ABB's DTC with respect to MPC-E with $N = 2$ over the grid of operating points



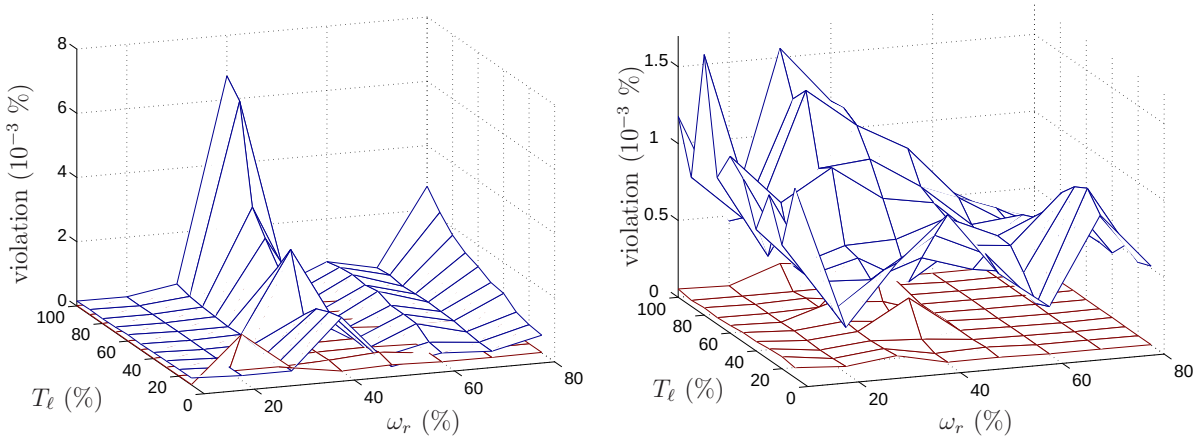
(a) Relative reduction of switching frequency (in per cent)

(b) Absolute reduction of switching frequency (in Hz)

Figure 7.13: Case Study I: Reduction of the switching frequency f of MPC-E with $N = 2$ relative to ABB's DTC over the grid of operating points

removes a priori input sequences with very high costs and poses in general no limitation on the performance.

For the first case study, Fig. 7.12 shows two comparisons of the switching frequencies over the above defined grid of operating points in two different ways, while Fig. 7.13 shows the relative and the absolute reduction of the switching frequency. Averaging the data



(a) Torque (b) Stator flux

Figure 7.14: Case Study I: Mean squared violation of torque and stator flux for ABB's DTC (upper surface) compared with MPC-E with $N = 2$ (lower surface) over the grid of operating points

Case study	Average reduction	Maximal reduction
I	25 % or 60 Hz	42 % or 120 Hz
II	23 % or 66 Hz	42 % or 132 Hz
III	23 % or 91 Hz	49 % or 195 Hz

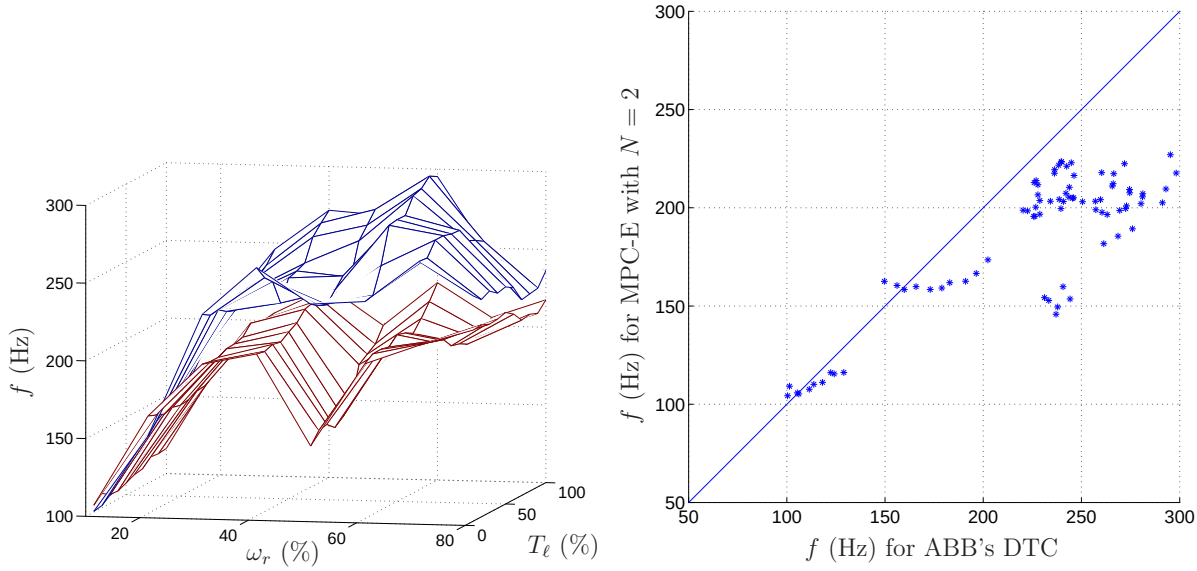
Table 7.12: Reduction of the switching frequency of MPC-E with $N = 2$ relative to ABB's DTC for the three case studies

in Fig. 7.13 over all grid points yields an average reduction of the switching frequency by 25 % (or 60 Hz), while the maximum improvement amounts to 42 % (or 120 Hz). Over the whole range of operating points, the bounds on the torque, stator flux and neutral point potential are at least as well respected as by ABB's DTC scheme as shown in Fig. 7.14. Nevertheless, as described in Section 7.4.1, also the MPC-E scheme allows for slight violations of the bounds.

The figures for the Case Studies II and III are very similar and thus omitted here. As shown in Table 7.12, the average and the maximal reduction for all three case studies are very similar indicating that the proposed control scheme works equally well for DTC drives with very different characteristics and ratings.

Horizon $N = 1$

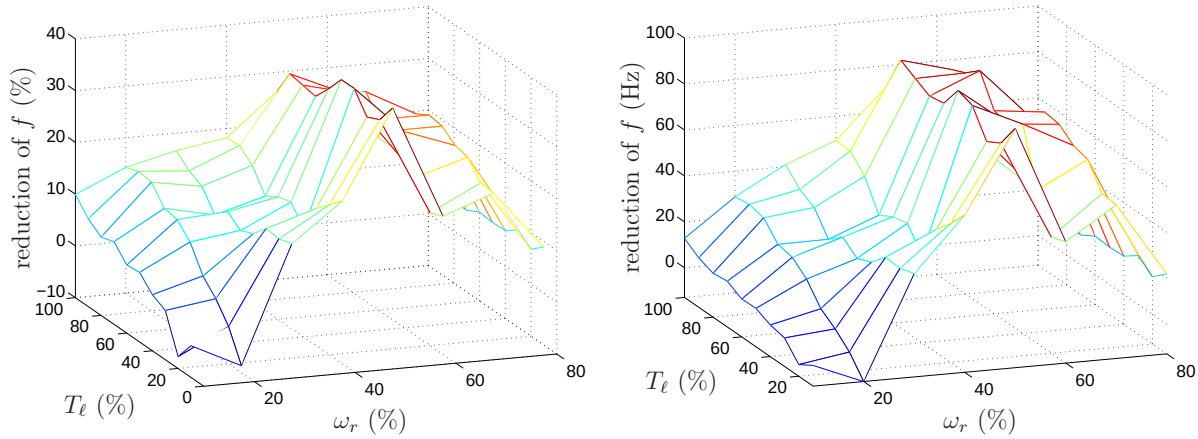
For the case of MPC-E with a horizon $N = 1$ as introduced in Section 7.4.2, we have considered only Case Study I. For Case Studies II and III, the performance is expected



(a) Switching frequency of ABB's DTC (upper surface) compared with MPC-E (lower surface)

(b) Switching frequency of ABB's DTC versus MPC-E

Figure 7.15: Case Study I: Comparison of switching frequency f of ABB's DTC with respect to MPC-E with $N = 1$ over the grid of operating points

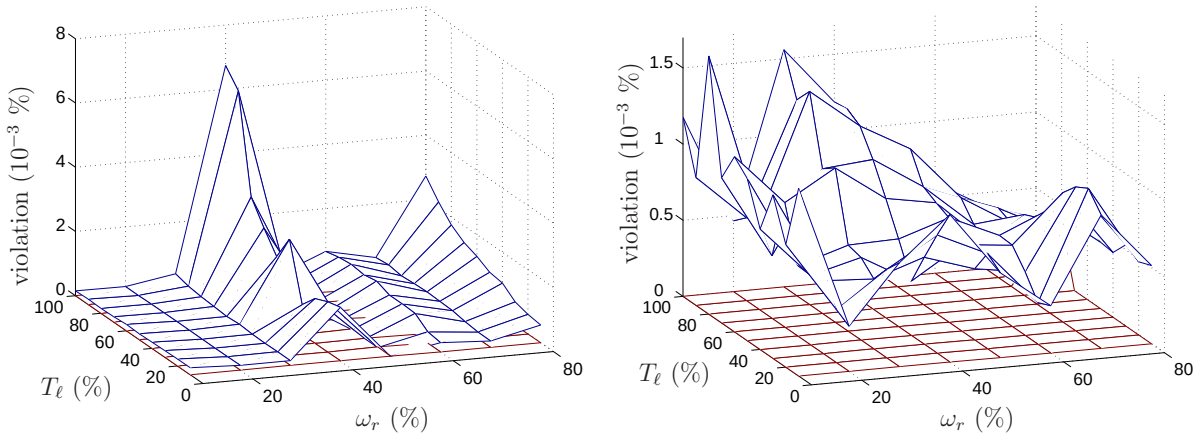


(a) Relative reduction of switching frequency (in per cent)

(b) Absolute reduction of switching frequency (in Hz)

Figure 7.16: Case Study I: Reduction of the switching frequency f of MPC-E with $N = 1$ relative to ABB's DTC over the grid of operating points

to be very similar, as it is the case for MPC-E with $N > 1$. Fig. 7.15 shows two comparisons of the switching frequencies over the grid of operating points in two different ways, while Fig. 7.16 depicts the relative and the absolute reduction of the switching frequency. Averaging the data in Fig. 7.16 over all grid points yields an average reduction of the switching frequency by 16 % (or 40 Hz), while the maximum improvement amounts to



(a) Torque (b) Stator flux

Figure 7.17: Case Study I: Mean squared violation of torque and stator flux for ABB's DTC (upper surface) compared with MPC-E with $N = 1$ (lower surface) over the grid of operating points

38 % (or 91 Hz). As shown in Fig. 7.17, the bounds on the torque, stator flux and neutral point potential are practically perfectly respected over the whole range of operating points.

Compared to the previous section showing the performance results for MPC-E with $N = 2$, the performance improvement in terms of an average reduction of the switching frequency is reduced by one third. However, at least in the absence of noise, MPC-E with $N = 1$ keeps the controlled variables practically strictly within their bounds, whereas MPC-E with $N > 1$ allows for small violations of the bounds. These differences make it hard to directly compare the switching frequencies and put the loss in performance improvement into perspective.

7.5 Infeasibilities

An inherent problem in DTC are infeasibilities, more precisely, situations in which the control problem cannot be solved and a new voltage vector is not found. In the proposed MPC schemes, infeasibilities manifest themselves when no sequence of manipulated variables exists that meets the constraints on the controlled variables. For MPC-E, this is equivalent to cases in which no candidate trajectory exists. In general, the overall reason for infeasibilities is the lack of enough degrees of freedom.

Factors limiting the degrees of freedom are. (i) The constraints induced by the upper and lower constraints on the three controlled variables are too tight or even conflicting, and (ii) the set of voltage vectors is discrete-valued and finite (eight for the two-level

inverter, 27 for the three-level inverter). For three-level inverters with only one snubber circuit per stack, two additional limitations arise. (iii) Constraints that are present on the allowed switch transitions limit the set of reachable voltage vectors within one time-step, while (iv) short prediction horizons N limit the set of reachable voltage vectors within N time-steps. Points (i)–(iii) are a result of the control objectives and the physics of the inverter, point (iv) is induced by the controller.

In the sequel, we provide a geometric insight in the nature of the problem focusing on a three-level inverter. The result of this analysis is that even when considering only two controlled variables and ignoring the switching constraints, the combination of points (i) and (ii) can lead to an infeasibility.

Geometric Analysis

This section aims at determining the set of voltage vectors that keep the torque and the length of the stator flux constant for a given state. To simplify the exposition, consider the voltage vectors in the dq frame[12] and neglect their discrete nature, i.e. assume that the inverter can produce at each phase any voltage between $-\frac{V_{dc}}{2}$ and $\frac{V_{dc}}{2}$, where V_{dc} denotes the dc-link voltage. As a result, the 27 discrete voltage vectors shown in Fig. 5.3(b) are relaxed to the set of vectors enclosed by a circle around the origin shown in Fig. 7.18(a). Neglecting the zero component and using (5.5) and the first two rows of (5.3), the radius of the circle is given by

$$\max_{u_{abc}\in\{-1,0,1\}^3} \left|\left|\frac{V_{dc}}{2}\frac{2}{3}\begin{bmatrix}1 & -\frac{1}{2} & -\frac{1}{2}\\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2}\end{bmatrix} u_{abc}\right|\right|_2 = \frac{2}{3}V_{dc}\,, \tag{7.22}$$

where $||\cdot||_2$ denotes the length of a vector. We denote the relaxed voltage vectors in the dq plane by $\tilde{v}_{dq} = [\tilde{v}_d\ \tilde{v}_q]^T$, where

$$||\tilde{v}_{dq}|| \leq \frac{2}{3}V_{dc} \tag{7.23}$$

according to (7.22).

Using (5.13) and (5.10), the time derivative of the torque as a function of the stator and rotor fluxes $\psi_s = [\psi_{ds}\ \psi_{qs}]^T$ and $\psi_r = [\psi_{dr}\ \psi_{qr}]^T$, respectively, the torque T_e (which is a function of the fluxes), the rotational speed ω_r and the relaxed voltage vector with components $\tilde{v}_d$ and $\tilde{v}_q$ is given by

$$\frac{dT_e}{dt} = -\frac{r_s x_{rr} + r_r x_{ss}}{D}T_e - \frac{x_m}{D}\psi_s^T\psi_r\omega_r + \frac{x_m}{D}\big(\psi_{dr}\tilde{v}_q - \psi_{qr}\tilde{v}_d\big)\,. \tag{7.24}$$

Neglecting the first expression which is very small compared with the other two (in the range of 5 to 10 %) and setting the derivative to zero, the expression simplifies to

$$\psi_{dr}\tilde{v}_q - \psi_{qr}\tilde{v}_d = \psi_s^T\psi_r\omega_r\,. \tag{7.25}$$

[12]We do not consider the zero component as the torque and the stator flux are independent from it.

Regarding the components of the relaxed voltage vector as free variables, the latter is the equation of a line on the dq plane that is parallel to the rotor flux vector. Assuming that the torque and the length of the stator flux are kept within tight bounds keeps the inner product of the fluxes nearly constant and renders a shifting of this line (with regard to the rotor flux vector) proportional to the speed. Therefore, the set of voltage vectors that produce a constant torque is affine in the dq plane. Voltage vectors between this line and the origin decrease, vectors beyond the line increase the torque. Note that increasing the speed shifts this line away from the origin thus reducing the number of available voltage vectors.

The derivative of the length of the stator flux with respect to time is derived from (5.14) and (5.10). It is a function of the fluxes, the length of the stator flux Ψ_s (which is a function of the fluxes) and the relaxed voltage vector with components $\tilde{v}_d$ and $\tilde{v}_q$

$$\frac{d\Psi_s}{dt} = r_s \frac{x_m}{D} \frac{\psi_s^T \psi_r}{\Psi_s} - \frac{r_s x_{rr}}{D} \Psi_s + \frac{\psi_{ds}\tilde{v}_d + \psi_{qs}\tilde{v}_q}{\Psi_s} \,. \tag{7.26}$$

Similar to the torque, we can neglect the first two expressions as they are very small with respect to the third one (in the range of 5 %). Setting the derivative to zero yields

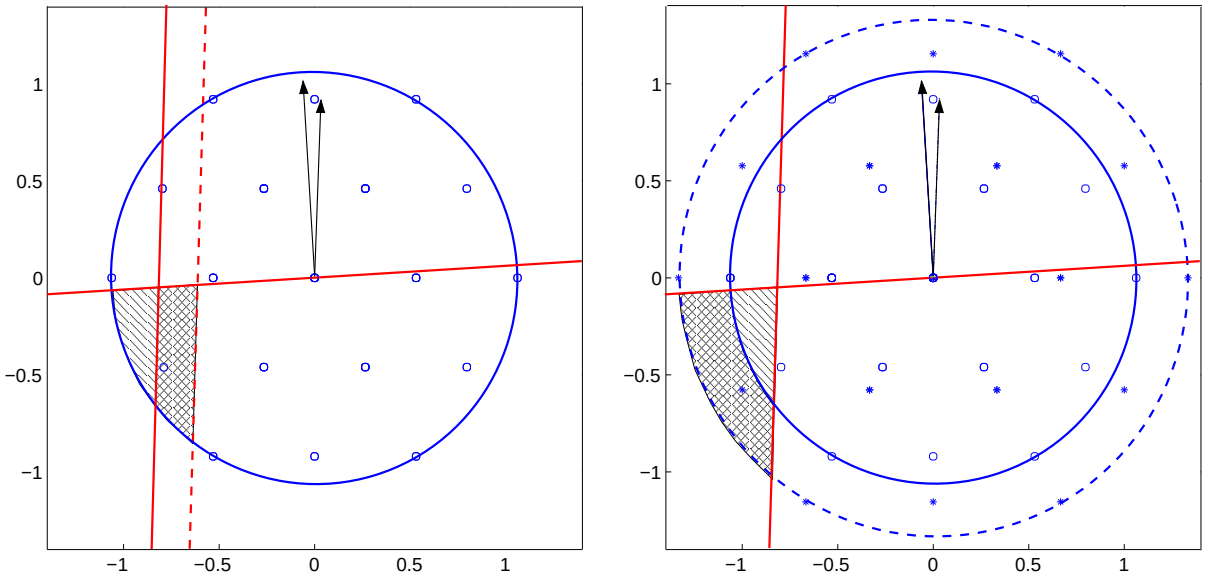
$$0 = \psi_{ds}\tilde{v}_d + \psi_{qs}\tilde{v}_q \,. \tag{7.27}$$

The set of relaxed voltage vectors fulfilling (7.27) is a line, which is perpendicular to the stator flux vector and contains the origin. This is an obvious result, as keeping the length of the stator flux constant can only be achieved by vectors normal to the stator flux. Vectors with an obtuse angle with respect to the stator flux increase, vectors with an acute angle decrease the length of the stator flux.

Because of the linearity, one can easily describe the set of voltage vectors for which the torque and the stator flux are increasing or decreasing in terms of an (unbounded) polyhedron. Often, we also add the (quadratic) constraint on the length of the relaxed voltage vectors (7.23) to the set.

Geometric Interpretation

Example 7.5 To visualize the above analysis, consider the following situation as an example. The stator and rotor fluxes are given by $\psi_s(k) = [-0.0622\ 1.0131]^T$ and $\psi_r(k) = [0.0321\ 0.9268]^T$, respectively, yielding the torque $T_e(k) = 0.3381$ and the length of the stator flux $\Psi_s(k) = 1.0150$. The lower and upper bounds on the torque and stator flux are given by $T_{e,min} = 0.3373$, $T_{e,max} = 0.4788$, $\Psi_{s,min} = 0.9670$ and $\Psi_{s,max} = 1.0150$. The neutral point potential is $\upsilon_n(k) = 0.0398$. As it is well between its lower and upper bounds $\upsilon_{n,min} = -0.07$ and $\upsilon_{n,min} = 0.07$, respectively, it is neglected here. The former control input is $u_{abc}(k-1) = [-1\ 0\ 0]^T$. The motor and inverter parameters are given in



(a) Reducing the speed moves the constant torque line to the right (vertical dashed line)

(b) Increasing the dc-link voltage increases the length of the voltage vectors (stars bounded by dashed circle)

Figure 7.18: Set of relaxed voltage vectors in the dq plane (hatched and cross hatched) for Example 7.5 that increase the torque and reduce the length of the stator flux for the given stator and rotor flux vectors

Table 7.13. Summing up, the torque is at its lower constraint, the stator flux at its upper constraint, and the neutral point potential is well within its bounds. Thus, a voltage vector needs to be found that increases the torque and reduces the length of the stator flux regardless of the neutral point potential. This task seems to be trivial.

In the following, we visualize the example in the dq plane. In Fig. 7.18, the stator and rotor flux vectors are pointing roughly along the positive q-axis. As the rotational speed is counter clockwise, the shorter vector to the right refers to the rotor flux. Furthermore, Fig. 7.18 shows for the speed $\omega_r = 0.8$ and the dc-link voltage $V_{dc} = 1.5937$ the sets of voltage vectors yielding constant torque (solid vertical line) and constant flux (solid horizontal line). The voltage vectors for $V_{dc} = 1.5937$ are given by small circles bounded by the outer solid circle, whereas the voltage vectors for $V_{dc} = 2$ are stars bounded by the outer dashed circle. The hatched area depicts the set of desired (relaxed) voltage vectors that increase the torque and reduce the length of the stator flux. For $\omega_r = 0.8$ and $V_{dc} = 1.5937$, this set contains no discrete voltage vector, i.e. there exists no discrete voltage vector that fulfills the constraints on the torque and the stator flux. Thus, the control problem for the given fluxes cannot be solved and is infeasible.

This is a physical problem stemming from the fact that the number of discrete voltage vectors is finite. In the following, we demonstrate the impact of the speed and the dc-link

voltage on the size of the hatched set.

- Operating at lower speed. This reduces the rotational speed of the rotor flux vector and allows one to apply shorter voltage vectors to produce sufficient torque. This situation is shown in Fig. 7.18(a), where the speed is reduced from $\omega_r = 0.8$ to 0.6 moving the line corresponding to constant torque to the right (dashed line). Thus, the selection process of the voltage vector is in general easier at lower speeds[13].

- Operating with higher dc-link voltage. As depicted in Fig. 7.18(b) for $V_{dc} = 2$, this would increase the length of the voltage vectors linearly. The new voltage vectors are shown as stars bounded by the dashed circle.

In both cases, the original hatched area is augmented by the cross hatched area that contains the discrete voltage vector $[-1\ 0\ 1]^T$ rendering the problem feasible.

Apart from that, a five-level inverter would increase the number of discrete voltage vectors and covers the dq space more densely with discrete voltage vectors thus enlarging the degrees of freedom and reducing the risk for infeasibilities.

Conclusions

It is important to note that the constraints on the allowed switch transitions in combination with short prediction horizons (as for MPC-E with $N > 1$ in Section 7.4.1) are not the only reason for infeasibilities. Even when relaxing the switching constraints as done here (and in MPC-E with $N = 1$ in Section 7.4.2), there may exist no discrete voltage vector that keeps the controlled variables within the imposed bounds. This dilemma is particularly emphasized in this example, where we are only considering two of the three controlled variables (the torque and the stator flux). Adding the neutral point potential with its bounds further reduces the set of voltage vectors that keep the controlled variables within their bounds.

To overcome this problem, the only and ultimate solution is to relax some of the bounds on the controlled variables. A reasonable choice is to start with the neutral point potential and to additionally relax if necessary also the bounds on the stator flux.

7.6 Case Studies

For the performance evaluation of the proposed MPC schemes, three case studies were considered with DTC drives of medium and low power. The respective ratings and parameters are detailed in the Tables 7.13, 7.14 and 7.15.

[13] Yet, the overall operation of the drive becomes more challenging at low speed due to the difficulties of the estimation procedure.

Induction Motor			
voltage	3300 V	r_s	0.0108 p.u.
current	356 A	r_r	0.0091 p.u.
real power	1.587 MW	x_{ls}	0.1493 p.u.
apparent power	2 MVA	x_{lr}	0.1104 p.u.
frequency	50 Hz	x_m	2.3489 p.u.
rotational speed	596 rpm		
Inverter			
dc-link voltage	4294 V	V_{dc}	1.5937 p.u.
current	500 A	x_c	4.3715 p.u.

Table 7.13: Case Study I: Rated values (left) and parameters (right) of the induction motor and the inverter

Induction Motor			
voltage	3100 V	r_s	0.0048 p.u.
current	1477.3 A	r_r	0.0050 p.u.
real power	6.6 MW	x_{ls}	0.1469 p.u.
apparent power	7.765 MVA	x_{lr}	0.0974 p.u.
frequency	65.6 Hz	x_m	2.8468 p.u.
rotational speed	979.1 rpm		
Inverter			
dc-link voltage	4294 V	V_{dc}	1.6965 p.u.
current	2865 A	x_c	3.4955 p.u.

Table 7.14: Case Study II: Rated values (left) and parameters (right) of the induction motor and the inverter

7.7 Conclusions and Future Research

Conclusions

In this chapter, we have proposed three novel model-based control schemes to tackle the DTC problem. In the following, we summarize each of them.

Induction Motor			
voltage	400 V	r_s	0.0323 p.u.
current	31 A	r_r	0.0314 p.u.
real power	15 kW	x_{ls}	0.1116 p.u.
apparent power	1.92 kVA	x_{lr}	0.1047 p.u.
frequency	50 Hz	x_m	2.0484 p.u.
rotational speed	970 rpm		
Inverter			
dc-link voltage	613 V	V_{dc}	1.8769 p.u.
current	38 A	x_c	0.7021 p.u.

Table 7.15: Case Study III: Rated values (left) and parameters (right) of the induction motor and the inverter

Based on an MLD model of the DTC drive, we have formulated and solved in Section 7.2 an optimal control problem tailored to the peculiarities of the DTC problem by employing three different penalty levels, the Late Switching Strategy and the Multiple-Rate Prediction Model Approach. The proposed MPC-PL scheme is based on a systematic design procedure allowing one to adapt it to other inverter topologies and induction motor characteristics. This is shown by applying the scheme to both the two- and the three-level inverter. Furthermore, this approach clearly demonstrates the potential for improving the performance of DTC with respect to state of the art industrial DTC look-up tables, as the comparison with ABB's ACS6000 drive emphasizes.

However, when solving the underlying optimization problem on-line to derive the control input, the corresponding computation times well exceed the sampling time of DTC. Therefore, the proposed controller cannot be directly implemented and experimentally verified making it mandatory to compute the (explicit) state-feedback control law, which can be stored in a look-up table. For a DTC drive with a two-level inverter, this computation has been carried out using Dynamic Programming. Optimal Complexity Reduction allowed us to reduce the number of resulting polyhedra by an order of magnitude. For the case of the three-level inverter, the combinatorial nature of the problem and the higher complexity of the model make the problem untractable using standard computational power at hand.

The MPC-PL scheme can be considered as a starting point indicating that there exists a potential for a significant performance improvement thus motivating further investigations.

Yet, the complexity of the control scheme is both on-line and off-line prohibitive, and it contains several tuning parameter in the cost function such as the weights on the soft constraints.

Exploiting the fact that the DTC objectives rather relate to feasibility than to optimality, we have proposed in Section 7.3 the MPC-FMB scheme, which allows for switch transitions only at the current time step and employs blocking control moves for the whole horizon, uses a time-varying prediction horizon, and avoids soft constraints using hard bounds instead. In particular, MPC-FMB is tailored to the off-line computation of the state-feedback control law, which features a complexity that is reduced by an order of magnitude with respect to MPC-PL and simultaneously enhances the performance. Moreover, it features only one tuning parameter, namely the maximal length of the horizon.

However, the state-feedback control law can only be derived for a particular operating point (speed, torque and bounds), and the extension to DTC drives with three-level inverters has not been tried due to the expected curse of dimensionality.

The last approach, MPC-E in Section 7.4 combines the advantages of MPC-PL and MPC-FMB while adding the notion of extrapolation and the feasible switching path and using a nonlinear motor model. The control scheme is available in two forms with horizons larger than or equal to one. They mainly differ in the computational burden, namely the upper bound on the computational burden of MPC-E with $N = 1$ is five times smaller than the one with $N = 2$, but they both significantly improve the performance with respect to state of the art industrial DTC schemes by reducing the average switching frequency and the violation of the bounds imposed on the controlled variables. More specifically, the simulation results for different DTC drives show that MPC-E with $N = 2$ reduces the switching frequency over the whole range of operating points by 24 % compared with ABB's ACS6000 scheme, and the simplified scheme with $N = 1$ yields a reduction of the switching frequency by 16 % while the bounds are at least as well respected as by ABB's DTC scheme.

The main advantages of MPC-E are the usage of on-line computations allowing for time-varying parameters, set points and bounds, the lack of tuning parameters, and the applicability to induction motors of very different characteristics driven by different inverter topologies. Summing up, the proposed control scheme provides superior performance combined with simplicity and flexibility. It is our opinion that the computational burden associated with MPC-E is close to the lowest possible that is achievable for a model-based approach. Moreover, performing 2000 operations (for the simplified scheme) within less than the sampling interval is demanding yet computationally feasible using standard hardware technology. This is emphasized by the fact that our industrial partner has filed the patent [GPM04d] for us, and is currently in the process of implementing MPC-E for commercial prototyping.

Summing up, from MPC-PL via MPC-FMB to MPC-E, the performance is constantly improving, while the controller complexity and the computational burden is decreasing by several orders of magnitude. The last scheme, MPC-E, is in terms of performance, simplicity, flexibility and applicability by far the most promising one.

Future Research

As an outlook, it would be interesting to investigate the following four points. Firstly, the analysis provided in Section 7.5 could be used to a priori rule out voltage vectors that move the torque or the stator flux in the undesired direction and to thus reduce the computation time. Secondly, the prediction model for MPC-E might be formulated in the rotating reference frame similar to the Stator Flux Dynamics Model in order to reduce the number of necessary operations. Thirdly, a major difficulty has not been addressed by any of the MPC approaches here: the choice of the bounds on the controlled variables. In practice, the bounds are not given a priori but rather must be tuned by the commissioning engineer so as to keep the switching frequency acceptably low. To do so, one might try to switch in such a way that the a given desired switching frequency is obtained within the prediction horizon. As a consequence, the bounds would become obsolete. Last, the state-feedback control laws for MPC-PL and MPC-FMB could by drastically simplified by replacing groups of similar hyperplanes by one hyperplane as detailed in Chapter 4.

Part IV

DC-DC Converters

8

Switch-Mode DC-DC Converters

8.1 Introduction

8.1.1 Switch-Mode DC-DC Conversion

Switch-mode DC-DC converters are switched circuits that transfer power from a DC input to a load. They are used in a large variety of applications due to their light weight, compact size, high efficiency and reliability. Specifically, they constitute the enabling technology in computer power supplies, battery chargers, variable speed DC motor drives, and sensitive and demanding aerospace and medical applications. Their analysis and design – both in the open and the closed loop – have attracted a wide research interest, and the quest for efficient control techniques is of interest for both the research and the industrial community. Because the DC voltage at the input is unregulated (consider for example the result of a coarse AC rectification) and the output power demand changes significantly over time constituting a time-varying load, the scope is to achieve output voltage regulation in the presence of input voltage and output load variations.

The difficulties in controlling DC-DC converters arise from their hybrid nature. In general, these converters feature three different modes of operation, where each mode is associated with a (different) linear continuous-time dynamic law. Furthermore, constraints are present resulting from the converter topology. In particular, the manipulated variable (duty cycle) is bounded between zero and one, and in the discontinuous current mode a state (inductor current) is constrained to be non-negative. Additional constraints are imposed as safety measures, such as current limiting or soft-starting, where the latter constitutes a constraint on the maximal derivative of the current during start-up. The control problem is further complicated by gross changes in the operating point due to input voltage and output load variations, and model uncertainties.

Fixed-frequency switch-mode DC-DC converters use semiconductor switches that are periodically switched on and off, followed by a low-pass filtering stage with an inductor and a capacitor to produce at the output a DC voltage with a small ripple. Specifically,

the switching stage comprises a primary semiconductor switch that is always controlled, and a secondary switch that is operated dually to the primary one. In its simplest form, the secondary switch is one-directional and not controlled. More sophisticated topologies, however, use a controlled bi-directional switch to avoid additional complications such as the so called discontinuous conduction mode. For details the reader is referred to the standard power electronics literature (e.g. [MUR89]).

The switches are driven by a pulse sequence of constant frequency (period), the *switching frequency* f_s (*switching period* T_s), which characterizes the operation of the converter. The DC component of the output voltage can be regulated through the duty cycle d, which is defined by $d = \frac{t_{on}}{T_s}$, where t_{on} represents the interval within the switching period during which the primary switch is in conduction.

The main control objective for DC-DC converters is to drive the primary switch with a duty cycle such that the DC component of the output voltage is equal to its reference. This regulation needs to be maintained despite variations in the load or the input voltage. The basic concept that is currently used for the control of DC-DC converters is the Pulse Width Modulation (PWM): The switch is turned on at the beginning of each switching period, and it is turned off by the controller when a certain condition is fulfilled. A latch keeps the switch turned off until the beginning of the next period. With this formulation, the control problem is to decide at which instant within the switching period the switch should be turned off.

8.1.2 Review of the State of the Art

The standard approach to model DC-DC converters is the method of state-space averaging [Mv76, EvM82]. In order to bypass the difficulties posed by the hybrid nature of the system, an averaged continuous-time model is obtained that uses the duty cycle as an input and describes the system's slow dynamic. This is done based on the basic assumption that the switching period is by far smaller than the time constant of the low pass filtering stage. The result of this procedure is still a nonlinear model due to the presence of multiplicative terms involving the state variables and the duty cycle. Often, the controller design is carried out using linear control techniques for an averaged model linearized around a specific operating point. Apart from the limitations of this approximation, the averaging procedure hides all information about the fast dynamics of the system, and fast-scale instabilities like subharmonic oscillations cannot be captured.

A more rigorous approach is to avoid the classic averaging technique. Here, mostly models given in the discrete-time domain have been reported, which produce an exact nonlinear mapping of the state variables from the beginning to the end of the switching period [KSV91, KMM00]. These methods successfully describe many aspects of the com-

plex dynamics of DC-DC converters, and they are very suitable for analyzing phenomena like subharmonic and chaotic oscillations that have been observed when DC-DC converters operate in closed loop [HDJ92]. Nevertheless, for design purposes they still carry the difficulty of being nonlinear with respect to the duty cycle, and making a systematic approach to the controller design problem a challenging task.

Concerning the operation of DC-DC converters in closed-loop, a variety of different control strategies are used in practice, categorized in voltage and current mode control schemes [MUR89]. Traditionally, they are all PI-type controllers tuned based on the above linearized average models. Simple rules, such as selecting a cross-over frequency an order of magnitude smaller than the switching frequency and a phase margin in the range of 45 to 60 degrees are used. Depending on the converter topology and the control strategy, these tuning guidelines result in step responses with typical overshoots of up to ten per cent and settling times in the range of 5 to 30 switching periods.

In the literature, a wide range of different strategies has been proposed for improving the controller design. In the following, we will classify the controller designs according to the models used by defining the categories *averaged and linearized (with extensions)*, *averaged and nonlinear* and *non-averaged* models. For each, we will refer only to some representative contributions.

Controller Design for Averaged & Linearized Models In [LTL91,LTL93], the authors propose a Linear Quadratic Regulator (LQR) based on a locally linearized discrete-time model of the averaged DC-DC converter using a Δu formulation. An additional integrator is included to remove a steady-state error in the output voltage, and an observer is added to provide estimates of the plant states. Even though the LQR approach introduces the notion of optimality, it carries a number of drawbacks: disturbances are handled only in an unsatisfactory way, the controller is designed for a locally linearized model, and constraints cannot be addressed in the controller design.

Controller Design for Averaged & Linearized Models with Extension Based on a family of transfer functions parameterized in the steady-state duty cycle, the author of [SR91] derives a family of PI controllers with parameters that depend nonlinearly on the steady-state duty cycle, where the tuning is done by following the guidelines of Ziegler and Nichols. Unfortunately, during transients the issue of switching among the different PI controllers is not addressed (a remedy might be for example a bumpless transfer scheme), an additional anti-windup scheme is needed to handle the constraints on the duty cycle, and a low-pass filter is added to the control loop, which leads to a deterioration of the closed-loop performance. Moreover, it is not clear how to choose the duty cycle used to define the controller parameters during transients.

Starting from a nonlinear and non-averaged discrete-time model, a controller with two components is proposed in [GMSV94]. The first component is an LQR controller that is designed after linearizing the model around the nominal operating point. This component is intended to provide good dynamic closed-loop performance. To better reject large disturbances in the input voltage, a nonlinear feedforward term is added that can be stored in a look-up table. To obtain variables that cannot be directly measured, an extended Kalman filter is suggested.

Controller Design for Averaged & Nonlinear Models To improve the controller performance, often a feedforward compensation term is added from the input voltage to the PWM controller. In [CPV92], the parameters of this term are derived such that the output voltage remains unaffected for a constant duty cycle. Unfortunately, this derivation is done by neglecting the parasitic elements of the converter. In a later stage, a rather ad hoc offset circuitry is suggested to cope with the effects of the parasitics.

For a continuous-time averaged nonlinear model, the authors of [KS99] propose a nonlinear suboptimal H_∞-controller with an additional integrator. What is remarkable about this approach is the fact that closed-loop stability is shown via a Lyapunov function.

Using a control methodology that extends the concept of Generalized Predictive Control [CMT87] to nonlinear systems, an unconstrained nonlinear predictive controller is formulated for a DC-DC converter in [LK04]. The prediction model is a discrete-time nonlinear averaged model neglecting the parasitics. The control horizon is set to one, the prediction horizon amounts to a few switching intervals. The nonlinear optimization problem is solved on-line in an iterative fashion. Assuming that the disturbances can be measured, this approach has the advantage that the prediction model can be updated on-line. However, an implementation becomes questionable due to the lack of convergence guarantees and the potentially excessive computation time. As an unconstrained optimization problem is solved, constraints on the duty cycle and the inductor current cannot be handled directly.

Controller Design for Non-Averaged Models In [SR03], the advances in sliding mode control for DC-DC converters are summarized. The key idea is to establish a sliding surface for the average inductor current that corresponds to the desired output voltage, and to indirectly regulate the output voltage through keeping the inductor current close to the sliding surface. This leads not only to slow closed-loop responses, but also to significant steady-state errors in the output voltage in the presence of unmodelled changes in the load resistance. The author suggest to cope with the latter problem by integrating the output voltage error and shifting the sliding surface accordingly. In general, changes in the input voltage and parasitics are not addressed here.

In more recent work, the hybrid nature of DC-DC converters is addressed for the modelling and controller design [SEK03, Lin03]. Yet in these approaches, the switching frequency of the converter is not kept constant. It is our belief, that such a problem setup does not necessarily correspond to the physical reality of the current industrial practice, since a variable switching frequency complicates the design of both the power stage of the converter and the Electromagnetic Interference (EMI) filter. However, the constant switching frequency might be sacrificed to increase efficiency, as is the case with resonant converters that use zero current or zero voltage switching to minimize the switching losses.

