

سایت اختصاصی مهندسی کنترل

 <https://controlengineers.ir>

 @controlengineers



HANDBOOK OF HYBRID SYSTEMS CONTROL

Setting out core theory and reviewing a range of new methods, theoretical problems, and applications, this handbook shows how hybrid dynamical systems can be modeled and understood. Sixty expert authors involved in the recent research activities and industrial application studies provide practical insights on topics ranging from the theoretical investigations over computer-aided design and verification tools to applications in several domains.

Structured into three parts, the book opens with a thorough introduction to hybrid systems theory, illustrating new dynamical phenomena through numerous examples and showing novel modeling, analysis, and design techniques that have been elaborated recently for this new system class. Part II then provides a survey of key tools and tool integration activities. Finally, Part III is dedicated to applications, implementation issues, and system integration, considering applications to energy management, the process industry, automotive systems, and digital networks.

Three running examples are referred to throughout the book, together with numerous illustrations, helping both researchers and industry professionals to understand complex theory, recognize problems, and find appropriate solutions.

JAN LUNZE is Head of the Institute of Automation and Computer Control at Ruhr-Universität Bochum, where he teaches systems and control theory. His research interests are in hybrid and discrete-event systems, in fault diagnosis, and in control theory of networked systems in which he currently coordinates a Priority Program of the German Science Foundation (Deutsche Forschungsgemeinschaft). He is the author or co-author of numerous technical papers and several monographs and textbooks.

FRANÇOISE LAMNABHI-LAGARRIGUE is Director of Research at the CNRS within the Signals and Systems Laboratory (L2S). She is an Associate Editor of the *International Journal of Control*, Scientific Manager of the HYCON Network of Excellence, and President of the European Embedded Control Institute (EECI). In 2008, she won the Michel Monpetit prize of the French Academy of Science.

controlengineers.ir

Handbook of Hybrid Systems Control

Theory, Tools, Applications

Edited by

JAN LUNZE

Ruhr-Universität Bochum, Germany

FRANÇOISE LAMNABHI-LAGARRIGUE

Centre National de la Recherche Scientifique (CNRS), France

controlengineers.ir



CAMBRIDGE
UNIVERSITY PRESS

CAMBRIDGE UNIVERSITY PRESS
Cambridge, New York, Melbourne, Madrid, Cape Town, Singapore,
São Paulo, Delhi, Dubai, Tokyo

Cambridge University Press
The Edinburgh Building, Cambridge CB2 8RU, UK

Published in the United States of America by Cambridge University Press, New York

www.cambridge.org

Information on this title: www.cambridge.org/9780521765053

© Cambridge University Press 2009

This publication is in copyright. Subject to statutory exception and to the provision of relevant collective licensing agreements, no reproduction of any part may take place without the written permission of Cambridge University Press.

First published in print format 2009

ISBN-13 978-0-511-64178-7 eBook (NetLibrary)

ISBN-13 978-0-521-76505-3 Hardback

Cambridge University Press has no responsibility for the persistence or accuracy of urls for external or third-party internet websites referred to in this publication, and does not guarantee that any content on such websites is, or will remain, accurate or appropriate.

Contents

List of contributors	page vii
Preface	xiii
Notation	xvi
<hr/>	
Part I Theory	
<hr/>	
1 Introduction to hybrid systems	3
<i>W. P. M. H. Heemels, D. Lehmann, J. Lunze, and B. De Schutter</i>	
2 Survey of modeling, analysis, and control of hybrid systems	31
<i>B. De Schutter, W. P. M. H. Heemels, J. Lunze, and C. Prieur</i>	
3 Hybrid automata	57
<i>S. Kowalewski, M. Garavello, H. Guéguen, G. Herberich, R. Langerak, B. Piccoli, J. W. Polderman, and C. Weise</i>	
4 Switched and piecewise affine systems	87
<i>J. Daafouz, M. D. Di Benedetto, V. D. Blondel, G. Ferrari-Trecate, L. Hetel, M. Johansson, A. L. Juloski, S. Paoletti, G. Pola, E. De Santis, and R. Vidal</i>	
5 Further switched systems	139
<i>A. Bemporad, M. K. Çamlibel, W. P. M. H. Heemels, A. J. van der Schaft, J. M. Schumacher, and B. De Schutter</i>	
6 Hybrid systems: quantization and abstraction	193
<i>J. Lunze, A. Bicchi, T. Moor, L. Palopoli, B. Picasso, J. Raisch, and A. Schild</i>	
7 Stochastic hybrid systems	249
<i>J. Lygeros and M. Prandini</i>	

Part II Tools

- 8 Overview of tools development and open problems** 279
D. A. van Beek and S. Engell
- 9 Verification tools for linear hybrid automata** 285
G. Frehse
- 10 Tools for modeling, simulation, control, and verification of piecewise affine systems** 297
A. Bemporad, S. Di Cairano, G. Ferrari-Trecate, M. Kvasnica, M. Morari, and S. Paoletti
- 11 Modeling, simulation, and optimization environments** 325
C. Sonntag
- 12 Interchange formats and tool integration** 361
D. A. van Beek, M. A. Reniers, J. E. Rooda, and R. R. H. Schiffelers

Part III Applications

- 13 Energy management** 377
M. Morari, A. G. Beccuti, S. Mariéthoz, and G. Papafotiou
- 14 Industrial controls** 405
S. Engell, S. Lohmann, T. Moor, C. de Prada, J. Raisch, D. Sarabia, and C. Sonntag
- 15 Automotive control** 439
L. Benvenuti, A. Balluchi, A. Bemporad, S. Di Cairano, B. Johansson, R. Johansson, A. Sangiovanni Vincentelli, and P. Tunestål
- 16 Networked control** 471
M. D. Di Benedetto, A. Bicchi, A. D'Innocenzo, K. H. Johansson, A. Robertsson, F. Santucci, U. Tiberi, and A. Tzes
- 17 Solar air conditioning – a benchmark for hybrid systems control** 501
E. F. Camacho and D. Zambrano
- References** 511
- Index** 553

Contributors

Andrea Balluchi

DANA ITALIA S.p.A.,
Off-Highway Products Group,
Zona Industriale Lifano,
38062 Arco (TN), Italy

Andrea Giovanni Beccuti

ETH Zurich, Automatic Control
Laboratory, Physikstrasse 3,
8092 Zurich, Switzerland

Bert van Beek

Eindhoven University of Technology,
Mechanical Engineering, Systems
Engineering,
PO Box 513, 5600 MB Eindhoven,
The Netherlands

Alberto Bemporad

Dipartimento di Ingegneria dell'
Informazione, Università di Siena,
Via Roma 56, 53100 Siena, Italy

Maria Domenica Di Benedetto

Department of Electrical and Informa-
tion Engineering, University of
L'Aquila, Centre of Excellence DEWS,
Poggio di Roio, 67040 L'Aquila, Italy

Luca Benvenuti

Università degli Studi di Roma "La
Sapienza," Dipartimento di Informatica
e Sistemistica "Antonio Ruberti,"
Via Ariosto 25, 00185 Roma, Italy

Antonio Bicchi

Dip. Sistemi Elettrici e Automazione
and Centro Interdipartimentale
"E. Piaggio," Università di Pisa,
Via Diotisalvi 2, 56100 Pisa, Italy

Vincent Blondel

Department of Mathematical Engineer-
ing, Université catholique de Louvain,
Avenue Georges Lemaître, 4,
B-1348 Louvain-la-Neuve, Belgium

Stefano Di Cairano

Dipartimento di Ingegneria dell'
Informazione Università di Siena,
Via Roma 56, 53100 Siena, Italy

Eduardo F. Camacho

Dpto. Ingeniería de Sistemas y
Automática, Escuela Superior de
Ingenieros, Camino de los Descubrim-
ientos s/n, 41092-Sevilla, Spain

Kanat Çamlıbel

University of Groningen, Institute of
Mathematics and Computer Science,
P.O. Box 407, 9700 AK Groningen,
The Netherlands

Jamal Daafouz

Institut National Polytechnique de Lorraine, Centre de Recherche en Automatique de Nancy, 2, avenue de la forêt de Haye, 54500 Vandoeuvre-lès-Nancy, France

Sebastian Engell

Technische Universität Dortmund, Bio- und Chemieingenieurwesen, Lehrstuhl für Systemdynamik und Prozessführung, 44221 Dortmund, Germany

Giancarlo Ferrari-Trecate

Dipartimento di Informatica e Sistemistica, Università degli Studi di Pavia, Via Ferrata 1, 27100 Pavia, Italy

Goran Frehse

Verimag, UMR Université Joseph Fourier/CNRS/INPG Verimag Centre Equation, 2, ave de Vignate 38610 Gieres, France

Mauro Garavello

Dipartimento di Scienze e Tecnologie Avanzate, Università del Piemonte Orientale "A. Avogadro," via Bellini 25/G, 15100 Alessandria, Italy

Hervé Guéguen

SUPELEC, CS 47601, 35576 Cesson-Sevigne cedex, France

Maurice Heemels

Eindhoven University of Technology, Mechanical Engineering, Control Systems Technology, PO Box 513, 5600 MB Eindhoven, The Netherlands