More specifically, the DC-DC control problem is interpreted in [SEK03] as a synthesis problem of a Hybrid Automaton [Hen00] satisfying safety properties and given performance criteria. Specifically, a ball is derived, which is a subset of the set of states satisfying the constraints, and switching is performed whenever the state vector reaches the surface of the ball, and no synchronization scheme or latch to keep the switching frequency constant is present. The proposed scheme resembles effectively a hysteresis type of controller. In an extension, the switch on and off transitions are restricted to fixed sampling instants, and a second inner ball is derived to account for this restriction. In this control approach, the notion of a performance index is missing and only a minimal performance is addressed.

In [Lin03], Relaxed Dynamic Programming, which is summarized in [Lin03], is used to derive an explicit controller for a power converter with a current sink as load. Starting from a non-averaged description, the model is expressed in discrete-time, and an objective function over an infinite horizon with a 'forgetting factor' is formulated. The complexity of the resulting controller is rather large, thus making an implementation difficult. Moreover, the switch on and off transitions are restricted to given sampling instants, and safety constraints on the current are not addressed. Apart from that, switching the polarity of the input voltage instead of switching between a positive potential and a ground seems to be rather uncommon and renders a problem setup that cannot be directly compared with the state of the art in the power electronics community.

Summary Except of [KS99] none of the above presented control approaches addresses the issue of closed-loop stability, and with the exception of [SEK03] constraints (e.g. on the duty cycle and the current) are not tackled in the design procedure. As a concluding remark, one might state that the common element in the academic state of the art is the use of simplified models for the description of the dynamic behavior of switch-mode DC-DC converters. It is obvious that approximations like the use of averaged or locally linearized models do not allow capturing the complex dynamics that stem from the hybrid nature of DC-DC converters, and unavoidably narrow the space of the explored phenomena, thus producing results of limited validity.

8.1.3 Outlook

Motivated by these difficulties, we present in this chapter a novel approach to the modelling and controller design problem of DC-DC converters using a synchronous step-down DC-DC converter as an illustrative example. In Section 8.2, we summarize the nonlinear continuous-time state-space equations of the converter, introduce the novel ν-resolution modelling approach that allows us to describe the converter as a hybrid model both in the MLD framework (Section 2.2.2) and in PWA form (Section 2.2.3). This leads to a model that is valid for the whole operating regime and captures the evolution of the state variables within the switching period. Based on the hybrid model, we formulate and solve a constrained finite time optimal control problem in Section 8.3. This results in a systematic controller design that achieves the objective of regulating the output voltage to the reference despite input voltage and output load variations while satisfying the constraints. In particular, the control performance does not degrade when changing operating points. Most important, in Section 8.3.3 we pre-solve the control problem off-line and derive the equivalent state-feedback control law parameterized over the whole state-space. This controller can be stored in a look-up table, hence allowing for the practical implementation of the proposed control scheme. A subsequent analysis for the nominal case in Section 8.4 shows that the considered state space is a control invariant set, implying that for all initial states within this set, a control input is always found and all constraints are met for all future time steps. Most importantly, a piecewise quadratic (PWQ) Lyapunov function is derived proving that the nominal closed-loop system is globally exponentially stable. Finally, Section 8.5 illustrates various aspects of the system's behavior with simulation results including start-up, a comparison with a current mode PI controller, and gross changes in the input voltage and the output resistance.

8.2 Modelling the Synchronous Converter

We start by modelling the synchronous step-down converter in continuous-time, and derive for each mode of operation the state-space equations. The model incorporates the parasitic elements, in particular the internal resistance of the inductor and the Equivalent Series Resistance (ESR) of the capacitor.

8.2.1 Continuous-Time Model

The circuit topology of the synchronous step-down converter is shown in Fig. 8.1. Using normalized quantities, r_o denotes the output load, which we assume to be ohmic, r_c is the ESR of the capacitor, r_ℓ is the internal resistance of the inductor, x_ℓ and x_c represent

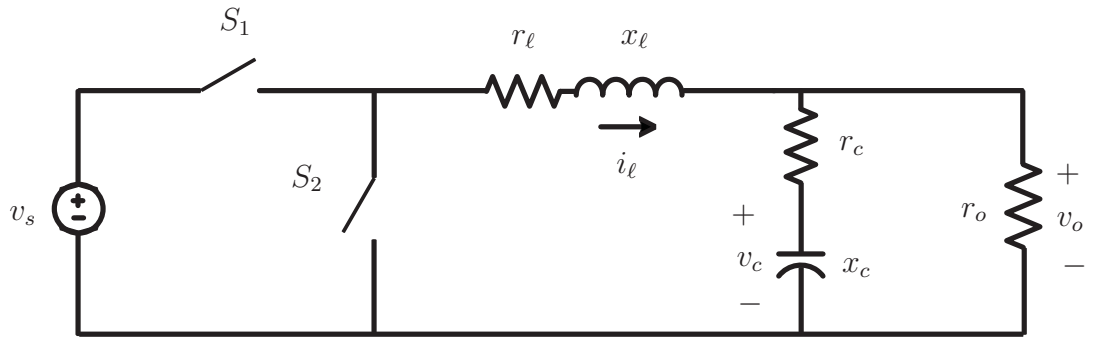


Figure 8.1: Topology of the step-down synchronous converter

the inductance and the capacitance of the low-pass filtering stage, and v_s denotes the input voltage. For every switching period k, a duty cycle $d(k)$, which is bounded between zero and one, is chosen by the controller. For the time interval $kT_s \leqslant t < (k + d(k))T_s$, the switch S_1 is on and power is transferred from the input directly to the load. While S_1 is on, the switch S_2, which is operated dually with respect to S_1, is off. At the end of this time interval (at time $t = (k + d(k))T_s$), S_1 is turned off and kept off for $(k + d(k))T_s \leqslant t < (k + 1)T_s$. Accordingly, S_2 is switched on thus providing a path for the inductor current i_ℓ regardless whether the latter is positive or negative. At the end of the switching period, at time $t = (k + 1)T_s$, S_1 is switched on and S_2 is switched off.

Plant Model Next, we present the standard modelling approach to a step-down converter. This model will be used later to simulate the behavior of the plant. Defining $x(t) = [i_\ell(t)\ v_c(t)]^T$ as the state vector, where $i_\ell(t)$ is the inductor current and $v_c(t)$ the capacitor voltage, and given the duty cycle $d(k)$ during the k-th period, the system is described by the following set of affine continuous-time state-space equations. While S_1 is conducting, they amount to

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = Fx(t) + fv_s, \quad kT_s \leqslant t < (k + d(k))T_s\,, \tag{8.1}$$

and if S_1 is off, the system evolves autonomously, i.e.

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = Fx(t), \quad (k + d(k))T_s \leqslant t < (k + 1)T_s\,, \tag{8.2}$$

where the matrices F and f are given by

$$F = \begin{bmatrix} -\frac{1}{x_\ell}(r_\ell + \frac{r_o r_c}{r_o + r_c}) & -\frac{1}{x_\ell}\frac{r_o}{r_o + r_c} \\ \frac{1}{x_c}\frac{r_o}{r_o + r_c} & -\frac{1}{x_c}\frac{1}{r_o + r_c} \end{bmatrix}, \quad f = \begin{bmatrix} \frac{1}{x_\ell} \\ 0 \end{bmatrix}. \tag{8.3}$$

The output voltage $v_o(t)$ across the load r_o is expressed as a function of the states through

$$v_o(t) = g^T x(t) \tag{8.4}$$

with

$$g = \begin{bmatrix} \frac{r_o r_c}{r_o + r_c} & \frac{r_o}{r_o + r_c} \end{bmatrix}^T . \tag{8.5}$$

The output variable, which is of main interest from a control point of view, however, is the output voltage error that is obtained by integrating the difference between the output voltage and its reference over the k-th switching period, i.e.

$$v_{o,err}(k) = \int_{kT_s}^{(k+1)T_s} (v_o(t) - v_{o,ref})\, dt\,, \tag{8.6}$$

where $v_{o,ref}$ denotes the reference of the output voltage, which is always set to one.

Summing up, the synchronous converter features two operation modes with two different affine dynamics. The two modes differ only in the affine expression and have the same output function. At the beginning of the switching period, the first mode with (8.1) is always active. The duty cycle $d(k)$ determines the time within the k-th switching period when changing from the first to the second mode. The latter evolves according to (8.2). At the end of the period, the converter transitions back to the first mode.

It is important to note that in current practice the inductor current $i_\ell(t)$ and the output voltage $v_o(t)$ can be directly measured. Variations in the input voltage $v_s(t)$ are considered to be measurable in accordance with common practice [MUR89], too.

The constraints present in the converter model arise from two different sources. By definition, the duty cycle $d(k)$ is constrained between zero and one. Moreover, the fact that the semiconductor devices and the load can physically handle only a certain maximal current, poses an additional upper bound on the inductor current, which is given by $i_\ell(t) < i_{\ell,max}$. This constraint is known as the current limit and is application specific.

In general, the parameters of a DC-DC converter are time-varying. However, these variations can be divided into two categories. First, the parameters of the low-pass filtering stage are only subject to slow deterioration over time or temperature changes. Specifically, these include the ESR of the capacitor r_c, the internal resistance of the inductor r_ℓ, and the inductance and capacitance of the low-pass filtering stage x_ℓ and x_c, respectively. Here, we assume these parameters to be time-invariant. On the other hand, the input voltage v_s and the load resistance r_o may vary step-wise and significantly. In particular the load resistance may vary by several orders of magnitude from a short circuit to open circuit conditions. Furthermore, in the problem set-up considered here, the output voltage reference $v_{o,ref}$ and the current limit $i_{\ell,max}$ are not restricted to be time-invariant.

Prediction Model In the following, we reformulate the above presented model to tailor it to the needs of the optimal control problem formulation.

First, from an implementation point of view, it is preferable that all states are directly measureable. Thus, we replace in the state vector the capacitor voltage by the output

voltage[1]. This leads to the redefined state vector $x(t) = [i_\ell(t)\ v_o(t)]^T$, and the matrices F, f and g turn into

$$\begin{aligned} F &= \begin{bmatrix} -\frac{r_\ell}{x_\ell} & -\frac{1}{x_\ell} \\ \frac{1}{x_c}\frac{r_o}{r_o+r_c}(1 - x_c r_c \frac{r_\ell}{x_\ell}) & -\frac{1}{x_c}\frac{1}{r_o+r_c}(1 + x_c r_c \frac{r_o}{x_\ell}) \end{bmatrix}, \\ f &= \begin{bmatrix} \frac{1}{x_\ell} \\ \frac{r_o}{r_o+r_c}\frac{r_c}{x_\ell} \end{bmatrix}, \qquad g = \begin{bmatrix} 0 & 1 \end{bmatrix}^T. \end{aligned} \tag{8.7}$$

Second, we require the input voltage v_s to be a parameter of the resulting optimal control law to handle changes in v_s straightforwardly. As will be motivated later, we thus remove v_s from the model equations by using v_s to normalize the physical quantities (states, output voltage reference and current limit) used in the model. Therefore, we introduce the state $x'(t) = \frac{x(t)}{v_s}$, which scales (8.1), (8.2), (8.4) and (8.6) over v_s. This yields the reformulated state-space equations

$$\frac{\mathrm{d}x'(t)}{\mathrm{d}t} = \begin{cases} Fx'(t) + f & \text{if } kT_s \leqslant t < (k + d(k))T_s \\ Fx'(t) & \text{if } (k + d(k))T_s \leqslant t < (k+1)T_s \end{cases} \tag{8.8a}$$

$$v'_o(t) = g^T x'(t)\,, \tag{8.8b}$$

where the matrices F, f and g are as in (8.7), and $v'_o = \frac{v_o}{v_s}$ is the scaled output voltage. The relation for the output voltage error is given by

$$v'_{o,err}(k) = \int_{kT_s}^{(k+1)T_s} (v'_o(t) - v'_{o,ref})\, dt \tag{8.9}$$

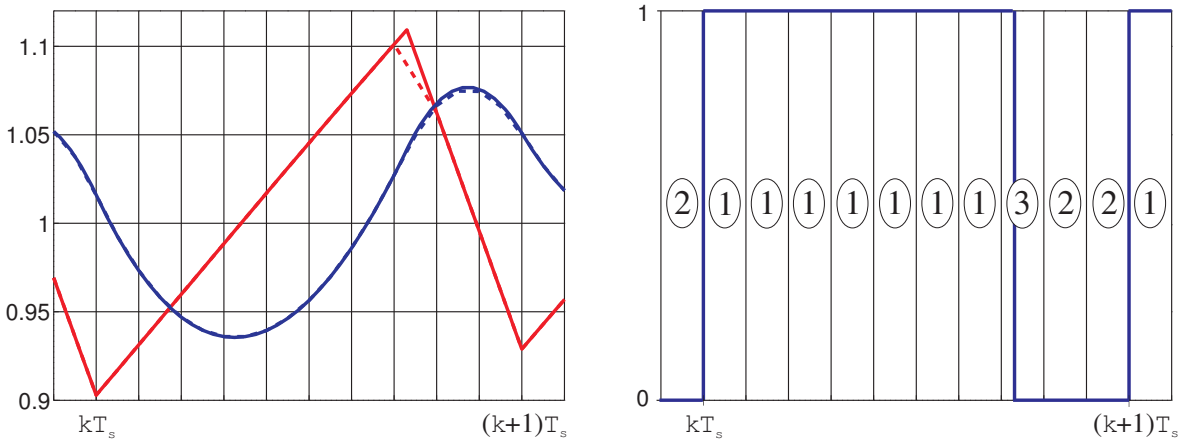
with the scaled output voltage reference $v'_{o,ref} = \frac{v_{o,ref}}{v_s}$, and the scaled output voltage error $v'_{o,err} = \frac{v_{o,err}}{v_s}$. Furthermore, we normalize the current limit

$$i'_{\ell,max} = \frac{i_{\ell,max}}{v_s}\,. \tag{8.10}$$

Before proceeding, we elaborate on the parameters of the reformulated model for the optimal control problem. As above, the system matrices F, f and g are assumed to be time-invariant. Here, we additionally assume that the load resistance r_o is constant[2]. Through the normalization, the model equations became independent from the time-varying v_s. Hence, the only time-varying parameters of the model are the scaled output voltage reference $v'_{o,ref}$ and the scaled current limit $i'_{\ell,max}$. Summing up, the model (8.7)–(8.10) uses scaled states and has the two parameters $v'_{o,ref}$ and $i'_{\ell,max}$. In the sequel, we will use the scaled model as prediction model for the constrained optimal control problem.

[1] In general, such a substitution is not performed, since the output voltage of most DC-DC converters is not continuous over time. For the step-down converter treated here, however, the output voltage is a continuous function of time. Furthermore, the MLD framework used for modelling could directly incorporate such discontinuities.

[2] Later, we will relax this assumption and introduce a Kalman filter to account for changes in r_o by manipulating the scaled output voltage reference $v'_{o,ref}$.



(a) Evolution of the states

(b) Position of the switch S_1 and the number of the mode which is active in the respective subperiod

Figure 8.2: The ν-resolution modelling approach visualized for the k-th period. The evolution of the states of the continuous-time nonlinear model (solid lines) is compared with the sequence of states of the discrete-time hybrid model (dashed lines) using $\nu = 10$ subperiods, where the saw tooth shaped line represents i'_ℓ and the smooth curve is v'_c.

8.2.2 ν-Resolution Discrete-Time Hybrid Model

Using the Prediction Model derived in the previous section as a starting point, the goal of this section is to derive a model of the synchronous step-down converter that is suitable to serve as a prediction model for the optimal controller. This model should include the following properties. First, it is natural to formulate the model and the controller in the discrete-time domain, as the manipulated variable given by the duty cycle is constant within the switching period of length T_s and changes only at the time-instants kT_s, $k \in \mathbb{N}_0$. Second, it would be beneficial to capture the evolution of the states also within one period, as this would enable us to impose constraints not only on the states at time-instants kT_s but also on intermediate values. This is particularly important for the inductor current, which can vary drastically within one period and would allow us to keep its peaks below the current limit. Third, the model needs to yield an approximation of the output voltage error given by the integral (8.9). Most important, as the converter is intrinsically hybrid in nature, we aim to retain the structure of the two operation modes and account for the hybrid character.

Motivated by these considerations, we introduce the ν-resolution modelling approach that accounts for all the above requested properties by dividing the period of length T_s into ν *subperiods* of length $\tau_s = T_s/\nu$ with $\nu \in \mathbb{N}$, $\nu \geq 1$. This concept is illustrated in Fig 8.2. Within a period, we use $\xi(n)$ to denote the states sampled with τ_s, and we

refer to the discrete time-instants of the subperiods by n, where $n \in \{0, 1, \ldots, \nu - 1\}$. Furthermore, by definition, $\xi(0) = x'(k)$ and $x'(k+1) = \xi(\nu - 1)$ hold. Note that ξ refers to the scaled x'.

Next, we introduce ν binary variables

$$\sigma_n = \text{true} \iff d(k) \geq \frac{n}{\nu}, \quad n = 0, \ldots, \nu - 1\,, \tag{8.11}$$

which represent the sampled switch position of S_1 at time-instants $n\tau_s$. Recall that the switch S_2 is dually operated with respect to S_1.

For each subperiod, we introduce the two modes discussed above (switch closed and open, respectively) plus an additional third mode that captures the transition from mode one to mode two. More specifically, the modes are (1) the switch S_1 remains closed for the whole subperiod, (2) the switch S_1 is open for the whole subperiod, and (3) the switch S_1 is opening within the subperiod. Hence, for the n-th subperiod, the state-update equations amount to

$$\xi(n+1) = \begin{cases} \Phi\ \xi(n) + \Psi & \text{if } \sigma_n \ \wedge\ \sigma_{n+1} \quad \text{(mode 1)} \\ \Phi\ \xi(n) & \text{if } \bar{\sigma}_n \quad \text{(mode 2)} \\ \Phi\ \xi(n) + \Psi(\nu d(k) - n) & \text{if } \sigma_n \ \wedge\ \bar{\sigma}_{n+1} \quad \text{(mode 3)}\,, \end{cases} \tag{8.12}$$

where Φ and Ψ are the discrete-time representations of F and f as defined in (8.7) with sampling time τ_s. The third (auxiliary) mode refers to the mode transition where the switch S_1 opens within a subperiod. Note that if we are in the third mode, i.e. $\sigma_n \ \wedge\ \bar{\sigma}_{n+1}$ holds, $(\nu d(k) - n)$ is bounded by zero and one. Thus, the third mode constitutes a weighted average of modes one and two. The error introduced by averaging can be made arbitrarily small by increasing ν.

On the evolution of the states $\xi(n)$, we impose the safety current limit by adding the constraints[3]

$$-i'_{\ell,max} \leq [1\ 0]\xi(n) \leq i'_{\ell,max}, \quad n = 0, 1, \ldots, \nu - 1\,. \tag{8.13}$$

The notion of the ν-resolution modelling approach thus allows us to not only impose the current limit at the time-instants kT_s, but also on the intermediate samples with the finer resolution $\frac{T_s}{\nu}$.

Using the sampled output voltage given by

$$v'_o(n) = g^T \xi(n)\,, \tag{8.14}$$

we approximate the voltage error integral (8.9) for the k-th period in the following way.

$$v'_{o,err}(k) = \sum_{n=0}^{\nu-2} \frac{v'_o(n) + v'_o(n+1)}{2(\nu - 1)} - v'_{o,ref} \tag{8.15}$$

[3]Here, we constrain the current between its upper and lower bounds. In practise, however, only the upper constraints are needed, since a negative current implies that power is drawn from the load to the source what might only occur during short transients.

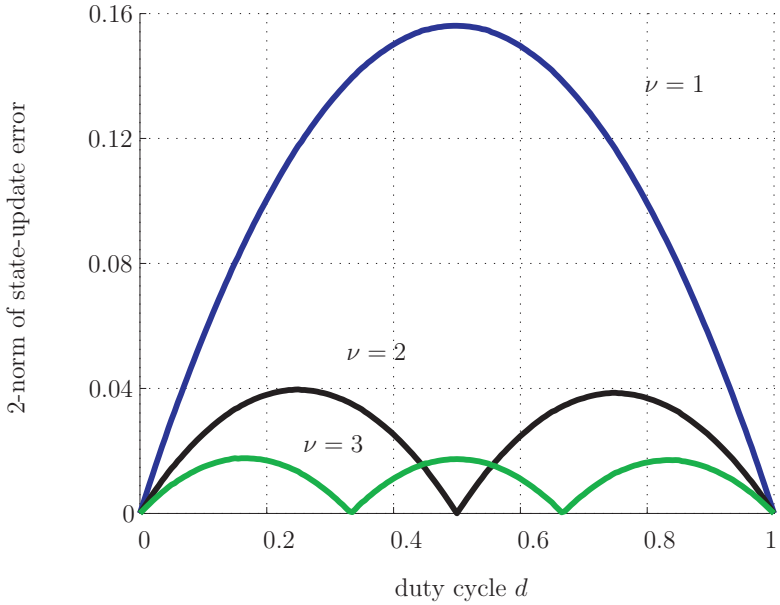


Figure 8.3: Accuracy (2-norm error) of the state-update function of the ν-resolution model with respect to the nonlinear dynamics

In summary, the ν-resolution modelling approach provides a description of the state evolution within one period. In particular, the discrete-time sequence $[\xi(0), \xi(1), \ldots, \xi(\nu-1)]$ is an accurate sampled representation of the continuous-time evolution of $x'(t)$ for $t \in [kT_s, (k+1)T_s]$. The only approximation that has been introduced appears in the third mode of (8.12) when the switch S_1 is turned off.

In particular, ν is a design parameter that can be chosen depending on the desired model accuracy. For the set of converter parameters in Table 8.1, which we will use at the end of the chapter for the simulations, Fig. 8.3 shows the accuracy of the state-update function (derived by exact time-discretization) of the ν-resolution model with respect to the nonlinear dynamics, by plotting the 2-norm of the error for various values of ν over the duty cycle $d(k)$. As the error is independent from the state $x'(k)$, this comparison holds for the whole state-space. The choice of $\nu = 1$ yields the standard average model, which is predominately used for the controller design for DC-DC converters. Obviously, the average model is perfectly accurate for $d(k) = 0$ and $d(k) = 1$, and it is at its worst for $d(k) = 0.5$. As one can see, setting $\nu = 2$ already significantly improves the accuracy of the model. We would like to stress once more that these results hold for the whole state-space making the model globally a valid approximation for all operating points, rather than locally for a specific operating point, as standard linearization would do.

Before proceeding, we define constraints on the states, the parameters and the input. For the states, we require $x'(k) \in \mathcal{X}' = [\underline{i}'_\ell, \bar{i}'_\ell] \times [\underline{v}'_o, \bar{v}'_o]$, and the model parameters are restricted to $[v'_{o,ref}\ i'_{\ell,max}] \in \mathcal{V}' = [\underline{v}'_{o,ref}, \bar{v}'_{o,ref}] \times [\underline{i}'_{\ell,max}, \bar{i}'_{\ell,max}]$, where the lower and upper

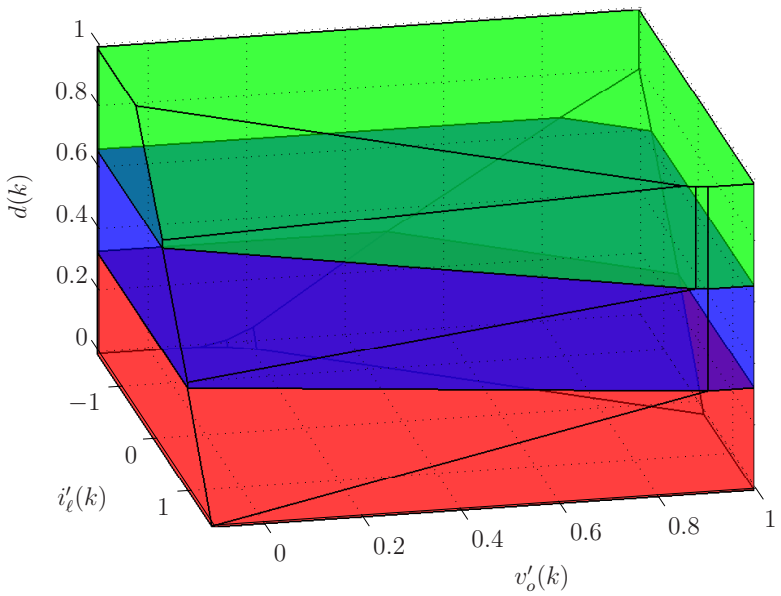


Figure 8.4: Polyhedral partition of the converter's PWA model for $\nu = 3$ (for the converter parameters in Table 8.1, intersected with $v'_{o,ref} = 0.556\,\text{p.u.}$ and $i'_{\ell,max} = 1.667\,\text{p.u.}$)

bounds are application specific. The input, on the other hand, is physically restricted to $d(k) \in \mathcal{U} = [0, 1]$.

8.2.3 ν-Resolution Model in MLD Form

The three operation modes of the ν-resolution model call for appropriate modelling using hybrid methodologies. For this, we use the HYSDEL modelling language [TB04] to conveniently describe the converter on a high-level textual basis; the code is summarized in Appendix A.4. The derivation of the MLD model given as in (2.8) is performed by the HYSDEL compiler, which generates the corresponding matrices. For details about the MLD framework, the reader is referred to Section 2.2.2. For the ν-resolution model, the above procedure yields an MLD model with two states, two parameters, (7ν) z-variables, $(2(\nu-1)+1)$ δ-variables and $(24\nu+8)$ mixed-integer linear inequality constraints.

8.2.4 ν-Resolution Model in PWA Form

For the computation of the explicit state-feedback control law, the converter model is required to be in PWA form. The mode enumeration algorithm allows for efficiently transforming HYSDEL models into their equivalent PWA representation. For more details about the algorithm, the reader is referred to Chapter 3.

The resulting PWA model is defined as a set of affine dynamics on a polyhedral partition of the five-dimensional space $\mathcal{X}' \times \mathcal{V}' \times \mathcal{U}$ given by the states, the parameters and the input. Since the binary variables σ_n, $n = 0, \ldots, \nu - 1$ define in (8.11) the modes along the axis of the duty cycle, we expect the $\mathcal{X}' \times \mathcal{V}' \times \mathcal{U}$ space to be partitioned along the duty cycle into ν polyhedra.

Example 8.1 To visualize the PWA model of the converter, consider as an example the set of converter parameters given in Table 8.1. Furthermore, we set $\mathcal{X}' = [-4, 4] \times [-0.1, 1]\,\text{p.u.}$, $\mathcal{V}' = [0.2, 1] \times [0.6, 3]\,\text{p.u.}$ and $\mathcal{U} = [0, 1]$. To visualize the polyhedral partition, we perform an intersection of $\mathcal{X}' \times \mathcal{V}' \times \mathcal{U}$ with $v'_{o,ref} = 0.556\,\text{p.u.}$ and $i'_{\ell,max} = 1.667\,\text{p.u.}$ thus removing the two dimensions corresponding to the parameter space $\mathcal{V}'$. Fig. 8.4 shows the resulting polyhedral partition of the state-input space $\mathcal{X}' \times \mathcal{U}$, where we have additionally restricted the first state i'_ℓ to $[-i'_{\ell,max}, i'_{\ell,max}]$. Note that small (large) capacitor voltages and large (small) duty cycles correspond to large (small) inductor currents. Since we have added the upper and lower safety current constraint in (8.13) as hard constraints to the model, these state-input combinations are removed from the $\mathcal{X}' \times \mathcal{V}' \times \mathcal{U}$ space.

Fig. 8.4 vividly visualizes the global character of the derived hybrid model. As the converter dynamics are linear in the states, there is no partitioning in the state-space. Yet, they are nonlinear in the duty cycle. We have approximated this nonlinearity by the ν-resolution modelling approach, where we partition the duty cycle in ν segments and approximate the transition from the first to the second mode by a third (auxiliary) mode. In particular, the hybrid model is continuous when moving from one polyhedron to a neighboring one. This follows from the state-update equations (8.12) and is confirmed by the continuity in Fig. 8.3.

8.3 Constrained Optimal Control

In this section, we first formulate the control objectives in terms of an objective function that leads in combination with the MLD model to a constrained finite time optimal control (CFTOC) problem, which can be solved on-line. In a second step, we compute off-line the state-feedback control law. For an overview of CFTOC, the reader is referred to Section 2.4. In a last step, we introduce a Kalman filter that adjusts the output voltage reference to address unmeasured changes in the load resistor.

In the sequel, we assume that the input and output voltages v_s and v_o, respectively, and the inductor current i_ℓ can be measured. The output reference voltage $v_{o,ref}$ and the current limit $i_{\ell,max}$ are given by the problem setup. Based on those measurements and

parameters, the scaled quantities v'_o, $v'_{o,ref}$, i'_ℓ and $i'_{\ell,max}$, which will be used as the inputs to the optimal controller, directly follow.

8.3.1 Objective Function

In general, the control objectives are to regulate the average output voltage to its reference as fast and with as little overshoot as possible, or equivalently, to (i) minimize the output voltage error $v'_{o,err}$ (ii) despite changes in the input voltage v_s or changes in the load resistance r_o and (iii) to respect the constraint on the inductor current. For now, we assume that the load resistance r_o is known. We will later relax this assumption.

To induce a steady state operation under a constant non-zero duty cycle, we introduce with

$$\Delta d(k) = d(k) - d(k-1) \tag{8.16}$$

the difference between two consecutive duty cycles.

Define the penalty matrix $Q = \text{diag}(q_1, q_2)$ with $q_1, q_2 \in \mathbb{R}^+$ and the vector[4] $\varepsilon(k) = [v'_{o,err}(k)\ \Delta d(k)]^T$, with $v'_{o,err}(k)$ as defined in (8.15), and consider the objective function

$$J(x'(k), d(k-1), v'_{o,ref}(k), i'_{\ell,max}(k), D(k)) = \sum_{\ell=0}^{N-1} \|Q\ \varepsilon(k+\ell|k)\|_1\,, \tag{8.17}$$

which penalizes the predicted evolution of $\varepsilon(k+\ell|k)$ from time-instant k on over the finite horizon N using the 1-norm. The objective function not only depends on the sequence of control inputs $D(k) = [d(k), \ldots, d(k+N-1)]^T$ and the current (measured) state $x'(k)$, but also on the last control input $d(k-1)$, the output voltage reference $v'_{o,ref}(k)$ and the current limit $i'_{\ell,max}(k)$, which are allowed to be time-varying to account for changes in the input voltage $v_s(k)$.

Summing up, objective (i) is incorporated in the objective function, and objective (ii) is handled by normalizing the prediction model by v_s, feeding the model with $v'_{o,ref}$, which is basically the inverse of v_s, and assuming for now that r_o is known and the Prediction Model can be updated accordingly. Objective (iii) is accounted for in the Prediction Model where it is imposed as a hard constraint[5] on the parameter $i'_{\ell,max}$.

[4] In [GPM04c] we have used a saturated version of $\Delta d(k)$ rather than $\Delta d(k)$ to allow for aggressive control moves when the voltage error is large but to force the controller to act cautiously if the output voltage is close to the reference and the voltage error is small. Since this introduces additional polyhedra in the PWA model increasing the complexity of the state-feedback controller, and as the gain in control performance is limited, we refrain from this concept here.

[5] In [GPM04c], however, we have imposed the upper bound on the inductor current as a soft constraints, which either leads to an additional model output and an additional binary variable or requires a slack variable. Here we refrain from doing so to not potentially induce additional polyhedra in the state-feedback control law.

8.3.2 On-Line Computation of Control Input

The control input at time-instant k is then obtained by minimizing the objective function (8.17) over the sequence of control inputs $D(k)$ subject to the mixed-integer linear inequality constraints of the MLD model (2.8) described in Section 8.2.3, the physical constraints on the sequence of duty cycles

$$0 \leq d(\ell) \leq 1\,, \quad \ell = k, ..., k+N-1\,, \tag{8.18}$$

and the expression (8.16). This amounts to the CFTOC

$$D^*(k) = \arg\min_{D(k)} J(x'(k), d(k-1), v'_{o,ref}(k), i'_{\ell,max}(k), D(k)) \tag{8.19a}$$

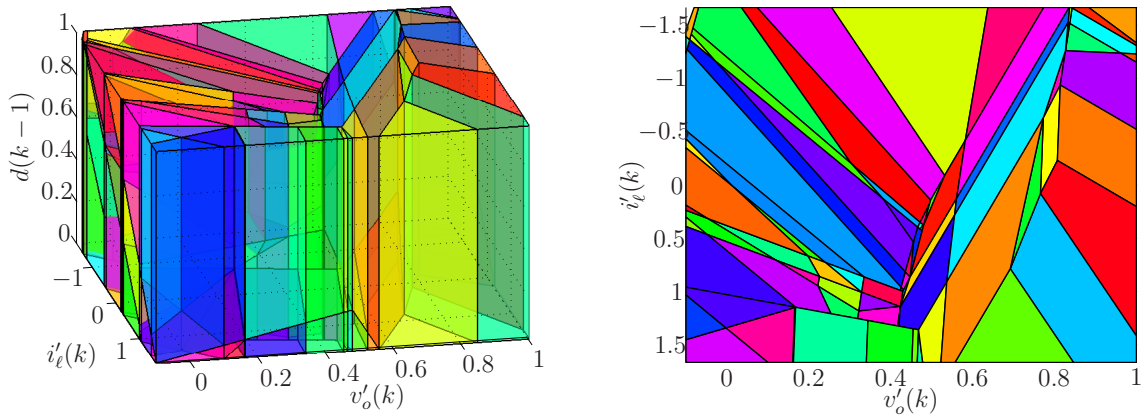
$$\text{subj. to MLD model, (8.18), (8.16)} \tag{8.19b}$$

leading to the sequence of optimal duty cycles $D^*(k)$, of which only the first duty cycle $d^*(k)$ is applied to the converter. At the next sampling interval, k is set to $k+1$, a new state measurement is obtained, and the CFTOC problem is solved again over the shifted horizon according to the receding horizon policy. As we are using the 1-norm in all cost expressions, the CFTOC problem amounts to solving a *Mixed-Integer Linear Program* (MILP) for which efficient solvers exist (like [ILO02]).

8.3.3 Off-Line Computation of State-Feedback Control Law

To allow for an implementation of the proposed controller despite the high switching frequency, the solution to the CFTOC problem (8.19) needs to be computed off-line. To do so, we replace in (8.19) the MLD model by the equivalent PWA model (2.9) derived in Section 8.2.4. For the PWA model, we assume the load resistance to be time-invariant and nominal ($r_o = 1\,\text{p.u.}$). Then, we can compute the PWA state-feedback control law employing Dynamic Programming and multi-parametric programming as outlined in Section 2.4.2, where the state vector $x'(k)$, the last control input $d(k-1)$, the output voltage reference $v'_{o,ref}(k)$ and the current limit $i'_{\ell,max}(k)$ are treated as parameters. In particular, we refrain from parameterizing the control law in terms of the load, as motivated in the next section. We summarize the parameters in the vector $p'(k) = [(x'(k))^T\ d(k-1)\ v'_{o,ref}(k)\ i'_{\ell,max}(k)]^T$ with $p'(k) \in \Pi = \mathcal{X}' \times \mathcal{U} \times \mathcal{V}'$. The resulting control law is a PWA function of $p'(k)$ defined on a polyhedral partition of the five-dimensional parameter space Π. For more details concerning the algorithm, the properties of its solution and techniques for implementation, the reader is referred to Section 2.4.2 and [Bor03].

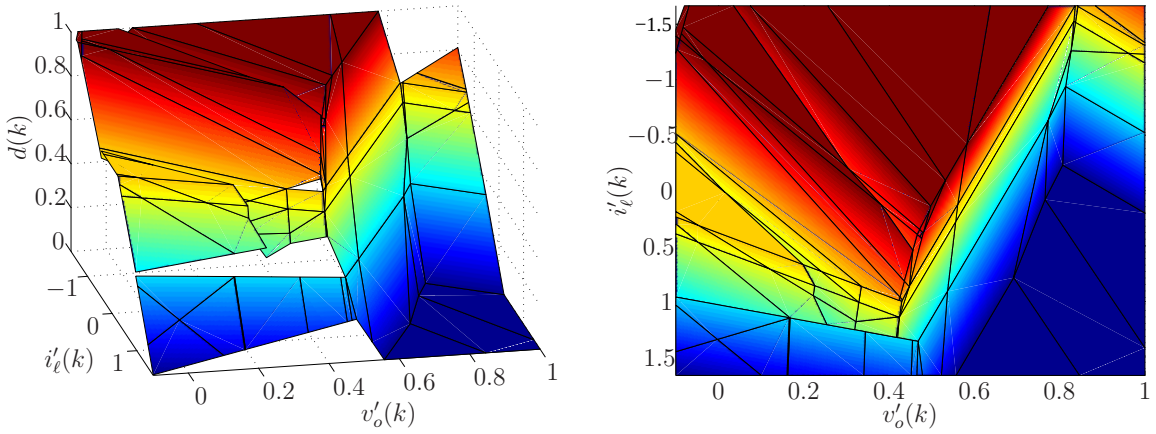
Example 8.2 For the PWA model derived in Example 8.1 with the model and control problem parameters given in Table 8.1, we compute the PWA state-feedback control law,



(a) Polyhedral partition of the $\mathcal{X}' \times \mathcal{U}$ space

(b) Polyhedral partition of the $\mathcal{X}'$ space (additionally intersected with $d(k-1) = 0.6$)

Figure 8.5: Polyhedral partitions of the state-feedback control law for $v'_{o,ref} = 0.556\,\text{p.u.}$ and $i'_{\ell,max} = 1.667\,\text{p.u.}$, where the colors are arbitrarily associated to the polyhedra



(a) Control law in $\mathcal{X}' \times \mathcal{U}$ space

(b) Control law in $\mathcal{X}'$ space

Figure 8.6: State-feedback control law $d(k)$ for $d(k-1) = 0.6$, $v'_{o,ref} = 0.556\,\text{p.u.}$ and $i'_{\ell,max} = 1.667\,\text{p.u.}$, where dark blue corresponds to $d(k) = 0$ and dark red to $d(k) = 1$

which is defined on 633 polyhedral regions in the five-dimensional parameter space Π. Using the optimal complexity reduction algorithm described in Chapter 4, the controller is simplified to 121 regions.

To visualize the state-feedback control law, we substitute $v'_{o,ref} = 0.556\,\text{p.u.}$ and $i'_{\ell,max} = 1.667\,\text{p.u.}$ into the control law. As a result, the control law, which refers now to the nominal case, is defined on the three-dimensional space $\mathcal{X}' \times \mathcal{U}$. Fig. 8.5 shows the corresponding

polyhedral partition, and Fig. 8.6 depicts the control input $d(k)$ as a PWA function of $x'(k)$, where we additionally set $d(k-1) = 0.6$. Note that the control law is well-defined, that is for each $x'(k) \in \mathcal{X}'$ and $d(k-1) \in \mathcal{U}$ exists a polyhedron and an associated affine control law such that $d(k)$ can be evaluated as can be seen from Fig. 8.6(b). Yet, the control law is discontinuous leading to the gaps visible in Fig. 8.6(a).

8.3.4 Load Variations

In addressing load changes, the following considerations arise from restrictions related to the controller's implementation. In an idealized simulation environment, one might assume that the load resistance is known and that sufficient computational power is at hand to solve the on-line optimization problem with the load resistance being a parameter that is updated on-line. In reality, however, the load resistance is not known and the computational power is rather limited. The latter makes it mandatory to refrain from solving the optimal control problem on-line and to derive the state-feedback control law as detailed in Section 8.3.3. For this controller, we have assumed that the load is time-invariant and nominal, since a parametrization of the control law over the load is not a feasible option, because the load enters the model equations nonlinearly, as can be seen from (8.7). To account for that, numerous PWA approximations would be necessary leading to an overly complex PWA model and an extremely complex state-feedback control law. Therefore, although the load might be theoretically estimated (e.g. by using an extended Kalman filter), this is not a viable option. What is needed, however, is a way to cope with load changes without introducing too much of an additional complexity. Hence, we aim at using the previously derived state-feedback controller with an additional loop.

As can be seen from (8.7), changes in the load resistor affect the converter dynamics and the DC gain. This is particularly the case, when the load decreases significantly below the nominal value (by 50 per cent or more). In this case, however, the only objective of the controller is to respect the safety constraint on the inductor current and to drop the output voltage accordingly. Predicting accurately the plant's output voltage over several switching periods is not needed. We conclude that this case is potentially easy to be treated. On the other hand, if the load is roughly nominal or increased beyond its nominal value, then the dynamics and the DC gain are subject only to minor changes. Yet, due to the accuracy requirement for the output voltage regulation (steady state error below one per cent), even small mismatches in the dynamics or the DC gain need to be addressed. We suggest to cope with these issues by adjusting the scaled output voltage reference fed into the controller such that the error between the output voltage and the *actual* reference is made small.

This can be achieved by augmenting the Prediction Model (8.7)–(8.10) by a third state

v'_e that tracks the output voltage error, and by using a Kalman filter [Jaz70] to estimate the augmented state vector

$$x'_a = \left[\; i'_\ell \; v'_o \; v'_e \; \right]^T . \tag{8.20}$$

The augmented model has the switched stochastic continuous-time state equation

$$\frac{\mathrm{d}x'_a(t)}{\mathrm{d}t} = \begin{bmatrix} F & 0 \\ 0 & 0 \end{bmatrix} x'_a(t) + \left\{ \begin{array}{ll} f & \text{if } kT_s \leqslant t < (k+d(k))T_s \\ 0 & \text{if } (k+d(k))T_s \leqslant t < (k+1)T_s \end{array} \right\} + G\omega_1(t) \tag{8.21}$$

and the measurement equation

$$\begin{bmatrix} i'_\ell(t) \\ v'_o(t) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix} x'_a(t) + H\omega_2(t)\,, \tag{8.22}$$

with the matrices $G = \mathrm{diag}(1,1,1)$ and $H = \mathrm{diag}(1,1)$. The random variables $\omega_1(t) \in \mathcal{R}^3$ and $\omega_2(t) \in \mathcal{R}^2$ represent the process and the measurement noise, respectively. They are assumed to be independent of each other with white and normal (Gaussian) probability distributions with $E[\omega_1\omega_1^T] = W_1$, $E[\omega_2\omega_2^T] = W_2$ and $E[\omega_1\omega_2^T] = 0$, where 0 is a zero matrix of appropriate dimension. We require $GW_1G^T \succeq 0$ and $W_2 + HW_1H^T \succ 0$. Note that the augmented model is detectable and uses the nominal value of the load resistor.

The Kalman filter provides an estimate of the augmented state $\hat{x}'_a(t)$, such that the (steady-state) covariance matrix of the error between the state vectors of the augmented plant and the estimator is minimized. The covariances of the noises are used to reflect the dominant source of uncertainty. In general, they express the trade off between the credibility of the obtained measurements with respect to the uncertainties on the dynamics of the augmented stochastic model (8.21). A further elaboration on the concept of the Kalman filter is beyond the scope of this thesis and the reader is referred to one of the numerous control theory textbooks.

To allow for an easy implementation of the Kalman filter, we use a time-invariant filter with a constant Kalman gain. Furthermore, for each mode of operation, one Kalman filter is needed, and we switch between them according to the switch transitions in the converter. Such an approach is possible as the mode transitions, which are imposed by the duty cycle, are precisely known. The initial state of each mode is set to the last state of the previous mode.

The vector on the left hand side of (8.22) holds the measurements from the plant with a generally disturbed load $r_o \neq 1\,\mathrm{p.u.}$. The states on the right hand side are estimated by the Kalman filter using a converter model with the nominal load $r_o = 1\,\mathrm{p.u.}$. Therefore, the third state v'_e primarily[6] relates to the output voltage error that is caused by the model

[6] As mentioned in Section 8.2.1, the converter model may be subject to additional uncertainties apart from r_o and v_s. Yet, those are in general minor with respect to the load changes. Furthermore, changes in v_s are addressed by the normalization and adjusting $i'_{\ell,max}$.

mismatch due to the different loads. By setting all noise covariances to small values expect for the process noise on v'_e, the output voltage error is tracked by v'_e. In a last step, we adjust the output voltage reference $v'_{o,ref}$ by the voltage error. Specifically, we replace $v'_{o,ref}$ in (8.9) by

$$\tilde{v}'_{o,ref} = v'_{o,ref} - \hat{v}'_e \,. \tag{8.23}$$

8.4 Analysis

In an a posteriori analysis, we aim at showing that the optimal controller leads to exponential closed-loop stability by deriving a piecewise quadratic (PWQ) Lyapunov function. This is possible since we have an explicit representation of the optimal control law at our disposal. We describe the plant by the PWA Prediction Model (8.7)–(8.10), and restrict ourselves to the nominal case with nominal load and nominal input voltage. Hence, a Kalman filter adjusting the output voltage reference is obsolete and the output voltage reference is time-invariant. Using the Prediction Model to close the loop, a closed-loop system results, which is PWA and autonomous by definition. Let $x'_c(k) = [i'_\ell(k)\ v'_o(k)\ d(k-1)]^T$, $x'_c(k) \in \mathcal{X}' \times \mathcal{U}$ denote its state vector, and assume that

$$\bar{x}'_c = \lim_{k\to\infty} x'_c(k) \tag{8.24}$$

exists with $\bar{v}'_o = v'_{o,ref}$. Since we aim at showing stability of the equilibrium point $\bar{x}'_c$, we perform the coordinate transformation

$$\zeta(k) = x'_c(k) - \bar{x}'_c \,. \tag{8.25}$$

In the sequel, we consider the autonomous system with state vector $\zeta(k) \in \mathcal{Z}$, $\mathcal{Z} = \{\zeta \,|\, \zeta + \bar{x}'_c \in \mathcal{X}' \times \mathcal{U}\}$. For this system, consider the control invariant subset $\mathcal{Z}_0 \subseteq \mathcal{Z}$ defined as

$$\mathcal{Z}_0 = \{\zeta(0) \in \mathcal{Z} \,|\, \zeta(k) \in \mathcal{Z} \ \ \forall k \geq 0\} \,. \tag{8.26}$$

We adopt the PWQ function

$$L(\zeta) = \zeta^T P_i \zeta \tag{8.27}$$

with i being the index of the polyhedron of the autonomous system that holds ζ. In particular, P_i is a time-invariant 3×3 matrix corresponding to the i-th polyhedron. We impose

$$L(\zeta(k)) \geq \varrho_i ||\zeta(k)||_2^2 \ \ \forall \zeta(k) \in \mathcal{Z}_0 \tag{8.28}$$

and

$$L(\zeta(k+1)) - L(\zeta(k)) \leq -\rho ||\zeta(k)||_2^2 \ \ \forall \zeta(k) \in \mathcal{Z}_0 \,, \tag{8.29}$$

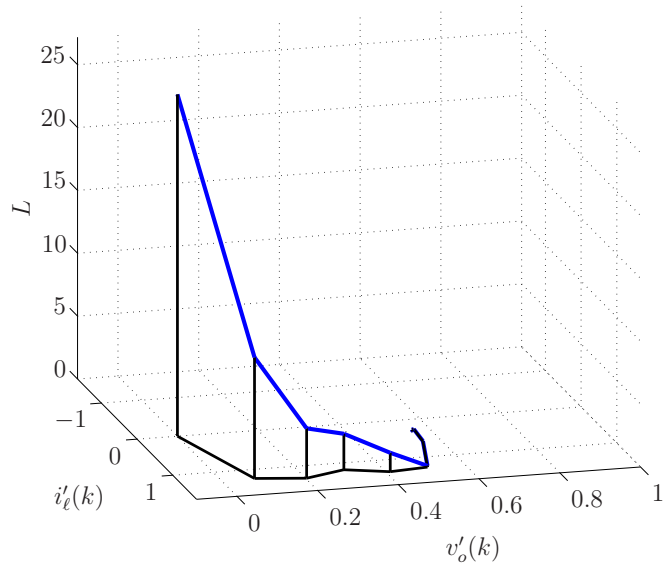


Figure 8.7: Value of Lyapunov function along closed-loop trajectory during start-up plotted on the $\mathcal{X}'$ plane

where $\varrho_i > 0$ and $\rho > 0$. Note that (8.28) does not imply that the matrices P_i are positive definite since the inequalities are required to hold only for the states in the i-th polyhedron. Furthermore, $L(\zeta)$ may be discontinuous across the polyhedral boundaries.

Theorem 8.1 *[FCMM02, Theorem 1] The equilibrium $\zeta = 0$ of the above autonomous system is exponentially stable on $\mathcal{Z}_0$ if there exists a PWQ Lyapunov function $L(\zeta)$ as in* (8.27)–(8.29).

For details on Lyapunov functions for PWA systems and computation approaches, the reader is referred to [FCMM02] and [GLPM03], respectively.

Example 8.3 Consider the nominal Prediction Model with $r_o = 1\,\text{p.u.}$, $v_s = 1.8\,\text{p.u.}$, $v_{o,ref} = 1\,\text{p.u.}$ and $i_{\ell,max} = 3\,\text{p.u.}$, and the corresponding state-feedback control law derived in Example 8.2. An analysis shows that the assumption in (8.24) holds, namely a steady state solution $\bar{x}'_c$ with $\bar{v}'_o = v'_{o,ref}$ exists allowing the derivation of the closed-loop autonomous system with $\zeta(k) \in \mathcal{Z}$. Evaluating the state-feedback controller shows that the control invariant subset is equal to $\mathcal{X}' \times \mathcal{U}$, and consequently $\mathcal{Z}_0 = \mathcal{Z}$. This implies that for any initial state within $\mathcal{X}' \times \mathcal{U}$, all constraints will be met at all future time-instants. In particular, a control input will be always found.

Using the Multi-Parametric Toolbox [KGBM04], a PWQ Lyapunov function $L(\zeta)$ with $\rho = 3.4 \cdot 10^{-5}$ is found within 2.9 min on a 2.8 GHz Pentium IV machine. Consequently, for the nominal system, the state $\bar{x}'_c$, which corresponds to the output voltage reference $v'_{o,ref}$, is exponentially stable for $\mathcal{X}' \times \mathcal{U}$. We would like to stress that the closed-loop system is indeed *globally* stable for all $x'_c(k) \in \mathcal{X}' \times \mathcal{U}$, and not only locally around $v'_{o,ref}$.

Parameters of the Converter					
x_c	70 p.u.	x_ℓ	3 p.u.	$i_{\ell,max}$	3 p.u.
r_c	0.001 p.u.	r_ℓ	0.05 p.u.	r_o	1 p.u.
Parameters of the Control Problem					
ν	3	N	2		
q_1	4	q_2	0.1		

Table 8.1: Model and controller parameters used for the simulation results

For the nominal start-up, the decaying value of the Lyapunov function along the closed-loop trajectory $x_c'(k)$, $k \in \mathbb{N}_0$, is depicted in Fig. 8.7. In this figure, $L(x_c'(k) - \bar{x}_c')$ is plotted over the two-dimensional state-space $\mathcal{X}'$, where the third dimension corresponding to $d(k-1) \in \mathcal{U}$ has been omitted. Note that for $\mathcal{X}'$ the same scaling is used as in Figs. 8.5 and 8.6 to allow for a straightforward comparison.

8.5 Simulation Results

In this section, simulation results demonstrating the potential advantages of the proposed control methodology are presented. Specifically, we examine the closed-loop dynamical behavior for the start-up, the response to step changes in the input voltage, and the response to step changes in the load resistance. These three cases are of prime interest in practical applications and pose performance challenges for any control scheme. The simulations were carried out using the nonlinear model of the converter as the real plant, closing the loop with the PWA state-feedback controller. The inductor current of the converter and the input and output voltages were regarded to be measurable as it is with the current industrial practise. Furthermore, we neglect measurement noise. All variables in the following figures are normalized to the *per unit* system, and one time unit of the time axis equals one switching period.

The circuit parameters of the plant used in the simulations were chosen to represent a realistic problem set-up, describing for example a 24 V to 12 V, 150 W step-down DC-DC converter. Table 8.1 summarizes the converter parameters expressed in the per unit system. If not otherwise stated, the input voltage is $v_s = 1.8$ p.u. and the output resistance is given by $r_o = 1$ p.u.. The output voltage reference is $v_{o,ref} = 1$ p.u..

The Prediction Model uses the same parameters as the plant models, with the difference that it is scaled with respect to v_s and that it always uses the nominal load $r_o = 1$ p.u.. Even though two subperiods in the ν-resolution modelling approach yield sat-

isfactory results, we chose $\nu = 3$ subperiods to accurately model the nonlinear dynamics. In Sections 8.2.3 and 8.2.4, we have derived the corresponding MLD and PWA model, respectively. The polyhedral partition of the PWA model is visualized in Fig. 8.4.

Regarding the optimal control scheme, the penalty matrix is chosen to be $Q = \mathrm{diag}(4, 0.1)$, putting a rather small weight on the changes of the manipulated variable. The prediction horizon is in all cases $N = 2$. As detailed in Section 8.3.3 and Example 8.2, the state-feedback control law is derived, which is shown in Figs. 8.5 and 8.6.

For the covariance matrices of the Kalman filter, we set

$$W_1 = \begin{bmatrix} 0.1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 100 \end{bmatrix} \qquad W_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{8.30}$$

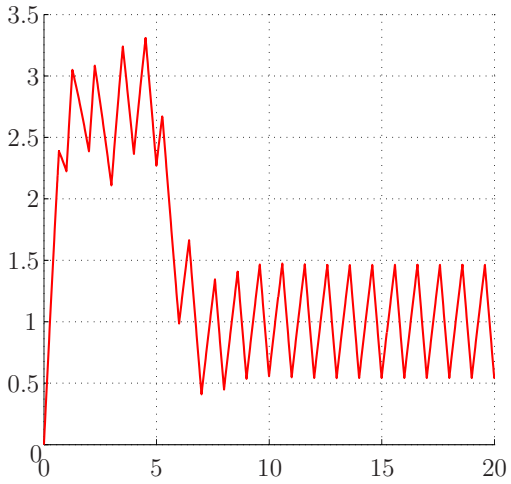
These are the same both for the on and the off mode (corresponding to S_1 being on and off, respectively).

8.5.1 Nominal Start-Up

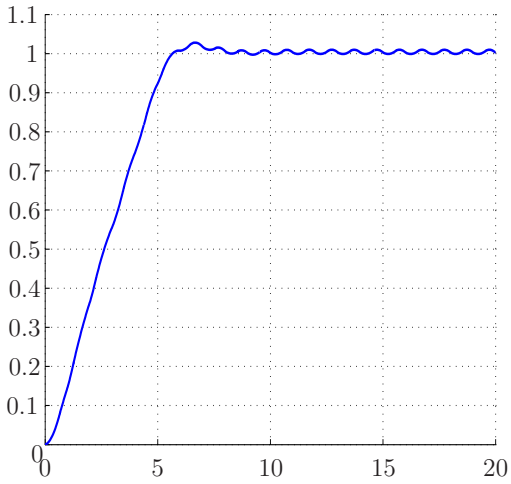
Fig. 8.8 shows the step response of the converter in nominal operation during start-up with the initial state $x(0) = [0 \;\; 0]^T$ and $d(-1) = 0$. The output voltage reaches its steady state within ten switching periods with an overshoot that does not exceed three per cent. As mentioned in the introduction, settling times of up to 30 periods and overshoots of five per cent are commonly encountered when using PI-type controllers. The current constraint is basically respected by the inductor current during start-up. The small violations are due to the coarse resolution chosen for the ν-resolution model, as the current constraint can be only imposed at ν time-instants within the switching period. The steady state error present in the output voltage is with 0.5 per cent sufficiently small and can be further reduced by increasing ν.

Note that the steady state ripple in the converter variables results from the fact that the converter is a switched circuit. Specifically, the ripple is an open-loop characteristic of the system that depends on the plant parameters and the operating point. In particular, it cannot be affected by the controller action. However, the ripple might be reduced by altering the design of the power stage of the converter, namely by increasing the switching frequency, or, at the expense of slower settling times, by increasing the capacitor and/or the inductor.

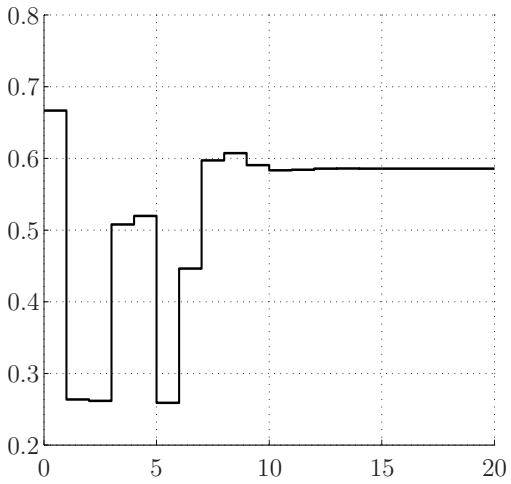
In Fig. 8.9, we compare the following three control schemes for the nominal start-up: MPC with a $\nu = 3$ model, MPC with an averaged model ($\nu = 1$) and the industrial standard. The latter is the so called current mode control, which is the standard control scheme used to handle current constraints. Since the generic scheme is known to be unstable for duty cycles above 0.5 [MM01], we have included a slope compensation scheme



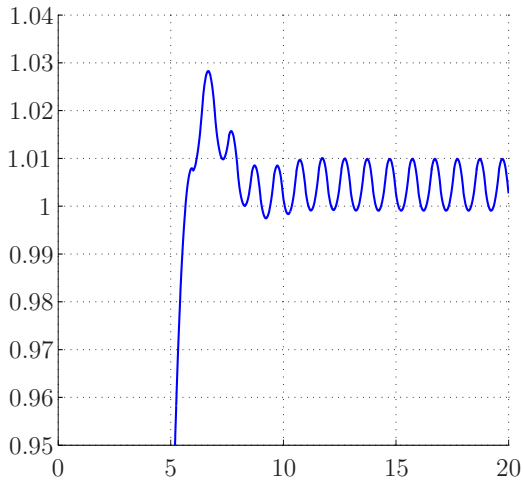
(a) Inductor current $i_\ell(t)$



(b) Output voltage $v_o(t)$



(c) Duty cycle $d(t)$

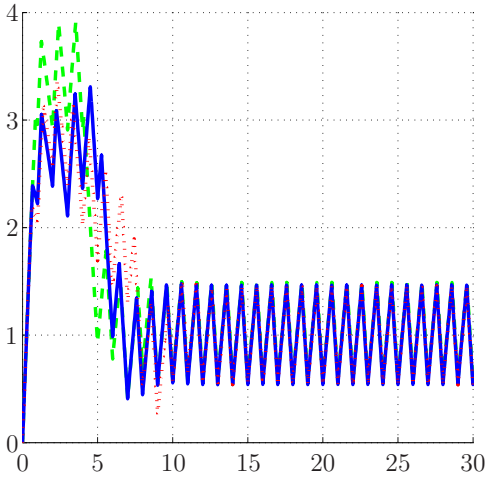


(d) Detail of $v_o(t)$

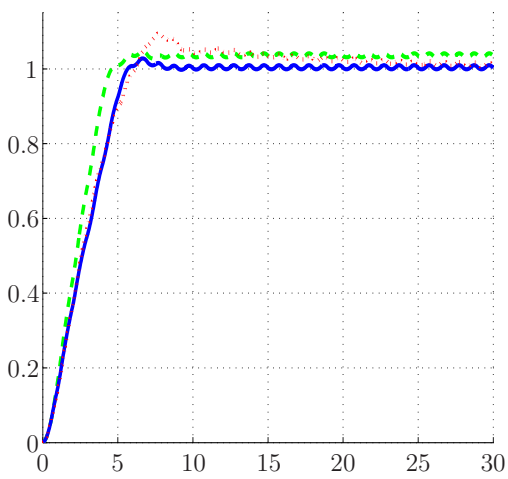
Figure 8.8: Closed-loop response during start-up in nominal operation

to remove this instability. This addition and the tuning of the PI controller is done in accordance with industrial practice following the design procedure summarized in [MM01].

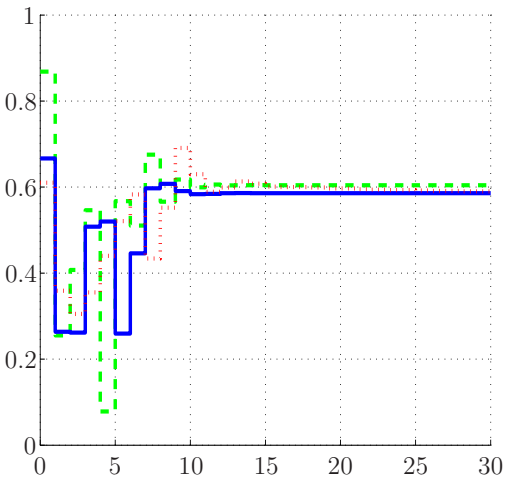
The MPC scheme using the averaged model with $\nu = 1$ leads to a steady state output error of 3.5 per cent. Moreover, the current constraint is violated by up to 30 per cent, thus making a shorter rise time possible. The inaccurate averaged model motivates the use of a hybrid model with $\nu > 1$. The current mode PI controller, on the other hand, respects the current constraint and yields the same rise time as MPC with the $\nu = 3$ Prediction Model. Yet it exhibits a large overshoot of almost ten per cent and a large settling time of approximately 30 switching periods.



(a) Inductor currents $i_\ell(t)$



(b) Output voltages $v_o(t)$

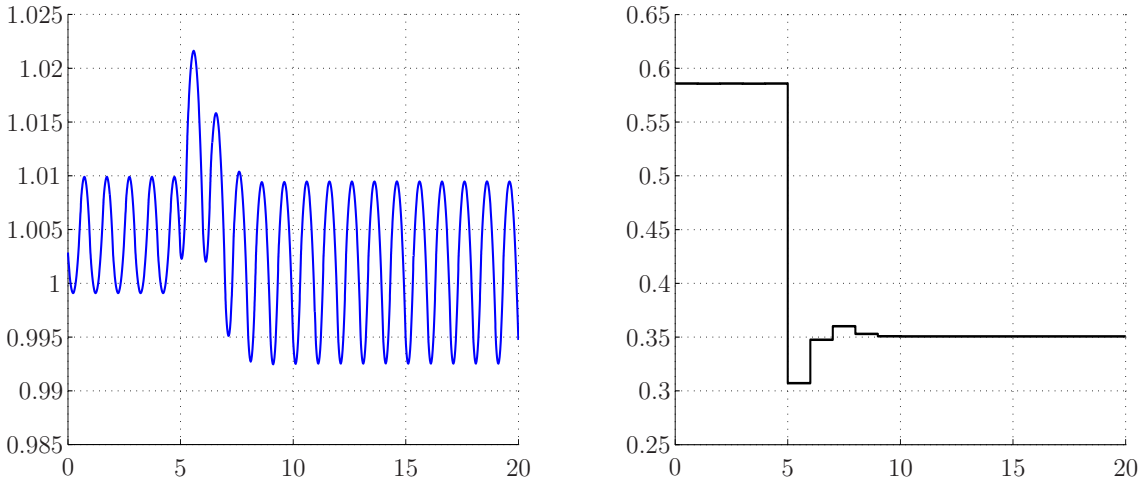


(c) Duty cycles $d(t)$

Figure 8.9: Comparison of closed-loop responses during start-up in nominal operation for MPC with $\nu = 3$ model (straight blue lines), MPC with $\nu = 1$ model (dashed green lines), and current mode PI controller (dotted red lines)

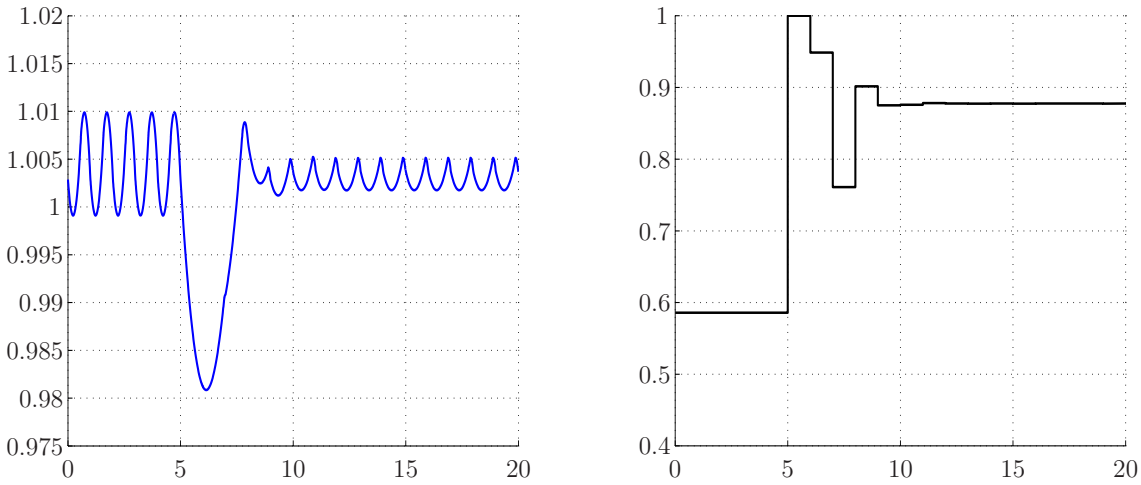
8.5.2 Step Change in Input Voltage

Initially, the converter is operating at steady state with the nominal input voltage $v_s = 1.8\,\text{p.u.}$ when step changes in the input voltage are applied. These measured disturbances occur at time-instant $k = 4$, but are fed to the controller with a delay of one switching period. We consider two step changes; one up to $v_s = 3\,\text{p.u.}$, and one down to $v_s = 1.2\,\text{p.u.}$. The step-up change is shown in Fig. 8.10. The output voltage remains practically unaffected and the controller finds the new steady-state duty cycle very quickly



(a) Output voltage $v_o(t)$ (b) Duty cycle $d(t)$

Figure 8.10: Closed-loop response to a step-up change in the input voltage from $v_s = 1.8\,\text{p.u.}$ to $v_s = 3\,\text{p.u.}$ at time-instant $k = 4$



(a) Output voltage $v_o(t)$ (b) Duty cycle $d(t)$

Figure 8.11: Closed-loop response to a step-down change in the input voltage from $v_s = 1.8\,\text{p.u.}$ to $v_s = 1.2\,\text{p.u.}$ at time-instant $k = 4$

within four switching periods. This new duty cycle is also responsible, due to the open-loop characteristics of the converter, for a larger ripple in the inductor current and the output voltage. Similarly, when applying the step-down change, the output voltage remains practically unaffected, and the relatively large undershoot results from the physical limitation of the duty cycle, as can be seen in Fig. 8.11.

Summing up, disturbances in the input voltage are rejected very effectively by the controller, and the output voltage is quickly restored to the reference. This is due to the fact that the state-feedback control law is indirectly parameterized over the input voltage by scaling the measured states, the output voltage reference and the current limit with respect to v_s. As a result, the performance of the controller is not affected by changes in v_s.

8.5.3 Step Change in Output Resistance

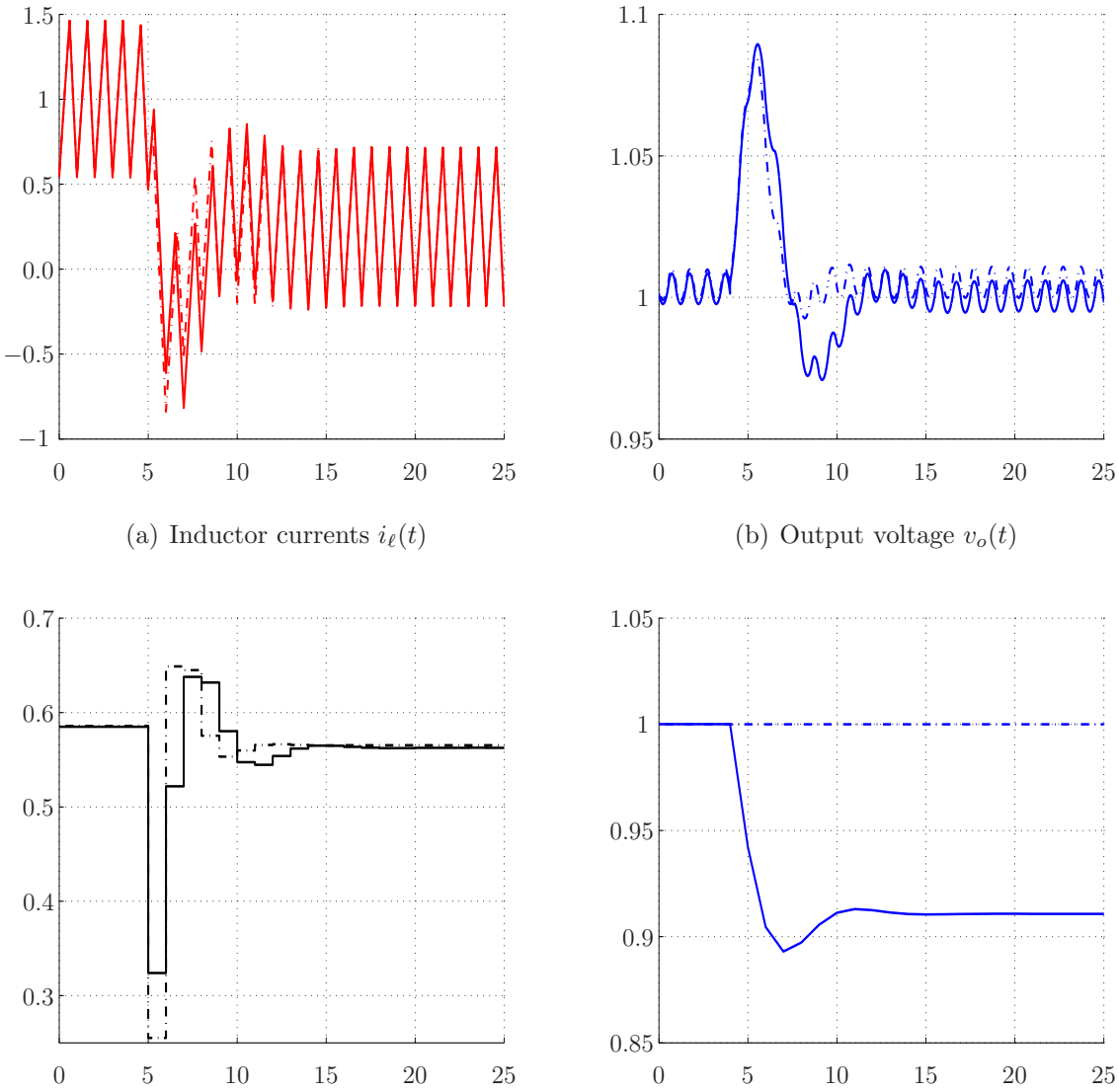
In a last step, we investigate the closed-loop performance in the presence of major step changes in the output resistance r_o. Starting from the nominal load $r_o = 1\,\text{p.u.}$, at time-instant $k = 4$, three steps to $r_o = 4\,\text{p.u.}$, $r_o = 0.5\,\text{p.u.}$ and $r_o = 0.05\,\text{p.u.}$ are applied. The last case corresponds to a short circuit and aims at activating the constraint on the current limit. In the sequel, we consider the following two control schemes.

- Implementable Scheme: For the explicit state-feedback control law, we assume r_o to be unknown. More specifically, the Prediction Model and the state-feedback control law are designed assuming nominal load conditions, and the Kalman filter is used to adjust the output voltage reference $v'_{o,ref}$. This corresponds to an implementable control scheme.

- Idealized Scheme: For comparison, we also consider a simulation set-up where we solve the CFTOC on-line. For this, we assume r_o to be known and update the Prediction Model, which is given in MLD form, accordingly. As for the other simulations above, we assume a time delay of one switching period. Obviously, the Kalman filter is obsolete for this setup and $v'_{o,ref}$ is thus time-invariant.

The three experiments are shown in Figs. 8.12, 8.13 and 8.14. The straight lines correspond to the implementable control scheme, the dash-dotted lines correspond to the idealized scheme.

Figs. 8.12 and 8.13 depict the closed-loop performance of the two schemes for the step-up and the step-down case. As can be observed, the dynamic behavior of the implementable scheme is similar to the idealized scheme – the major difference being the larger settling time, which results from the dynamics of the Kalman filter. These depend on the noise covariance matrices that allow for a trade-off between speed and sensitivity. Apart from that, the Kalman filter in the implementable scheme tends to yield smaller steady-state errors in the output voltage due to its inherent integrating action. Note that the large over- and undershoots observed in these two cases are equal for both control schemes, since they stem from the open-loop characteristics of the converter (relatively small output capacitor)



(a) Inductor currents $i_\ell(t)$

(b) Output voltage $v_o(t)$

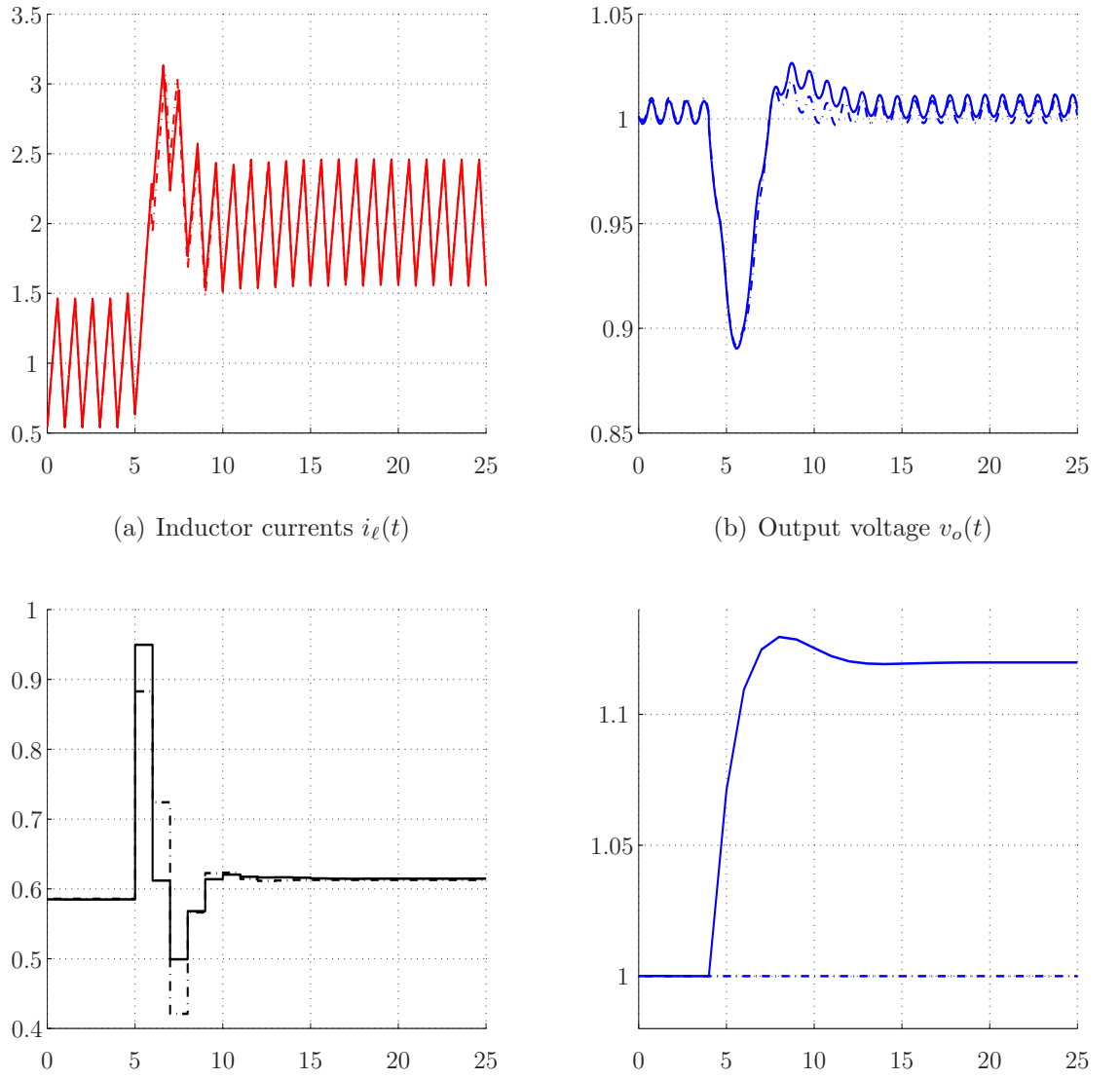
(c) Duty cycle $d(t)$

(d) Adjusted output voltage reference $v_{o,ref}(t)$

Figure 8.12: Closed-loop response to a step-up change in the load resistor from $r_o = 1\,\text{p.u.}$ to $r_o = 4\,\text{p.u.}$ at time-instant $k = 4$. The straight lines correspond to the implementable control scheme (state-feedback controller and Kalman filter), the dash-dotted lines correspond to the idealized scheme (on-line solution with model adaptation)

and the fact that the worst-case scenario is examined here, where the disturbance is fed to the control scheme with the maximum time delay of one switching period.