Gerlind Herberich

RWTH Aachen, Lehrstuhl für Informatik 11 Ahornstr. 55, 52074 Aachen, Germany

Laurentiu Hetel

Institut National Polytechnique de Lorraine, Centre de Recherche en Automatique de Nancy, 2, avenue de la forêt de Haye, 54500 Vandoeuvre-lès-Nancy, France

Alessandro D’Innocenzo

Department of Electrical and Information Engineering, University of L’Aquila, Centre of Excellence DEWS, Poggio di Roio, 67040 L’Aquila, Italy

Bengt Johansson

Dept. of Energy Sciences, Lund University, S-221 00 Lund, Sweden

Karl Henrik Johansson

School of Electrical Engineering, Automatic Control, Royal Institute of Technology, Osquidas väg 10, SE-100 44 Stockholm, Sweden

Mikael Johansson

Department of Signals, Systems and Sensors, Royal Institute of Technology SE-100 44 Stockholm, Sweden

Rolf Johansson

Dept. Automatic Control, Lund Institute of Technology, Lund University, PO Box 118, S-221 00 Lund, Sweden

Aleksandar Juloski

Siemens AG, Corporate Technology, Gunter-Scharowsky-Str. 1, 91058 Erlangen, Germany

Stefan Kowalewski

RWTH Aachen, Lehrstuhl für Informatik 11 Ahornstr. 55, 52074 Aachen, Germany

Michal Kvasnica

Slovak University of Technology,
 Radlinskeho 9, 81237 Bratislava,
 Slovakia

Françoise Lamnabhi-Lagarrigue

Laboratoire des Signaux
 et Systèmes, SUPELEC,
 3 rue Joliot Curie,
 91192 Gif-sur-Yvette, France

Rom Langerak

University of Twente, Department of
 Computer Science,
 P.O. Box 217, 7500 AE Enschede,
 The Netherlands

Daniel Lehmann

Ruhr-Universität Bochum,
 Lehrstuhl für Automatisierungstechnik
 und Prozessinformatik,
 44780 Bochum, Germany

Sven Lohmann

Technische Universität Dortmund,
 Bio- und Chemieingenieurwesen,
 Lehrstuhl für Systemdynamik und
 Prozessführung,
 44221 Dortmund, Germany

Jan Lunze

Ruhr-Universität Bochum,
 Lehrstuhl für Automatisierungstechnik
 und Prozessinformatik,
 44780 Bochum, Germany

John Lygeros

ETH Zurich, Automatic Control
 Laboratory, Physikstrasse 3,
 8092 Zurich, Switzerland

Sébastien Mariéthoz

ETH Zurich, Automatic Control
 Laboratory, Physikstrasse 3,
 8092 Zurich, Switzerland

Thomas Moor

Lehrstuhl für Regelungstechnik,
 Friedrich-Alexander Universität,
 D-91058 Erlangen, Germany

Manfred Morari

ETH Zurich, Automatic Control
 Laboratory, Physikstrasse 3,
 8092 Zurich, Switzerland

Luigi Palopoli

Dipartimento di Ingegneria e Scienza
 dell'Informazione, Università di
 Trento, Via Sommarive 14,
 38100 Povo (TN), Italy

Simone Paoletti

Dipartimento di Ingegneria dell'
 Informazione, Università di Siena,
 Via Roma 56, 53100 Siena, Italy

Georgios Papafotiou

ETH Zurich, Automatic Control
 Laboratory, Physikstrasse 3,
 8092 Zurich, Switzerland

Bruno Picasso

Centro Interdipartimentale "E. Piaggio,"
 Università di Pisa and Dipartimento
 di Elettronica e Informazione,
 Politecnico di Milano,
 Via Ponzio 34/5, 20133 Milano, Italy

Benedetto Piccoli

Istituto per le Applicazioni del
 Calcolo "Mauro Picone", Viale del
 Policlinico 137, I-00161 Roma, Italy

Giordano Pola

Department of Electrical and Informa-
 tion Engineering, University of
 L'Aquila, Centre of Excellence DEWS,
 Poggio di Roio, 67040 L'Aquila, Italy

Jan Willem Polderman

University of Twente, Department
 of Applied Mathematics,
 P.O. Box 217, 7500 AE Enschede,
 The Netherlands

Cesar de Prada

Departamento de Ingenieria de Sistemas y Automatica, Universidad de Valladolid, c/Real de Burgos s/n. 47011 Valladolid, Spain

Maria Prandini

Dipartimento di Elettronica e Informazione, Politecnico di Milano, Piazza Leonardo da Vinci 32, 20133 Milano, Italy

Christophe Prieur

LAAS-CNRS, University of Toulouse, 7, avenue du Colonel Roche, 31077 Toulouse, France

Jörg Raisch

Technische Universität Berlin, Fachgebiet Regelungssysteme, Einsteinufer 17, D-10587 Berlin, and Max-Planck-Institut für Dynamik komplexer technischer Systeme, Systems and Control Theory Group

Michel Reniers

Eindhoven University of Technology, Computer Science, Design and Analysis of Systems, PO Box 513, 5600 MB Eindhoven, The Netherlands

Anders Robertsson

Dept. Automatic Control Lund Institute of Technology Lund University, PO Box 118, SE-221 00 Lund, Sweden

Jacobus Rooda

Eindhoven University of Technology, Mechanical Engineering, Systems Engineering, PO Box 513, 5600 MB Eindhoven, The Netherlands

Alberto Sangiovanni Vincentelli

University of California at Berkeley, Berkeley, CA, USA

Elena De Santis

Department of Electrical and Information Engineering, University of L'Aquila, Centre of Excellence DEWS, Poggio di Roio, 67040 L'Aquila, Italy

Fortunato Santucci

Department of Electrical and Information Engineering, University of L'Aquila, Centre of Excellence DEWS, Poggio di Roio, 67040 L'Aquila, Italy

Daniel Sarabia

Departamento de Ingenieria de Sistemas y Automatica, Universidad de Valladolid, c/Real de Burgos s/n. 47011 Valladolid, Spain

Arjan van der Schaft

University of Groningen, Institute of Mathematics and Computer Science, P.O. Box 407, 9700 AK Groningen, The Netherlands

Ramon Schiffelers

Eindhoven University of Technology, Mechanical Engineering, Systems Engineering, PO Box 513, 5600 MB Eindhoven, The Netherlands

Axel Schild

Ruhr-Universität Bochum, Lehrstuhl für Automatisierungstechnik und Prozessinformatik, 44780 Bochum, Germany

Hans Schumacher

Department of Econometrics and Operations Research, Tilburg University, P.O. Box 90153, 5000 LE Tilburg, The Netherlands

Bart De Schutter

Delft Center for Systems and Control and Marine and Transport Technology, Delft University of Technology, Mekelweg 2, 2628 CD Delft, The Netherlands

Christian Sonntag

Technische Universität Dortmund, Bio- und Chemieingenieurwesen, Lehrstuhl für Systemdynamik und Prozessführung, 44221 Dortmund, Germany

Ubaldo Tiberi

Department of Electrical and Information Engineering, University of L'Aquila, Centre of Excellence DEWS, Poggio di Roio, 67040 L'Aquila, Italy

Per Tunestål

Dept. of Energy Sciences, Lund University, S-221 00 Lund, Sweden

Anthony Tzes

Department of Electrical and Computer Engineering, University of Patras, University campus, 26504 Rio, Patras, Greece

Rene Vidal

Center for Imaging Science, Department of Biomedical Engineering, Johns Hopkins University, 3400 N. Charles St., Baltimore MD 21218, USA

Carsten Weise

RWTH Aachen
Lehrstuhl für Informatik 11
Ahornstr. 55, 52074 Aachen, Germany

Darine Zambrano

Dpto. Ingeniería de Sistemas y Automática, Escuela Superior de Ingenieros, Camino de los Descubrimientos s/n, 41092-Sevilla, Spain

controlengineers.ir

Preface

Hybrid systems are dynamical systems that consist of components with continuous and discrete behavior. Modeling, analysis, and design of such systems raise severe methodological questions, because they necessitate the combination of continuous-variable system descriptions like differential and difference equations with discrete-event models like automata or Petri nets. Consequently, hybrid systems methodology is based on the principles and results of the theories of continuous and discrete systems, which, until recently, have been elaborated separately, with contributions coming from different disciplines, such as control theory, computer science, and mathematics.

This handbook reviews the new phenomena and theoretical problems brought about by the combination of continuous and discrete dynamics and surveys the main approaches, methods, and results that have been obtained during the last decade of research in this field. It is structured into three main parts:

- *Part I: Modeling, analysis, and control design methods:* The first part gives a thorough introduction to hybrid systems theory. The material is classified by the modeling approaches used to represent hybrid systems in a form that is convenient for analysis and control design. Hybrid automata and switched systems are well-studied system classes, which are extensively described, but other approaches like mixed logical dynamical systems, complementarity systems, quantized systems, and stochastic hybrid systems are also explained.
- *Part II: Tools:* The second part is concerned with computer-aided systems analysis, control design, and verification. After a survey of the variety of relevant tools, selected tools are described in more detail. This part concludes with a presentation of current tool integration activities.
- *Part III: Applications:* The third part is devoted to applications, implementation issues, and system integration. Concentrating on energy management, industrial control, automotive control, and networked control systems, important application domains are considered.

This bridge from the theory over the tools towards applications shows the advances made in hybrid systems theory over the last 10 years as well as the main gaps that have to be closed in the near future to improve the multidisciplinary design of modern technological systems. Due to its restricted size, the handbook has to concentrate on the most successful lines of current research and on the fields of applications where hybrid systems theory has facilitated significant progress.

This handbook is considerably different from the increasing number of publications that have recently appeared. Monographs and paper collections concerning specific types of hybrid systems focus on the theoretical background and place little importance on applications. The handbook fills the current lack of a basic introduction to hybrid systems from a control perspective, gives a survey of the field as a whole, and indicates the importance of recent theoretical developments using examples of applications.

Readership and structure The handbook has been written for all those who require an overview of the theory of hybrid systems. Potential readers are researchers in academia and industrial professionals who are engaged in the development of control systems for complex applications, as well as graduate and PhD students. They are assumed to be familiar with dynamical systems theory and should have some knowledge of both continuous and discrete-event systems.

For this readership, the handbook provides basic introductory information, detailed descriptions of selected topics, and extensive references to the rapidly growing literature. In each chapter, the first section gives a **broad introduction** to the subject followed by sections that provide a more detailed development of important sub-topics. Three **running examples** are used for the illustration of the methods and tools described in Parts I and II. Each chapter ends with **bibliographical notes**.

There is no uniform theory of hybrid systems yet, but the development of this field has been guided by different modeling approaches, which have in turn led to corresponding analysis and design methods. Although some relations among these approaches have been established, each approach is particularly suitable for its specific class of hybrid systems. The handbook, in particular Part I, is structured accordingly. As hybrid automata and piecewise affine systems have gained particular interest in research, a whole chapter is devoted to each of them, whereas further modeling ideas are summarized in additional theory chapters.

In addition, the field does not yet have a uniform terminology and notation, but instead has a mixture of notions and symbols with their origins in control theory and computer science. An important aim of this handbook is to bring them closer together.

Hybrid systems research, training, and innovation This handbook is a result of four years of cooperation within the FP6 *Network of Excellence HYCON*—“Hybrid Control: Taming Heterogeneity and Complexity of Networked Embedded Systems” (EU project IST-2004-511368, <http://www.ist-hycon.org>), gathering 26 partners from European academic institutions, industry, and private and public research agencies. It summarizes the research interests and common expertise of more

than 60 authors in this network and gives, in this sense, a *European* view on the field with emphasis on the research results of the participating institutions and groups. All sections are co-authored by researchers who have contributed to the corresponding topic during recent years. They are mentioned in alphabetic order with the responsible author first.

The Network of Excellence will end in the spring of 2009. The goals of the HYCON Network include the coordination of the efforts of the European scientific community through the creation of a common research program on the control of networked embedded systems, the creation of a research infrastructure shared among the members of the network in order to facilitate long-term scientific collaborations, and the implementation of new working methodologies and improving the research efficiency by several coordination, education, and mobility programs. A further important aspect of HYCON is the collaboration of academic research institutions with industrial partners in the fields of process and energy control, automotive applications, and communication networks, the result of which is presented in Part III of this handbook.

As a tangible and durable result of HYCON, the **European Embedded Control Institute** (EECI) was incorporated (under the French Association Law 1901) in 2006 to provide a sustainable structure to support the research community in this field (<http://www.eeci-institute.eu>). EECI is rapidly becoming a worldwide leading focal point in hybrid systems research and is helping to attract the brightest minds in the field to the European research area. After the HYCON Network of Excellence has finished, the EECI will remain and will continue to attract young and senior researchers and practitioners, thus yielding strong long-term collaborations among the participating members. EECI is currently located in Gif-sur-Yvette, France, and has a networked embedded control laboratory in L'Aquila, Italy.

Acknowledgement Several organizations and persons have helped to make this handbook idea a reality. The funding of the HYCON Network of Excellence by the European Union created the organizational environment. Mr. Daniel Lehmann has structured the writing process by creating guidelines and templates and by retaining the communication among the co-authors. Ms. Andrea Marschall has drawn and re-drawn many figures and Ms. Hannelore Hupp has helped to give the book a uniform layout.

Finally, the close cooperation of the editors with Cambridge University Press, in particular with Dr. Phil Meyler, is gratefully acknowledged.

Notation

The symbols are chosen according to the following conventions. Scalar values or signals are denoted by lower-case letters such as x , a or t , vectors by boldface letters such as \mathbf{x} or \mathbf{y} and matrices by boldface upper-case letters such as \mathbf{A} , \mathbf{B} . Accordingly, the elements x_1, \dots, x_n of a vector \mathbf{x} or $a_{11}, a_{12}, \dots, a_{mn}$ of a matrix \mathbf{A} are represented by italics. Sets are symbolized by calligraphic letters such as \mathcal{F} and \mathcal{Z} .

Unless the current literature uses other notations, q is the discrete state and \mathbf{x} the continuous state of a hybrid system. \mathbf{u} and \mathbf{y} denote the continuous input or continuous output, respectively, of the system.

The inequality $\mathbf{P} > 0$ is interpreted in two different ways, which is explicitly mentioned. In connection with optimal control or linear matrix inequalities, it states that the matrix \mathbf{P} is positive definite. Alternatively, the sign $>$ has to be interpreted as an elementwise relation saying that all elements of the matrix \mathbf{P} are positive ($p_{ij} > 0$ for all i, j).

Book homepage See book homepage at <http://www.rub.de/atp> → Books for further information and computer programs for the animation of important hybrid systems phenomena.

Part I

controlengineers.ir Theory

controlengineers.ir

1

Introduction to hybrid systems

W. P. M. H. Heemels, D. Lehmann, J. Lunze, and B. De Schutter

This chapter gives an informal introduction to hybrid dynamical systems and illustrates by simple examples the main phenomena that are encountered due to the interaction of continuous and discrete dynamics. References to numerous applications show the practical importance of hybrid systems theory.

Chapter contents

1.1	What is a hybrid system?	page	4
1.1.1	Three reasons to study hybrid systems		4
1.1.2	Behavior of hybrid systems		6
1.1.3	Hybrid dynamical phenomena		9
1.2	Models of hybrid systems		14
1.2.1	Model ingredients		14
1.2.2	Model behavior		16
1.2.3	Hybrid automata		17
1.3	Running examples		17
1.3.1	Two-tank system		17
1.3.2	Automatic gearbox		22
1.3.3	DC-DC converter		26

1.1 What is a hybrid system?

Wherever continuous and discrete dynamics interact, hybrid systems arise. This is especially profound in many technological systems, in which logic decision making and embedded control actions are combined with continuous physical processes. To capture the evolution of these systems, mathematical models are needed that combine in one way or another the dynamics of the continuous parts of the system with the dynamics of the logic and discrete parts. These mathematical models come in all kinds of variations, but basically consist of some form of differential or difference equations on the one hand and automata or other discrete-event models on the other hand. The collection of analysis and synthesis techniques based on these models forms the research area of hybrid systems theory, which plays an important role in the multi-disciplinary design of many technological systems that surround us.

1.1.1 Three reasons to study hybrid systems

The reasons to study hybrid systems can be quite diverse. Here we will provide three sources of motivation, which are related to (i) the design of technological systems, (ii) networked control systems, and (iii) physical processes exhibiting non-smooth behavior.

Challenges of multi-disciplinary design When designing a technological system (Fig. 1.1) such as a wafer stepper, electron microscope, copier, robotic system, fast component moulder, medical system, etc., multiple disciplines need to make the overall design in close cooperation. For instance, the electronic design, mechanical design, and software design together have to result in a consistent, functioning machine. The designs are typically made in parallel by multiple groups of people, where the communication between these groups is often hampered by lack of common understanding and common models. The lack of common models complicates the making of cross-disciplinary design decisions that may have advantages for one discipline, but disadvantages for others. To make a good trade-off, the overall effect of such a design decision has to be evaluated as early as possible. As the complexity of a technological system with typically millions of lines of codes and tens of thousands of mechanical components gives rise to many cross-disciplinary design decisions, a framework is required that supports efficient evaluation of design decisions incorporating quantitative information and models from multiple disciplines.

Hybrid systems theory studies the behavior of dynamical systems, including the technological systems mentioned above described by modeling formalisms that involve both continuous models such as differential or difference equations describing the physical and mechanical part, and discrete models such as finite-state machines or Petri nets that describe the software and logical behavior.



Fig. 1.1 Example of a technological system with hybrid dynamics (courtesy of Daimler).

This theory is one of the few scientific research directions that aim at approaching the design problem of technological systems in a rigorous manner and at developing a complete design framework. As such, hybrid systems theory combines ideas originating in the computer science and the software engineering disciplines on one hand, and systems theory and control engineering on the other. This mixed character explains the terminology “hybrid systems,” which was used in this context for the first time by Witsenhausen in 1966 [665].