In the last case, we examine a crucial aspect of the controller operation, namely the system's protection against excessive load currents. The load drops at $k = 4$ from its nominal value to a very small one (namely to $r_o = 0.05$), almost creating a short circuit



(a) Inductor currents $i_\ell(t)$

(b) Output voltage $v_o(t)$

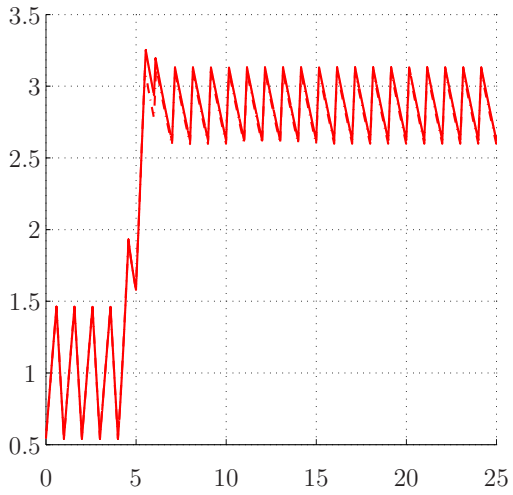
(c) Duty cycle $d(t)$

(d) Adjusted output voltage reference $v_{o,ref}(t)$

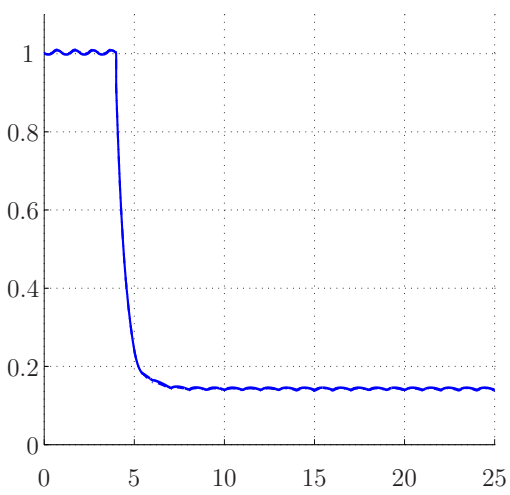
Figure 8.13: Closed-loop response to a step-down change in the load resistor from $r_o = 1\,\text{p.u.}$ to $r_o = 0.5\,\text{p.u.}$ at time-instant $k = 4$. The straight lines correspond to the implementable control scheme (state-feedback controller and Kalman filter), the dash-dotted lines correspond to the idealized scheme (on-line solution with model adaptation)

at the output. The simulation results in Fig. 8.14 show that the controller respects the current limit and forces the output voltage v_o to drop to the level that is needed in order to respect the constraint.

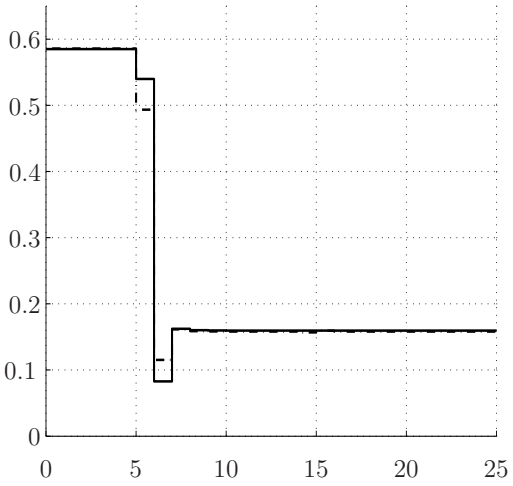
Such a feature is utilized in all practical applications through various protection schemes, but is usually not explicitly considered as part of the controller design. The proposed



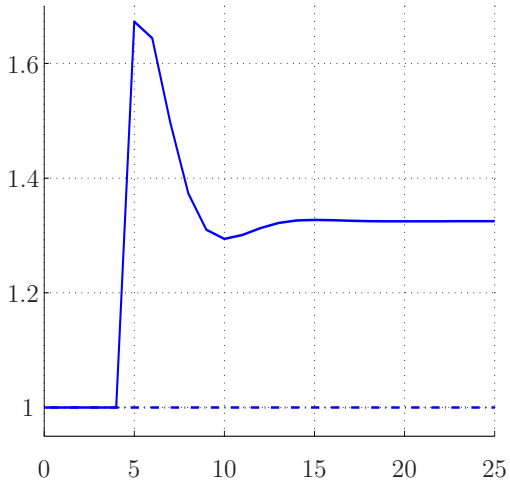
(a) Inductor currents $i_\ell(t)$



(b) Output voltage $v_o(t)$



(c) Duty cycle $d(t)$



(d) Adjusted output voltage reference $v_{o,ref}(t)$

Figure 8.14: Closed-loop response to a short circuit (the load resistance is reduced from $r_o = 1\,\text{p.u.}$ to $r_o = 0.05\,\text{p.u.}$) at time-instant $k = 4$. The straight lines correspond to the implementable control scheme (state-feedback controller and Kalman filter), the dash-dotted lines correspond to the idealized scheme (on-line solution with model adaptation)

approach, however, addresses the current constraint (as well as the duty cycle constraint) explicitly during the controller design, as can be also seen from the start-up simulation.

Considering the comparison of the two control schemes, we observe that they both practically yield the same performance, since the Kalman filter and the inaccuracy introduced by the usage of the nominal r_o are overshadowed by the presence of the current

constraint. The argument that follows is that for small changes in r_o, a Kalman filter is needed leading to a zero steady-state error, but possibly to a small deterioration of the dynamic performance. Yet for large load drops, the Kalman filter has hardly any effect on the closed-loop dynamics due to the activation of the current constraint. In particular, the Kalman filter does not lead to violation of this constraint. This argument justifies the reasoning in Section 8.3.4, where we have proposed the use of a prediction model with nominal r_o in combination with a Kalman filter.

8.6 Conclusions and Future Research

Conclusions

In this chapter, we have presented a new control approach for fixed frequency switch-mode DC-DC converters by formulating an optimal control problem using hybrid systems methodologies. More specifically, a novel ν-resolution hybrid model was introduced to avoid averaging and to model the converter arbitrarily accurate, and a constrained finite time optimal control problem was formulated and solved on-line and off-line. This control methodology allowed us to explicitly take into account during the design phase physical constraints, such as the restriction of the duty cycle to the interval between zero and one, and safety constraints, such as current limiting. The off-line solution to the control problem yielded an explicit state-feedback controller defined over a polyhedral partition of the state-space that allows for the practical implementation of the proposed scheme. This controller is parameterized not only over the measured states $i'_\ell(k)$ and $v'_o(k)$, which are scaled by the input voltage v_s, and the previous duty cycle $d(k-1)$, but also over the scaled output voltage reference $v'_{o,ref}(k)$ and the scaled current limit $i'_{\ell,max}(k)$. This allowed us to efficiently reject disturbances in the input voltage of any magnitude. Moreover, the addition of a Kalman filter estimating the output voltage error and adjusting the voltage reference accordingly provides disturbance rejection to large changes in the output resistance. These include short circuits, for which the output voltage is dropped such that the safety constraints is respected. Most importantly, we were able to compute the control invariant set and showed that the controller renders the nominal system exponentiallly stable. Simulation results have been provided demonstrating that the proposed controller leads to a closed-loop system with very favorable dynamical properties.

Simplification

These favorable properties come to the expense of a rather complex control law that is yet expected to be implementable using today's available hardware. Nevertheless, when assuming a time-invariant output voltage reference as it is the case in a large number of

applications, the controller can be simplified at the expense of a slight deterioration in the performance with regard to the current limit. More specifically, let $v_{o,ref}$ be constant and, without loss of generality, equal to one. Then, one may rewrite (8.10) as

$$i'_{\ell,max} = i_{\ell,max}\frac{v'_{o,ref}}{v_{o,ref}} = i_{\ell,max}v'_{o,ref}\,. \tag{8.31}$$

With the reasonable assumption that $i_{\ell,max}$ is time-invariant, and taking into account that the control law uses $v'_{o,ref}$ as a parameter, this reformulation removes the parameter $i'_{\ell,max}$ from the controller thus reducing the dimension of its parameter space from five to four. The number of polyhedra of the controller is reduced roughly by a factor of two. However, since the Kalman filter adjusts the output voltage reference, we have replaced at the end of Section 8.3.4 $v'_{o,ref}$ by $\tilde{v}'_{o,ref} = v'_{o,ref} - \hat{v}'_e$ as in (8.23). This can lead to an ill-computation of the current limit within the prediction model, and subsequently to a violation of the actual current constraint. This violation is a linear function of the estimated output voltage offset $\hat{v}'_e$. Therefore, the adverse influence of the above simplification can be easily limited by imposing an upper constraint on $\hat{v}'_e$.

Future Research

Rather than deriving a discrete-time PWA model of the converter, another approach would be to consider the nonlinear discrete-time averaged model, which is nonlinear but not hybrid. Using this model, we propose to formulate a nonlinear optimal control problem, and to derive the explicit control law using multi-parametric nonlinear programming (mp-NLP) [Fia83]. Such a problem has been tackled in an approximate way in [Joh02] by locally approximating the mp-NLP with multi-parametric quadratic programming (mp-QP) sub-problems. Further results are reported in [Joh04]. The ideas reported in [BF02] and [Bem04] might prove to be useful and inspiring, too. Based on these concepts, an mp-NLP approach suitable for DC-DC converters should emerge that might be extended in a later stage to a more general problem class.

Alternatively, one might refrain from computing an explicit state-feedback controller altogether and solve the control problem on-line. The main advantage of such an approach would be that parameters such as the input voltage, the output voltage reference, and an estimate of the load resistance could be updated on-line in the prediction model. Such an approach might be computationally feasible when restricting oneself to a very short control horizon. The prediction horizon, however, might be chosen to be rather large if necessary, since it does not significantly increase the computational burden. In particular, for a control horizon of one, the optimizer of the minimization problem would be scalar. By deriving a solution technique that exploits the system structure and is tailored to the problem, a very short computation time is expected, which even might lie in the range

of the time needed to search through a look-up table with 100 or more polyhedra in a four or five-dimensional space. With such an approach, we would gain an enormous flexibility and might be able to tackle any converter topology without major changes in the problem set-up. This idea is inspired by the MPC-E scheme in Section 7.4 and the on-line optimization reported in [LK04].

Part V

Power Systems

9

Emergency Voltage Control in Power Systems

9.1 Introduction

9.1.1 Voltage Stability in Power Systems

An electrical power system consists of numerous components connected together to form a complex system generating, transmitting and distributing electrical power. Apart from being among the largest man-made systems, power systems are substantially nonlinear, the time-constants of its dynamics comprise several orders of magnitude, and many of its control inputs are discrete-valued. Examples for the latter include capacitor banks and tap changers, which must be switched using fixed step sizes, and load-shedding, which must be carried out by disconnecting whole feeders since most utilities lack direct load control schemes. Furthermore, generators include saturations to protect them from over-excitation, and propositional logic, finite state machines and discrete events in general are often part of embedded control loops. As an example, consider a transformer equipped with an on-load tap changer (OLTC) and a rule-based controller to regulate the secondary voltage by adjusting the transformer's tap position. Concluding the above, power systems are inherently hybrid systems.

Electric power systems and additional preventive control schemes are designed in such a way that the system should be able to withstand any single contingency, that is, outage of any single component without loss of stability and with all system variables kept within predefined ranges [Kun94]. This is usually referred to as N-1 security. Not all possible disturbances, however, can be foreseen at the planning stage and these may result in instability leading eventually to collapse or islanding of the system. Guaranteeing that a power system can cope with one outage is in many cases not sufficient, since the initial outage triggers often additional outages thus requiring N-2 or N-3 security that is hard and costly to ensure. Furthermore, because of environmental constraints on the extension of

the transmission capacity, increased electricity consumption and new economic constraints imposed by the liberalized power market, power systems are operated closer and closer to their stability limits.

If a power system is heavily stressed, it can become unstable exhibiting slow voltage drops that may lead to a voltage collapse resulting in a blackout if appropriate countermeasures are not taken. The main factors contributing to a voltage collapse are the generator limits on the reactive power leading to a loss of voltage control at the generators, and the self-restoring dynamic behavior evident in most loads. Moreover, additional primary control loops regulating the voltage like OLTC can lead to instabilities. In general, the main problem is the inability of the power system to meet its reactive demands. A voltage collapse generally manifests itself as a slow decay of voltage over a time frame of several minutes [Kun94].

In the past, a number of severe voltage instability incidents have occurred around the globe [Tay94, vV98, TE97], most notably the recent blackouts in northeastern US and southern Canada in August 2003 [US 04], in southern Sweden and eastern Denmark in September 2003, followed only within a few days by the major blackout in Italy that has affected most of the country's 58 million people. As a consequence, voltage stability has become a major concern in power system planning and operation and the need for emergency control schemes that ensure stability – even during cascaded or multiple outages – has increased. For an introduction to voltage stability, the reader is referred to the textbooks [Kun94, Tay94, vV98] and the extensive bibliography [AL98].

In this chapter, we focus on the emergency voltage control problem on the level of the transmission system. Emergency control needs to be applied in case of severe contingencies that result in the violation of operational limits or even a voltage collapse unless proper emergency control actions are taken. In the next section, we provide a brief review of some of the related control methodologies.

9.1.2 Review of Emergency Voltage Control

Currently, most practical implementations of protection systems against voltage collapses are purely rule-based and most often rely only on local criteria [CIG00]. General rules are to disconnect load and to connect any available capacitor bank if the voltage drops to abnormally low levels. Traditionally, these rules are tuned on an *ad-hoc* basis [CIG00]. In [vML02] however, a *systematic* approach has been reported, where an optimally tuned rule is derived by solving a combinatorial optimization problem. Preliminary work by the same authors includes [MLv00], where a Genetic Algorithm is used.

While local protection schemes have the obvious advantage of simplicity – they do not require wide-area communication – it has been been shown that substantial benefits can

be gained by coordinating the actions taken in different parts of the system [LK03]. In particular, proper coordination of all control measures minimizes the amount of load shed. Additionally, because of the nonlinearity of the power system, it is very difficult to specify a single appropriate rule for the complete range of operating conditions.

Recent advances in computation, communication and power system instrumentation technology, more specifically Phasor Measurement Units and Wide-Area Measurement Systems [Reh01], have made coordinated and model-based approaches tractable. They are highly attractive since the use of a model in combination with on-line optimization allows for optimal coordination of different control moves and automatic adaption to changing operating conditions. Thanks to this, they are less conservative than non-adaptive local schemes – even if the local schemes have been optimally tuned – and avoid unnecessary operations of the protection schemes thus minimizing the amount of load being shed.

Hence, in the sequel, we focus on emergency voltage control schemes adopting a centralized approach based on wide-area measurements. These schemes can be grouped into static and dynamic control approaches. Static approaches such as power flow methods are based on static models, and aim at restoring a lost equilibrium by applying adequate control actions. These approaches can only guarantee the existence of such an equilibrium, but not that the system will be actually attracted to it. Therefore, we restrict ourself to control schemes that are based on dynamic models to account for load restoration phenomena and tap changer dynamics that constitute an integral part of the voltage instability phenomena. Moreover, we consider only schemes that are model-based and afford a systematic design procedure excluding heuristic methods.

Model Predictive Control

The concept of Model Predictive Control (MPC), which is summarized in Section 2.4, is very suitable to be applied to power systems by formulating the emergency control problem as an optimal control problem over a receding horizon. Specifically, the control objectives – like keeping the voltages close to their nominal values, minimizing the control moves, and avoiding load-shedding unless absolutely necessary – can be easily mapped into a cost function. In particular, priorities in the control actions can be easily specified to distinguish between cheap and expensive (emergency) control actions like load-shedding. Since the optimal control input is obtained by solving the optimal control problem on-line, the model can be adapted on-line in the presence of faults or changes in the topology of the power system. Most importantly, MPC can directly handle soft and hard constraints, and deal with integer manipulated variables and linear hybrid systems. The disadvantages of MPC are that a rather accurate model of the power system is necessary, and that the computation times for hybrid systems may explode when increasing the problem complexity.

Linearized Models In [LHO02], a control scheme for optimal coordination of load-shedding, capacitor switching and tap changer operation is proposed that uses a dynamic system model to preserve long-term voltage stability. This paper is the first to apply MPC to the emergency voltage control problem. The hybrid behavior of the power system is taken into account at the current sampling instant, but neglected within the prediction interval, since a single linearized model is used for the whole prediction horizon. To fully take the hybrid behavior into account, it is suggested to obtain the system response for each admissible control move by simulating the system model over the prediction interval, what is in general computationally very demanding. Moreover, the control scheme is restricted to discrete-valued control variables thus leading to a purely combinatorial optimization problem, which is solved by an exhaustive tree search. The computational burden of the underlying optimization problem is reduced by employing a control horizon of one and limiting the depth of the tree, or equivalently, restricting the maximal number of control moves.

To render the problem tractable for large-scale applications, this work is extended in [LK03] by heuristic search enhancements. In particular, a transposition table, branch and bound techniques, search ordering, and iterative broadening are used. Moreover, the scheme uses a simpler prediction model that is a linearized about the current operating point of the power system. Yet, the scheme is successfully applied to the Nordic test system from [CIG95] yielding impressive improvements with respect to the best available local control strategies.

Models based on Trajectory Sensitivity In [ZA03], an emergency control scheme is proposed that uses MPC and the notion of trajectory sensitivity [HP00, HP02], which provide first-order approximations around the voltage trajectories (rather than an operating point). In the event of a contingency, the nominal system trajectory is predicted over a given time horizon. If any voltage violates a given bound, the emergency control scheme is triggered, and the sensitivities of the voltage trajectories are computed with respect to changes in the control inputs at the current time-instant. The control inputs include (constrained) real-valued inputs such as voltage references for Automatic Voltage Regulators (AVR) in generators, and discrete-valued inputs such as load-shedding and tap changer commands. Even though the hybrid character of the power system is partly accounted for by allowing for discrete-valued inputs, OLTC automata are neglected. An MPC formulation is devised with a control horizon of one, where the control actions are prioritized as follows: tap changes, changes in the generator reference voltages and load-shedding. Since the 1-norm is used, the underlying optimization problem amounts to a Mixed-Integer Linear Program (MILP).

A related approach is pursued in [HG04], where an emergency control scheme is pre-

sented aiming at using load-shedding in a non-disruptive way. This is achieved by exploiting the fact that in many cases load (like air conditioning) can be shed for a short time with hardly any noticeable effect to the consumer. A hierarchical control scheme is proposed, where a lower level scheme monitors and adjusts the controllable loads, and a higher level controller establishes a coordinated load shedding strategy. For the latter, MPC is used with a control horizon of one. The internal model of MPC incorporates a static voltage dependent load model, and constraints on the control inputs and the controlled voltages are introduced. The objective is to shed the minimal amount of load such that the voltage constraints are met. This leads to a nonlinear optimization problem, which is turned into an easy-to-solve Linear Program (LP) by approximating the voltage behavior using the methodology of trajectory sensitivity [HP00, HP02]. Summing up, by not explicitly addressing the hybrid nature of the power system, assuming real-valued bounded control inputs and using first-order approximations around the voltage trajectories leads to a promising control approach with an underlying optimization problem that can be efficiently solved for problems of very large scale.

Contributions of *Control and Computation* Project

In the *Control and Computation (CC)* project of the IST research framework IST-2001-33520, two power systems defined in [Lar02] and [Lar03] were proposed as case studies. These systems exhibit hybrid features, and are intended to develop and evaluate hybrid control methodologies. The system [Lar03] is a two bus power system with an infinite bus, a transmission corridor, a transformer equipped with an OLTC, a capacitor bank and a dynamic load. The case study in [Lar02] features four buses and extends the aforementioned power system by two more transmission corridors and a generator that is equipped with an AVR. Before outlining our approach to tackle the case study [Lar02], we provide for completeness a summary of the contributions proposed by the CC project partners.

The approach described in [AAC05] tackles the four bus power system [Lar02], and aims at formulating a nonlinear optimal control problem with the objective to keep the voltages within the imposed bounds while minimizing load-shedding. A number of simplifications are introduced, namely the problem is reformulated such that the power system features only discrete-valued control inputs, and some of the control inputs are required to remain constant over the prediction horizon. The first simplification leads to a discrete-valued set of admissible control sequences, and the second one drastically reduces the cardinality of this set. The control sequences are ordered such that non-disruptive control actions like tap changing and capacitor switching are preferred over load-shedding. The simplified nonlinear optimal control problem is solved at each time-step by simulating the system responses to all admissible control sequences, using a Matlab/Simulink representation

of the model. Previously, the same approach has been used to tackle the simplified benchmark problem [Lar03]. Results can be found in [AAC04].

Obviously, such an approach is only feasible since all control inputs are discrete-valued, and the set of control sequences has been considerably reduced. Unfortunately, an internal model to obtain the predictions is not available, and hence a closed form of the optimization problem cannot be derived. Moreover, classic solver techniques for combinatorial optimization problems such as branch and bound are abandoned. Initially, in 2001, we have implemented precisely such an approach, too, but considered all control sequences and used branch and bound techniques to cut down on the computation time. The approach was dropped since it cannot handle real-valued control inputs and it scales inherently poorly to large problems.

Another line of research is pursued in [Sol04], where the author proposes for the two bus power system [Lar03] a controller consisting of three control loops. The first loop aims at driving the power system to a stable equilibrium point (provided that such a point exists). It manipulates the voltage reference of the OLTC, and consists of a feed-forward compensator from the line impedance, and a feedback controller from the line and load impedances, and the turn ratio of the transformer. The second and the third control loop shed load and connect capacitors to the power system, respectively, if certain thresholds on a stability margin are exceeded. It is our impression that this control scheme not only circumvents the hybrid features of the power system, but the tuning and prioritization of the control moves appears to be non-obvious and non-intuitive given that the controller consists of three interacting control loops. Instead of being a wide-area emergency control scheme, this approach rather resembles a local scheme that coordinates the control moves in the OLTC and the load-shedding in the attached load, using only local information.

9.1.3 Outlook

This chapter is structured in the following way. After summarizing in Section 9.2 the nonlinear model of the example power system, we detail the derivation of the prediction model in Section 9.3. For the modelling, we use the *Mixed Logical Dynamical (MLD)* framework (see Section 2.2.2) to account for the hybrid character of the power system. Even though nonlinear functions need to be approximated by piecewise affine (PWA) functions, the MLD methodology allows for modelling the power system in an arbitrarily accurate fashion. Most importantly, all hybrid features of the power system such as OTLC controllers with thresholds, propositional logic and state automata can be described by introducing integer variables and mixed-integer linear inequality constraints. Moreover, real-valued control inputs, and not only discrete-valued ones, can be directly addressed.

The optimal control problem is stated in Section 9.4, where we set up an MPC prob-

lem. In its objective function, the deviation of the bus voltages from their references, the violation of given soft constraints on these bus voltages, and the switching of the manipulated variables are penalized. The control moves are categorized as nominal and emergency control moves, with switching the capacitor bank and changing the voltage reference of the tap changer being nominal moves, while load-shedding constitutes the emergency control action. The latter is only applied if the soft constraints on the bus voltages cannot be met when only resorting to nominal control moves. Note that MPC allows for straightforwardly setting up the cost function. Specifically, tuning is avoided since the control moves are prioritized according to the aforementioned objectives. Moreover, our MPC scheme is capable of predicting the hybrid evolution of the power system, it features control horizons larger than one, and the underlying optimization problem is given in a closed form.

In Section 9.5, the optimal control problem is solved on-line and simulation results are provided. For the given example, the bus voltages can be stabilized by the proposed MPC scheme using only nominal control moves. The chapter is concluded in Section 9.6, where some future research directions are provided, too.

We would like to stress that in comparison to the previous work [LHO02, LK03], the MLD framework allows us to model the *hybrid* behavior of the power system and thus provides better accuracy since effectively the global behavior of the system is accounted for in the prediction interval. It is believed that this is the first time a general-purpose optimal control framework for hybrid systems is applied successfully to the emergency voltage control problem.

9.1.4 Notation

Throughout this chapter, we assume that the voltages V, the currents I and the apparent power S can be represented by phasors. The voltage V for example may be written as

$$V = V_m \mathrm{e}^{\delta} = V_d + jV_q\,, \tag{9.1}$$

where the absolute value (or magnitude) of the voltage is $V_m = |V|$, the angle is $\delta = \arg(V)$, and the direct and quadrature components are $V_d = \Re\{V\}$ and $V_q = \Im\{V\}$, respectively. As it is common practice in the power system community, all equations, variables and parameters are normalized using the per-unit system (see e.g. [MBB97]).

9.2 Nonlinear Model

In this chapter, we will focus on the aforementioned power system [Lar02, GLM02], which is contained in the *Control and Computation (CC)* project of the IST research framework

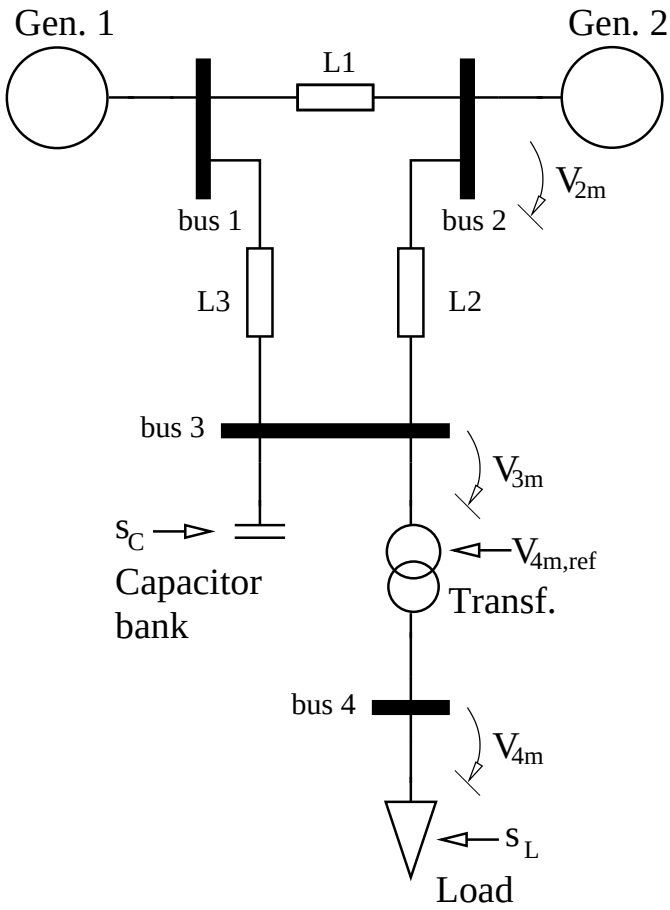


Figure 9.1: Example power system, where the number of capacitor banks connected to the system s_C, the amount of load shed s_L and the load voltage reference $V_{4m,ref}$ are the three input variables. The output variables are the three bus voltages V_{2m}, V_{3m}, V_{4m} at Bus 2, 3 and 4, respectively

IST-2001-33520 as a case study. This power system incorporates all the components of a mature power system and can be driven unstable exhibiting a voltage collapse. Nevertheless, it is small enough to serve as a starting point allowing for a successful derivation of an emergency control scheme. Moreover, as shown in [LRB03], such a small system can be used to model sensitive and vulnerable parts of a large power system. As an example, consider a transmission corridor between two subsystems as described in [LRB03].

9.2.1 Overview

As depicted in Fig. 9.1, the power system contains two generators. Generator 1 is modelled as an infinite bus, i.e. a large power system, whereas Generator 2 can only produce a limited amount of reactive power. The latter includes an internal controller regulating the voltage at Bus 2. The two generators are connected with each other and the transformer by the three transmission corridors L1, L2 and L3. All these components form the transmission

system of the power system.

The transformer incorporates an internal controller regulating the load voltage V_{4m} within a dead-band around the voltage reference $V_{4m,ref}$. This controller is a finite state machine and allows for changes in the tap position n_T only every 30 s by one discrete step. In addition, by manipulating s_C, parts of the capacitor bank can be used to support the power system by producing reactive power close to the load.

The distribution system that, in general, consists of numerous loads on different voltage levels connected with each other by transformers is modelled using one load model aggregating and approximating the whole distribution system. It is connected to the secondary side of the transformer. Discrete parts of the load can be disconnected by using s_L.

Originally, the model of the example power system has been given in MODELICA [Til01] describing the different component separately and connecting them with each other by algebraic network equations. In the next section, the model is restated and reformulated for convenience and to define a consistent notation.

9.2.2 Component Models

Hereafter, we present in detail the component models of the power system, namely the two generators, the capacitor bank, the transformer, the load and the network. The latter connects the components with each other via algebraic equality constraints thus forming the overall model. In this, the bus voltages $V_i = V_{id} + jV_{iq}$, $i \in \{1, 2, 3, 4\}$, and the currents $I_i = I_{id} + jI_{iq}$, $i \in \{1, 2, 3, 4, 5\}$ and $I_c = I_{cd} + jI_{cq}$ may be considered as "global" variables. Apart from these, additional "local" variables are necessary to model the internal behavior of the components. The respective parameters of the component models are given in Section 9.2.5.

Generator 1

Generator 1 represents a strong surrounding power grid that is modelled as an infinite bus with the fixed voltage V_1.

$$V_{1d} = V_{1m,ref} \tag{9.2a}$$

$$V_{1q} = 0 \tag{9.2b}$$

Generator 2

According to Fig. 9.2, Generator 2 is made up by three parts, namely the turbine, the synchronous machine and the automatic voltage regulator (AVR). The turbine delivers the constant mechanical power P_{m0} and drives the synchronous machine, which turns the mechanical power into electrical power. As the time-scales of voltage instability scenarios

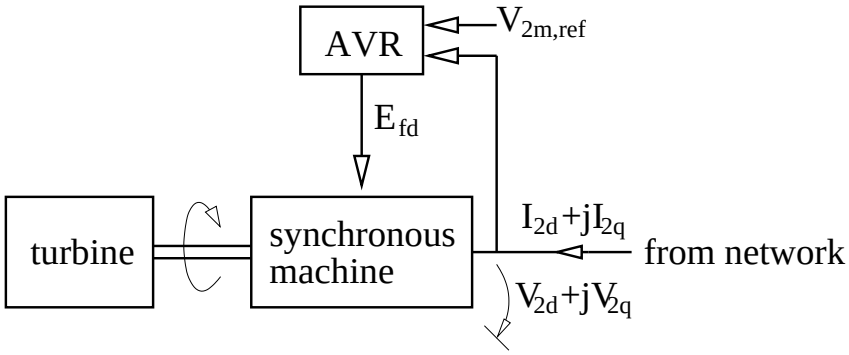


Figure 9.2: Scheme of Generator 2

are well above one second, the transient and subtransient dynamics of the synchronous machine may be neglected. Therefore, the synchronous machine exhibits simply a *static* nonlinear input-output behavior.

The swing equation together with the transient EMF equation and the equation for the electrical power amount to

$$P_{m0} = E_{fd} i_q + (x_d - x_q) i_d i_q \,, \tag{9.3}$$

where the field voltage E_{fd} is controlled by the AVR, and i_d and i_q are the rotating armature currents on the d- and q-axis, respectively. The armature voltages v_d and v_q are given by

$$v_d = -x_q i_q \tag{9.4a}$$

$$v_q = E_{fd} + x_d i_d \,. \tag{9.4b}$$

Depending on the rotor angle ϕ, the Park transformation relates the rotating armature entities to the terminal current $I_2 = I_{2d} + jI_{2q}$ and to the terminal voltage $V_2 = V_{2d} + jV_{2q}$.

$$-I_{2d} = -\sin(\phi) i_d + \cos(\phi) i_q \tag{9.5a}$$

$$-I_{2q} = \cos(\phi) i_d + \sin(\phi) i_q \tag{9.5b}$$

$$V_{2d} = -\sin(\phi) v_d + \cos(\phi) v_q \tag{9.5c}$$

$$V_{2q} = \cos(\phi) v_d + \sin(\phi) v_q \tag{9.5d}$$

The equations of the synchronous machine (9.3)–(9.5) can be simplified to the algebraic constraint

$$E_{fd} = \frac{V_{2m}^4 + V_{2m}^2 Q_2 (x_d + x_q) + (P_2^2 + Q_2^2) x_d x_q}{V_{2m} \sqrt{V_{2m}^4 + (P_2^2 + Q_2^2) x_q^2 + 2 V_{2m}^2 Q_2 x_q}} \tag{9.6}$$

with

$$P_2 = -V_{2d} I_{2d} - V_{2q} I_{2q} = P_{m0} \tag{9.7a}$$

$$Q_2 = V_{2d} I_{2q} - V_{2q} I_{2d} \tag{9.7b}$$

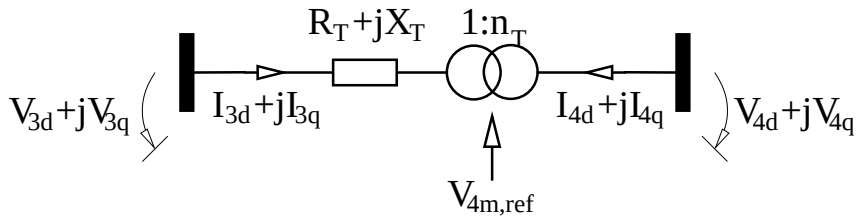


Figure 9.3: Scheme of the transformer

and

$$V_{2m} = \sqrt{(V_{2d})^2 + (V_{2q})^2}\,. \tag{9.8}$$

Assuming that the generator is operating in voltage control mode, an AVR is used to adjust the field voltage E_{fd} and consequently the current in the field winding of the synchronous machine. Specifically, the AVR is a saturating P-controller with an input nonlinearity controlling the absolute value of the terminal voltage V_{2m} by manipulating E_{fd}.

$$dE_{fd} = 50(V_{2m,ref} - V_{2m}) \tag{9.9a}$$

$$E_{fd} = \min(E_{f,max}, dE_{fd} + E_{f0}) \tag{9.9b}$$

The saturation element protects the generator against too high a field current and thus from overheating. Yet, when E_{fd} saturates, the control over the terminal voltage is lost, and in case of a voltage instability scenario, this is one of the driving forces leading to a voltage collapse if no appropriate countermeasures are taken.

Capacitor Bank

The capacitor bank can be used to inject reactive power close to the load. This enhances the apparent power factor of the load, which is usually inductive, and reduces the amount of reactive power to be transferred over the transmission network. Thus, the generators are relieved and additional active power can be transferred via the transmission lines. As a result, in general, the voltage at Bus 4 increases. According to the integer input $s_C \in \{0, 1, 2, 3\}$ to the capacitor bank, different discrete parts c_{step} of the capacitor banks are connected to Bus 3.

$$I_{cd} = -c_{step} \cdot s_C \cdot V_{3q} \tag{9.10a}$$

$$I_{cq} = c_{step} \cdot s_C \cdot V_{3d} \tag{9.10b}$$

Transformer

The transformer depicted in Fig. 9.3 connects the transmission system with the load by stepping down the voltage. Moreover, the transformer is equipped with a tap on one of the

windings to allow for controlling the load voltage by adapting the turns ratio. Since the tap can be changed whilst the transformer is energized, one refers to such a transformer as *under-load tap changing* (ULTC) or *on-load tap changing* (OLTC) transformer.

Modelling the transformer as an ideal transformer with the tap position n_T and the additional impedance $R_T + jX_T$, the primary voltages and currents can be expressed in terms of the secondary voltages and currents in the following way.

$$V_{3d} = \frac{1}{n_T} V_{4d} - n_T(R_T I_{4d} - X_T I_{4q}) \tag{9.11a}$$

$$V_{3q} = \frac{1}{n_T} V_{4q} - n_T(X_T I_{4d} + R_T I_{4q}) \tag{9.11b}$$

$$I_{3d} = -n_T I_{4d} \tag{9.11c}$$

$$I_{3q} = -n_T I_{4q} \tag{9.11d}$$

The absolute value of the secondary voltage V_{4m}, which is equal to the load voltage, is given by

$$V_{4m} = \sqrt{(V_{4d})^2 + (V_{4q})^2}\,. \tag{9.12}$$

This secondary voltage is required to vary only within small bounds around the given reference of the secondary voltage $V_{4m,ref}$ in order to ensure an adequately constant load voltage. This is accomplished by an internal voltage controller that manipulates n_T according to the evolution of the error $V_{4m} - V_{4m,ref}$. Specifically, a tolerance band centered around the reference voltage $V_{4m,ref}$ with the width d_T is introduced. If the load voltage V_{4m} exceeds this tolerance band, one of the two binary variables $\mathrm{lwTol}, \mathrm{upTol} \in \{0, 1\}$ is set to one, depending on whether the lower or the upper bound has been hit. These binary variables are defined by the logic implications

$$[\mathrm{lwTol} = 1] \quad \longleftrightarrow \quad [V_{4m} < V_{4m,ref} - \frac{d_T}{2}] \tag{9.13a}$$

$$[\mathrm{upTol} = 1] \quad \longleftrightarrow \quad [V_{4m} > V_{4m,ref} + \frac{d_T}{2}]\,. \tag{9.13b}$$

Combining both variables, we define the binary variable

$$\mathrm{excTol} = \mathrm{lwTol} \vee \mathrm{upTol} \tag{9.14}$$

stating whether the tolerance band has been exceeded or not.

The controller performs a tapping action, if the tolerance band has been exceeded for a certain time, provided that the lower or upper physical limits of the tap changer have not been reached yet. In [Lar02], the controller has been modelled as a finite state machine. Here, we model the controller in an equivalent yet simplified way by switched affine dynamics and logic implications as shown hereafter.

First, we introduce the timer t_T and the binary variable excDel stating whether the time delay has been exceeded or not.

$$[\text{excDel} = 1] \quad \longleftrightarrow \quad [t_T \geq \tau_{tapDelay} + \tau_{mechDelay}] \tag{9.15}$$

The time delay is composed of two parts. The tapping delay $\tau_{tapDelay}$, which is controller specific and a tuning parameter, is introduced to avoid excessive tapping actions. Usually, it lies in the range of 30 s to 60 s. The mechanical delay $\tau_{mechDelay}$ accounts for the time the transformer requires to tap one step up or down. This takes usually less than 1 s. Formulating the controller in the discrete-time domain, the timer is increased by the sampling interval T_s if the load voltage violates the tolerance band and the time delay has not been exceeded; else the timer is set to zero. This leads to the discrete-time switched dynamic

$$t_T(k+1) = \begin{cases} t_T(k) + T_s & \text{if } [\text{excTol}(k) \wedge \overline{\text{excDel}}(k) = 1]\,, \\ 0 & \text{else}\,, \end{cases} \tag{9.16}$$

where $k \in \mathbb{N}_0$ represents the discrete time-instant. In particular, the timer is reset to zero when the time delay has been exceeded and, as we will see later, a tapping action has been performed.