Hybrid systems theory is a relatively young research field as opposed to the more conventional mono-disciplinary research areas such as mechanical, electrical, or software engineering. The urgent need for multi-disciplinary design and development methods for technological systems has spurred the growth of hybrid systems theory in recent years. However, due to the inherent complexity of hybrid systems, many issues still remain unsolved at present, at least at the scale needed for industrial applications. The current status of hybrid systems theory is surveyed in this handbook, which can be used as a starting point for future developments in this appealing and challenging research domain.

Adding communication: networked control systems Besides merging software (discrete) and physical (continuous) aspects of systems, another important aspect of many technological systems is communication. Within one single system, many subsystems interact through communication networks. For systems-of-systems the coordination plays an even larger role, resulting in extremely complicated networks of communication. One might think of examples such as automated highways [426] and air-traffic management [629]. As the many control, computation, communication, sensing, or actuation actions take place through shared network or processor resources, another dimension is added to the design of these systems. Within the context of these *networked control systems* (Chapter 15), the asynchronous and event-driven nature of the data transmission caused by varying delays, varying sampling intervals, package loss, etc., and the implementation of the networks and protocols

complicate their analysis and design even further. Also in this domain hybrid systems theory plays an essential role as a foundation to understand the behavior of these complex systems.

Physical processes modeled as hybrid systems In the technological and networked systems mentioned above, the digital and logic (embedded control) aspects are typically brought in by design in order to control the physics and mechanics of the system. However, hybrid system theory is not only useful within these domains. Many physical processes, exhibiting both fast and slow changing behaviors, can often be well described by using (simple) hybrid models. For instance, in non-smooth mechanics [123], the evolution of impacting rigid bodies can be captured in hybrid models. Indeed, as the impacts occur at a much smaller time scale than the unconstrained motion, the behavior can be described well by introducing discrete events and actions in a smooth model. The bouncing ball presented in Example 2.4 is a simple demonstration of this. Also the vector fields defining the behavior of the system might be different over time as they depend crucially on the fact whether a contact is active or not. The dynamics of a robot arm moving freely in space is completely different from the situation in which it is striking the surface of an object. Other examples in mechanics with hybrid behavior include motion systems with friction models that distinguish between stick and slip modes, backlash in gears, and dead zones in cog wheels.

Examples are not only found in the mechanical domain. Nowadays, switches such as thyristors and diodes are used in electrical networks for a wide variety of applications in both power engineering and signal processing. Examples include switched-capacitor filters, modulators, analog-to-digital converters, power converters, and choppers. In the ideal case, diodes are considered as elements with two (discrete) modes: the blocking mode and the conducting mode. Mode transitions for diodes are governed by state events, where currents or voltages change their sign. This indicates that hybrid modeling and analysis offer an attractive perspective on these switched circuits [306]. The DC-DC converter discussed in Section 1.3.3 forms a simple example of this.

Also many biological and chemical systems can often be efficiently described by hybrid models. For example, simulating moving bed processes, which are a special kind of chromatographic separation processes, have to be switched regularly among different structures in order to avoid that the separation process will eventually stop. Like in DC-DC converters, the switching is an integral part of the physical principle utilized in such processes. For the analysis of these systems and for control design, the model has to be switched accordingly, which demonstrates the necessity to extend continuous models towards hybrid models.

1.1.2 Behavior of hybrid systems

The previous section indicates that multi-disciplinary design of technological systems and the study of several non-smooth physical processes require the understanding of

the complex interaction between discrete dynamics and continuous dynamics. To provide some insight in this interaction, let us consider the following example.

Example 1.1 *Thermostat*

As a textbook example of a simple hybrid system consider the regulation of the temperature in a house. In a simplified description, the heating system is assumed either to work at its maximum power or to be turned off completely. This is a system that can operate in two modes: “on” and “off.” In each mode of operation (given by the discrete state $q \in \{\text{on}, \text{off}\}$) the evolution of the temperature T can be described by a different differential equation. This is illustrated in Fig. 1.2 in which each mode corresponds to a node of a directed graph, while the edges indicate the possible discrete state transitions. As such, this system has a hybrid state (q, T) consisting of a discrete state q taking the discrete values “on” and “off” and a continuous state T taking values in the real numbers.

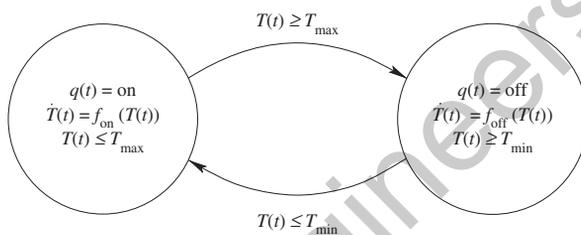


Fig. 1.2 Model of a temperature control system.

Clearly, the value of the discrete state q affects the evolution of the continuous state T as a different vector field is active in each mode. Vice versa, the switching between the two modes of operations is controlled by a logical device (the embedded controller) called the *thermostat* and depends on the value of the continuous state T . The mode is changed from “on” to “off” whenever the temperature T reaches the value T_{\max} (determined by the desired temperature). Vice versa, when the temperature T reaches a minimum value T_{\min} , the heating is switched “on.”

This example already shows some of the main features of hybrid systems:

- The thermostat is a hybrid system, because its state consists of a discrete state q and a continuous state T .
- The continuous behavior of the system depends on the discrete state, i.e. depending on whether the mode is “on” or “off” a different dynamics $\dot{T}(t) = f_{\text{on}}(T(t))$ or $\dot{T}(t) = f_{\text{off}}(T(t))$, respectively, governs the evolution of the temperature T .
- The changes of the discrete state q are determined by the continuous state T and different conditions on T might trigger the change of the discrete state (e.g. when the discrete state is “on,” $T = T_{\max}$ triggers the mode change, while $T = T_{\min}$ triggers the change when the discrete state is “off.”) \square

Although the thermostat example is rather simple, it already contains some of the basic ingredients that are needed to properly model hybrid systems. A proper modeling format must involve (at least) the description of the evolution of both

continuous-valued signals (temperatures, positions, velocities, currents, voltages, etc.) and discrete-valued signals (operation mode, position of switch, alarm on or off, etc.) over time and their mutual influence, see Fig. 1.3 for an abstraction of this perspective.

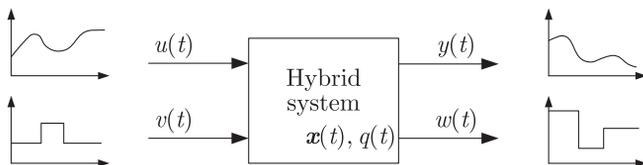


Fig. 1.3 Hybrid dynamical system.

The system depicted in Fig. 1.3 has six types of signals:

- $y(t)$ is a continuous output signal;
- $w(t)$ is a discrete output signal;
- $x(t)$ is a continuous (n -dimensional) state vector;
- $q(t)$ is a discrete state;
- $u(t)$ is a continuous input signal;
- $v(t)$ is a discrete input signal.

The input and output signals may be scalar or vector-valued, but for explaining the main idea of hybrid systems this distinction is not important.

Whereas the discrete signals (such as the “on” and “off” modes of the thermostat example) are typically piecewise constant, the continuous signals may change their value continuously or discontinuously. In the thermostat example the continuous signal representing the temperature is only changing continuously. There are no jumps (discontinuities) in the temperature. The state of the hybrid system is described by the pair (x, q) consisting of the continuous state vector x and the discrete state q . An important characteristic of hybrid systems lies in the fact that this pair influences the future behavior of the system. Moreover, the evolution of the system may also be influenced by a continuous as well as a discrete input, which are denoted by u and v , respectively, and one may receive some information on the hybrid state (x, q) from the discrete and continuous outputs w and y , respectively.

Figure 1.4 displays the typical behavior of an autonomous hybrid system (i.e. a hybrid system without an input), where the scalar continuous state x and the discrete state q are identical to the outputs. It shows that the evolution consists of smooth phases in which the discrete state remains constant and the continuous state changes continuously. At the transition times t_1, t_2, t_3, \dots the discrete state changes from its current value to a new value. Simultaneously, the continuous state may jump as shown in the figure for the time t_1 . At time t_1 the state changed abruptly from $x(t_1^-)$ to $x(t_1^+)$, where $x(t_1^-)$ and $x(t_1^+)$ denote the (limit) values of x just before and just after the state jump, respectively. It is important to realize that the transition times are

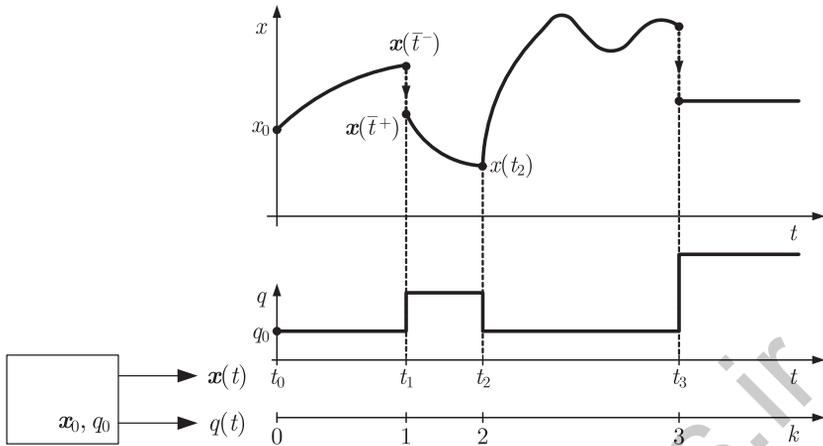


Fig. 1.4 Behavior of an autonomous hybrid system.

not necessarily prescribed by some clock (time events), but usually depend on both the discrete and the continuous state. For instance, in the thermostat example these transition times were determined by the temperature T reaching the values T_{\min} or T_{\max} (state events). In summary:

The trajectories of hybrid systems are partitioned into several time intervals. At the interval borders, the discrete state changes and/or jumps of the continuous state occur, whereas within all intervals the continuous signals change smoothly and the discrete state remains constant.

For a hybrid system with inputs, the behavior also depends upon the input signals. In this case the time instant at which the discrete state jumps, the new discrete and continuous states that are assumed afterwards as well as the continuous state evolution are all affected by these inputs.

1.1.3 Hybrid dynamical phenomena

Appropriate models for hybrid systems are often obtained by adding new dynamical phenomena to the classical description formats of the mono-disciplinary research areas. Indeed, continuous models represented by differential or difference equations, as adopted by the dynamics and control community, have to be extended to be suitable for describing hybrid systems. On the other hand, the discrete models used in computer science, such as automata or finite-state machines, need to be extended by concepts like time, clocks, and continuous evolution in order to capture the mixed discrete and continuous evolution in hybrid systems. The hybrid system models explained in Part I of this handbook combine both ideas. Here we will describe

the phenomena one has to add to the continuous models based on the differential equations

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t)). \quad (1.1)$$

Roughly speaking, as also argued in the previous discussion, four new phenomena that are typical for hybrid systems are required to extend the dynamics of purely continuous systems as in (1.1):

- autonomous switching of the dynamics;
- autonomous state jumps;
- controlled switching of the dynamics;
- controlled state jumps.

These phenomena are first explained for autonomous hybrid systems.

Autonomous switching of the dynamics This reflects the fact that the vector field \mathbf{f} that occurs in (1.1) is changed discontinuously. The switching may be invoked by a clock if the vector field \mathbf{f} depends explicitly on the time t :

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), t).$$

For instance, if periodic switching between two different modes of operation is used with period $2T$, we would have

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), t) := \begin{cases} f_1(\mathbf{x}(t)), & \text{if } t \in [2kT, (2k+1)T) \text{ for some } k \in \mathbb{N}, \\ f_2(\mathbf{x}(t)), & \text{if } t \in [(2k+1)T, (2k+2)T) \text{ for some } k \in \mathbb{N}. \end{cases}$$

This is an example of *time-driven switching*.

The switching can also be invoked when the continuous state \mathbf{x} reaches some *switching set* \mathcal{S} . As the situation $\mathbf{x}(t) \in \mathcal{S}$ is considered to be a state event, this kind of switching is said to be *event-driven*. The thermostat example provided an illustration of event-driven switching as the transition from the “on” mode to the “off” mode was triggered by the temperature reaching the value T_{\max} .

The following example also illustrates event-driven switching.

Example 1.2 Hybrid tank system

The tank systems shown in Fig. 1.5 illustrate two situations in which the dynamics of a process changes in dependence upon the state (liquid level). The tank in the left part of the figure is filled by the pump, which is assumed to deliver a constant flow Q_P , and emptied by two outlet pipes, whose outflows $Q_1(t)$ and $Q_2(t)$ depend upon the level $h(t)$. As the flow $Q_2(t)$ vanishes if the liquid level is below the threshold h_p given by the position of the upper pipe, the dynamical properties of the tank change if the level $h(t)$ exceeds this threshold.

The dependence of the vector field upon the state can be simply written down. For $h(t) < h_v$, the differential equation is given by

$$\dot{h}(t) = \frac{1}{A}(Q_P - \sqrt{2gh(t)}) = f_1(h(t)),$$

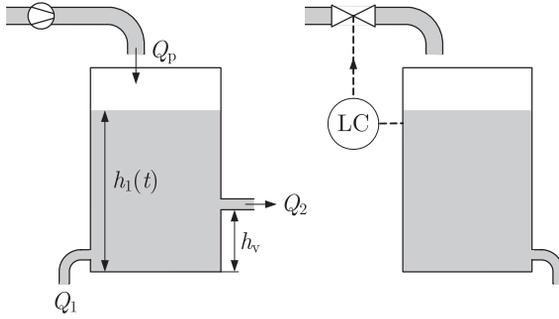


Fig. 1.5 Hybrid tank systems.

where g denotes the gravity constant. For $h(t) \geq h_v$ this equation changes into

$$\dot{h}(t) = \frac{1}{A} (Q_P - \sqrt{2gh(t)} - \sqrt{2g(h(t) - h_v)}) = f_2(h(t)).$$

Hence, the model can be written as

$$\dot{h}(t) = \begin{cases} f_1(h(t)) & \text{if } h(t) < h_v, \\ f_2(h(t)) & \text{if } h(t) \geq h_v, \end{cases}$$

which shows that the vector field switches between two different functions f_1 and f_2 in dependence upon the state $h(t)$ with the switching surface

$$\mathcal{S} = \{h \in \mathbb{R} \mid h = h_v\}.$$

Now assume that the pump is switched on and off at different time instances t_1 and t_2 . Then the function f occurring in the differential equation changes at these time points but this switching does not depend upon the state $h(t)$, but is time-driven.

The tank in the right part of Fig. 1.5 illustrates that autonomous switching is a typical phenomenon introduced by safety measures. In the tank system the level controller is equipped with a safety switch-off. If the liquid level is below the corresponding threshold, the dynamics is given by the controlled tank. If the level exceeds the threshold, the pump is switched off, which brings about a corresponding switching of the differential equation of the tank. \square

Switching among different dynamics has important consequences for the behavior of the hybrid system. For instance, Example 2.3 shows that switching between two linear stable vector fields can result in an unstable overall system.

Autonomous state jumps These constitute the second hybrid phenomenon. At some time \bar{t} , the state may jump from the value $\mathbf{x}(\bar{t}^-)$ towards the value $\mathbf{x}(\bar{t}^+)$. An illustrative example is a bouncing ball (see also Example 2.4 later). If the ball touches the ground at time \bar{t} , then its velocity is instantaneously reversed.

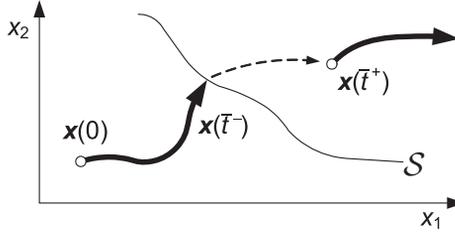


Fig. 1.6 Autonomous state jump.

A simple representation of state jumps is given as follows. An *autonomous jump set* is a set \mathcal{S} on which a state jump is invoked (Fig. 1.6). Some relation \mathcal{R} , which often is called a *reset map*, determines where the state jumps goes to:

$$(\mathbf{x}(\bar{t}^-), \mathbf{x}(\bar{t}^+)) \in \mathcal{R}.$$

Here \bar{t} is the time instant at which the trajectory $\mathbf{x}(\cdot)$ reaches the set \mathcal{S} :

$$\mathbf{x}(\bar{t}) \in \mathcal{S}.$$

The reset map may depend on the discrete state $q(\bar{t}^-)$ of the hybrid system just before the reset. Including such state jumps in a continuous system described by the differential equation (1.1) results in the extended model

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{f}(\mathbf{x}(t)), & \text{for } \mathbf{x}(t) \notin \mathcal{S}, \\ (\mathbf{x}(t^-), \mathbf{x}(t^+)) &\in \mathcal{R}(q(\bar{t}^-)), & \text{for } \mathbf{x}(t) \in \mathcal{S}. \end{aligned}$$

Example 1.3 Reset oscillator

Consider the reset oscillator described by the affine state space model

$$\frac{d}{dt} \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -1 & 2\delta \end{pmatrix} \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

together with the reset map defined by

$$x_1(\bar{t}^+) = -x_1(\bar{t}^-),$$

where \bar{t} denotes any time instant at which the state is on the switching set

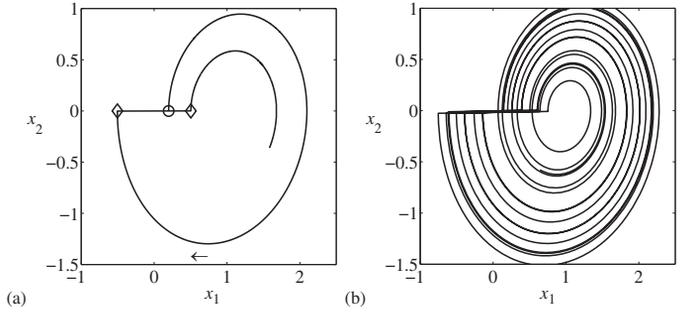


Fig. 1.7 Behavior of the reset oscillator.

$$S = \{x \in \mathbb{R}^2 \mid x_1 = 0, x_2 < 0\}.$$

Such reset systems find application, for instance, in data transmission over highly disturbed communication channels and reset control systems.