Since the tap position is constrained to $n_T \in [n_{min}, n_{max}]$, two additional binary variables are needed to evaluate if tapping can be physically performed, or more specifically, whether the tap position is not at its lower or upper limit n_{min} or n_{max}, respectively.

$$[\text{lwLmt} = 1] \quad \longleftrightarrow \quad [n_T = n_{min}] \tag{9.17a}$$

$$[\text{upLmt} = 1] \quad \longleftrightarrow \quad [n_T = n_{max}] \tag{9.17b}$$

If the load voltage is too large (small), the time delay has been exceeded and the transformer can physically tap down (up), the internal controller steps the transformer down (up) by one tap step n_{step}. Summing up, the discrete-time switched dynamic for the tap position n_T is

$$n_T(k+1) = \begin{cases} n_T(k) - n_{step} & \text{if } [\text{upTol}(k) \wedge \text{excDel}(k) \wedge \overline{\text{lwLmt}}(k)]\,, \\ n_T(k) + n_{step} & \text{if } [\text{lwTol}(k) \wedge \text{excDel}(k) \wedge \overline{\text{upLmt}}(k)]\,, \\ n_T(k) & \text{else}\,. \end{cases} \tag{9.18}$$

The controller can be further simplified if the sampling interval T_s is equal to 30 s. In this case, the timer along with (9.14)–(9.16) is not needed, and the term excDel is removed from (9.18).

Load

The whole distribution network with different voltage levels and various loads like motors, heating, lighting, etc. is aggregated in one model shown schematically in Fig. 9.4. The

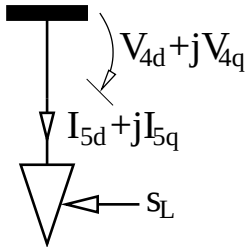


Figure 9.4: Scheme of the load

power consumed by most load devices is voltage dependent. Hence, the load admittance varies dynamically with the voltage. When the supply voltage decreases, most loads exhibit an instantaneous load relief. Yet, following such a disturbance, internal controllers of the various loads like thermostats of electrical heating and power electronics regulating the rotational speed of machines restore the power demand, which settles usually below the pre-disturbance level. Following [KH94] and references therein, this self-restoring behavior can be described by two first-order continuous-time dynamical systems – one for the active power and one for the reactive power. The model of the active power

$$\frac{\mathrm{d}x_{Lp}}{\mathrm{d}t} = -\frac{x_{Lp}}{T_p} + P_{L0}\big((V_{4m})^{a_s} - (V_{4m})^{a_t}\big) \tag{9.19a}$$

$$P_L = (1 - s_L \cdot \ell_{step})\big(\frac{x_{Lp}}{T_p} + P_{L0} \cdot (V_{4m})^{a_t}\big) \tag{9.19b}$$

has the internal real-valued state x_{Lp}, which models the load recovery dynamic with the time constant T_p. The instantaneous voltage dependency is expressed by $(V_{4m})^{a_t}$ and the steady-state voltage dependency is given by $(V_{4m})^{a_s}$. In this model, the absolute value of the load voltage V_{4m} can be considered as an input, the actual active power P_L as an output, and the rated power of the load P_{L0} as a parameter. Discrete parts of the load of size ℓ_{step} can be disconnected from the feeder according to the input variable $s_L \in \{0, 1, 2, 3\}$. The load model for the reactive power is defined accordingly.

$$\frac{\mathrm{d}x_{Lq}}{\mathrm{d}t} = -\frac{x_{Lq}}{T_q} + Q_{L0}\big((V_{4m})^{b_s} - (V_{4m})^{b_t}\big) \tag{9.20a}$$

$$Q_L = (1 - s_L \cdot \ell_{step})\big(\frac{x_{Lq}}{T_q} + Q_{L0} \cdot (V_{4m})^{b_t}\big) \tag{9.20b}$$

Note that the absolute value of the load voltage V_{4m} is given in (9.12).

The currents depend on the active and the reactive powers as well as on the direct and quadrature voltages.

$$I_{5d} = \frac{P_L V_{4d} + Q_L V_{4q}}{(V_{4m})^2} \tag{9.21a}$$

$$I_{5q} = \frac{P_L V_{4q} - Q_L V_{4d}}{(V_{4m})^2} \tag{9.21b}$$

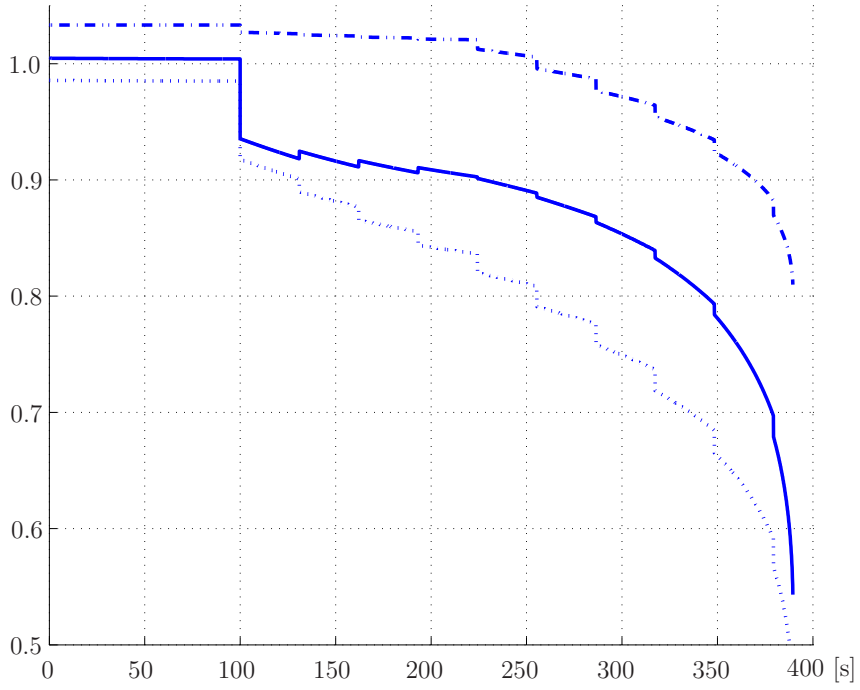


Figure 9.5: Open-loop response of the bus voltages following a fault in corridor L3 at time $t = 100\,\mathrm{s}$: V_{2m} is the dash-dotted, V_{3m} is the dashed and V_{4m} is the solid line

Network

The different system components are connected with each other by the network. Let Y_{L1}, Y_{L2} and Y_{L3} denote the admittances of the equivalent lines of the transmission corridors L1, L2 and L3, respectively. At each bus, according to Kirchhoff's law, the sum of the currents is required to sum up to zero yielding two linear algebraic equations per bus.

$$-I_{1d} = Y_{L1}(V_{1q} - V_{2q}) + Y_{L3}(V_{1q} - V_{3q}) \tag{9.22a}$$
$$-I_{1q} = Y_{L1}(-V_{1d} + V_{2d}) + Y_{L3}(-V_{1q} + V_{3d}) \tag{9.22b}$$
$$-I_{2d} = Y_{L1}(-V_{1q} + V_{2q}) + Y_{L2}(V_{2q} - V_{3q}) \tag{9.22c}$$
$$-I_{2q} = Y_{L1}(V_{1d} - V_{2d}) + Y_{L2}(-V_{2q} + V_{3d}) \tag{9.22d}$$
$$-(I_{cd} + I_{3d}) = Y_{L2}(V_{3q} - V_{2q}) + Y_{L3}(V_{3q} - V_{1q}) \tag{9.22e}$$
$$-(I_{cq} + I_{3q}) = Y_{L2}(-V_{3d} + V_{2d}) + Y_{L3}(-V_{3q} + V_{1d}) \tag{9.22f}$$
$$I_{4d} + I_{5d} = 0 \tag{9.22g}$$
$$I_{4q} + I_{5q} = 0 \tag{9.22h}$$

9.2.3 Fault Scenario

At time $t = 100\,\mathrm{s}$, a short circuit occurs in two of the three transmission lines of the heavily loaded transmission corridor L3. The two affected lines are immediately disconnected. As

a consequence, the admittance of the equivalent line (of the corridor) is reduced from 2 p.u. to $\frac{2}{3}$ p.u., which increases the reactive power losses on the corridor and also constricts the active power transmitted from Generator 1 to the load. Hence the load voltage drops, and due to the instantaneous voltage dependence of the load, the power drawn by the load drops, too. Yet, the AVR in Generator 2 quickly restores the terminal voltage by increasing the field voltage and the excitation. Moreover, the OLTC of the transformer tries to restore the load voltage by tapping up, and the power demand of the load recovers due to internal control loops. Both, the internal controller of the tap changer and the load dynamics drive Generator 2 to its loading limit. At time $t = 224\,\mathrm{s}$, the over-excitation limiter of the second generator is activated leading to a loss of voltage control and to a fast voltage collapse within less than 200 s. The corresponding bus voltages are depicted in Fig. 9.5.

9.2.4 Characteristics and Summary

Summing up, the power system features three input variables, namely the real-valued and constrained reference voltage $V_{4m,ref} \in [0.8, 1.2]$ of the tap changer and the two integer variables $s_C \in \{0, 1, 2, 3\}$ and $s_L \in \{0, 1, 2, 3\}$ denoting the number of capacitors connected to the system and the amount of load shed, respectively. The outputs of main interest are the three real-valued bus voltages $V_{im} \in \mathbb{R}$, $i \in \{2, 3, 4\}$. Besides that, the power system has the following real and discrete-valued states. The two continuous states x_{Lp}, $x_{Lq} \in \mathbb{R}$ model the dynamics of the active and reactive load power. The discrete-valued variable $n_T \in \{0.8, 0.82, 0.84, \ldots, 1.2\}$ denotes the tap position of the transformer. The OLTC controller has another real-valued state, namely the timer $t_T \in \mathbb{R}_{\geq}$. As mentioned before, the timer becomes obsolete if the sampling interval equals the time delay of the tap changer (here roughly 30 s).

Concluding the above, the power system under consideration is a hybrid system containing integer manipulated variables, a saturation element (in the AVR), two switched dynamics and six logic implications (in the tap changer controller), two ordinary differential equations (in the load model) and a total of 31 algebraic equations[1], of which 18 are nonlinear.

9.2.5 Parameters

All parameters of the example power system are given in Table 9.1 using the notation introduced above. For completeness, the initial values of the load states are given by $x_{Lp}(0) = x_{Lq}(0) = 0$, the initial position of the tap changer is $n_T(0) = 1.02$, and the

[1] They can be easily reduced to 27 by setting $I_5 = -I_4$ and by putting the two equations of Generator 1 into the network equations.

Component	Parameter	Value	Description
Generator 1	$V_{1m,ref}$	1.03	Reference voltage [p.u.]
Generator 2	$V_{2m,ref}$	1.03	Reference voltage [p.u.]
Generator 2	E_{f0}	1.55	Default field voltage [p.u.]
Generator 2	$E_{f,max}$	2.2	Maximal field voltage [p.u.]
Generator 2	x_d	2.38	Direct axis synchronous reactance [p.u.]
Generator 2	x_q	2.27	Quadrature axis synchronous reactance [p.u.]
Generator 2	P_{m0}	0.352	Mechanical power [p.u.]
Transformer	R_T	0	Leakage resistance [p.u.]
Transformer	X_T	0.0031	Leakage reactance [p.u.]
Transformer	d_T	0.03	Deadband around $V_{4m,ref}$ [p.u.]
Transformer	$\tau_{tapDelay}$	30	Time-delay between tapping [s]
Transformer	$\tau_{mechDelay}$	1	Mechanical time-delay during tapping [s]
Transformer	n_{min}	0.8	Minimal tap position [p.u.]
Transformer	n_{max}	1.2	Maximal tap position [p.u.]
Transformer	n_{step}	0.02	Tapping step size [p.u.]
Capacitor bank	c_{step}	0.1	Capacitor bank switching step size [p.u.]
Load	P_{L0}	1	Rated active power load [p.u.]
Load	Q_{L0}	0.2	Rated reactive power load [p.u.]
Load	a_s	0	Steady-state active power voltage dep.
Load	a_t	2	Transient active power voltage dependency
Load	b_s	1	Steady-state reactive power voltage dep.
Load	b_t	2	Transient reactive power voltage dependency
Load	T_p	60	Active power recovery time constant [s]
Load	T_q	60	Reactive power recovery time constant [s]
Load	ℓ_{step}	0.05	Load shedding step size [p.u.]
Network	Y_{L1}	$\frac{4}{3}$	Admittance of Corridor 1 [p.u.]
Network	Y_{L2}	$\frac{1}{0.7}$	Admittance of Corridor 2 [p.u.]
Network	$Y_{L3,noFault}$	2	Admittance of nominal Corridor 3 [p.u.]
Network	$Y_{L3,withFault}$	$\frac{2}{3}$	Admittance of faulted Corridor 3 [p.u.]

Table 9.1: Parameters of the example power system

timer is set to $t_T(0) = 0$. Furthermore, the initial inputs to the system are $V_{4m,ref}(0) = 1$, $s_C(0) = 2$ and $s_L(0) = 0$.

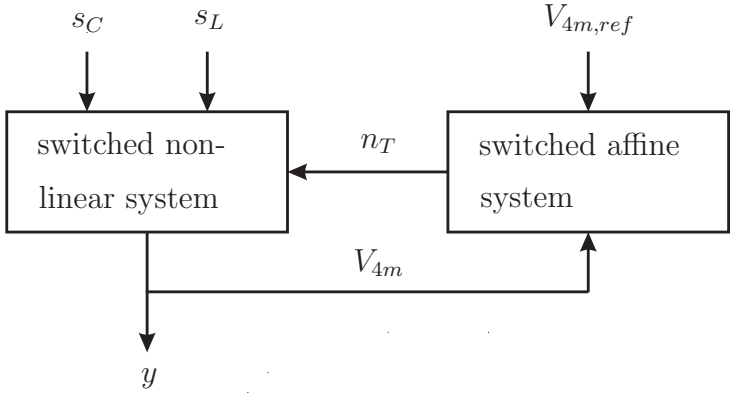


Figure 9.6: Model decomposition

9.3 Hybrid Modelling

9.3.1 Decomposition

Before modelling the power system in the MLD framework, which is summarized in Section 2.2.2, we briefly show that the power system can be decomposed into two parts, specifically into a switched nonlinear and a switched affine dynamical system.

(i) The switched nonlinear system. This part is a differential algebraic equation (DAE) system formed by the two ordinary differential equations (ODE) (9.19a) and (9.20a) of the load and the algebraic equations (equality constraints) of the generators (9.2)–(9.5) and (9.8), the capacitor bank (9.10), the transformer (9.11)–(9.12), the load (9.19b), (9.20b) and (9.21), and the network (9.22). The ODE as well as most of the algebraic equations are nonlinear. Additionally, the saturation of the internal AVR (9.9) of Generator 2 is included as part of the switched nonlinear dynamics. The discrete-valued n_T and the integers s_C and s_L form the inputs to this subsystem, the three bus voltages V_{im}, $i \in \{2, 3, 4\}$ constitute the outputs, and x_{Lp}, x_{Lq} are the real-valued states. Note that this subsystem is a switched nonlinear system since each combination of discrete-valued inputs is associated with a (different) nonlinear dynamic.

(ii) The switched affine system. It combines the thresholds, the logic implications and the switched affine dynamics modelling the internal OLTC controller (9.13)–(9.18) of the transformer. $V_{4m,ref}$ and the bus voltage V_{4m} are the continuous-valued inputs, the tap position n_T is both the discrete-valued output and a state, and the timer t_T is a real-valued state.

The model decomposition is shown in Fig. 9.6, where we have used $y = [V_{2m}\ V_{3m}\ V_{4m}]^T$.

9.3.2 MLD Model

Before deriving the MLD model, we observe that in the load model the transient voltage dependencies of the active and reactive powers are linearly dependent. Moreover, the steady-state voltage dependency is almost linearly dependent – at least in a neighborhood around the nominal operating point. This is confirmed through simulations indicating that the ratio between the load states is almost constant, i.e. $\frac{x_{Lp}}{x_{Lq}} \in [9.995, 10.4]$. The function $10 + 5(\frac{1}{V_{4m}} - 1)$ approximates $\frac{x_{Lp}}{x_{Lq}}$ accurately for $x_{Lp} \in [-1, 10]$ observed during experiments. Hence, we reduce the number of model states by one by replacing the state x_{Lp} by the algebraic relation

$$x_{Lp}(V_{4m}) = \big(10 + 5(\frac{1}{V_{4m}} - 1)\big)x_{Lq}\,. \tag{9.23}$$

Subsequently, we outline two different approaches to cast the nonlinear model of the power system into MLD form.

The first approach is to consider each nonlinear function (dynamic or algebraic constraint) individually. Since the MLD framework is defined in the discrete time domain, all continuous-time dynamics need to be first represented by discrete-time dynamics. Next, each nonlinear function is approximated by a PWA function. This can been done using an algorithm based on [FMLM03] that exploits the combined use of clustering, linear identification and classification techniques allowing one to identify at the same time both the affine functions and the polyhedral partition of the domain. However, as most of the involved expressions are strongly nonlinear and defined over two- or three-dimensional domains and since the interactions between the nonlinear algebraic constraints are tight, this approach leads to a disproportionally large number of polyhedra and unacceptably large approximation errors [GLM02]. Hence, this approach is not pursued here.

The second approach exploits the structure of the example power system, which was shown in the previous section. The switched affine subsystem can be directly translated into MLD form by introducing binary and auxiliary continuous variables [BM99b]. Additional approximations are not needed – in particular, the switched affine system is already given in discrete time. On the other hand, the switched nonlinear subsystem features after the model reduction the continuous-valued state x_{Lq}, the discrete-valued input n_T and the integer inputs s_C and s_L. Rather than approximating each nonlinearity individually, we suggest to approximate the system behavior as a whole. In particular, we propose to partition the two-dimensional state-input space spanned by x_{Lq} and n_T into polyhedra, and to sample the system response for each combination of integer inputs. This yields the discrete-time state-update functions of the load as well as the output functions of the bus voltages. Note that these are implicitly defined by the DAE and not explicitly given. Due to the fact that the switched nonlinear system (as a whole) is only mildly nonlinear (in

contrast to the individual nonlinearities), it is sufficient to partition the two-dimensional state-input space into 24 triangular polyhedra or simplices. The toolbox [Jul00] is then used to compute the PWA state-update and output functions for a given integer input combination. Note that the partition does not depend on the integer inputs. For each binary input combination, however, the polyhedra refer in general to different PWA state-update and output functions. Additionally, to reduce the complexity of the MLD model, we reduce the number of available control inputs. Since $s_C = 0$ or $s_C = 1$ would destabilize the system and $s_L \geq 2$ is never necessary to stabilize the system, we only consider $s_C = \{2, 3\}$ and $s_L = \{0, 1\}$. This second modelling approach is clearly preferable since it reduces not only the number of polyhedra of the PWA approximation but it also leads to significantly smaller approximation errors compared to the first approach.

The second modelling procedure yields an MLD system with three states (n_T, x_{Lq} and t_T), 302 z-variables, 31 δ-variables and 1660 inequality constraints. The derivation of the MLD system is performed by the compiler HYSDEL (HYbrid System DEscription Language) generating the matrices of the MLD system starting from a high-level description of the system [TB04].

9.4 Constrained Optimal Control

After stating the control objectives, we will cast them into a cost function, and formulate a constrained finite time optimal control (CFTOC) problem, which is solved on-line. Moreover, to avoid accidental tap changes due to small model mismatches, we will introduce a cascaded controller scheme.

9.4.1 Control Objectives

The control objectives are to bring the load voltage V_{4m} as close to its reference value one as possible while fulfilling the soft constraints on the bus voltages $V_{2m} \in [0.95, 1.05]$ p.u., $V_{3m} \in [0.9, 1.1]$ p.u. and $V_{4m} \in [0.9, 1.1]$ p.u., and while switching the manipulated variables as little as possible.

The control moves can be classified as nominal and emergency control. During nominal control, the soft constraints on the bus voltages can be fulfilled by using only "cheap" control moves, i.e. capacitor bank switching s_C and changing the tap changer voltage reference $V_{4m,ref}$. If the soft constraints cannot be met by only applying cheap control moves, the controller has to switch to emergency control and use the full range of available control moves including load-shedding, which is considered to be very expensive.

9.4.2 Cascaded Controller Scheme

The reference voltage $V_{4m,ref}$ is used to communicate to the internal controller of the tap changer whether the tap position shall be kept, increased or decreased by one step, in accordance to whether V_{4m} lies in the tolerance band centered around $V_{4m,ref}$, below or above it. Obviously, small variations in $V_{4m,ref}$ can trigger major changes (tapping actions) making $V_{4m,ref}$ prone to modelling errors. To circumvent this, we introduce the tapping strategy $\Delta n_T \in \{0, n_{step}, -n_{step}\}$, and we consider for the control problem formulation that Δn_T is manipulated directly rather than $V_{4m,ref}$. For the implementation, however, we convert back from Δn_T, which is calculated by the controller, to $V_{4m,ref}$, which is actually applied to the power system.

$$V_{4m,ref}(k) = \begin{cases} V_{4m,ref}(k-1) & \text{if } \Delta n_T(k) = 0 \\ 0.8\,\text{p.u.} & \text{if } \Delta n_T(k) = -n_{step} \\ 1.2\,\text{p.u.} & \text{if } \Delta n_T(k) = n_{step} \end{cases} \tag{9.24}$$

Note that 0.8 p.u. and 1.2 p.u. constitute the minimal and maximal admissible values of $V_{4m,ref}$ and $n_{step} = 0.02$ p.u. is the physical step size of the tap changer.

9.4.3 Objective Function

In accordance with the aforementioned cascaded controller structure, we define the vector of manipulated variables as

$$u(k) = \begin{bmatrix} \Delta n_T(k) & s_C(k) & s_L(k) \end{bmatrix}^T , \tag{9.25}$$

which is constrained to $u(k) \in \mathcal{U}$ with $\mathcal{U} = \{-n_{step}, 0, n_{step}\} \times \{2, 3\} \times \{0, 1\}$.

The control objectives of Section 9.4.1 are cast into the objective function

$$\begin{aligned} J(x(k), u(k-1), U(k)) \;=\; & \sum_{\ell=0}^{N-1} \Big(\|V_{4m}(k+\ell|k) - 1\| + \|Q\,\Delta u(k+\ell)\|_\infty \Big) \\ & + \sum_{\ell=1}^{N-1} \Big(\underline{S}(k+\ell|k) + \overline{S}(k+\ell|k) \Big) , \end{aligned} \tag{9.26}$$

which penalizes the evolution of three terms over the prediction horizon N given the sequence of manipulated variables $U(k) = [(u(k))^T, \ldots, (u(k+N-1)^T]^T$. Hereafter, we will elaborate on these three terms in the objective function. The first term penalizes the deviation of the load voltage V_{4m} from its reference one using the 1-norm. The second term penalizes the switching of the manipulated variables using the diagonal weight matrix Q and the ∞-norm. Here, we are using a Δu formulation with

$$\Delta u(k) = u(k) - u(k-1) \tag{9.27}$$

to penalize the switching rather than the absolute value. This approach adds the integer state vector $u(k-1)$ to the CFTOC formulation. Additionally, due to the cascaded controller scheme, the tap changer strategy Δn_T is weighted rather than the reference voltage $V_{4m,ref}$. From a physical point of view this is reasonable, too, as the mechanical wear of the transformer results from tapping and not from changes in the reference voltage. Third, the soft constraints on the bus voltages are penalized. To take the soft constraints on the lower bounds into account, we introduce the binary variables $\underline{b}_i$, $i \in \{2,3,4\}$,

$$[\underline{b}_i(k) = 1] \longleftrightarrow [V_{im}(k) < \underline{V}_{im}], \tag{9.28}$$

which indicate the violation of the i-th lower soft constraint. Here, $\underline{V}_{2m} = 0.95\,\text{p.u.}$, $\underline{V}_{3m} = \underline{V}_{4m} = 0.9\,\text{p.u.}$ are the lower bounds on the respective bus voltages as defined in Section 9.4.1. Additionally, the slack variables $\underline{s}_i \geq 0$, $i \in \{2,3,4\}$,

$$\underline{s}_i(k) = \begin{cases} 0 & \text{if } \underline{b}_i(k) = 0 \\ \underline{V}_{im} - V_{im}(k) & \text{if } \underline{b}_i(k) = 1 \end{cases} \tag{9.29}$$

denote the degree of the violation. Penalizing the violation as well as the degree of the violation yields

$$\underline{S}(k+\ell|k) = p \cdot \sum_{i=2}^{4} \Big(\underline{b}_i(k+\ell|k) + \underline{s}_i(k+\ell|k) \Big), \tag{9.30}$$

where p is a large penalty. The soft constraints on the upper bound $\overline{S}(k+\ell|k)$ are defined accordingly.

Choosing the weights in the objective function is rather straightforward, and time consuming fine tuning is avoided as shown hereafter. After setting the weight on the voltage deviation $\|V_{4m}(k+\ell|k) - 1\|$ to one, the weight $Q = \text{diag}(q_1, q_2, q_3)$ on the manipulated variables is chosen such that control moves are performed when the penalty on the voltage deviation exceeds a certain limit. Therefore, Q is derived by comparing the cost on the voltage deviation $\|V_{4m}(k+\ell|k) - 1\|$ when refraining from using a control move with the cost of performing a control action [LHO02]. Assume, that the tap changer may be moved by one step up or down, if this results in a reduction of the voltage deviation by at least $0.004\,\text{p.u.}$. As one step is given by $n_{step} = 0.02\,\text{p.u.}$, we get $q_1 = \frac{0.004}{0.02} = 0.2$. Analogously, assuming that one part of the capacitor bank may be switched if this reduces the voltage deviation by $0.03\,\text{p.u.}$ or more leads to $q_2 = \frac{0.03}{1} = 0.03$. As the lower and upper voltage constraints on V_{4m} are equal to $0.9\,\text{p.u.}$ and $1.1\,\text{p.u.}$, respectively, the maximal cost resulting from the voltage deviation is 0.1. Load-shedding has to be avoided unless a soft constraint on the bus voltages is violated. This implies $q_3 \gg \frac{0.1}{1} = 0.1$ and therefore $q_3 = 1$ is chosen. Concerning the soft constraints, the weight p is chosen such that the full range of control moves can be used to remove the violation. This reasoning leads to

$$p \gg \max_{u(k),u(k-1)\in\mathcal{U}} \|Q \cdot \Delta u(k)\|_\infty = 3 \tag{9.31}$$

and to the choice $p = 10$.

This choice of the penalties will cause emergency actions to be triggered as soon as one of the soft constraints is either violated or predicted to be violated when refraining from emergency control moves. Hence, violations of the soft constraints are avoided unless no appropriate (emergency) control move is available. This is achieved by three penalty levels, where the penalties on the load voltage deviation, tapping and capacitor switching are the lowest. The penalty on load-shedding is of medium size, and the soft constraints have large penalties. Moreover, to avoid an unnecessary activation of emergency control actions in case of small measurement noise and/or model mismatch, the soft constraints are not imposed on the current time-instant k but only on the next time-instant $k+1$ as can be seen from (9.26). This enables the controller to remove unpredicted violations of the soft constraints within one sampling interval.

9.4.4 On-Line Computation of Control Input

The control input at time-instant k is obtained by minimizing the objective function (9.26) over the sequence of control inputs $U(k)$ subject to the mixed-integer linear inequality constraints of the MLD model (2.8) described in Section 9.3.2, the constraints on the sequence of manipulated variables

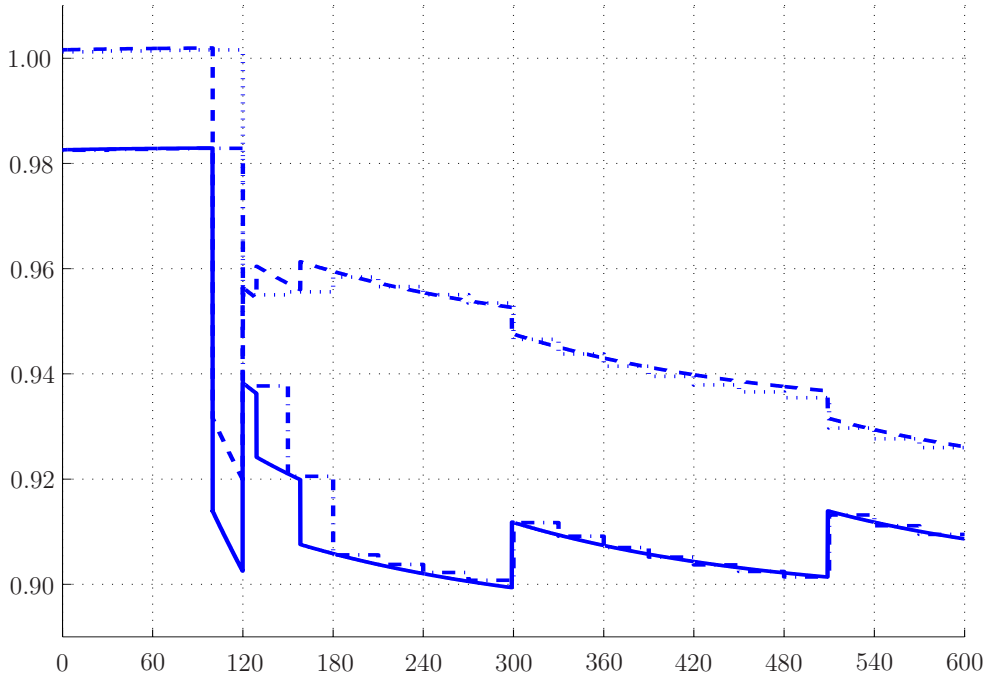
$$u(\ell) \in \mathcal{U}\,, \quad \ell = k, ..., k+N-1\,, \tag{9.32}$$

the expression (9.27), and the terms (9.28)–(9.30) modelling the soft constraints. This amounts to the CFTOC

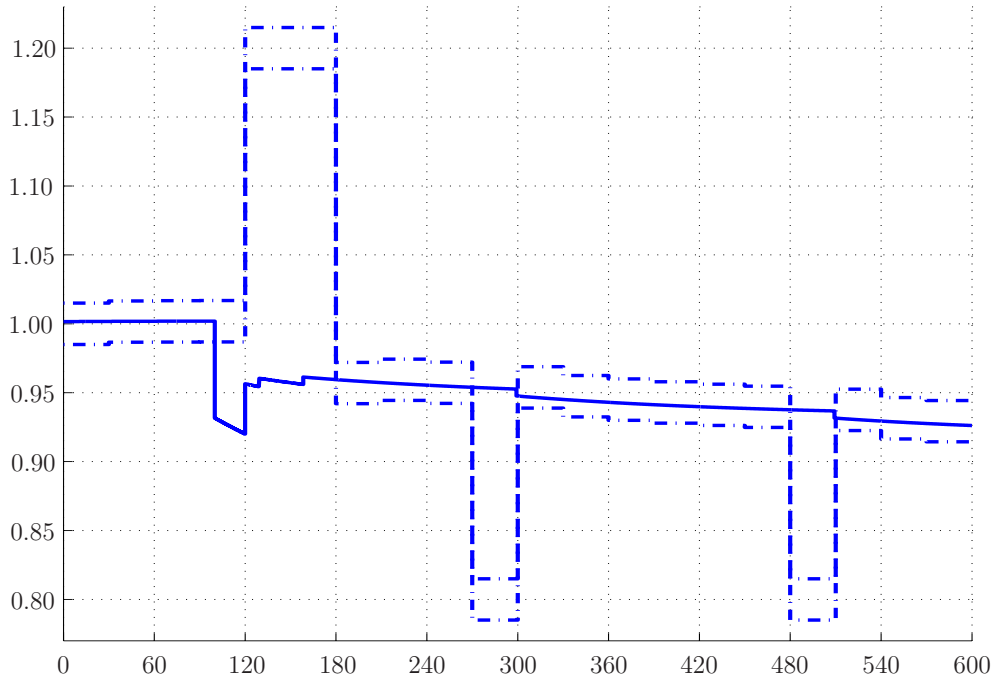
$$U^*(k) = \arg\min_{U(k)} J(x(k), u(k-1), U(k)) \tag{9.33a}$$

$$\text{subj. to MLD model (2.8), (9.27)–(9.30), (9.32)} \tag{9.33b}$$

leading to the sequence of optimal control inputs $U^*(k)$. Of this sequence, only the first element $u^*(k)$ is considered, which is, after being translated according to (9.24) into an input vector of the form $[V_{4m}\ s_C\ s_L]^T$, applied to the power system. At the next sampling interval, k is set to $k+1$, a new state measurement (or estimate) is obtained, and the CFTOC problem is solved again over the shifted horizon according to the receding horizon policy. As we are using linear norms in all cost expressions, the CFTOC problem amounts to solving a *Mixed-Integer Linear Program* (MILP), for which efficient solvers exist like [ILO02].

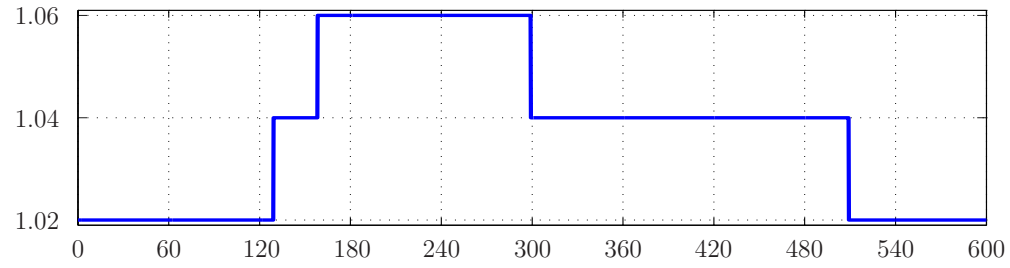


(a) The voltage V_{3m} at Bus 3 of the nonlinear (MLD) model is the solid (dash-dotted) line; the voltage V_{4m} at Bus 4 of the nonlinear (MLD) model is the dashed (dotted) line

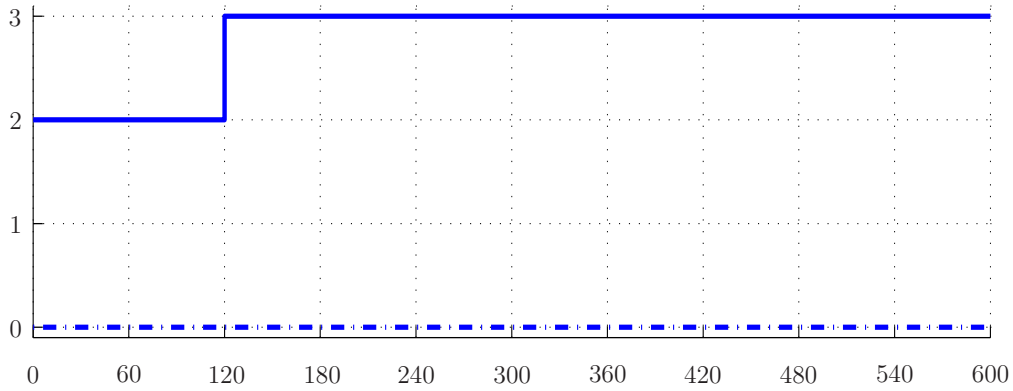


(b) The upper and lower bounds of the tolerance band (centered around $V_{4m,ref}$) are the dash-dotted lines; the voltage V_{4m} at Bus 4 of the nonlinear model is represented by the solid line

Figure 9.7: Closed-loop evolution of the bus voltages [p.u.] over time [s] for $T_s = 30\,\mathrm{s}$



(a) Tap changer position n_T



(b) Integer manipulated variables: Capacitor switching s_C is a solid, load-shedding s_L is a dash-dotted line

Figure 9.8: Closed-loop evolution of the tap changer position and the integer manipulated variables over time [s] for $T_s = 30\,\mathrm{s}$

9.5 Simulation Results

This section presents control experiments using the cascaded controller scheme. MPC employs the derived MLD model as prediction model, whereas the "real" power system is represented by the more accurate nonlinear model described in Section 9.2. Since the load state(s) and the tap changer position can be easily measured or estimated, we assume that they are available for the MPC scheme. Furthermore, setting the sampling interval to $T_s = 30\,\mathrm{s}$ allows us to use the simplified OLTC controller. The length of the prediction horizon N is set to three, making sure that the prediction time interval amounts to 90 s and thus exceeds the dominant time constants $T_p = T_q = 60\,\mathrm{s}$ of the load. Then, given the initial states and inputs of Section 9.3.1 and applying the fault in transmission corridor L3 at time $t = 100\,\mathrm{s}$, we obtain the hereafter discussed closed-loop results.

Fig. 9.7(a) shows the bus voltages V_{3m} and V_{4m} of the original nonlinear model as well as the ones of the MLD model. V_{2m} is not depicted, as it remains within $[1.02, 1.03]$ during the control experiment and is thus only of minor interest. The manipulated variable $V_{4m,ref}$ together with its tolerance band is shown in Fig. 9.7(b), whereas Fig. 9.8(a) displays the

trajectory of the tap position n_T. The integer manipulated variables, namely capacitor switching s_C and load-shedding s_L, are visualized in Fig. 9.8(b).

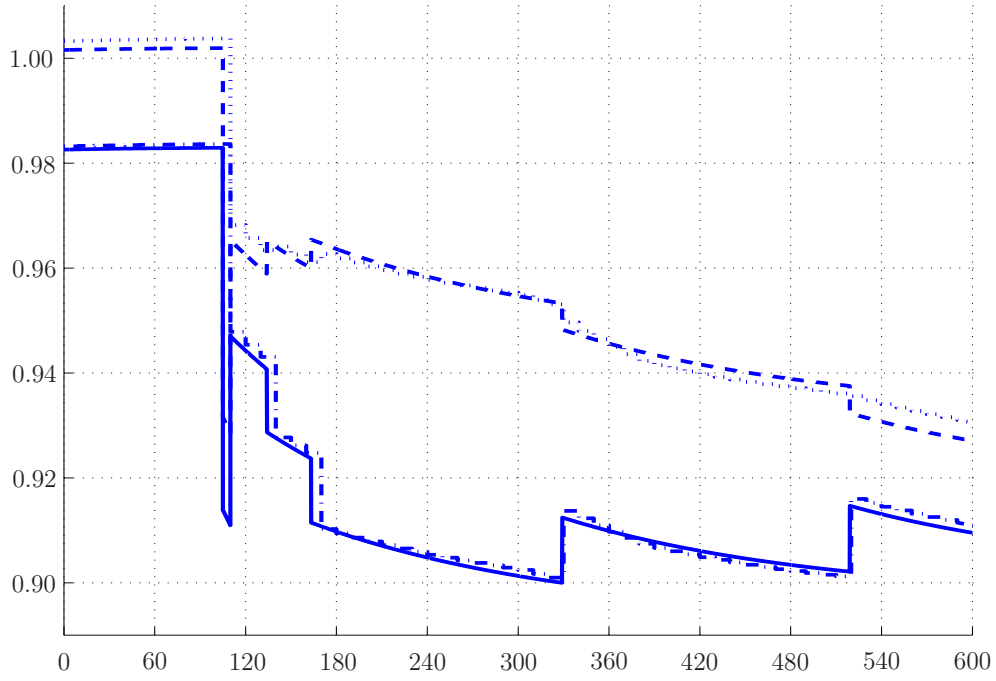
The fault at time $t = 100\,\mathrm{s}$ reduces the bus voltages instantaneously leading to a voltage collapse if no appropriate control actions are taken as shown Fig. 9.5. At the first sampling instant after the fault at time $t = 120\,\mathrm{s}$, MPC predicts the potential collapse and connects the remaining part of the capacitor bank to the power system by setting s_C to its maximum value three. This control move is both necessary and sufficient to stabilize the system and therefore, thanks to the proper timing, only nominal control moves are needed to prevent a voltage collapse. In particular, load-shedding is avoided and s_L is kept at zero. Note that the proper timing of the control actions is important, since connecting the capacitor bank a few sampling instants later would lead to a severe violation of the lower soft constraint on V_{3m} that could not be removed by nominal control moves only (within one sampling interval). As a result, emergency control would be applied and part of the load would be shed in order to meet the soft constraints.

Apart from the capacitor switching, $V_{4m,ref}$ is set to its maximum at time $t = 120\,\mathrm{s}$ to step up the tap changer twice thus reducing the deviation of V_{4m} from its reference. At $t = 270\,\mathrm{s}$ and $t = 480\,\mathrm{s}$, however, MPC issues tap down commands to avoid a violation of the lower soft constraint on V_{3m}. According to the objective function and the horizon, this is the optimal sequence of control moves. If desired, a different tuning of the objective function and a longer prediction horizon can avoid the tap up and down actions and keep the tap changer at its initial position.

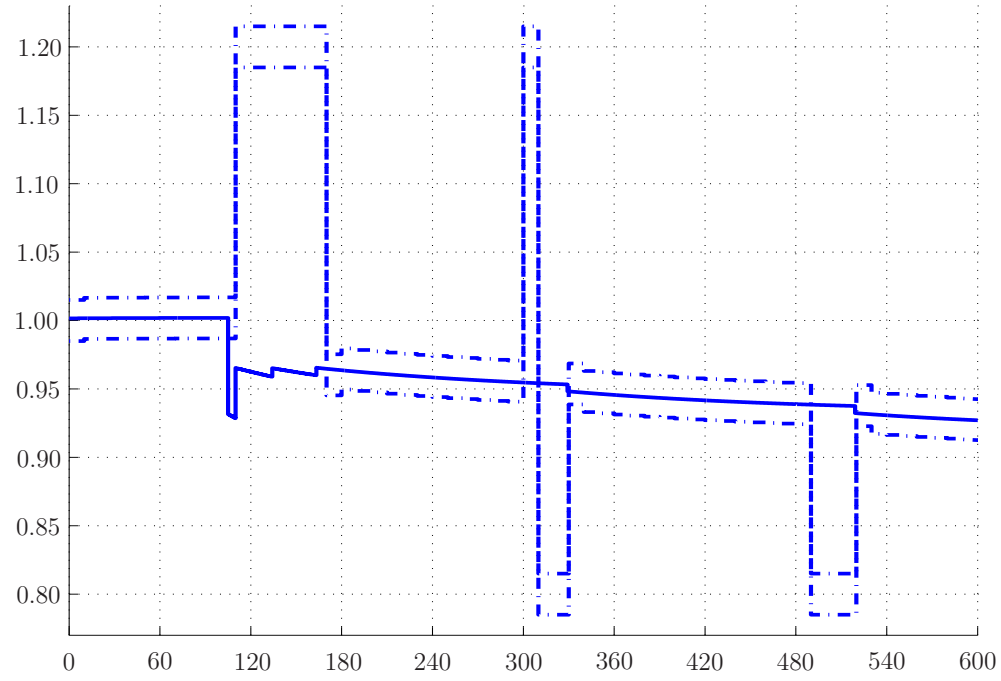
In Fig. 9.7(a), the modelling error, which results from approximating the switched nonlinear system by a PWA representation, can be seen as a small mismatch between the respective bus voltages of the nonlinear and the MLD model. Increasing the number of partitions of the PWA approximation would reduce the errors arbitrarily. The major error, however, is introduced by discretizing the time with $T_s = 30\,\mathrm{s}$. Using a sampling interval of 10 s reduces this error significantly.

Hence, in a second simulation, we set $T_s = 10\,\mathrm{s}$ and $N = 7$. The complete OLTC model (9.13)–(9.18) is included in the MLD model and the timer t_T is added as additional state. As can be seen from the simulations in Figs. 9.9 and 9.10, the obtained closed-loop results are similar to the previous case with $T_s = 30\,\mathrm{s}$ with the main difference that the controller issues the first stabilizing control moves already at time $t = 110\,\mathrm{s}$. However, since only eight polyhedra were used to approximate the switched nonlinear system, the approximation error is pronounced.

The computation times for solving the optimal control problem at each time-step when running CPLEX 8.0 on a Pentium IV 2.8 GHz machine are as follows. For $T_s = 30\,\mathrm{s}$ and a horizon $N = 3$, the computation time is on average 1.8 s and always less than 3.5 s. For $T_s = 10\,\mathrm{s}$ and $N = 7$, the respective times are 11 s and 160 s.

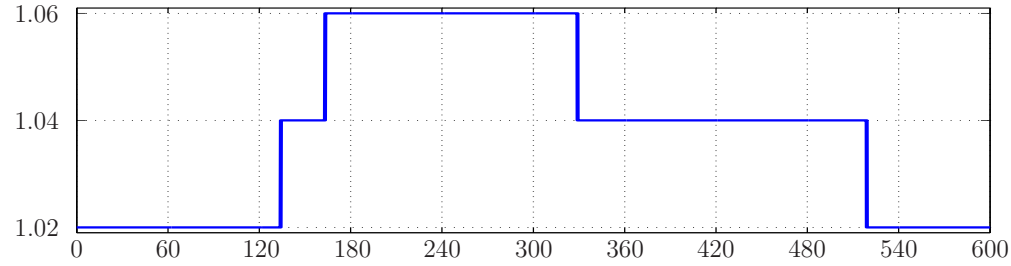


(a) The voltage V_{3m} at Bus 3 of the nonlinear (MLD) model is the solid (dash-dotted) line; the voltage V_{4m} at Bus 4 of the nonlinear (MLD) model is the dashed (dotted) line

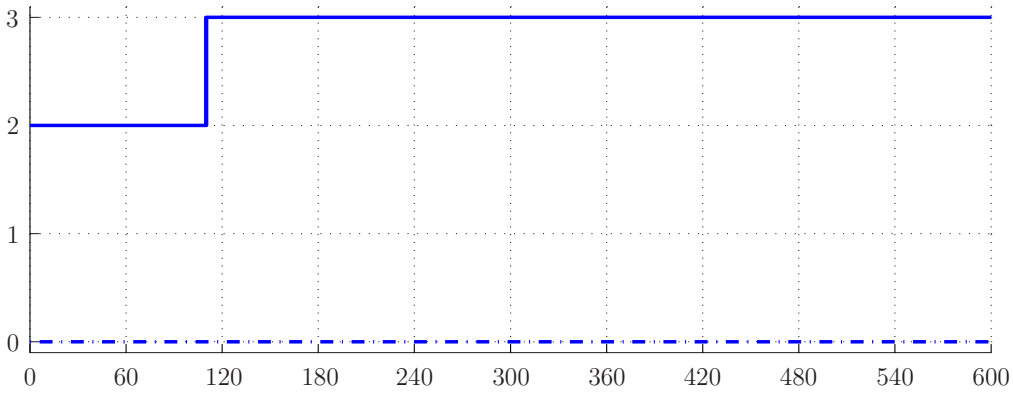


(b) The upper and lower bounds of the tolerance band (centered around $V_{4m,ref}$) are the dash-dotted lines; the voltage V_{4m} at Bus 4 of the nonlinear model is represented by the solid line

Figure 9.9: Closed-loop evolution of the bus voltages [p.u.] over time [s] for $T_s = 10\,\mathrm{s}$



(a) Tap changer position n_T



(b) Integer manipulated variables: Capacitor switching s_C is a solid, load-shedding s_L is a dash-dotted line

Figure 9.10: Closed-loop evolution of the tap changer position and the integer manipulated variables over time [s] for $T_s = 10\,\mathrm{s}$

Temporal Decomposition of CFTOC

Given a CFTOC problem for a hybrid system in the MLD framework, we have proposed in [BGM04] a temporal decomposition scheme that efficiently derives the control actions by performing Lagrangian decomposition on the prediction horizon. More specifically, the algorithm translates the original optimal control problem into a temporal sequence of independent subproblems of smaller dimension. The solution to the Lagrangian problem yields a sequence of control actions for the full horizon that is approximate in nature due to the non-convexity of the hybrid optimal control problem formulation and the consequent duality gap. For cases, however, where the duality gap is sufficiently narrow, the approximate control law yields basically the same closed-loop behavior as the one obtained from the original optimal controller, but with a considerably smaller computational burden.

In [BGM04], we have considered as an illustrative example the CFTOC (9.33) for the above presented power system with $N = 6$ and $T_s = 30\,\mathrm{s}$. When running the above control experiment (with the only difference being the prediction horizon of six), a reduction of the computation time by an order of magnitude is achieved. In particular, the maximal computation time is reduced from 486 s to 25.2 s, while the average computation time

is cut down from 276 s to 18.6 s. The duality gap is always less than 12 %. Because the duality gap is small, combined with the fact that the control inputs are purely integer preventing small deviations in the input variables that would otherwise occur, the Lagrangian decomposition scheme actually attains the optimal solution to the CFTOC (9.33). For the case study considered, the control problem may thus be tackled with a substantially lower computational effort, so that the effective potential for future applications appears to be encouraging. For details about the decomposition scheme, the reader is referred to [BGM04].

9.6 Conclusions and Future Research

Conclusions

To account for the hybrid character, the example power system was modelled in the MLD framework. Replacing the nonlinearities by PWA approximations introduces only small modelling errors hardly manifesting themselves in the control experiments thus proving the usefulness of the MLD modelling approach. We have shown that the load voltage can be stabilized by an MPC controller using only nominal control moves. The tuning of the controller is straightforward and systematic allowing us to easily distinguish between nominal and emergency control moves.

Future Research

Two more topics might be investigated for the case study under consideration: The derivation of the (explicit) state-feedback control law in accordance to Section 2.4.2, and the verification of closed-loop stability by performing an a posteriori analysis of the state-feedback control law. Yet, these results would only be meaningful for the specific fault. For a general emergency control scheme, they are only of limited value.

To extend the applicability of the scheme to power system of larger scale, one viable way is to consider decomposition schemes, in particular Lagrangian decomposition in time or in space. Decomposition in time, as proposed in [BGM04] for the above treated case study, might be also applied to larger systems. Decomposition in space is motivated by the fact that even though power systems are (highly) meshed grids, they can be often subdivided into several areas interconnected by a few tie lines. As an example consider the Nordic power system with the production located mainly in the north and several consumer areas in the south. A spatial decomposition scheme might exploit this topology by splitting the large optimization problem into smaller (and easier) to solve subproblems that are linked by dual variables. Unfortunately, since four variables[2] are associated with

[2]For example real and imaginary parts of voltage and current.

each tie line, one would need to introduce four dual variables for each tie line that is cut.

In general, we have experienced two major problems when casting a nonlinear power system in the MLD framework by approximating the nonlinearities in a PWA fashion. Firstly, from a control point of view the model complexity is unnecessarily increased by introducing a large number of binary variables associated with the approximations. As a result, the computation time for solving the underlying MILP becomes disproportionately large. Secondly, the approximations deteriorate the model accuracy. In particular for power systems, this is a drawback since apart from the load most system parameters are accurately known. Therefore, we suggest to avoid PWA approximations, to consider the original nonlinear hybrid model, and to directly tackle the resulting mixed-integer nonlinear optimization problem.

Probably the most promising approach is to combine the advantages of the MLD framework with the concept of trajectory sensitivities by considering optimal control schemes with prediction models that use mixed-integer linear inequality constraints to capture thresholds, logic and automata, and adopt first-order approximations around the nominal open-loop trajectories. Such an approach would allow one to capture the hybrid features of power systems, but the resulting optimization problem would comprise only integers that are necessary to model load-shedding, capacitor switching and OLTC dynamics.

Part VI

Appendix

A

HYSDEL Code

In this appendix, the HYSDEL codes used in this thesis are provided. Specifically, the paperboy example from Section 3.5.2, the model of the DTC drive from Section 6.5 both with a two- and a three-level inverter, the model of the synchronous step-down DC-DC converter from Section 8.2.3, and the example power system from Section 9.3.2 are given hereafter.

A.1 Paperboy Example

```
SYSTEM paperboy {

INTERFACE {
    PARAMETER {
        /* DHA 1: */
        REAL width = 4;                              /* width of the road in m */
        /* friction per region */
        REAL mu1x = 2, mu2x = 1.5, mu3x = -0.5, mu4x = -1,  mu5x = 0;
        REAL mu1y = 0, mu2y = 0,   mu3y = 0,    mu4y = 0,   mu5y = 0;
        REAL nu1x = 2, nu2x = 0.5, nu3x = 1,    nu4x = 1.5, nu5x = 0.05;
        REAL nu1y = 2, nu2y = 0.5, nu3y = 1,    nu4y = 1.5, nu5y = 0.05;

        /* DHA 2: */
        REAL size = 10;                              /* size of a house in m */
        REAL xh1 = -40, yh1 = -40;                   /* position of the House 1 in m */
        REAL xh2 =  40, yh2 =  40;                   /* position of the House 2 in m */

        /* DHA 3: */
        REAL Mb = 90;                                /* mass of the paperboy and his bike in kg */
        REAL Mm = 10;                                /* mass of one mail item in kg */
        REAL Ts = 1;                                 /* sampling time in s */

        /* constraints: */
        REAL Fmax = 162;                             /* maximal force in N */
        REAL space = 1000;                           /* size of x-y space in m */
        REAL vmax = 15;                              /* maximal speed in m/s */
    }
    INPUT {
        REAL Fbx [-Fmax, Fmax], Fby [-Fmax, Fmax];  /* force applied by paperboy in N */
    }
```

```
    STATE {
        REAL x [-space, space], y [-space, space];  /* position of paperboy in m*/
        REAL vx [-vmax, vmax], vy [-vmax, vmax];    /* speed of paperboy in m/s */
        BOOL d1, d2;                                /* status of mail delivery */
    }
    OUTPUT {
        REAL xo, yo;                                /* position of paperboy in m */
        REAL n;                                     /* number of delivered mail items */
    }
}

IMPLEMENTATION {
    AUX {
        BOOL dx1, dx2, dy,                          /* variables to characterize the topology */
             h1x1, h1x2, h1y1, h1y2,                /* one variable per 'border' of House 1 */
             h2x1, h2x2, h2y1, h2y2,                /* one variable per 'border' of House 2 */
             dh1, dh2,                              /* true if paperboy inside of respective house */
             i1, i2, i3, i4, i5,                    /* true if paperboy inside of the respective region */
             no_del;                                /* true if both d1 and d2 false*/
        REAL F_i1x, F_i2x, F_i3x, F_i4x,            /* auxiliary forces in x */
             F_i1y, F_i2y, F_i3y, F_i4y;            /* auxiliary forces in y */
        REAL Fx, Fy;                                /* effective force */
        REAL F_d1x, F_d2x, F_d3x,                   /* auxiliary forces in x */
             F_d1y, F_d2y, F_d3y;                   /* auxiliary forces in y */
    }

    /* DHA 1: */
    AD {
        dx1 = x <= -width/2;
        dx2 = x >=  width/2;
        dy  = y >= 0;
    }
    LOGIC {
        i1 =  dx1 &         dy;
        i2 =         dx2 &  dy;
        i3 =  dx1 &        ~dy;
        i4 =         dx2 & ~dy;
        i5 = ~dx1 & ~dx2;
    }
    DA {
        F_i1x = {IF i1 THEN Fbx - mu1x - nu1x * vx};
        F_i1y = {IF i1 THEN Fby - mu1y - nu1y * vy};
        F_i2x = {IF i2 THEN Fbx - mu2x - nu2x * vx};
        F_i2y = {IF i2 THEN Fby - mu2y - nu2y * vy};
        F_i3x = {IF i3 THEN Fbx - mu3x - nu3x * vx};
        F_i3y = {IF i3 THEN Fby - mu3y - nu3y * vy};
        F_i4x = {IF i4 THEN Fbx - mu4x - nu4x * vx};
        F_i4y = {IF i4 THEN Fby - mu4y - nu4y * vy};
        /* Trick: the definition of F_i5 is omitted */
        Fx = {IF i5 THEN Fbx - mu5x - nu5x * vx
                    ELSE F_i1x + F_i2x + F_i3x + F_i4x};
        Fy = {IF i5 THEN Fby - mu5y - nu5y * vy
                    ELSE F_i1y + F_i2y + F_i3y + F_i4y};
    }

    /* DHA 2: */
    AD {
        h1x1 = x - xh1 <=  size/2;      h1x2 = x - xh1 >= -size/2;
```

```
        h1y1 = y - yh1 <=  size/2;      h1y2 = y - yh1 >= -size/2;

        h2x1 = x - xh2 <=  size/2;      h2x2 = x - xh2 >= -size/2;
        h2y1 = y - yh2 <=  size/2;      h2y2 = y - yh2 >= -size/2;
    }
    LOGIC {
        dh1 = h1x1 & h1x2 & h1y1 & h1y2;
        dh2 = h2x1 & h2x2 & h2y1 & h2y2;
    }
    AUTOMATA {
        d1 = d1 || dh1;
        d2 = d2 || dh2;
    }

    /* DHA 3: */
    LOGIC {
        no_del = ~d1 & ~d2;
    }
    DA {
        F_d1x = {IF ~d1 THEN Fx / (Mb + Mm) - Fx / Mb};
        F_d1y = {IF ~d1 THEN Fy / (Mb + Mm) - Fy / Mb};
        F_d2x = {IF ~d2 THEN Fx / (Mb + Mm) - Fx / Mb};
        F_d2y = {IF ~d2 THEN Fy / (Mb + Mm) - Fy / Mb};
        F_d3x = {IF no_del THEN Fx / (Mb + 2*Mm) - F_d1x - F_d2x - Fx / Mb};
        F_d3y = {IF no_del THEN Fy / (Mb + 2*Mm) - F_d1y - F_d2y - Fy / Mb};
    }
    CONTINUOUS {
        vx = vx + Ts *(F_d1x + F_d2x + F_d3x + Fx / Mb);
        vy = vy + Ts *(F_d1y + F_d2y + F_d3y + Fy / Mb);
        x = x + vx * Ts;
        y = y + vy * Ts;
    }

    OUTPUT {
        xo = x;     yo = y;
        n = (REAL d1) + (REAL d2);
    }
    MUST {
        Fbx >= -Fmax;   Fbx <=  Fmax;
        Fby >= -Fmax;   Fby <=  Fmax;

        x >= -space;    x <=  space;
        y >= -space;    y <=  space;

        vx >= -vmax;    vx <=  vmax;
        vy >= -vmax;    vy <=  vmax;
    }
}
}
```

A.2 DTC Drive with Two-Level Inverter

```
SYSTEM drive_2level {

INTERFACE {
    STATE {
```

```
        REAL    psi_ds  [0, 1.2],      /* d-component of stator flux       */
                psi_qs  [0, 1.2],      /* q-component of stator flux       */
                alpha   [0.4, 1],      /* alpha = cos(phi)                 */
                u_a_old [-1, 1],       /* last switch position (stack a)   */
                u_b_old [-1, 1],       /* last switch position (stack b)   */
                u_c_old [-1, 1];       /* last switch position (stack c)   */
    }
    INPUT {
        BOOL    u_ap, u_bp, u_cp,      /* current switch position (zero or positive) */
                u_an, u_bn, u_cn;      /* current switch position (zero or negative) */
    }
    OUTPUT {
        REAL    Te_out,                /* torque              */
                psi_s_sq_out;          /* squared stator flux */
    }
    PARAMETER {
        REAL    /* d-component of rotor flux */
                psi_dr,

                /* parameters depending on sampling interval */
                am_11 = 0.9997, am_12 = 0, am_21 = 0, am_22 = 0.9997,
                b_11 = 0.0078, b_12 = 0, b_21 = 0, b_22 = 0.0078,

                /* parameters depending on operating point */
                ar_11,      /* ar_11=cos(Ts*wr) */
                ar_12,      /* ar_12=sin(Ts*wr) */

                /* motor and inverter model parameters */
                Vdc  = 1.5937,
                xm   = 2.3489,
                D    = 0.6265,
                rs   = 0.0108,

                /* bounds and references */
                Te_min, Te_max,
                psi_s_sq_min, psi_s_sq_max, psi_s_sq_ref,

                /* penalties */
                Q_torque_bound,
                Q_flux_bound, Q_flux_ref,
                Q_du;
    }
}

IMPLEMENTATION {
    AUX {
        /* inputs */
        REAL    in1, in2,
                u_a, u_b, u_c,
                u_d, u_q;
        REAL    zap_Y_d, zan_Y_d, zap_Y_q, zan_Y_q,
                zbp_Y_d, zbn_Y_d, zbp_Y_q, zbn_Y_q,
                zcp_Y_d, zcn_Y_d, zcp_Y_q, zcn_Y_q,
                za_Y_d, zb_Y_d, zc_Y_d,
                za_Y_q, zb_Y_q, zc_Y_q;

        /* 1-dim approximation of \psi_{ds}^2 as a function of \psi_{ds} */
        BOOL    d10;
```

```
    REAL    psi_ds_sq,
            z10, z11;

    /* 1-dim approximation of \psi_{qs}^2 as a function of \psi_{qs} */
    BOOL    d20, d21;
    REAL    psi_qs_sq,
            z20, z21, z22;

    /* v1-dim approximation of \beta as a function of \alpha */
    BOOL    d30, d31, d32;
    REAL    beta [0, 0.95],
            z30, z31, z32, z33;

    /* outputs */
    REAL    Te [-0.5, 1.2],
            psi_s_sq [0.5, 1.5];

    /* penalties */
    REAL    e_objective,
            e_switching,
            e_Te,
            e_psi_s_sq, e_psi_s_sq_ref,
            e_dua, e_dub, e_duc;
}

/********** Voltage vector section **********/
LINEAR {
    /* each phase is described by an integer variable u_x */
    u_a = (REAL u_ap) - (REAL u_an);
    u_b = (REAL u_bp) - (REAL u_bn);
    u_c = (REAL u_cp) - (REAL u_cn);
}
DA {
    /* Park transformation for the d-component of the stator voltage */
    zap_Y_d = {IF u_ap THEN  alpha ELSE 0};
    zan_Y_d = {IF u_an THEN -alpha ELSE 0};

    zbp_Y_d = {IF u_bp THEN -0.5*alpha + 0.866*beta ELSE 0};
    zbn_Y_d = {IF u_bn THEN  0.5*alpha - 0.866*beta ELSE 0};

    zcp_Y_d = {IF u_cp THEN -0.5*alpha - 0.866*beta ELSE 0};
    zcn_Y_d = {IF u_cn THEN  0.5*alpha + 0.866*beta ELSE 0 };

    /* Park transformation for the q-component of the stator voltage */
    zap_Y_q = {IF u_ap THEN -beta ELSE 0};
    zan_Y_q = {IF u_an THEN  beta ELSE 0};

    zbp_Y_q = {IF u_bp THEN  0.5*beta + 0.866*alpha ELSE 0};
    zbn_Y_q = {IF u_bn THEN -0.5*beta - 0.866*alpha ELSE 0};

    zcp_Y_q = {IF u_cp THEN -0.866*alpha + 0.5*beta ELSE 0};
    zcn_Y_q = {IF u_cn THEN  0.866*alpha - 0.5*beta ELSE 0};
}
LINEAR {
    /* d-component of the voltage vector at each phase is calculated by: */
    za_Y_d  = zap_Y_d + zan_Y_d;
    zb_Y_d  = zbp_Y_d + zbn_Y_d;
    zc_Y_d  = zcp_Y_d + zcn_Y_d;
```

```
    /* q-component of the voltage vector at each phase is calculated by: */
    za_Y_q  = zap_Y_q + zan_Y_q;
    zb_Y_q  = zbp_Y_q + zbn_Y_q;
    zc_Y_q  = zcp_Y_q + zcn_Y_q;

    /* d- and q-components of the stator voltage: */
    u_d = (2/3)*(Vdc/2)*(za_Y_d + zb_Y_d + zc_Y_d);
    u_q = (2/3)*(Vdc/2)*(za_Y_q + zb_Y_q + zc_Y_q);
}
MUST {
    /* restrict switch position to {-1, 0, 1} */
    (REAL u_ap) + (REAL u_an) <= 1;
    (REAL u_bp) + (REAL u_bn) <= 1;
    (REAL u_cp) + (REAL u_cn) <= 1;

    /* restrict switch position to {-1, 1} to yield a 2-level inverter */
    1 -(REAL u_ap) - (REAL u_an)  <= 0;
    1 -(REAL u_bp) - (REAL u_bn)  <= 0;
    1 -(REAL u_cp) - (REAL u_cn)  <= 0;
}

/********** Motor section **********/
AD {
    /* 1-dim approximation of \psi_{ds}^2 as a function of \psi_{ds} */
    d10 = psi_ds -0.925 <= 0;

    /* 1-dim approximation of \psi_{qs}^2 as a function of \psi_{qs} */
    d20 = psi_qs -0.13 <= 0;
    d21 = psi_qs -0.27 <= 0;

    /* 1-dim approximation of \beta as a function of \alpha */
    d30 = alpha -0.61084 <= 0;
    d31 = alpha -0.79118 <= 0;
    d32 = alpha -0.93104 <= 0;
}
DA {
    /* 1-dim approximation of \psi_{ds}^2 as a function of \psi_{ds} */
    z10 = {IF  d10 THEN 1.72*psi_ds -0.7387};
    z11 = {IF ~d10 THEN 1.98*psi_ds -0.9792};

    /* 1-dim approximation of \psi_{qs}^2 as a function of \psi_{qs} */
    z20 = {IF  d20       THEN 0.12*psi_qs -0.002};
    z21 = {IF ~d20 & d21 THEN 0.40*psi_qs -0.0384};
    z22 = {IF ~d21       THEN 0.68*psi_qs -0.114};

    /* 1-dim approximation of \beta as a function of \alpha */
    z30 = {IF  d30       THEN -0.5661*alpha +1.143};
    z31 = {IF ~d30 & d31 THEN -0.9871*alpha +1.4001};
    z32 = {IF ~d31 & d32 THEN -1.6232*alpha +1.9034};
    z33 = {IF ~d32       THEN -5.6862*alpha +5.6862};
}
LINEAR {
    /* building beta: */
    beta  = z30 + z31 + z32 + z33;

    /* inputs to the state update equations (stator voltage and rotor flux): */
    in1 = b_11*(u_d + (rs*xm/D)*psi_dr) + b_12*u_q;
```

```
    in2 = b_21*(u_d + (rs*xm/D)*psi_dr) + b_22*u_q;
}
CONTINUOUS {
    /* state update equations for the motor */
    psi_ds = ( ar_11*am_11+ar_12*am_21)*psi_ds + ( ar_11*am_12+ar_12*am_22)*psi_qs + in1;
    psi_qs = (-ar_12*am_11+ar_11*am_21)*psi_ds + (-ar_12*am_12+ar_11*am_22)*psi_qs + in2;
    alpha  = ar_11*alpha - ar_12*beta;
}
LINEAR {
    /* squared stator flux: */
    psi_ds_sq = z10 + z11;
    psi_qs_sq = z20 + z21 + z22;
    psi_s_sq  = psi_ds_sq + psi_qs_sq ;

    /* torque (linear in psi_qs): */
    Te = (xm/D)*psi_dr*psi_qs;
}

/********** Output section **********/
OUTPUT {
    /* torque and squared stator flux: */
    Te_out       = Te;
    psi_s_sq_out = psi_s_sq;
}

/********** Cost section **********/
CONTINUOUS {
    /* previous switching for penalizing */
    u_a_old = u_a;
    u_b_old = u_b;
    u_c_old = u_c;
}
MUST {
    /* penalty on torque */
    Q_torque_bound*( Te - Te_max) <= e_Te;
    Q_torque_bound*(-Te + Te_min) <= e_Te;
    0                             <= e_Te;

    /* penalties on flux */
    Q_flux_bound*( psi_s_sq - psi_s_sq_max) <= e_psi_s_sq;
    Q_flux_bound*(-psi_s_sq + psi_s_sq_min) <= e_psi_s_sq;
    0                                       <= e_psi_s_sq;

    Q_flux_ref*( psi_s_sq - psi_s_sq_ref) <= e_psi_s_sq_ref;
    Q_flux_ref*(-psi_s_sq + psi_s_sq_ref) <= e_psi_s_sq_ref;

    /* penalties on switching */
    Q_du*( u_a - u_a_old) <= e_dua;
    Q_du*(-u_a + u_a_old) <= e_dua;

    Q_du*( u_b - u_b_old) <= e_dub;
    Q_du*(-u_b + u_b_old) <= e_dub;

    Q_du*( u_c - u_c_old) <= e_duc;
    Q_du*(-u_c + u_c_old) <= e_duc;
}
LINEAR {
    /* penalties */
```

```
        e_objective = e_Te + e_psi_s_sq + e_psi_s_sq_ref;          /* with time-invariant penalties */
        e_switching = e_dua + e_dub + e_duc;                       /* with time-varying penalties   */
    }
}
}
```

A.3 DTC Drive with Three-Level Inverter

To improve the readability and to shorten the exposition of the HYSDEL code for the three-level inverter, we present in the following only the parts of the code that differ from the one of the two-level inverter. To the INTERFACE section, the following is added.

```
    STATE {
        REAL    npp     [-0.2, 0.2],    /* neutral point potential          */
                lamda   [-20, 20],      /* distribution of switching losses */
    }

    PARAMETER {
        REAL    /* parameters depending on sampling interval */
                Ts  = 0.0078,

                /* motor and inverter model parameters */
                xrr = 2.4593,
                xCt = 4.3715,

                /* bounds and references */
                npp_min, npp_max,

                /* penalties */
                Q_npp,
                Q_lamda;
    }
```

In the IMPLEMENTATION section, the following modifications need to be done. First, the restriction

```
    MUST {
        /* restrict switch position to {-1, 1} to yield a 2-level inverter */
        1 -(REAL u_ap) - (REAL u_an)  <= 0;
        1 -(REAL u_bp) - (REAL u_bn)  <= 0;
        1 -(REAL u_cp) - (REAL u_cn)  <= 0;
    }
```

is removed. Second, in the cost section, the sums of the penalties in

```
    LINEAR {
        /* penalties */
        e_objective = e_Te + e_psi_s_sq + e_psi_s_sq_ref;          /* with time-invariant penalties */
        e_switching = e_dua + e_dub + e_duc;                       /* with time-varying penalties   */
    }
```

are replaced by

```
LINEAR {
    /* penalties */
    e_objective = e_Te + e_psi_s_sq + e_psi_s_sq_ref + e_npp + e_lamda; /* with time-invariant penalties */
    e_switching = e_dua + e_dub + e_duc;                                /* with time-varying penalties   */
}
```

Third, the following parts are added.

```
AUX {
    /* approximation of neutral point potential */
    BOOL    d40, d41, d42, d43, d44;
    REAL    term1 [0, 0.6],
            z40, z41, z42, z43, z44, z45, z46, z47;
    BOOL    d50, d51, d52, d53, d54;
    REAL    term2 [0, 0.6],
            z50, z51, z52, z53, z54, z55, z56, z57;
    BOOL    d60, d61, d62, d63, d64;
    REAL    term3 [0, 1.2],
            z60, z61, z62, z63, z64, z65, z66, z67;
    BOOL    d70, d71, d72, d73, d74;
    REAL    term4 [0, 1.2],
            z70, z71, z72, z73, z74, z75, z76, z77;
    REAL    id [0, 0.6], iq [0, 1.2],
            first_row [-2, 2], second_row [-2, 2], third_row [-2, 2],
            i_npp_1, i_npp_2, i_npp_3, i_npp;

    /* penalties */
    REAL    e_npp,
            e_lamda;

    /* switching constraints */
    BOOL    d_sw_a, d_sw_a_low, d_sw_a_hig,
            d_sw_b, d_sw_b_low, d_sw_b_hig,
            d_sw_c, d_sw_c_low, d_sw_c_hig,
            d_sw_ab, d_sw_ac, d_sw_bc;
    REAL    z_sw_ab_new, z_sw_ab_old,
            z_sw_ac_new, z_sw_ac_old,
            z_sw_bc_new, z_sw_bc_old;
}

/********** Inverter section **********/
LINEAR {
     /* stator current: */
    id = (xrr/D)*psi_ds - (xm/D)*psi_dr;
    iq = (xrr/D)*psi_qs;
}

/* the modeling of the neutral point potential requires the pwa approximation of 4 terms:
1. the term \alpha*i_d = term1
2. the term \beta *i_d = term2
3. the term \alpha*i_q = term3
4. the term \beta *i_q = term4 */
AD {
    /* 2-dim approximation of \alpha*i_d */
    d40 = id -0.3 <= 0;
    d41 = 0.3*alpha -0.3*id -0.12 <= 0;
    d42 = alpha -0.7 <= 0;
```

```
    d43 = 0.3*alpha -0.3*id -0.03 <= 0;
    d44 = 0.3*alpha -0.3*id -0.21 <= 0;

    /* 2-dim approximation of \beta*i_d */
    d50 = id -0.3 <= 0;
    d51 = 0.3*beta -0.455*id <= 0;
    d52 = beta -0.455 <= 0;
    d53 = 0.3*beta -0.455*id +0.1365 <= 0;
    d54 = 0.3*beta -0.455*id -0.1365 <= 0;

    /* 2-dim approximation of \alpha*i_q */
    d60 = iq -0.6 <= 0;
    d61 = 0.6*alpha -0.3*iq -0.24 <= 0;
    d62 = alpha -0.7 <= 0;
    d63 = 0.6*alpha -0.3*iq -0.06 <= 0;
    d64 = 0.6*alpha -0.3*iq -0.42 <= 0;

    /* 2-dim approximation of \beta*i_q */
    d70 = iq -0.6 <= 0;
    d71 = 0.6*beta -0.455*iq <= 0;
    d72 = beta -0.455 <= 0;
    d73 = 0.6*beta -0.455*iq +0.273 <= 0;
    d74 = 0.6*beta -0.455*iq -0.273 <= 0;
}
DA {
    /* 2-dim approximation of \alpha*i_d */
    z40 = {IF  d40 &  d41        THEN 0.3006200*alpha + 0.40286*id -0.12849};
    z41 = {IF ~d41 &  d42        THEN 0.0028604*alpha + 0.70062*id -0.0093802};
    z42 = {IF  d43               THEN 0.5749200*alpha + 0.42508*id -0.24487};
    z43 = {IF ~d40 &  d42 & ~d43 THEN 0.3006200*alpha + 0.69938*id -0.21744};
    z44 = {IF  d40 & ~d42 &  d44 THEN 0.2993800*alpha + 0.70062*id -0.21694};
    z45 = {IF ~d44               THEN 0.0250790*alpha + 0.97492*id -0.024933};
    z46 = {IF  d41 & ~d42        THEN 0.5971400*alpha + 0.69938*id -0.425};
    z47 = {IF ~d40 & ~d41        THEN 0.2993800*alpha + 0.99714*id -0.3059};

    /* 2-dim approximation of \beta*i_d */
    z50 = {IF  d50 &  d51        THEN 0.3050000*beta + 0.0087773*id -0.014692};
    z51 = {IF ~d51 &  d52        THEN 0.0057872*beta + 0.4625900*id -0.014692};
    z52 = {IF  d53               THEN 0.5745100*beta + 0.0386600*id -0.023657};
    z53 = {IF ~d50 &  d52 & ~d53 THEN 0.3050000*beta + 0.4474100*id -0.14628};
    z54 = {IF  d50 & ~d52 &  d54 THEN 0.2950000*beta + 0.4625900*id -0.14628};
    z55 = {IF ~d54               THEN 0.0254900*beta + 0.8713400*id -0.023657};
    z56 = {IF  d51 & ~d52        THEN 0.5942100*beta + 0.4474100*id -0.27787};
    z57 = {IF ~d50 & ~d51        THEN 0.2950000*beta + 0.9012200*id -0.27787};

    /* 2-dim approximation of \alpha*i_q */
    z60 = {IF  d60 &  d61        THEN 0.610010*alpha + 0.40579*iq -0.26338};
    z61 = {IF ~d61 &  d62        THEN 0.011574*alpha + 0.70500*iq -0.024004};
    z62 = {IF  d63               THEN 1.149000*alpha + 0.42549*iq -0.4908};
    z63 = {IF ~d60 &  d62 & ~d63 THEN 0.610010*alpha + 0.69500*iq -0.4369};
    z64 = {IF  d60 & ~d62 &  d64 THEN 0.589990*alpha + 0.70500*iq -0.4289};
    z65 = {IF ~d64               THEN 0.050981*alpha + 0.97451*iq -0.051588};
    z66 = {IF  d61 & ~d62        THEN 1.188400*alpha + 0.69500*iq -0.84179};
    z67 = {IF ~d60 & ~d61        THEN 0.589990*alpha + 0.99421*iq -0.60242};

    /* 2-dim approximation of \beta*i_q */
    z70 = {IF  d70 &  d71        THEN 0.610010*beta + 0.0087773*iq -0.029384};
    z71 = {IF ~d71 &  d72        THEN 0.011574*beta + 0.4625900*iq -0.029384};
```

```
    z72 = {IF  d73               THEN 1.149000*beta + 0.0386600*iq -0.047314};
    z73 = {IF ~d70 &  d72 & ~d73 THEN 0.610010*beta + 0.4474100*iq -0.29256};
    z74 = {IF  d70 & ~d72 &  d74 THEN 0.589990*beta + 0.4625900*iq -0.29256};
    z75 = {IF ~d74               THEN 0.050981*beta + 0.8713400*iq -0.047314};
    z76 = {IF  d71 & ~d72        THEN 1.188400*beta + 0.4474100*iq -0.55574};
    z77 = {IF ~d70 & ~d71        THEN 0.589990*beta + 0.9012200*iq -0.55574};
}
LINEAR {
    /* building term1, term2, term3 and term4: */
    term1 = z40 + z41 + z42 + z43 + z44 + z45 + z46 + z47;
    term2 = z50 + z51 + z52 + z53 + z54 + z55 + z56 + z57;
    term3 = z60 + z61 + z62 + z63 + z64 + z65 + z66 + z67;
    term4 = z70 + z71 + z72 + z73 + z74 + z75 + z76 + z77;

    /* auxiliary variables for expression P^(-1)*[id iq 0]': */
    first_row  =        term1                              -       term4;
    second_row = -0.500*term1 + 0.866*term2 + 0.866*term3 + 0.500*term4;
    third_row  = -0.500*term1 - 0.866*term2 - 0.866*term3 + 0.500*term4;
}
DA {
    /* components of current flowing through npp (depending on switch positions): */
    i_npp_1 = {IF u_ap || u_an THEN first_row  ELSE 0};
    i_npp_2 = {IF u_bp || u_bn THEN second_row ELSE 0};
    i_npp_3 = {IF u_cp || u_cn THEN third_row  ELSE 0};
}
LINEAR {
    /* npp current: */
    i_npp = i_npp_1 + i_npp_2 + i_npp_3;
}
CONTINUOUS {
    /* state update equations for the inverter */
    npp   = npp   + ((Ts/(2*xCt))*i_npp);         /* the plus sign in these two equations holds for    */
    lamda = lamda + (u_a + u_b + u_c);            /* the even sectors (0,2,4); for odd sectors (1,3,5) */
                                                  /* we need a minus sign instead */
}

/********** Cost section **********/
MUST {
    /* penalty on neutral point potential */
    Q_npp*( npp - npp_max) <= e_npp;
    Q_npp*(-npp + npp_min) <= e_npp;
    0                      <= e_npp;

    /* penalty on distribution of switching losses */
    Q_lamda*( lamda ) <= e_lamda;
    Q_lamda*(-lamda ) <= e_lamda;
    0                 <= e_lamda;
}

/********** Switching constraints section **********/
AD {
    /* detecting switch change in phase: */
    d_sw_a_low =  u_a - u_a_old <= 0;
    d_sw_a_hig = -u_a + u_a_old <= 0;

    d_sw_b_low =  u_b - u_b_old <= 0;
    d_sw_b_hig = -u_b + u_b_old <= 0;
```

```
        d_sw_c_low =  u_c - u_c_old <= 0;
        d_sw_c_hig = -u_c + u_c_old <= 0;
    }
    LOGIC {
        /* detecting case of no switch transition within phase: */
        d_sw_a = d_sw_a_low & d_sw_a_hig;       /* phase a did NOT switch */
        d_sw_b = d_sw_b_low & d_sw_b_hig;       /* phase b did NOT switch */
        d_sw_c = d_sw_c_low & d_sw_c_hig;       /* phase c did NOT switch */

        /* detecting case of two transitions: */
        d_sw_ab = ~d_sw_a & ~d_sw_b &  d_sw_c;  /* phases a and b switched, c did not switch */
        d_sw_ac = ~d_sw_a &  d_sw_b & ~d_sw_c;  /* phases a and c switched, b did not switch */
        d_sw_bc =  d_sw_a & ~d_sw_b & ~d_sw_c;  /* phases c and b switched, a did not switch */
    }
    DA {
        /* case of two switch transitions: */
        z_sw_ab_new = {IF d_sw_ab THEN u_a + u_b ELSE 0};
        z_sw_ac_new = {IF d_sw_ac THEN u_a + u_c ELSE 0};
        z_sw_bc_new = {IF d_sw_bc THEN u_b + u_c ELSE 0};

        z_sw_ab_old = {IF d_sw_ab THEN u_a_old + u_b_old ELSE 0};
        z_sw_ac_old = {IF d_sw_ac THEN u_a_old + u_c_old ELSE 0};
        z_sw_bc_old = {IF d_sw_bc THEN u_b_old + u_c_old ELSE 0};
    }
    MUST {
        /* transitions between -1 and 1 are not allowed */
         u_a - u_a_old <= 1;
        -u_a + u_a_old <= 1;

         u_b - u_b_old <= 1;
        -u_b + u_b_old <= 1;

         u_c - u_c_old <= 1;
        -u_c + u_c_old <= 1;

        /* with two transitions, the sum of the previous and the new states must be 0 */
        z_sw_ab_new + z_sw_ab_old <= 0;
       -z_sw_ab_new - z_sw_ab_old <= 0;

        z_sw_ac_new + z_sw_ac_old <= 0;
       -z_sw_ac_new - z_sw_ac_old <= 0;

        z_sw_bc_new + z_sw_bc_old <= 0;
       -z_sw_bc_new - z_sw_bc_old <= 0;

        /* all switches changing (three transitions) not allowed */
        1 - (REAL d_sw_a) - (REAL d_sw_b) - (REAL d_sw_c) <= 0;
    }
```