Figure 1.7 shows the behavior of the reset oscillator for $\delta = 0.1$. Fig. 1.7(a) includes the trajectory in the state space for the short time interval $t \in [0, 10]$. The trajectory starts in the initial state $x(0) = (0.2 \ 0)^T$ depicted by the small circle. The switching surface S is hit in the point $(-0.504, 0)^T$ as indicated by the left diamond. Next, a state jump occurs that brings the state to the right diamond. Fig. 1.7(b) shows the oscillator behavior for a longer time horizon.

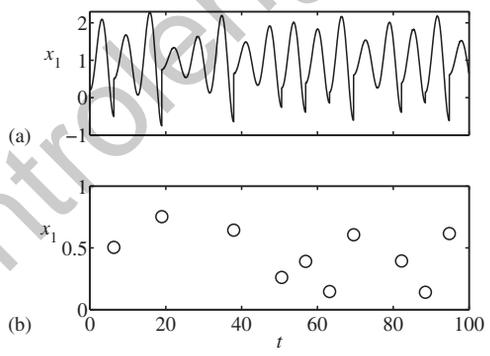


Fig. 1.8 Trajectory of the reset oscillator: (a) state trajectory $x_1(t)$ and (b) destination state sequence $x_1(t_k^+)$ of the state jumps.

The state jump has two consequences:

- Although the oscillator has an affine state space model, the behavior of the reset oscillator is chaotic.
- Although the oscillator without state jump is an unstable system (with eigenvalues $\lambda_{1,2} = 0.1 \pm j0.995$) the reset oscillator state remains bounded.

The irregular (chaotic) behavior can be seen in Fig. 1.8, where in part (a) the state trajectory x_1 is shown. Figure 1.8(b) extracts the state jumps from the evolution. The circles depict the points $x_1(\bar{t}_k^+)$ just after the occurrence of the state jumps at the time instants \bar{t}_k^+ , ($k = 0, 1, \dots$). Neither the temporal distance $\bar{t}_{k+1}^+ - \bar{t}_k^+$ between these jumps nor the endpoints $x_1(\bar{t}_k^+)$ show a regular behavior. \square

The above example shows that state jumps may considerably change the dynamical properties of a system in comparison to the same system without state jumps.

Controlled switching This occurs if the system has a discrete input v that is used to invoke the switching among different continuous dynamics. If the value of the discrete input is changed at time \bar{t} , then the vector field $f(x(t), v(t))$ changes abruptly at time \bar{t} as well.

Systems with discrete control inputs represent a relevant system class from a practical point of view. The DC-DC converter is a simple example of such systems that will be used as running example throughout this handbook (Section 1.3.3).

Controlled state jumps These are discontinuities in the state trajectory that occur as a response to a control command. An example in which such a state jump is necessary for satisfying performance requirements is the automatic gearbox described in Section 1.3.2. A state jump in the gearbox controller must be invoked whenever the gearing is changed in order to avoid a jump in the acceleration of the vehicle.

1.2 Models of hybrid systems

Although many different models have been proposed in literature, as will be seen in the following chapters, the model ingredients (including the main dynamical phenomena as seen in the previous section) are basically the same.

1.2.1 Model ingredients

The structure of hybrid systems introduced so far shows that every model of a hybrid system has to define at least the following elements (Fig. 1.9):

- \mathcal{X} is the continuous state space, for which often $\mathcal{X} = \mathbb{R}^n$ holds;
- \mathcal{Q} is the discrete state space, for example $\mathcal{Q} = \{0, 1, 2, \dots, Q\}$;
- f is a set of vector fields describing the continuous dynamics for all $q \in \mathcal{Q}$;
- $Init$ is a set of initial values (q_0, x_0) of the hybrid state;
- δ is the discrete state transition function;
- \mathcal{G} is a set of guards prescribing when a discrete state transition occurs.

To simplify the considerations, hybrid systems without external inputs will be investigated here.

The model elements lead to a graphical representation of the hybrid system, which will later be extended to get the hybrid automaton representation dealt with in Chapter 3. The discrete part of the dynamics is modeled by means of a graph whose vertices represent the discrete states (also called *operation modes* or *locations*) and whose edges represent state transitions. Every vertex is associated with the vector field

$$f : Q \times \mathbb{R}^n \rightarrow \mathbb{R}^n$$

that belongs to the corresponding value q of the discrete state. It describes the evolution of the continuous state if the discrete state is $q(t)$:

$$\dot{x}(t) = f(q(t), x(t)).$$

The trajectories that can be obtained for all possible initial continuous states is also called the set of *activities*. As the discrete state $q(t)$ remains constant over some time interval ($q(t) = q$), the vector field $f(q, x(t))$ is alternatively denoted by $f_q(x(t))$, which shows that the q -th vector field is valid in the operation mode q .

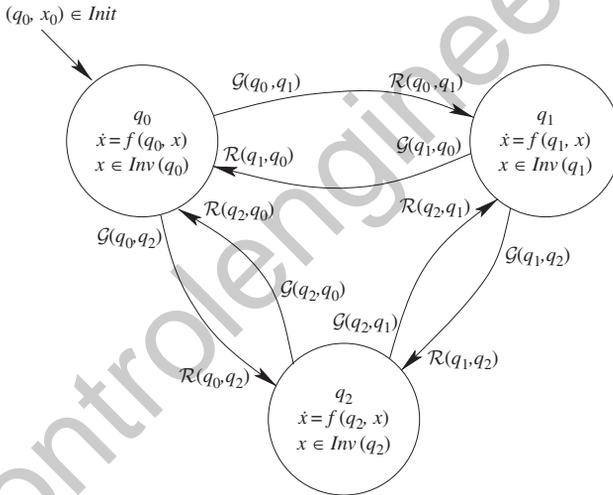


Fig. 1.9 Schematic representation of a hybrid automaton with three discrete states. Each node of the directed graph represents a mode (operating point) given by a system of differential (or difference) equations. The arrows indicate the possible discrete transitions that correspond to a change of the mode.

Whereas the discrete state q influences the continuous dynamics by selecting a specific vector field $f(q, \cdot)$, the influence of the continuous dynamics on the discrete state evolution is represented by a set of guards. A guard describes a region in the state space \mathcal{X} . If the state x is in this region, a discrete state transition *may* occur. For example, in Fig. 1.9, the guard $G(q_0, q_1)$ poses a condition on the state x that has to be satisfied in order to invoke the discrete state transition $q_0 \rightarrow q_1$.

The change of the discrete state is described by the state transition function δ , which determines the discrete successor state q' if the system is in a given discrete state q . This function is graphically represented by the arrows among the discrete states in Fig. 1.9. As the figure shows, the question of which successor state is assumed depends upon the guard condition \mathcal{G} that is satisfied by the continuous state x at the switching time.

The model explained above can be extended by the following elements in order to include state jumps and to complete the representation of the interaction between the continuous and the discrete dynamics:

- \mathcal{R} is a reset map defining the state jumps;
- Inv denotes the invariants.

Each mode has an invariant associated to it, which describes the conditions that the continuous state has to satisfy at this mode. Invariants and guards play complementary roles: whereas invariants describe when a transition *must* take place (namely when otherwise the motion of the continuous state would lead to violation of the conditions given by the invariant), the guards serve as “enabling conditions” that describe when a particular transition *may* take place.

The reset map is, in general, a set-valued function that specifies how new continuous states are related to previous continuous states for a particular transition.

1.2.2 Model behavior

To provide insight in the evolution of the dynamical system defined above, we give a short, rather informal description. The initial hybrid state (q_0, x_0) of “trajectories” of a hybrid automaton lies in the initial set $Init$. From this hybrid state the continuous state x evolves according to the differential equation

$$\dot{x} = f(q_0, x) \quad \text{with } x(0) = x_0$$

and the discrete state q remains constant: $q(t) = q_0$. The continuous evolution can go on as long as x stays in $Inv(q_0)$. If at some point the continuous state x reaches the guard $\mathcal{G}(q_0, q_1)$, we say that the transition (q_0, q_1) is enabled. The discrete state may then change to q_1 , and the continuous state jumps from the current value x^- to a new value x^+ with $(x^-, x^+) \in \mathcal{R}(q_0, q_1)$. After this transition, the continuous evolution resumes according to the mode q_1 and the whole process is repeated. Note that the invariants and guards are related to the switching sets and jumps sets introduced earlier, as all these concepts are related to triggering discrete actions such as resets of the continuous states or changes in the discrete state.