A.4 Synchronous Step-Down DC-DC Converter

```
SYSTEM syn_nu3 {

INTERFACE {
    STATE {
        REAL    iL [-3, 3],
```

```
                vo [-1, 1],
                d_old [0, 1],
                vo_ref [0.2, 1],
                iL_max [0.6, 3];
    }
    INPUT {
        REAL    d [0, 1];
    }
    OUTPUT {
        REAL    vo_errSum_out,
                du_out;
    }
    PARAMETER {
        REAL    a_11 = 0.9557,  a_12 = -0.6724,  a_21 = 0.0285,  a_22 =  0.9573,
                b_1  = 0.6826,  b_2  =  0.0135,
                Ts = 1/3;
    }
}

IMPLEMENTATION {
    AUX {
        REAL    iL_upd_0, vo_upd_0;
        BOOL    s1, s1_2;
        REAL    iL1, iL_upd0_1, iL_upd1_1, vo1, vo_upd0_1, vo_upd1_1;
        BOOL    s2, s2_3;
        REAL    iL2, iL_upd0_2, iL_upd1_2, vo2, vo_upd0_2, vo_upd1_2;
        BOOL    s3;
        REAL    iL3, vo3;
        REAL    vo_errSum;
    }

    /* n = 0 */
    DA {
        iL_upd_0 = { IF s1 THEN b_1 ELSE b_1*(d/Ts-0) };
        vo_upd_0 = { IF s1 THEN b_2 ELSE b_2*(d/Ts-0) };
    }
    LINEAR {
        iL1 = a_11*iL + a_12*vo + iL_upd_0;
        vo1 = a_21*iL + a_22*vo + vo_upd_0;
    }

    /* n = 1 */
    AD {
        s1 = d >= 1*Ts;
    }
    LOGIC {
        s1_2 = s1 & ~s2;
    }
    DA {
        iL_upd0_1 = { IF s1_2 THEN b_1*(d/Ts-1) };
        iL_upd1_1 = { IF s2   THEN b_1          };
        vo_upd0_1 = { IF s1_2 THEN b_2*(d/Ts-1) };
        vo_upd1_1 = { IF s2   THEN b_2          };
    }
    LINEAR {
        iL2 = a_11*iL1 + a_12*vo1 + iL_upd0_1 + iL_upd1_1;
        vo2 = a_21*iL1 + a_22*vo1 + vo_upd0_1 + vo_upd1_1;
    }
```

```
    /* n = 2 */
    AD {
        s2 = d >= 2*Ts;
    }
    LOGIC {
        s2_3 = s2 & ~s3;
    }
    DA {
        iL_upd0_2 = { IF s2_3 THEN b_1*(d/Ts-2) };
        iL_upd1_2 = { IF s3 THEN b_1 };
        vo_upd0_2 = { IF s2_3 THEN b_2*(d/Ts-2) };
        vo_upd1_2 = { IF s3 THEN b_2 };
    }
    LINEAR {
        iL3 = a_11*iL2 + a_12*vo2 + iL_upd0_2 + iL_upd1_2;
        vo3 = a_21*iL2 + a_22*vo2 + vo_upd0_2 + vo_upd1_2;
    }

    /* n = 3 */
    AD {
        s3 = d >= 3*Ts;
    }

    /* state-update from k to k+1 */
    CONTINUOUS {
        iL = iL3;
        vo = vo3;
        d_old = d;
        vo_ref = vo_ref;
        iL_max = iL_max;
    }

    /* output voltage error */
    LINEAR {
        vo_errSum = (0.5*vo0 + vo1 + vo2 + 0.5*vo3)*Ts - vo_ref;
    }

    /* outputs */
    OUTPUT {
        vo_errSum_out = vo_errSum;
        du_out = d-d_old;
    }

    /* hard constraints on variables */
    MUST {
        iL  >= -iL_max;  iL  <= iL_max;
        iL1 >= -iL_max;  iL1 <= iL_max;
        iL2 >= -iL_max;  iL2 <= iL_max;
        iL3 >= -iL_max;  iL3 <= iL_max;
        vo >= -1;        vo <= 1;
        d_old >= 0;      d_old <= 1;
        vo_ref >= 0.2;   vo_ref <= 1;
        iL_max >= 0.6;   iL_max <= 3;
        d >= 0;          d <= 1;
    }
}
}
```

A.5 Power System

This section presents the HYSDEL code of the example power system from Section 9.3.2 with the sampling interval $T_s = 10\,\mathrm{s}$ and the fault in the tie line being applied. Since the OLTC is modelled here by a finite state machine, the model differs slightly from the representation (9.13)–(9.18).

```
SYSTEM powerSystem_Ts10_withFault {

INTERFACE {
    STATE {
        REAL    n [0.8, 1.2],       /* tap position */
                x [-0.5, 10],       /* load state */
                timer [-0.1, 1000]; /* timer of OLTC */
        BOOL    idle, tapdelay,     /* discrete states for OLTC automaton */
                sC_old, sL_old;     /* previous capacitor switching, load-shedding */
    }
    INPUT {
        BOOL    n_up, n_down,       /* tap command */
                sC, sL;             /* capacitor switching, load-shedding */
    }
    OUTPUT {
        REAL    V3m_out, V4m_out;   /* bus voltages 3 and 4 */
        REAL    sc_slSUM_out;       /* sum of slacks on bus voltages */
        BOOL    sc_violated_out;    /* at least one soft constraint on bus voltage violated */
    }
    PARAMETER {
        REAL    Ts = 10,
                Eps = 1e-3;
        REAL    n_min = 0.8,        /* constants of OLTC model */
                n_max = 1.2,
                n_step = 0.02,
                t_tapdelay = 29;
        REAL    V3m_min = 0.9,      /* lower soft constraints on bus voltages */
                V4m_min = 0.9;
    }
}

IMPLEMENTATION {
    AUX {
        /* delta u(k) */
        REAL    Dn [-0.03, 0.03], DsC [-1, 1], DsL [-1, 1];

        /* OLTC */
        BOOL    n_updown,
                n_upbound, n_lobound,
                wait_tapdelay, ready2tap;
        REAL    timer_z1, timer_z2, timer_z3;
        REAL    n_z1[-0.03, 0.01], n_z2[-0.01, 0.03];

        /* soft constraints on bus voltages */
        REAL    sc3lo_sl [-0.2, 0], sc3lo_slPOS[0, 0.1],
                sc4lo_sl [-0.2, 0], sc4lo_slPOS[0, 0.1],
                sc_slSUM [0, 0.2];
        BOOL    sc3lo_d, sc4lo_d,
                sc_violated;
```

```
    /* PWA approximation of switched nonlinear system (DAE with AVR) */
    BOOL    reg_d1, reg_d2, reg_d3, reg_d4, reg_d5, reg_d6, reg_d7, reg_d8;
    BOOL    d1, d2, d3, d4, d5, d6, d7, d8, d9, d10, d11, d12, d13, d14, d15, d16, d17, d18, d19,
            d20, d21, d22, d23, d24, d25, d26, d27, d28, d29, d30, d31, d32;
    REAL    xt[-0.5, 10], xt_z1, xt_z2, xt_z3, xt_z4, xt_z5, xt_z6, xt_z7, xt_z8, xt_z9, xt_z10,
            xt_z11, xt_z12, xt_z13, xt_z14, xt_z15, xt_z16, xt_z17, xt_z18, xt_z19, xt_z20, xt_z21,
            xt_z22, xt_z23, xt_z24, xt_z25, xt_z26, xt_z27, xt_z28, xt_z29, xt_z30, xt_z31, xt_z32;
    REAL    V3m[0.8, 1.1], V3m_z1, V3m_z2, V3m_z3, V3m_z4, V3m_z5, V3m_z6, V3m_z7, V3m_z8, V3m_z9,
            V3m_z10, V3m_z11, V3m_z12, V3m_z13, V3m_z14, V3m_z15, V3m_z16, V3m_z17, V3m_z18,
            V3m_z19, V3m_z20, V3m_z21, V3m_z22, V3m_z23, V3m_z24, V3m_z25, V3m_z26, V3m_z27,
            V3m_z28, V3m_z29, V3m_z30, V3m_z31, V3m_z32;
    REAL    V4m[0.8, 1.1], V4m_z1, V4m_z2, V4m_z3, V4m_z4, V4m_z5, V4m_z6, V4m_z7, V4m_z8, V4m_z9,
            V4m_z10, V4m_z11, V4m_z12, V4m_z13, V4m_z14, V4m_z15, V4m_z16, V4m_z17, V4m_z18,
            V4m_z19, V4m_z20, V4m_z21, V4m_z22, V4m_z23, V4m_z24, V4m_z25, V4m_z26, V4m_z27,
            V4m_z28, V4m_z29, V4m_z30, V4m_z31, V4m_z32;
}

/********** OLTC **********/
AD {
    n_upbound = n >= n_max;
    n_lobound = n <= n_min;
    wait_tapdelay = timer -t_tapdelay +Eps <= 0;
}
LOGIC {
    n_updown = n_up | n_down;
    ready2tap = tapdelay & ~wait_tapdelay;
}
DA {
    timer_z1 = {IF idle THEN Ts
                        ELSE timer};
    timer_z2 = {IF tapdelay THEN Ts};
    timer_z3 = {IF ready2tap THEN -t_tapdelay};
    n_z1 = {IF (ready2tap & n_down & ~n_lobound) THEN -n_step};
    n_z2 = {IF (ready2tap & n_up   & ~n_upbound) THEN  n_step};
}
CONTINUOUS {
    timer = timer_z1 + timer_z2 + timer_z3;
    n = n + n_z1 + n_z2;
}
AUTOMATA {
    idle     = (idle | tapdelay | ready2tap) & ~n_updown;
    tapdelay = (idle | (tapdelay & wait_tapdelay) | ready2tap) & n_updown;
}

/********** PWA Approximation of Switched Nonlinear System (DAE with AVR) **********/
/* delta variables associated with polyhedra (simplices) */
MUST {
    n -1 -0.21 +0.21*(REAL reg_d1) <= 0;
    0.2*x -5.25*n +4.3 -2.11 +2.11*(REAL reg_d1) <= 0;
    x -4.75 -5.26 +5.26*(REAL reg_d2) <= 0;
    -0.2*x +5.25*n -4.3 -2.11 +2.11*(REAL reg_d2) <= 0;
    0.2*x -5.25*n +5.35 -3.16 +3.16*(REAL reg_d3) <= 0;
    x -4.75 -5.26 +5.26*(REAL reg_d4) <= 0;
    -n +1 -0.21 +0.21*(REAL reg_d4) <= 0;
    -0.2*x +5.25*n -5.35 -1.06 +1.06*(REAL reg_d4) <= 0;
    -x +4.75 -5.26 +5.26*(REAL reg_d5) <= 0;
    n -1 -0.21 +0.21*(REAL reg_d5) <= 0;
    0.2*x -5.25*n +3.25 -1.06 +1.06*(REAL reg_d5) <= 0;
```

```
    -0.2*x +5.25*n -3.25 -3.16 +3.16*(REAL reg_d6) <= 0;
    -x +4.75 -5.26 +5.26*(REAL reg_d7) <= 0;
    0.2*x -5.25*n +4.3 -2.11 +2.11*(REAL reg_d7) <= 0;
    -n +1 -0.21 +0.21*(REAL reg_d8) <= 0;
    -0.2*x +5.25*n -4.3 -2.11 +2.11*(REAL reg_d8) <= 0;
}
LINEAR {
    (REAL reg_d1) +(REAL reg_d2) +(REAL reg_d3) +(REAL reg_d4) +(REAL reg_d5) +(REAL reg_d6) +
    (REAL reg_d7) +(REAL reg_d8) <= 1;
}

/* delta variables associated with polyhedron and control input */
LOGIC {
    d1  = reg_d1 & ~sC & ~sL;
    d2  = reg_d2 & ~sC & ~sL;
    d3  = reg_d3 & ~sC & ~sL;
    d4  = reg_d4 & ~sC & ~sL;
    d5  = reg_d5 & ~sC & ~sL;
    d6  = reg_d6 & ~sC & ~sL;
    d7  = reg_d7 & ~sC & ~sL;
    d8  = reg_d8 & ~sC & ~sL;

    d9  = reg_d1 & ~sC &  sL;
    d10 = reg_d2 & ~sC &  sL;
    d11 = reg_d3 & ~sC &  sL;
    d12 = reg_d4 & ~sC &  sL;
    d13 = reg_d5 & ~sC &  sL;
    d14 = reg_d6 & ~sC &  sL;
    d15 = reg_d7 & ~sC &  sL;
    d16 = reg_d8 & ~sC &  sL;

    d17 = reg_d1 &  sC & ~sL;
    d18 = reg_d2 &  sC & ~sL;
    d19 = reg_d3 &  sC & ~sL;
    d20 = reg_d4 &  sC & ~sL;
    d21 = reg_d5 &  sC & ~sL;
    d22 = reg_d6 &  sC & ~sL;
    d23 = reg_d7 &  sC & ~sL;
    d24 = reg_d8 &  sC & ~sL;

    d25 = reg_d1 &  sC &  sL;
    d26 = reg_d2 &  sC &  sL;
    d27 = reg_d3 &  sC &  sL;
    d28 = reg_d4 &  sC &  sL;
    d29 = reg_d5 &  sC &  sL;
    d30 = reg_d6 &  sC &  sL;
    d31 = reg_d7 &  sC &  sL;
    d32 = reg_d8 &  sC &  sL;
}

/* state-update for load state */
DA {
    xt_z1  = {IF d1  THEN 0.91644*x -9.4548*n +10.7348};
    xt_z2  = {IF d2  THEN 0.88369*x -8.5953*n +10.0308};
    xt_z3  = {IF d3  THEN 0.98589*x -4.2417*n +5.5564};
    xt_z4  = {IF d4  THEN 0.91644*x -2.4185*n +3.6985};
    xt_z5  = {IF d5  THEN 0.9329*x -8.5953*n +9.7971};
    xt_z6  = {IF d6  THEN 0.89168*x -7.5133*n +9.1273};
```

```
    xt_z7  = {IF d7  THEN 1.0203*x -2.4185*n +3.2053};
    xt_z8  = {IF d8  THEN 0.9329*x -0.12495*n +1.3267};
    xt_z9  = {IF d9  THEN 0.91061*x -9.8792*n +10.9583};
    xt_z10 = {IF d10 THEN 0.88207*x -9.1301*n +10.3447};
    xt_z11 = {IF d11 THEN 0.96404*x -5.4483*n +6.554};
    xt_z12 = {IF d12 THEN 0.91061*x -4.0458*n +5.1249};
    xt_z13 = {IF d13 THEN 0.92411*x -9.1301*n +10.145};
    xt_z14 = {IF d14 THEN 0.88891*x -8.2062*n +9.5731};
    xt_z15 = {IF d15 THEN 0.99518*x -4.0458*n +4.7231};
    xt_z16 = {IF d16 THEN 0.92411*x -2.1801*n +3.195};
    xt_z17 = {IF d17 THEN 0.92027*x -9.6151*n +10.2334};
    xt_z18 = {IF d18 THEN 0.88873*x -8.7872*n +9.5552};
    xt_z19 = {IF d19 THEN 0.98645*x -4.2027*n +4.8541};
    xt_z20 = {IF d20 THEN 0.92027*x -2.4656*n +3.0839};
    xt_z21 = {IF d21 THEN 0.93663*x -8.7872*n +9.3277};
    xt_z22 = {IF d22 THEN 0.89629*x -7.7282*n +8.6721};
    xt_z23 = {IF d23 THEN 1.0204*x -2.4656*n +2.6084};
    xt_z24 = {IF d24 THEN 0.93663*x -0.26755*n +0.80808};
    xt_z25 = {IF d25 THEN 0.91751*x -10.206*n +10.5757};
    xt_z26 = {IF d26 THEN 0.8859*x -9.3761*n +9.896};
    xt_z27 = {IF d27 THEN 0.9607*x -5.1994*n +5.5907};
    xt_z28 = {IF d28 THEN 0.91751*x -4.0657*n +4.4353};
    xt_z29 = {IF d29 THEN 0.92771*x -9.3761*n +9.6973};
    xt_z30 = {IF d30 THEN 0.8937*x -8.4833*n +9.1446};
    xt_z31 = {IF d31 THEN 0.99247*x -4.0657*n +4.0793};
    xt_z32 = {IF d32 THEN 0.92771*x -2.3657*n +2.6869};
}
LINEAR {
    xt = xt_z1+xt_z2+xt_z3+xt_z4+xt_z5+xt_z6+xt_z7+xt_z8+xt_z9+xt_z10+xt_z11+xt_z12+xt_z13+xt_z14+
         xt_z15+xt_z16+xt_z17+xt_z18+xt_z19+xt_z20+xt_z21+xt_z22+xt_z23+xt_z24+xt_z25+xt_z26+xt_z27+
         xt_z28+xt_z29+xt_z30+xt_z31+xt_z32;
}
CONTINUOUS {
    x = xt;
}

/* output for V3m */
DA {
    V3m_z1  = {IF d1  THEN -0.004198*x -0.45468*n +1.3815};
    V3m_z2  = {IF d2  THEN -0.0034351*x -0.47471*n +1.3979};
    V3m_z3  = {IF d3  THEN -0.0068194*x -0.5657*n +1.4912};
    V3m_z4  = {IF d4  THEN -0.004198*x -0.63451*n +1.5613};
    V3m_z5  = {IF d5  THEN -0.0053059*x -0.47471*n +1.4068};
    V3m_z6  = {IF d6  THEN -0.0039272*x -0.5109*n +1.4292};
    V3m_z7  = {IF d7  THEN -0.0087429*x -0.63451*n +1.5829};
    V3m_z8  = {IF d8  THEN -0.0053059*x -0.72473*n +1.6568};
    V3m_z9  = {IF d9  THEN -0.0038391*x -0.43929*n +1.378};
    V3m_z10 = {IF d10 THEN -0.0032119*x -0.45576*n +1.3915};
    V3m_z11 = {IF d11 THEN -0.0057266*x -0.5303*n +1.468};
    V3m_z12 = {IF d12 THEN -0.0038391*x -0.57984*n +1.5185};
    V3m_z13 = {IF d13 THEN -0.0047388*x -0.45576*n +1.3987};
    V3m_z14 = {IF d14 THEN -0.0036339*x -0.48476*n +1.4167};
    V3m_z15 = {IF d15 THEN -0.0073669*x -0.57984*n +1.5353};
    V3m_z16 = {IF d16 THEN -0.0047388*x -0.64883*n +1.5918};
    V3m_z17 = {IF d17 THEN -0.0043015*x -0.51951*n +1.4829};
    V3m_z18 = {IF d18 THEN -0.00364*x -0.53687*n +1.4971};
    V3m_z19 = {IF d19 THEN -0.0067065*x -0.61076*n +1.573};
    V3m_z20 = {IF d20 THEN -0.0043015*x -0.67389*n +1.6373};
```

```
    V3m_z21 = {IF d21 THEN -0.0053781*x -0.53687*n +1.5054};
    V3m_z22 = {IF d22 THEN -0.0040903*x -0.57068*n +1.5263};
    V3m_z23 = {IF d23 THEN -0.0085055*x -0.67389*n +1.6573};
    V3m_z24 = {IF d24 THEN -0.0053781*x -0.75599*n +1.7245};
    V3m_z25 = {IF d25 THEN -0.0040495*x -0.49926*n +1.4764};
    V3m_z26 = {IF d26 THEN -0.0033708*x -0.51707*n +1.491};
    V3m_z27 = {IF d27 THEN -0.0055011*x -0.585*n +1.5615};
    V3m_z28 = {IF d28 THEN -0.0040495*x -0.62311*n +1.6003};
    V3m_z29 = {IF d29 THEN -0.0048139*x -0.51707*n +1.4979};
    V3m_z30 = {IF d30 THEN -0.0038027*x -0.54361*n +1.5143};
    V3m_z31 = {IF d31 THEN -0.0070844*x -0.62311*n +1.6147};
    V3m_z32 = {IF d32 THEN -0.0048139*x -0.68271*n +1.6635};
}
LINEAR {
    V3m = V3m_z1+V3m_z2+V3m_z3+V3m_z4+V3m_z5+V3m_z6+V3m_z7+V3m_z8+V3m_z9+V3m_z10+V3m_z11+V3m_z12+
          V3m_z13+V3m_z14+V3m_z15+V3m_z16+V3m_z17+V3m_z18+V3m_z19+V3m_z20+V3m_z21+V3m_z22+V3m_z23+
          V3m_z24+V3m_z25+V3m_z26+V3m_z27+V3m_z28+V3m_z29+V3m_z30+V3m_z31+V3m_z32;
}

/* output for V3m */
DA {
    V4m_z1  = {IF d1  THEN -0.0042148*x +0.5742*n +0.35671};
    V4m_z2  = {IF d2  THEN -0.0027113*x +0.53474*n +0.38904};
    V4m_z3  = {IF d3  THEN -0.007896*x +0.23665*n +0.69242};
    V4m_z4  = {IF d4  THEN -0.0042148*x +0.14002*n +0.79089};
    V4m_z5  = {IF d5  THEN -0.0053157*x +0.53474*n +0.40141};
    V4m_z6  = {IF d6  THEN -0.0032767*x +0.48121*n +0.43454};
    V4m_z7  = {IF d7  THEN -0.010209*x +0.14002*n +0.81936};
    V4m_z8  = {IF d8  THEN -0.0053157*x +0.011585*n +0.92456};
    V4m_z9  = {IF d9  THEN -0.0038537*x +0.59557*n +0.34678};
    V4m_z10 = {IF d10 THEN -0.0025578*x +0.56155*n +0.37464};
    V4m_z11 = {IF d11 THEN -0.0066124*x +0.29265*n +0.64831};
    V4m_z12 = {IF d12 THEN -0.0038537*x +0.22024*n +0.72211};
    V4m_z13 = {IF d13 THEN -0.0047327*x +0.56155*n +0.38497};
    V4m_z14 = {IF d14 THEN -0.0030512*x +0.51741*n +0.41229};
    V4m_z15 = {IF d15 THEN -0.0086266*x +0.22024*n +0.74478};
    V4m_z16 = {IF d16 THEN -0.0047327*x +0.11802*n +0.8285};
    V4m_z17 = {IF d17 THEN -0.0043121*x +0.55748*n +0.41025};
    V4m_z18 = {IF d18 THEN -0.0029022*x +0.52047*n +0.44056};
    V4m_z19 = {IF d19 THEN -0.00775*x +0.21904*n +0.74697};
    V4m_z20 = {IF d20 THEN -0.0043121*x +0.12879*n +0.83893};
    V4m_z21 = {IF d21 THEN -0.0053703*x +0.52047*n +0.45228};
    V4m_z22 = {IF d22 THEN -0.0034356*x +0.46968*n +0.48372};
    V4m_z23 = {IF d23 THEN -0.0099521*x +0.12879*n +0.86572};
    V4m_z24 = {IF d24 THEN -0.0053703*x +0.0085202*n +0.96423};
    V4m_z25 = {IF d25 THEN -0.0040795*x +0.58536*n +0.3958};
    V4m_z26 = {IF d26 THEN -0.002698*x +0.54909*n +0.4255};
    V4m_z27 = {IF d27 THEN -0.0063244*x +0.26507*n +0.71496};
    V4m_z28 = {IF d28 THEN -0.0040795*x +0.20614*n +0.77501};
    V4m_z29 = {IF d29 THEN -0.0047908*x +0.54909*n +0.43544};
    V4m_z30 = {IF d30 THEN -0.0032191*x +0.50783*n +0.46098};
    V4m_z31 = {IF d31 THEN -0.0083022*x +0.20614*n +0.79507};
    V4m_z32 = {IF d32 THEN -0.0047908*x +0.11397*n +0.87057};
}
LINEAR {
    V4m = V4m_z1+V4m_z2+V4m_z3+V4m_z4+V4m_z5+V4m_z6+V4m_z7+V4m_z8+V4m_z9+V4m_z10+V4m_z11+V4m_z12+
          V4m_z13+V4m_z14+V4m_z15+V4m_z16+V4m_z17+V4m_z18+V4m_z19+V4m_z20+V4m_z21+V4m_z22+
          V4m_z23+V4m_z24+V4m_z25+V4m_z26+V4m_z27+V4m_z28+V4m_z29+V4m_z30+V4m_z31+V4m_z32;
```

```
    }

    /********** Costs on Switching and Slacks on Soft Constraints **********/
    /* slack variables on lower soft constraints on bus voltages 3 and 4 */
    LINEAR {
        sc3lo_sl = V3m_min - V3m;
        sc4lo_sl = V4m_min - V4m;
    }
    AD {
        sc3lo_d = -sc3lo_sl <= 0;
        sc4lo_d = -sc4lo_sl <= 0;
    }
    DA {
        sc3lo_slPOS = {IF sc3lo_d THEN sc3lo_sl};
        sc4lo_slPOS = {IF sc4lo_d THEN sc4lo_sl};
    }
    LINEAR {
        sc_slSUM = sc3lo_slPOS + sc4lo_slPOS;   /* sum of slacks */
    }

    /* violation of a soft constraint on bus voltage */
    LOGIC {
        sc_violated = sc3lo_d | sc4lo_d;
    }

    /* costs for switching */
    AUTOMATA {
        sC_old = sC;
        sL_old = sL;
    }
    LINEAR {
        Dn = -n_z1 + n_z2 + ( (REAL n_up) + (REAL n_down) )*n_step*0.001;   /* change in tap position */
        DsC = (REAL sC) - (REAL sC_old);                                    /* capacitor switching */
        DsL = (REAL sL) - (REAL sL_old);                                    /* change in load-shedding */
    }

    /********** Outputs **********/
    OUTPUT {
        V3m_out = V3m;
        V4m_out = V4m;

        sc_slSUM_out = sc_slSUM;
        sc_violated_out = sc_violated;
    }

    /********** Constraint on Input **********/
    MUST {
        (REAL n_up) + (REAL n_down) <= 1.1;
    }
}
}
```

B

Nomenclature

Mathematical Definitions

Throughout this thesis, as a general rule, scalars and vectors are denoted with the lower case letters (e.g., $a, b, \ldots$), matrices are denoted with the upper case letters (e.g., $A, B, \ldots$), and sets are denoted with the upper case calligraphic letters (e.g., $\mathcal{A}, \mathcal{W}, \ldots$). If not otherwise noted, all vectors are column vectors. In the following let $c \in \mathbb{C}$ be a complex number, $s \in \mathbb{R}^n$ a column vector, $S \in \mathbb{R}^{n \times n}$ a square matrix, and $R \in \mathbb{R}^{n \times m}$ a rectangular matrix.

General

$\cdot$	general placeholder (for any variable)
$\triangleq$	definition
$\mid$ or :	such that
$\in$	is element of (belongs to)
$\forall$	for all
$\exists$	exists at least one
$\notin, \nexists, \ldots$	/ denotes negation
$\rightarrow$	mapping
$\mapsto$	maps to

Spaces

$\mathbb{N}$	positive integers, $\mathbb{N} \triangleq \mathbb{Z}_{>}$
$\mathbb{N}_0$	non-negative integers, $\mathbb{N} \triangleq \mathbb{Z}_{\geq}$
$\mathbb{Z}$	integers
$\mathbb{Z}_{\geq}$	non-negative integers
$\mathbb{Z}_{>}$	positive integers
$\mathbb{R}$	real numbers
$\mathbb{R}^n$	space of n-dimensional real vectors
$\mathbb{R}^{n\times m}$	space of n by m real matrices
$\mathbb{R}_{\geq}$	non-negative real numbers
$\mathbb{C}$	complex numbers

Operations with Logic Variables

$\vee$	or
$\wedge$	and
$\Rightarrow$	implies
$\Leftrightarrow$	if and only if

Operations with Complex Numbers

$\Re\{c\}$	real part of a complex number
$\Im\{c\}$	imaginary part of a complex number

Operations with Vectors

$\mathbb{O}_n$	vector of zeros, $\mathbb{O}_n \triangleq [0\ 0\ \ldots\ 0]^T \in \mathbb{R}^n$
$\mathbb{1}_n$	vector of ones, $\mathbb{1}_n \triangleq [1\ 1\ \ldots\ 1]^T \in \mathbb{R}^n$
$<$	element-wise comparison, similar for $\leq, =, \geq, >$
s^T	row vector
$\lvert s\rvert$	element-wise absolute value
$\lVert s\rVert$	any vector norm
$\lVert s\rVert_1$	sum of absolute elements of a vector
$\lVert s\rVert_2$	Euclidian norm
$\lVert s\rVert_\infty$	largest absolute element of a vector

Operations with Matrices

I	identity matrix (of appropriate dimension)
S $(\succeq) \succ 0$	positive (semi)definite matrix
S $(\preceq) \prec 0$	negative (semi)definite matrix
R^T	matrix transpose
S^{-1}	inverse of the square matrix
$\det(S)$	determinant of the square matrix
$\operatorname{rank}(R)$	rank of a matrix

Operations with Sets

$\emptyset$	the empty set
$(\subset) \subseteq$	(strict) subset
$(\supset) \supseteq$	(strict) superset
$\cap$	intersection
$\cup$	union

Optimization

min	minimum
max	maximum
inf	infimum
sup	supremum

Variables

In the sequel, we provide an alphabetical list of the variables used in this thesis. Care has been taken to avoid multiple definitions, yet these could not be avoided in all cases.