This framework leads to the behavior of a hybrid system as depicted in Fig. 1.4: continuous phases separated by events at which maybe multiple discrete actions (jumps of the continuous state x and/or changes in the discrete state q) take place. It is obvious that these systems may switch between many operating modes where each mode is governed by its own vector field (Fig. 1.9). Mode transitions are triggered by variables crossing specific thresholds (state events) and by the elapse of certain time periods (time events) due to the invariants and guards. With a change of mode, discontinuities in the continuous variables may occur as given by the reset map.

1.2.3 Hybrid automata

The model ingredients introduced above lead directly to one of the main modeling formalisms used in hybrid systems theory: the hybrid automaton. The formal definition given and explained in Section 3.1 summarizes the modeling elements introduced here in the following form, where $2^{\mathcal{X}}$ denotes the power set of \mathcal{X} , i.e. the collection of all subsets of \mathcal{X} :

A hybrid automaton H is an 8-tuple

$$H = (Q, \mathcal{X}, f, Init, Inv, \mathcal{E}, \mathcal{G}, \mathcal{R}),$$

where

- $Q = \{q_1, \dots, q_k\}$ is a finite set of *discrete states (control locations)*;
- \mathcal{X} is the continuous state space;
- $f : Q \times \mathbb{R}^n \rightarrow \mathbb{R}^n$ is a vector field;
- $Init \subset Q \times \mathbb{R}^n$ is the set of *initial states*;
- $Inv : Q \rightarrow 2^{\mathbb{R}^n}$ describe the *invariants* of the locations;
- $\mathcal{E} \subseteq Q \times Q$ is the *transition relation*;
- $\mathcal{G} : \mathcal{E} \rightarrow 2^{\mathbb{R}^n}$ is the *guard condition*;
- $\mathcal{R} : \mathcal{E} \rightarrow 2^{\mathbb{R}^n} \times 2^{\mathbb{R}^n}$ is the *reset map*.

The hybrid state of the system H is given by $(q, x) \in Q \times \mathcal{X}$.

Based on the description of this general hybrid system model various ramifications and extensions can be created as well as other more specific models of hybrid systems such as piecewise affine systems, mixed logical dynamical systems, complementarity systems, and so on. This handbook will provide an overview of the available results for all these model classes and will also pinpoint various open issues for future research. Before doing so, we will introduce some running examples that will be used throughout the handbook to illustrate the main ideas.

1.3 Running examples

This section introduces three simple examples that illustrate the main new phenomena that are introduced by the interaction of continuous and discrete dynamics. These examples will be referred to frequently throughout this handbook.

1.3.1 Two-tank system

Process description The two-tank system is a hybrid system with autonomous switching. The main control task is to stabilize its state. The system represents a simplified version of systems that are widely used in the process industry to provide a customer with a continuous liquid flow by maintaining the liquid levels of the tanks at prescribed values.

This example consists of two coupled cylindrical tanks T_1 and T_2 connected by pipes (Fig. 1.10). The water flow between the tanks and out of the tanks can

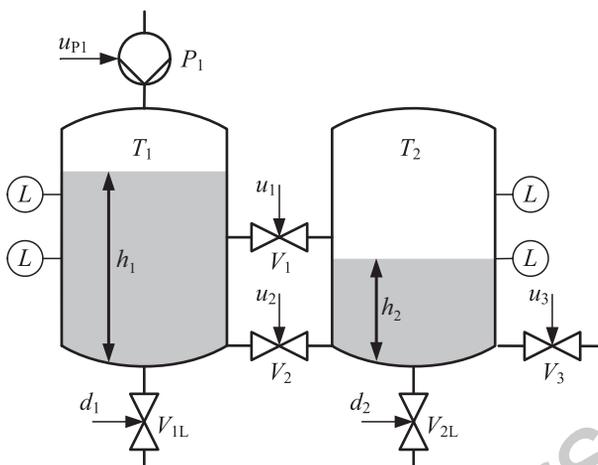


Fig. 1.10 Two-tank system.

be controlled by the valves V_1 , V_2 , V_3 , V_{1L} , and V_{2L} , each of which can only be completely opened or closed (on/off valves). The connection pipes between the tanks are placed at the bottom of the tanks (with valve V_2) and at the height h_0 above the bottom (with valve V_1).

The maximum water level of each tank is denoted by h_{\max} . All tanks have the same cross-sectional area A and are located at the same level.

In a typical situation, the valves V_1 , V_2 , and V_3 are opened and the valves V_{1L} and V_{2L} closed. Liquid is filled into the left tank by the pump P_1 . Measurements concern the levels $h_1(t)$ and $h_2(t)$ in tanks T_1 and T_2 respectively. Discrete sensors (denoted by L in the figure) yield a qualitative characterization of the liquid levels as *low*, *medium*, and *high*.

The system has both continuous and discrete inputs. The continuous input is the inflow through the pump $u_{P1}(t) = Q^{P1}(t)$ and the discrete inputs are the positions of the valves V_1 , V_2 , and V_3 , so that

$$\mathbf{u}(t) = (u_{P1}(t) \ u_1(t) \ u_2(t) \ u_3(t))^T$$

holds. Disturbances affecting the system can be induced by changing the positions of the valves V_{1L} and V_{2L} .

Hybrid phenomena The two-tank is a typical hybrid system, as it has a continuous dynamics with state-dependent and controlled switching. If the valve positions remain constant the continuous dynamics switches autonomously between four discrete modes $q(t)$ depending on whether or not the liquid levels exceed the height h_0 of the upper connection pipe. The discrete system behavior is represented by the automaton shown in Fig. 1.11, where each node represents one discrete operation mode.

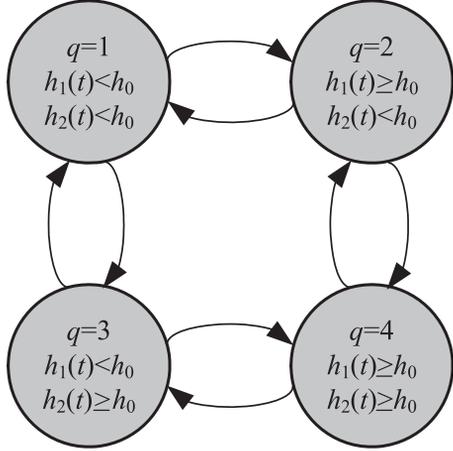


Fig. 1.11 Discrete behavior of the two-tank system.

Dynamical model The two-tank system has two continuous state variables

$$\mathbf{x}(t) = (h_1(t) \ h_2(t))^T, \ h_i \in \mathbb{R},$$

and four discrete states

$$q(t) \in \{1, 2, 3, 4\}$$

that depend on the levels as shown in Table 1.1. The nonlinear dynamics follows from Torricelli's law:

Table 1.1 Discrete modes in dependence of the continuous states.

$q(t)$	$h_1(t)$	$h_2(t)$
1	$< h_0$	$< h_0$
2	$\geq h_0$	$< h_0$
3	$< h_0$	$\geq h_0$
4	$\geq h_0$	$\geq h_0$

$$Q_{ij}^{V_i}(t) = c \cdot \text{sgn}(h_i(t) - h_j(t)) \cdot \sqrt{2g \cdot |h_i(t) - h_j(t)|} \cdot u_i(t),$$

where $Q_{ij}^{V_i}(t)$ is the water flow from tank T_i into tank T_j through the pipe with valve V_i , c the flow constant of the valves, $u_i(t) \in \{0,1\}$ the position of valve V_L (0 means the valve is closed and 1 the valve is opened), and g the gravity constant.

The change of water volume $V(t)$ in a tank can be described by

$$\dot{V}(t) = \dot{h}(t) \cdot A = \sum Q_{\text{in}}(t) - \sum Q_{\text{out}}(t),$$

where $\sum Q_{\text{in}}(t)$ is the sum of all inflows into the tank and $\sum Q_{\text{out}}(t)$ is the sum of all outflows. By applying this equation to the two tanks, the following nonlinear differential equations are obtained:

$$\begin{aligned} \dot{h}_1(t) &= \frac{u_{P_1}(t) - Q_{12}^{V_1}(t) - Q_{12}^{V_2}(t) - Q_L^{V_{1L}}(t)}{A}, \\ \dot{h}_2(t) &= \frac{Q_{12}^{V_1}(t) + Q_{12}^{V_2}(t) - Q_L^{V_{2L}}(t) - Q_N^{V_3}(t)}{A}. \end{aligned}$$

The flow $Q_{12}^{V_1}(t)$ depends on the mode $q(t)$ as follows:

$$Q_{12}^{V_1}(t) = \begin{cases} 0, & q(t) = 1, \\ c \cdot \text{sgn}(h_1(t) - h_0) \cdot \sqrt{2g \cdot |h_1(t) - h_0|} \cdot u_1(t), & q(t) = 2, \\ c \cdot \text{sgn}(h_0 - h_2(t)) \cdot \sqrt{2g \cdot |h_0 - h_2(t)|} \cdot u_1(t), & q(t) = 3, \\ c \cdot \text{sgn}(h_1(t) - h_2(t)) \cdot \sqrt{2g \cdot |h_1(t) - h_2(t)|} \cdot u_1(t), & q(t) = 4. \end{cases}$$

The following equations hold in all four modes:

$$\begin{aligned} Q_{12}^{V_2}(t) &= c \cdot \text{sgn}(h_1(t) - h_2(t)) \cdot \sqrt{2g \cdot |h_1(t) - h_2(t)|} \cdot u_2(t), \\ Q_N^{V_3}(t) &= c \cdot \sqrt{2g \cdot h_2(t)} \cdot u_3(t), \\ Q_L^{V_{iL}} &= c \cdot \sqrt{2g \cdot h_i(t)} \cdot d_i(t), \quad i = 1, 2, \end{aligned}$$

where $Q_N^{V_3}(t)$ is the water flow exiting from tank T_2 through the pipe with valve V_3 , and $Q_L^{V_{iL}}$ is the water flow exiting from tank T_i through the pipe with valve V_{iL} . If these differential and algebraic equations are associated with the discrete model shown in sec1.2:fig:tank2, a hybrid automaton results as overall model.