Symbol	**Meaning**
a_i	normal vector of the i-th hyperplane in $\mathbb{R}^d$ (of a collection of n hyperplanes)
a	matrix $\mathbb{R}^{d \times n}$ of normal vectors of a collection of n hyperplanes in $\mathbb{R}^d$
α	$\cos(\varphi)$, in DTC
A	state-space matrix (system), in MLD model
A_i	state-space matrix (system) corresponding to mode i, e.g. in DHA and in PWA model
$\mathcal{A}$	hyperplane arrangement
b_i	offset of the i-th hyperplane from the origin (of a collection of n hyperplanes)

Symbol	Meaning
b	vector $\mathbb{R}^n$ of offsets of a collection of n hyperplanes
β	$\sin(\varphi)$, in DTC
B_1, B_2, B_3	state-space matrices (input, δ and z) in MLD model
B_i	state-space matrix (input) corresponding to mode i, e.g. in DHA and in PWA model
C	state-space matrix (output), in MLD model
C_i	state-space matrix (output) corresponding to mode i, e.g. in DHA and in PWA model
d	dimension of Euclidian space
d	duty cycle, in DC-DC converter
δ	binary (or Boolean) vector, e.g. in MLD model
δ	rotor angle, in Power System
δ_e	event (binary signal), in DHA
δ_i	i-th Boolean variable, in OCR
Δ	set of Boolean vectors, in OCR
D_1, D_2, D_3	state-space matrices (direct feedthrough, δ and z), in MLD model
D_i	state-space matrix (direct feedthrough) corresponding to mode i, e.g. in DHA and in PWA model
$\mathcal{D}$	set of δ_e, in DHA
$E_1, \ldots, E_5$	inequality constraint matrices, in MLD model
$\mathscr{E}$	edge of a digraph $\mathscr{G}$, in Mode Enumeration
f	state-space vector (offset) in continuous-time, in DC-DC converter
f_i	state-space vector (offset in state-update) corresponding to mode i, e.g. in DHA and in PWA model
$f_{\mathrm{B}}, f_{\mathrm{H}}, f_{\mathrm{M}}$	logic state-update, affine threshold and mode selector function, in DHA
f_{W}	Boolean function, in OCR
f_{PWA}	state-update function, in PWA model
F	state-space matrix (system) in continuous-time, in DC-DC converter
F_j	matrix of j-th state-feedback control law
$\mathscr{F}$	feedback arc set for digraph $\mathscr{G}$, in Mode Enumeration
g	nonlinear output function, in DTC
g_{B}	binary output function, in DHA
g_i	state-space vector (offset in output) corresponding to mode i, e.g. in DHA and in PWA model
g_j	offset vector of j-th state-feedback control law

Symbol	Meaning
g_{PWA}	output function, in PWA model
$\mathscr{G}$	digraph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$, in Mode Enumeration
H_i	i-th hyperplane (in a collection of hyperplanes)
i	index, e.g. for hyperplane arrangement, mode
i_{abc}	vector of stator currents with components i_a, i_b, i_c, in DTC
i_ℓ	inductor current, in DC-DC converter
I_{id}, I_{iq}	d- and q-component of i-th current, in Power System
$\mathcal{I}$	set of indices, $i \in \mathcal{I}$, e.g. for modes, hyperplanes
j	index, e.g. for polyhedron, mode, hyperplane arrangement
J	cost function
$\mathcal{J}$	set of indices, $j \in \mathcal{J}$, e.g. for modes in PWA model
k	discrete-time instants, $k \in \mathbb{N}_0$
l	current computational order or step, in Mode Enumeration
ℓ	time-step within prediction interval
λ	distribution of switching effort, in DTC
L	Lyapunov function, in DC-DC converter
m	marking (of a cell in a hyperplane arrangement)
m_r, m_b	dimension of real and binary input vectors u_r and u_b
μ	sector number, in DTC
M	set of markings (of cells in a hyperplane arrangement)
M_b, M_w	set of markings referring to black and white polyhedra (in a hyperplane arrangement)
n	number of hyperplanes in hyperplane arrangement
n	discrete-time instants of subperiods, $n \in \{0, 1, \ldots, \nu - 1\}$, in DC-DC converter
n_r, n_b	dimensions of real and binary state vectors x_r and x_b
n_e	dimension of event vector δ_e, in DHA
n_δ, n_z	dimensions of δ and z, in MLD model
n_T	tap position, in Power System
ν	sector number, in DTC
ν	number of subperiods, in DC-DC converter
N	length of prediction horizon
O	order of complexity
p	number of polyhedra in a set of polyhedra

Symbol	Meaning
p_r, p_b	length of real and binary output vectors y_r and y_b
φ	angle between a-axis of three-phase system and d-axis of reference frame, and angular position of rotating reference frame, in DTC
ϕ	rotor angle, in Power System
ψ_s	stator flux vector with components ψ_{ds}, ψ_{qs}, in DTC
ψ_r	rotor flux vector with components ψ_{dr}, ψ_{qr}, in DTC
P	Park transformation, in DTC
P_2, P_L	active power of Generator 2 and load, in Power System
$\mathcal{P}_j$	j-th polyhedron, e.g. in PWA model
$\mathcal{P}_m$	polyhedron (cell) in hyperplane arrangement (corresponding to marking m)
Ψ_s	length of stator flux vector, in DTC
Ψ	state-space matrix (system) referring to sampling interval τ_s, in DC-DC converter
Φ	state-space vector (input) referring to sampling interval τ_s, in DC-DC converter
Π	mapping matrix, in DTC
q	number of polyhedra in a set of polyhedra
Q	penalty matrix
Q_2, Q_L	reactive power of Generator 2 and load, in Power System
$\mathcal{Q}_u^c$, $\mathcal{Q}_u^r$	core and ring (set of polyhedra) for control input u, in DTC
$\mathcal{Q}_j$	j-th polyhedron, e.g. in OCR
r_o	output resistance, in DC-DC converter
R	rotation matrix, in DTC
$\mathcal{R}$	polyhedral domain (a hyperplane arrangement is restricted to)
s	number of DHAs in a composition, in Mode Enumeration
s_C	number of capacitor banks connected, in Power System
s_L	amount of load shed, in Power System
σ_n	sampled switch position at time-instants $n\tau_s$, in DC-DC converter
$\underline{S}$, $\overline{S}$	soft constraints on lower and upper bounds, in Power System
$\mathcal{S}_j$	j-th PWA dynamic, i.e. the collection $\{A_j, B_j, f_j, C_j, D_j, g_j\}$, in Mode Enumeration
Σ	hybrid model, e.g. a DHA or a PWA model
t	continuous-time axis, also clock variable in EG
t_T	timer in OLTC, in Power System

Symbol	Meaning
τ_s	sampling interval of subperiods, in DC-DC converter
ϑ	angle corresponding to flux rotation, in DTC
T_e	electromagnetic torque, in DTC
T_ℓ	load torque, in DTC
T_s	sampling interval
u	input vector (holding both real and binary components)
u_{abc}	vector of switch positions of inverter with components u_a, u_b, u_c, in DTC
u_{dq0}	voltage vector with components u_d, u_q, u_0, in DTC
u_r, u_b	real and binary input vectors
U	sequence of input vectors (within prediction horizon)
$\mathcal{U}$	set of input vector u
$\mathcal{U}_r$, $\mathcal{U}_b$	sets of real and binary input vectors u_r and u_b
v	auxiliary input vector (holding both real and binary components), for compositions of DHAs with loops, in Mode Enumeration
v_r, v_b	auxiliary real and binary input vectors, for compositions of DHAs with loop, in Mode Enumeration
v_c, v_s, v_o	capacitor, input and output voltages, in DC-DC converter
υ_n	neutral point potential, in DTC
V_{dc}	dc-link voltage, in DTC
V_{im}	absolute value of i-th bus voltage, in Power System
V_{id}, V_{iq}	d- and q-component of i-th bus voltage, in Power System
$\mathcal{V}$	set of auxiliary input vector v, in Mode Enumeration
$\mathcal{V}$	set of parameters, in DC-DC converter
$\mathcal{V}_r$, $\mathcal{V}_b$	set of auxiliary real and binary input vectors, in Mode Enumeration
$\mathscr{V}$	vertex of a digraph $\mathscr{G}$, in Mode Enumeration
w_{ij}	element of matrix W, in Mode Enumeration
W	adjacency matrix of digraph G, in Mode Enumeration
ω_b	base angular velocity, in DTC
ω_{fr}	angular velocity of frame, in DTC
ω_r	angular velocity of rotor, in DTC
x	state vector (holding both real and binary components)
x_m	motor state vector, in DTC
x_r, x_b	real and binary state vectors
x_{Ld}, x_{Lq}	d- and q-components of load state, in Power System
ξ	state supersampled with τ_s, in DC-DC converter

Symbol	Meaning
X	"don't care" in Boolean minimization, in OCR
$\mathcal{X}$	set of state vector x
$\mathcal{X}_r$, $\mathcal{X}_b$	sets of real and binary state vectors x_r and x_b
y	output vector (holding both real and binary components)
y_r, y_b	real and binary output vectors
$\mathcal{Y}$	set of output vector y
$\mathcal{Y}_r$, $\mathcal{Y}_b$	sets of real and binary output vectors y_r and y_b
z	auxiliary real vector, in MLD model
z	upper bound, in OCR
ζ	optimization variable
ζ	state of closed-loop (autonomous) system, in DC-DC converter
$\mathcal{Z}$	set of states ζ, in DC-DC converter
$\mathcal{Z}_0$	control invariant subset of $\mathcal{Z}$, in DC-DC converter

Acronyms

AVR	Automatic Voltage Regulator
CFTOC	Constrained Finite Time Optimal Control
DAE	Differential Algebraic Equation
DHA	Discrete Hybrid Automaton
DP	Dynamic Program(ming)
DTC	Direct Torque Control
EG	Event Generator
FAS	Feedback Arc Set
FSM	Finite State Machine
HYSDEL	HYbrid System DEscription Language
LP	Linear Program(ming)
LQR	Linear Quadratic Regulator
MILP	Mixed Integer Linear Program(ming)
MIQP	Mixed Integer Quadratic Program(ming)
MLD	Mixed Logical Dynamical
MPC	Model Predictive Control
mp-LP	multi-parametric Linear Program(ming)
mp-MILP	multi-parametric Mixed Integer Linear Program(ming)
mp-MIQP	multi-parametric Mixed Integer Quadratic Program(ming)
mp-QP	multi-parametric Quadratic Program(ming)
MS	Mode Selector
OCR	Optimal Complexity Reduction
p.u.	per unit
PWA	Piecewise Affine
PWM	Pulse Width Modulation
PWQ	Piecewise Quadratic
QP	Quadratic Program(ming)
RHC	Receding Horizon Control
SAS	Switched Affine System

C

Curriculum Vitae

Tobias Geyer
Born on May 25, 1975 in Biberach, Germany

07/2000 – 03/2005	Doctorate at Automatic Control Laboratory, ETH Zürich (Dr. sc. ETH)
10/1999 – 03/2000	Diploma project at Bell Labs, Homldel, NJ
10/1995 – 03/2000	Undergraduate studies in Electrical Engineering, ETH Zürich (Dipl. El.-Ing. ETH)
09/1994 – 10/1995	Civil service at Red Cross Biberach
09/1982 – 06/1994	Pestalozzi Gymnasium Biberach (Abitur)

Bibliography

[AAC04] Alamir, M., S.A. Attia and C. Canudas de Wit: *Nonlinear Predictive Controller for the Simplified ABB Test Power System Stabilitzation Problem.* In *Proceedings of the World Automation Congress*, Sevilla, Spain, 2004.

[AAC05] Attia, S.A., M. Alamir and C. Canudas de Wit: *Voltage Collapse Avoidance in Power Systems: A Receding Horizon Approach.* International Journal on Intelligent Automation and Soft Computing, 2005. to appear.

[ABB] ABB Asea Brown Boveri Ltd: *Product webpage of ACS 6000.* online document. www.abb.com/motors&drives.

[ABI+01] Alur, R., C. Belta, F. Ivančić, V. Kumar, M. Mintz, G.J. Pappas, H. Rubin and J. Schug: *Hybrid Modeling and Simulation of Biomolecular Networks.* In Di Benedetto, M.D. and A. Sangiovanni Vincentelli (editors): *Hybrid Systems: Computation and Control*, volume 2034 of *Lecture Notes in Computer Science*, pages 19–33. Springer-Verlag, 2001.

[ADE+01] Alur, R., T. Dang, J. Esposito, R. Fierro, Y. Hur, F. Ivančić, V. Kumar, I. Lee, P. Mishra, G. Pappas and O. Sokolsky: *Hierarchical Hybrid Modeling of Embedded Systems.* In Henzinger, T.A. and C.M. Kirsch (editors): *Embedded Software First International Workshop. Proceedings*, volume 2211 of *Lecture Notes in Computer Science*, pages 14–31. Springer-Verlag, 2001.

[AF96] Avis, D. and K. Fukuda: *Reverse Search for Enumeration.* Discrete Applied Mathematics, 65:21–46, 1996.

[AH97] Alur, R. and T. A. Henzinger: *Modularity for Timed and Hybrid Systems.* In Mazurkiewicz, A. and J. Winkowski (editors): *Proceedings of CONCUR 97: Concurrency Theory. 8th International Conference*, volume 1243 of *Lecture Notes in Computer Science*, pages 74–88. Springer-Verlag, 1997.

[AL98] Ajjarapu, V. and B. Lee: *Bibliography on voltage stability.* IEEE Transactions on Power Systems, 13:115–125, February 1998.

[BBB+00] Balluchi, A., L. Benvenuti, M. Di Benedetto, C. Pinello and A. Sangiovanni-Vincentelli: *Automotive Engine Control and Hybrid Systems: Challenges and Opportunities.* Proceedings IEEE, 88(7):888–912, 2000.

[BBBM05] Borrelli, F., M. Baotić, A. Bemporad and M. Morari: *Dynamic Programming for Constrained Optimal Control of Discrete-Time Linear Hybrid Systems.* Automatica, 41, 2005. accepted, preprint available from http://control.ee.ethz.ch/.

[BBFH01] Borrelli, F., A. Bemporad, M. Fodor and D. Hrovat: *A Hybrid Approach to Traction Control.* In Di Benedetto, M.D. and A. Sangiovanni-Vincentelli (editors): *Hybrid Systems: Computation and Control*, volume 2034 of *Lecture Notes in Computer Science*, pages 162–174. Springer-Verlag, 2001.

[BBM00] Bemporad, A., F. Borrelli and M. Morari: *Piecewise Linear Optimal Controllers for Hybrid Systems.* In *Proceedings of the American Control Conference*, pages 1190–1194, Chicago, IL, June 2000.

[BBM03] Borrelli, F., A. Bemporad and M. Morari: *Geometric Algorithm for Multiparametric Linear Programming.* Journal of Optimization Theory and Applications, 118(2):515–540, March 2003.

[BCM03a] Baotić, M., F.J. Christophersen and M. Morari: *Infinite Time Optimal Control of Hybrid Systems with a Linear Performance Index.* In *Proceedings of the 42nd IEEE Conference on Decision and Control*, pages 3191–3196, Maui, Hawaii, USA, December 2003.

[BCM03b] Baotić, M., F.J. Christophersen and M. Morari: *A new Algorithm for Constrained Finite Time Optimal Control of Hybrid Systems with a Linear Performance Index.* In *Proceedings of the European Control Conference*, Cambridge, UK, September 2003.

[Bem02] Bemporad, A.: *An Efficient Technique for Translating Mixed Logical Dynamical Systems into Piecewise Affine Systems.* In *Proceedings of the 41st IEEE Conference on Decision and Control*, Las Vegas, NV, December 2002.

[Bem04] Bemporad, A.: *Multiparametric Nonlinear Integer Programming and Explicit Quantized Optimal Control*. Technical Report, Deptartement of Information Engineering, University of Siena, 2004.

[BF02] Bemporad, A. and C. Filippi: *Suboptimal Explicit RHC via Approximate Multiparametric Quadratic Programming*. Technical Report AUT02-07, Automatic Control Laboratory ETH Zurich, http://control.ee.ethz.ch/, 2002.

[BFT01] Bemporad, A., K. Fukuda and F.D. Torrisi: *Convexity Recognition of the Union of Polyhedra*. Computational Geometry: Theory and Applications, 18:141–154, April 2001.

[BGKH02] Bemporad, A., N. Giorgetti, I.V. Kolmanovsky and D. Hrovat: *A Hybrid Systems Approach to Modeling and Optimal Control of DISC Engines*. In *Proceedings of the 41st IEEE Conference on Decision and Control*, Las Vegas, NV, December 2002.

[BGM04] Beccuti, A.G., T. Geyer and M. Morari: *Temporal Lagrangian Decomposition of Model Predictive Control for Hybrid Systems*. In *Proceedings of the 43rd IEEE Conference on Decision and Control*, Atlantis, Bahamas, December 2004.

[BHMS84] Brayton, R.K., G.D. Hachtel, C.T. McMullen and A.L. Sangiovanni-Vincentelli: *Logic Minimization Algorithms for VLSI Synthesis*. Kluwer Academic Publishers, 1984.

[BM99a] Bemporad, A. and M. Morari: *Control of Systems Integrating Logic, Dynamics and Constraints*. Automatica, 35(3):407–427, March 1999.

[BM99b] Bemporad, A. and M. Morari: *Robust Model Predictive Control: A Survey*. In Garulli, A., A. Tesi and A. Vicino (editors): *Robustness in Identification and Control*, volume 245 of *Lecture Notes in Control and Information Sciences*, pages 207–226. Springer-Verlag, 1999.

[BM00] Bemporad, A. and D. Mignone: *MIQP.M: A Matlab Function for Solving Mixed Integer Quadratic Programs*. ETH Zurich, 2000. code available at http://control.ethz.ch/~hybrid/miqp.

[BMDP02] Bemporad, A., M. Morari, V. Dua and E.N. Pistikopoulos: *The Explicit Linear Quadratic Regulator for Constrained Systems*. Automatica, 38(1):3–20, January 2002.

[BMFN99] Brungger, A., A. Marzetta, K. Fukuda and J. Nievergelt: *The Parallel Search Bench ZRAM and its Applications*. Annals of Operations-Research, 90:45–63, 1999.

[Bor03] Borrelli, F.: *Constrained Optimal Control of Linear and Hybrid Systems*, volume 290 of *Lecture Notes in Control and Information Sciences*. Springer, 2003.

[Bro81] Brown, D.W.: *A State-Machine Synthesizer – SMS*. In *Proceedings of the 18th Design Automation Conference*, pages 301–304, June 1981.

[BT03] Baotić, M. and F.D. Torrisi: *Polycover*. Technical Report AUT03-11, Automatic Control Laboratory ETH Zurich, http://control.ee.ethz.ch/, 2003.

[Buc43] Buck, R.C.: *Partition of Space*. American Mathematical Monthly, 50:541–544, 1943.

[BV04] Boyd, S. and L. Vandenberghe: *Convex Optimization*. Cambridge University Press, 2004.

[CB99] Camacho, E.F. and C. Bordons: *Model Predictive Control*. Springer, 1999.

[CB00] Celanovic, N. and D. Boroyevich: *A Comprehensive Study of Neutral-Point Voltage Balancing Problem in Three-Level Neutral-Point-Clamped Voltage Source PWM Inverters*. IEEE Transactions on Power Electronics, 15(2):242–249, March 2000.

[Cha84] Chazelle, B.: *Convex Partitions of Polyhedra: A Lower Bound and Worst-Case Optimal Algorithm*. SIAM Journal of Computing, 13:488–507, August 1984.

[CIG95] CIGRE task force 38.02.08: *Long Term Dynamics Phase II*, 1995.

[CIG00] CIGRE Task Force: *System Protection Schemes in Power Networks*. Technical Report, 2000.

[CMT87] Clarke, D.W., C. Mohtadi and P.S. Tuffs: *Generalized Predictive Control – Part I. The Basic Algorithm*. Automatica, 23(2):137–148, 1987.

[CPST02] Casadei, D., F. Profumo, G. Serra and A. Tanni: *FOC and DTC: Two Viable Schemes for Induction Motors Torque Control*. IEEE Transactions on Power Electronics, 17(5):779–787, September 2002.

[CPV92] Calderone, L., L. Pinola and V. Varoli: *Optimal Feed-Forward Compensation for PWM DC/DC Converters with "Linear"and "Quadratic"Conversion Ratio.* IEEE Transactions on Power Electronics, 7(2):349–355, April 1992.

[CST00] Casadei, D., G. Serra and A. Tanni: *Implementation of a Direct Torque Control Algorithm for Induction Motors Based on Discrete Space Vector Modulation.* IEEE Transactions on Power Electronics, 15(4):769–777, July 2000.

[DBP02] Dua, V., N.A. Bozinis and E.N. Pistikopoulos: *A Multiparametric Programming Approach for Mixed-Integer Quadratic Engineering Problems.* Computers and Chemical Engineering, 26(4–5):715–733, 2002.

[Deo74] Deo, N.: *Graph Theory with Applications to Engineering and Computer Science.* Prentice-Hall, 1974.

[DoE82] EECS, University of California, Berkeley Departement of: *webpage of Espresso-II.* online document, 1982. http://www-cad.eecs.berkeley.edu/Software/software.html.

[DP00] Dua, V. and E.N. Pistikopoulos: *An Algorithm for the Solution of Multiparametric Mixed Integer Linear Programming Problems.* Annals of Operations-Research, pages 123–139, 2000.

[dTM02] Toit Mouton, H. du: *Natural Balancing of Three-Level Neutral-Point-Clamped PWM Inverters.* IEEE Transactions on Industrial Electronics, 49(5):1017–1025, October 2002.

[Dv01] De Schutter, B. and T. van den Boom: *On Model Predictive Control for Max-min-plus-scaling Discrete Event Systems.* Automatica, 37(7):1049–1056, 2001.

[ED02] Earl, M. G. and R. D'Andrea: *Modeling and Control of a Multi-Vehicle System using Mixed Integer Linear Programming.* In *Proceedings of the 41st IEEE Conference on Decision and Control*, Las Vegas, NV, December 2002.

[Ede87] Edelsbrunner, H.: *Algorithms in Combinatorial Geometry.* Springer-Verlag, 1987.

[ELS93] Eades, P., X. Lin and W.F. Smyth: *A Fast and Effective Heuristic for the Feedback Arc Set Problem.* Information Processing Letters, 47:319–323, 1993.

[EvM82] Erickson, R.W., S. Čuk and R.D. Middlebrook: *Large Signal Modeling and Analysis of Switching Regulators*. In *Proceedings of the IEEE Power Electronics Specialists Conference*, pages 240–250, 1982.

[FCMM02] Ferrari-Trecate, G., F.A. Cuzzola, D. Mignone and M. Morari: *Analysis of Discrete-Time Piecewise Affine and Hybrid Systems*. Automatica, 38(12):2139–2146, March 2002.

[FF02] Ferrez, J.A. 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 cdd-lib Libraries*. Technical Report, Mcgill, http://www.cs.mcgill.ca/~fukuda/download/mink/RS_TOPE020713.tar.gz, July 2002.

[FFL01] Ferrez, J.A., K. Fukuda and Th.M. Liebling: *Cuts, Zonotopes and Arrangements*. Technical Report, EPF Lausanne, Switzerland, November 2001.

[Fia83] Fiacco, A.V.: *Introduction to Sensitivity and Stability Analysis in Nonlinear Programming*. Academic Press, 1983.

[Flo95] Floudas, C.A.: *Nonlinear and Mixed-Integer Optimization*. Oxford University Press, 1995.

[FMLM03] Ferrari-Trecate, G., M. Muselli, D. Liberati and M. Morari: *A Clustering Technique for the Identification of Piecewise Affine Systems*. Automatica, 39(2):205–217, February 2003.

[FMM02] Ferrari-Trecate, G., D. Mignone and M. Morari: *Moving Horizon Estimation for Hybrid Systems*. IEEE Transactions on Automatic Control, 47(10):1663–1676, 2002.

[GLM02] Geyer, T., M. Larsson and M. Morari: *Hybrid Control of Voltage Collapse in Power Systems*. Technical Report AUT02-12, Automatic Control Laboratory ETH Zurich, http://control.ee.ethz.ch/, 2002.

[GLM03] Geyer, T., M. Larsson and M. Morari: *Hybrid Emergency Voltage Control in Power Systems*. In *Proceedings of the European Control Conference*, Cambridge, UK, September 2003.

[GLPM03] Grieder, P., M. Lüthi, P. Parrilo and M. Morari: *Stability & Feasibility of Constrained Receding Horizon Control*. In *Proceedings of the European Control Conference*, Cambridge, UK, September 2003.

[GMSV94] Garofalo, F., P. Marino, S. Scala and F. Vasca: *Control of DC/DC Converters with Linear Optimal Feedback and Nonlinear Feedforward*. IEEE Transactions on Power Electronics, 9(6):607–615, November 1994.

[GP05] Geyer, T. and G. Papafotiou: *Direct Torque Control for Induction Motor Drives: A Model Predictive Control Approach based on Feasibility*. In Morari, M. and L. Thiele (editors): *Hybrid Systems: Computation and Control*, volume 3414 of *Lecture Notes in Computer Science*, pages 274–290. Springer-Verlag, 2005.

[GPM89] Garcia, C.E., D.M. Prett and M. Morari: *Model Predictive Control: Theory and Practice – a Survey*. Automatica, 25(3):335–348, 1989.

[GPM04a] Geyer, T., G. Papafotiou and M. Morari: *Direct Torque Control for Induction Motor Drives: A Model Predictive Control Approach based on Feasibility*. Technical Report AUT04-09, Automatic Control Laboratory ETH Zurich, http://control.ee.ethz.ch/, 2004.

[GPM04b] Geyer, T., G. Papafotiou and M. Morari: *Model Predictive Direct Torque Control Minimizing the Switching Frequency*, 2004. unpublished.

[GPM04c] Geyer, T., G. Papafotiou and M. Morari: *On the Optimal Control of Switch-Mode DC-DC Converters*. In Alur, R. and G. Pappas (editors): *Hybrid Systems: Computation and Control*, volume 2993 of *Lecture Notes in Computer Science*, pages 342–356. Springer-Verlag, 2004.

[GPM04d] Geyer, T., G. Papafotiou and M. Morari: *Verfahren zum Betrieb einer rotierenden elektrischen Maschine*. European patent application pending, 2004.

[GPM05] Geyer, T., G. Papafotiou and M. Morari: *Global Optimal Control of Switch-Mode DC-DC Converters*, 2005. submitted to journal.

[GTM03a] Geyer, T., F.D. Torrisi and M. Morari: *Efficient Mode Enumeration of Compositional Hybrid Systems*. In Pnueli, A. and O. Maler (editors): *Hybrid Systems: Computation and Control*, volume 2623 of *Lecture Notes in Computer Science*, pages 216–232. Springer-Verlag, 2003.

[GTM03b] Geyer, T., F.D. Torrisi and M. Morari: *Efficient Mode Enumeration of Compositional Hybrid Models*. Technical Report AUT03-01, Automatic Control Laboratory ETH Zurich, http://control.ee.ethz.ch/, 2003.

[GTM04] Geyer, T., F.D. Torrisi and M. Morari: *Optimal Complexity Reduction of Piecewise Affine Models Based on Hyperplane Arrangements*. In *Proceedings of the American Control Conference*, pages 1190–1195, Boston, MA, June 2004.

[HBOL01] Hespanha, J. P., S. Bohacek, K. Obraczka and J. Lee: *Hybrid Modeling of TCP Congestion Control*. In Di Benedetto, M.D. and A. Sangiovanni-Vincentelli (editors): *Hybrid Systems: Computation and Control*, volume 2034 of *Lecture Notes in Computer Science*, pages 291–304. Springer-Verlag, 2001.

[HCO74] Hong, S.J., R.G. Cain and D.L. Ostapko: *MINI: A Heuristic Approach for Logic Minimization*. IBM Journal of Research and Development, 18:443–458, 1974.

[HDJ92] Hamill, D.C., J.H.B. Deane and D. Jefferies: *Modeling of Chaotic DC-DC Converters by Iterated Nonlinear Mappings*. IEEE Transactions on Power Electronics, 7(1):25–36, January 1992.

[Hen00] Henzinger, T.A.: *The Theory of Hybrid Automata*. In Inan, M.K. and R.P. Kurshan (editors): *Verification of Digital and Hybrid Systems*, volume 170 of *NATO ASI Series F: Computer and Systems Sciences*. Springer-Verlag, 2000.

[HG04] Hiskens, I.A. and B. Gong: *Voltage Stability Enhancement via Model Predictive Control of Load*. In *Proceedings of the Symposium on Bulk Power System Dynamics and Control VI*, Cortina d'Ampezzo, Italy, 2004.

[HMP01] Henzinger, T.A., M. Minea and V. Prabhu: *Assume-Guarantee Reasoning for Hierarchical Hybrid Systems*. In Di Benedetto, M.D. and A. Sangiovanni-Vincentelli (editors): *Hybrid Systems: Computation and Control*, volume 2034, pages 275–290. Springer-Verlag, March 2001.

[HP00] Hiskens, I.A. and M.A. Pai: *Trajectory Sensitivity Analysis of Hybrid Systems*. IEEE Transactions on Circuits and Systems-I: Fundamental Theory and Applications, 47(2):204–220, February 2000.

[HP02] Hiskens, I.A. and M.A. Pai: *Power System Applications of Trajectory Sensitivities*. In *Proceedings of the IEEE Power Engineering Society Winter Meeting*, volume 2, pages 1200–1205, 2002.

[HSB01] Heemels, W.P.M.H., B. De Schutter and A. Bemporad: *Equivalence of Hybrid Dynamical Models*. Automatica, 37(7):1085–1091, July 2001.

[ILO02] ILOG, Inc.: *CPLEX 8.0 User Manual*. Gentilly Cedex, France, 2002.

[Jaz70] Jazwinski, A. H.: *Stochastic Processes and Filtering Theory*. Academic Press, 1970.

[Joh00] Johansson, K.H.: *Hybrid Systems: Modeling, Analysis and Control – Composition of Hybrid Automata*. Lecture notes of the class EECS 291e, Lecture 5, 2000.

[Joh02] Johansen, T.A.: *On Multi-Parametric Nonlinear Programming and Explicit Nonlinear Model Predictive Control*. In *Proceedings of the 41st IEEE Conference on Decision and Control*, pages 2768–2773, Las Vegas, NV, December 2002.

[Joh04] Johansen, T.A.: *Approximate Explicit Receding Horizon Control of Constrained Nonlinear Systems*. Automatica, 40(2):293–300, February 2004.

[Jul00] Julian, P.: *A Toolbox for the Piecewise Linear Approximation of Multidimensional Functions*, July 2000.

[Kam01] Kamau, S.I.: *Different Approaches to Modelling of Hybrid Systems*. Forschungsbericht 2001.11, TU Hamburg-Harburg, April 2001.

[Kar53] Karnaugh, M.: *A Map Method for Synthesis of Combinational Logic Circuits*. AIEE Transactions on Communications and Electronics, 72:593–599, November 1953.

[Kat94] Katz, R.H.: *Contemporary Logic Design*. Benjamin/Cummings Publishing Company, 1994.

[KGBM04] Kvasnica, M., P. Grieder, M. Baotić and M. Morari: *Multi Parametric Toolbox (MPT)*. In Alur, R. and G. Pappas (editors): *Hybrid Systems: Computation and Control*, volume 2993 of *Lecture Notes in Computer Science*, pages 448–462. Springer-Verlag, 2004. http://control.ee.ethz.ch/~mpt.

[KH94] Karlsson, D. and D.J. Hill: *Modelling and Identification of Nonlinear Dynamic Loads in Power Systems*. IEEE Transactions on Power Systems, 9:157–166, February 1994.

[KL00] Kennel, R. and A. Linder: *Predictive Control of Inverter Supplied Electrical Drives*. In *Proceedings of the IEEE Power Electronics Specialists Conference*, volume 2, pages 761–766, Galway, Ireland, 2000.

[KLL01] Kennel, R., A. Linder and M. Linke: *Generalized Predictive Control (GPC) – Ready for Use in Drive Applications?* In *Proceedings of the IEEE Power Electronics Specialists Conference*, volume 4, pages 1839–1844, Vancouver, Canada, 2001.

[KMM00] Kostakis, G.T., S.N. Manias and N.I. Margaris: *A Generalized Method for Calculating the RMS Values of Switching Power Converters.* IEEE Transactions on Power Electronics, 15(4):616–625, July 2000.

[Kra86] Krause, P.C.: *Analysis of Electric Machinery.* McGraw-Hill, NY, 1986.

[KS99] Kugi, A. and K. Schlacher: *Nonlinear H_∞-Controller Design for a DC-to-DC Power Converter.* IEEE Transactions on Control Systems Technology, 7(2):230–237, December 1999.

[KSV91] Kassakian, J.G., M.F. Schlecht and G.C. Verghese: *Principles of Power Electronics.* Addison-Wesley, 1991.

[Kun94] Kundur, P.: *Power System Stability and Control.* Power System Engineering Series. McGraw Hill Inc., 1994.

[Lar02] Larsson, M.: *The ABB Power Transmission Test Case.* www.dii.unisi.it/~hybrid/cc/, February 2002.

[Lar03] Larsson, M.: *A Simple Test System Illustrating Load-Voltage Dynamics in Power Systems.* www.dii.unisi.it/~hybrid/cc/, March 2003.

[LHO02] Larsson, M., D.J. Hill and G. Olsson: *Emergency Voltage Control using Search and Predictive Control.* International Journal of Electrical Power and Energy Systems, 24(2):121–130, 2002.

[Lin03] Lincoln, B.: *Dynamic Programming and Time-Varying Delay Systems.* PhD thesis, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden, 2003.

[LK03] Larsson, M. and D. Karlsson: *Coordinated System Protection Scheme against Voltage Collapse using Heuristic Search and Predictive Control.* IEEE Transactions on Power Systems, 18:1001–1006, August 2003.

[LK04] Lazar, M. and R. De Keyser: *Non-Linear Predictive Control of a DC-to-DC Converter.* In *Symposium on Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM)*, Capri, Italy, 2004.

[LRB03] Larsson, M., C. Rehtanz and J. Bertsch: *Monitoring and Operation of Transmission Corridors*. In *Proceedings of the IEEE Power Tech Conference*, volume 3, pages 8–15, Bologna, Italy, 2003.

[LSV01] Lynch, N., R. Segala and F. Vaandrager: *Hybrid I/O Automata Revisited*. In Di Benedetto, M.D. and A. Sangiovanni-Vincentelli (editors): *Hybrid Systems: Computation and Control*, volume 2034 of *Lecture Notes in Computer Science*, pages 403–417. Springer-Verlag, Rome, Italy, 2001.

[LTL91] Leung, F.H.F., P.K.S. Tam and C.K. Li: *The Control of Switching DC-DC Converters – A General LQR Problem*. IEEE Transactions on Industrial Electronics, 38(1):65–71, February 1991.

[LTL93] Leung, F.H.F., P.K.S. Tam and C.K. Li: *An Improved LQR-based Controller for Switching DC-DC Converters*. IEEE Transactions on Industrial Electronics, 40(5):521–528, October 1993.

[Mac02] Maciejowski, J.M.: *Predictive Control*. Prentice Hall, 2002.

[May01] Mayne, D.Q.: *Control of Constrained Dynamic Systems*. European Journal of Control, 7(2–3):87–99, 2001.

[MBB97] Machowski, J., J.W. Bialek and J.R. Bumby: *Power System Dynamics and Stability*. Interscience Publishers John Wiley & Sons Inc., 1997.

[MBB03] Morari, M., M. Baotić and F. Borrelli: *Hybrid Systems Modeling and Control*. European Journal of Control, 9(2–3):177–189, 2003.

[McC56] McCluskey, E.J.: *Minimization of Boolean Functions*. Bell Systems Technical Journal, 35:1417–1444, November 1956.

[Mig02] Mignone, D.: *Control and Estimation of Hybrid System with Mathematical Optimization*. PhD thesis, Automatic Control Laboratory ETH Zurich, Zurich, 2002.

[MLv00] Moors, C., D. Lefebvre and T. van Cutsem: *Design of Load Shedding Schemes against Voltage Instability*. In *Proceedings of the IEEE Power Engineering Society Winter Meeting*, volume 2, pages 1495–1500, 2000.

[MM01] Mitchell, D. and B. Mammano: *Designing Stable Control Loops*. Unitrode Design Seminars, 2001. http://focus.ti.com/lit/ml/slup173/slup173.pdf.

[MRMC02] Martins, C.A., X. Roboam, T. Meynard and A. Carvalho: *Switching Frequency Imposition and Ripple Reduction in DTC Drives by Using a Multilevel Converter*. IEEE Transactions on Power Electronics, 17(2):286–297, March 2002.

[MRRS00] Mayne, D.Q., 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.

[MUR89] Mohan, N., T.M. Undeland and W.P. Robbins: *Power Electronics: Converters, Applications and Design*. Wiley, 1989.

[Mv76] Middlebrook, R.D. and S. Čuk: *A General Unified Approach to Modeling Switching Power Converter Stages*. In *Proceedings of the IEEE Power Electronics Specialists Conference*, pages 18–34, 1976.

[PA01] Purcell, A. and P.P. Acarnley: *Enhanced Inverter Switching for Fast Response Direct Torque Control*. IEEE Transactions on Power Electronics, 16(3):382–389, May 2001.

[PB97] Park, T. and P.I. Barton: *Implicit Model Checking of Logic-Based Control Systems*. Aiche Journal, 43(9):2246–2260, 1997.

[PGM04a] Papafotiou, G., T. Geyer and M. Morari: *Hybrid Modelling and Optimal Control of Switch-mode DC-DC Converters*. In *IEEE Workshop on Computers in Power Electronics (COMPEL)*, Champaing, IL, USA, 2004.

[PGM04b] Papafotiou, G., T. Geyer and M. Morari: *Optimal Direct Torque Control of Three-Phase Symmetric Induction Motors*. In *Proceedings of the 43rd IEEE Conference on Decision and Control*, Atlantis, Bahamas, December 2004.

[PGM04c] Papafotiou, G., T. Geyer and M. Morari: *Optimal Direct Torque Control of Three-Phase Symmetric Induction Motors*, 2004. submitted to journal.

[PTL94] Pohjalainen, P., P. Tiitinen and J. Lulu: *The Next Generation Motor Control Method - Direct Torque Control, DTC*. In *Proceedings of the European Power Electronics Chapter Symposium*, volume 1, pages 115–120, Lausanne, Switzerland, 1994.

[QB03] Qin, S.J. and T.A. Badgwell: *A Survey of Industrial Model Predictive Control Technology*. Control Engineering Practice, 11:733–764, 2003.

[Qui55] Quine, W.V.: *A Way to Simplify Truth Functions*. American Mathematical Monthly, 62:627–631, November 1955.

[Raw00] Rawlings, J.B.: *Tutorial Overview of Model Predictive Control*. IEEE Control Systems Magazine, 20(3):38–52, June 2000.

[Reh01] Rehtanz, C.: *Wide Area Protection and Online Stability Assessment based on Phasor Measurements*. In *Proceedings of the Bulk Power System Dynamics and Control*, Onomichi, Japan, 2001.

[RG91] Raman, R. and I.E. Grossmann: *Relation between MILP modeling and logical inference for chemical process synthesis*. Computers and Chemical Engineering, 15(2):73–84, 1991.

[RL99] Rashid, S. and J. Lygeros: *Hybrid systems: Modeling, Analysis and Control – Open Hybrid Automata and Composition*. UCB/ERL M99/34, Lecture notes of the class EECS 291e, Lecture 8, 1999.

[RLSLM04] Retif, J.M., X. Lin-Shi, A.M. Llor and F. Morand: *New Hybrid Direct-Torque Control for a Winding Rotor Synchronous Machine*. In *Proceedings of the IEEE Power Electronics Specialists Conference*, volume 1, pages 1438–1442, Aachen, Germany, 2004.

[RPS+04] Rodriguez, J., J. Pontt, C. Silva, P. Cortes, U. Amman and S. Rees: *Predictive Direct Torque Control of an Induction Machine*. In *Proceedings of the 11th IEEE Power Electronics and Motion Control Conference*, Riga, Latvia, 2004.

[SEK03] Senesky, M., G. Eirea and T.J. Koo: *Hybrid Modelling and Control of Power Electronics*. In Pnueli, A. and O. Maler (editors): *Hybrid Systems: Computation and Control*, volume 2623 of *Lecture Notes in Computer Science*, pages 450–465. Springer-Verlag, 2003.

[Sol04] Solyom, S.: *Control of Systems with Limited Capacity*. PhD thesis, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden, 2004.

[Son81] Sontag, E.D.: *Nonlinear Regulation: The Piecewise Linear Approach*. IEEE Transactions on Automatic Control, 26(2):346–358, April 1981.

[Son96] Sontag, E.D.: *Interconnected Automata and Linear Systems: A Theoretical Framework in Discrete-Time*. In Alur, R., T.A. Henzinger and E.D.

Sontag (editors): *Hybrid Systems III Verification and Control*, volume 1066 of *Lecture Notes in Computer Science*, pages 436–448. Springer-Verlag, 1996.

[SR91] Sira-Ramírez, H.: *Nonlinear P-I Controller Design for Switchmode DC-to-DC Power Converters.* IEEE Transactions on Circuits and Systems-I: Fundamental Theory and Applications, 38(4):410–417, April 1991.

[SR03] Sira-Ramírez, H.: *On the Generalized PI Sliding Mode Control of DC-to-DC Power Converters: A Tutorial.* International Journal of Control, 76(9–10):1018–1033, 2003.

[SS98] Schaft, A.J. van der and J.M. Schumacher: *Complementarity Modelling of Hybrid Systems.* IEEE Transactions on Automatic Control, 43:483–490, 1998.

[Tay94] Taylor, C.W.: *Power System Voltage Stability.* McGraw Hill Inc., 1994.

[TB01] Torrisi, F.D. and A. Bemporad: *Discrete-Time Hybrid Modeling and Verification.* In *Proceedings of the 40th IEEE Conference on Decision and Control*, pages 2899–2904, Orlando, Florida, December 2001.

[TB04] Torrisi, F.D. and A. Bemporad: *HYSDEL — A Tool for Generating Computational Hybrid Models for Analysis and Synthesis Problems.* IEEE Transactions on Control Systems Technology, 12(2):235–249, March 2004.

[TE97] Taylor, C.W. and D.C. Erickson: *Recording and Analyzing the July 2 Cascading Outage.* IEEE Computer Applications in Power, 10:26–30, January 1997.

[Til01] Tiller, M.: *Introduction to Physical Modelling with Modelica.* Kluwer Academic Publishers, 2001.

[TJB01] Tøndel, P., T.A. Johansen and A. Bemporad: *An Algorithm for Multi-Parametric Quadratic Programming and Explicit MPC Solutions.* In *Proceedings of the 40th IEEE Conference on Decision and Control*, Orlando, Florida, December 2001.

[TJB03] Tøndel, P., T.A. Johansen and A. Bemporad: *Evaluation of Piecewise Affine Control via Binary Search Tree.* Automatica, 39(5):945–950, May 2003.

[TN86] Takahashi, I. and T. Noguchi: *A New Quick Response and High Efficiency Control Strategy for the Induction Motor.* IEEE Transactions on Industry Applications, 22(2):820–827, September/October 1986.

[TO89] Takahashi, I. and Y. Ohmori: *High-Performance Direct Torque Control of an Induction Motor*. IEEE Transactions on Industry Applications, 25(2):257–264, March/April 1989.

[US 04] US - Canada Power System Outage Task Force: *Final Report on the August 14, 2003 Blackout in the United States and Canada: Causes and Recommendations*, April 2004.

[Vei52] Veitch, E.W.: *A Chart Method for Simplifying Boolean Functions*. In *Proceedings of the Association for Computing Machinery*, pages 127–133, May 1952.

[vML02] van Cutsem, T., C. Moors and D. Lefebvre: *Design of Load Shedding Schemes against Voltage Instability using Combinatorial Optimization*. In *Proceedings of the IEEE Power Engineering Society Winter Meeting*, volume 2, pages 848–853, 2002.

[vV98] van Cutsem, T. and C. Vournas: *Voltage Stability of Electric Power Systems*. Power Electronics and Power Systems Series. Kluwer Academic Publishers, 1998.

[ZA03] Zima, M. and G. Andersson: *Stability Assessment and Emergency Control Method using Trajectory Sensitivities*. In *Proceedings of the IEEE Power Tech Conference*, volume 2, pages 7–13, Bologna, Italy, 2003.