All relevant parameter values are given in Table 1.2.

Hybrid behavior Figure 1.12 shows the behavior of the two-tank system with the initial state $x_0 = (0.25 \ 0.45)^T$ and constant inflow $u_{P_1}(t) = 0.03 \text{ m}^3 \text{ s}^{-1}$. Fig. 1.12(a) shows the trajectories of the tank levels and Fig. 1.12(b) the related modes demonstrating the autonomous switching of the system, when the height h_0 is reached.

The two-tank system offers various possibilities to illustrate the main analysis and design concepts presented in the handbook. One potential question is whether or not the system state depicted in Fig. 1.13 can be reached by an appropriate control input. This example will be used throughout the book to illustrate different modeling

Table 1.2 Parameter values and ranges of the two-tank system.

Parameter	Value
c	$3.6 \times 10^{-5} \text{ m}^2$
h_0	0.3 m
h_{\max}	0.6 m
A	0.0154 m^2
g	9.81 m s^{-2}
Q_{\max}	$0.1 \times 10^{-3} \text{ m}^3 \text{ s}^{-1}$
Qualitative value	Range
<i>low</i>	[0...20] cm
<i>medium</i>	[20...25] cm
<i>high</i>	[25...60] cm
State	Value
h_1, h_2	$\in \mathbb{R} \text{ cm}$
q	$\in \{1, 2, 3, 4\}$
Input	Possible value/range
u_1, u_2, u_3	$\in \{0,1\}$
$u_{P1} = Q^{P1}$	$\in [0, Q_{\max}] \text{ m}^3 \text{ s}^{-1}$
Disturbance	Value
d_1, d_2	$\in \{0,1\}$

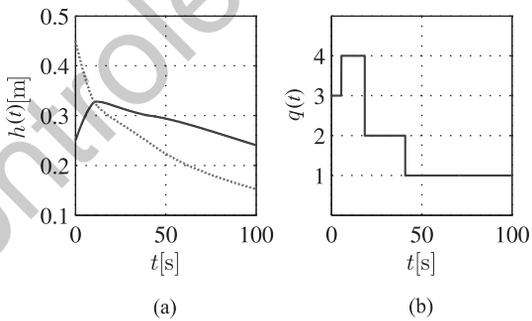


Fig. 1.12 Simulation results of the two-tank system.

paradigms, reachability analysis, and abstraction-based modeling methods. In Example 3.1 the system is represented as a hybrid automaton, in Example 5.2 as a linear complementarity system. Example 3.3 develops optimal control for this system, whereas Example 11.1 uses this system to illustrate simulation tools. More complex

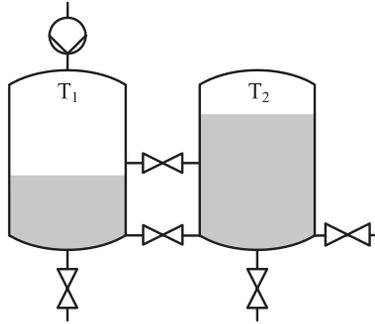


Fig. 1.13 Is this a potential state of the two-tank system?

technological processes, which likewise have a hybrid dynamics, are considered in Chapter 14.

1.3.2 Automatic gearbox

Process description The automatic gearbox is a switched system with a discontinuous evolution of the continuous state. State jumps occur together with controlled switching.

Automatic gearboxes are used to change gear ratios automatically. This example presents an automatic transmission with four gears. It consists of the gearbox and a controller comprised of a continuous and a discrete-event part as depicted in Fig. 1.14.

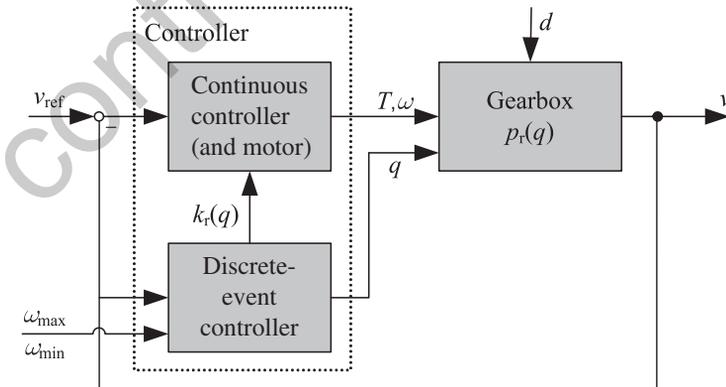


Fig. 1.14 Automatic gearbox.

The gearbox and its controller interact dependent on the vehicle velocity $v(t)$. The continuous inputs of the gearbox are the torque $u(t) = T(t)$ and the angular velocity $\omega(t)$ of the motor. Disturbances $d(t)$ are induced by the road, e.g. by different coefficients of friction.

Hybrid phenomena and dynamical model The automatic gearbox has four discrete modes $q(t)$

$$q(t) \in \{1, 2, 3, 4\},$$

which affect the continuous dynamics by changing the transmission ration $p_r(q)$. Table 1.3 presents all modes with their specific parameters $p_r(q)$ and $k_r(q)$, where $p_r(1) > p_r(2) > p_r(3) > p_r(4)$ holds. The mode is automatically changed by the discrete inputs selected by the controller.

Table 1.3 Modes and specific parameter.

$q(t)$	Transmission ration	Controller gain
1	$p_r(1)$	$k_r(1)$
2	$p_r(2)$	$k_r(2)$
3	$p_r(3)$	$k_r(3)$
4	$p_r(4)$	$k_r(4)$

The continuous part of the controller consists of a PI-controller with the integrator state $T_1(t)$. To obtain a comfortable ride, restrictions are imposed on the derivative of the acceleration $\ddot{v}(t)$, which make it necessary to switch the controller parameters $k_r(q)$ dependent on the gearing $q(t)$ and to impose state jumps in the integrator state whenever the gear is changed. In Fig. 1.15 the switching scheme of the automatic gearbox is depicted.

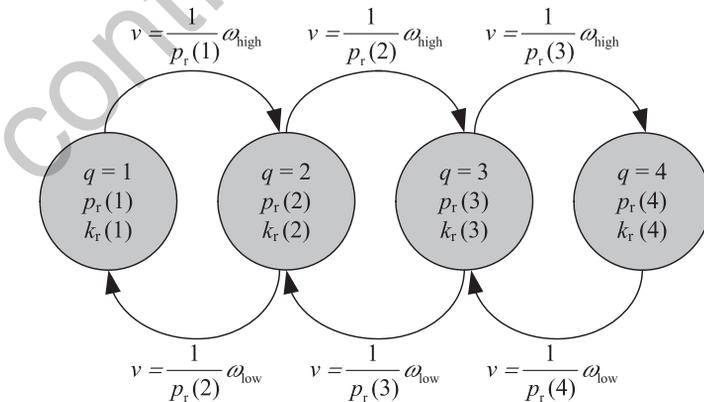


Fig. 1.15 Hybrid automaton of the automatic gearbox.

Dynamical model The gearbox is modeled here with only the velocity as a continuous state variable

$$\mathbf{x}(t) = (v(t) \ T_1(t))^T.$$

The transmission influences the torque $T(t)$ and the angular velocity $\omega(t)$ of the motor according to

$$\begin{aligned} T_w(t) &= p(q) T(t) = p(q) u(t), \\ \omega_w(t) &= \frac{1}{p(q)} \omega(t), \end{aligned}$$

where $T_w(t)$, $\omega_w(t)$ are respectively the torque and the velocity of the wheels, and $p(q)$ denotes the transmission ratio of the gearbox, which is obtained from the ratio $p_r(q)$ used in Table 1.3 by

$$p_r(q) = \frac{p(q)}{r},$$

where r is the wheel radius.

The relations between the torque and the force $F(t)$ accelerating the vehicle and between the velocity of the vehicle and the angular velocity are given by

$$\begin{aligned} F(t) &= \frac{T_w(t)}{r}, \\ v(t) &= r \omega_w(t). \end{aligned}$$

Applying Newton's law of motion leads to the vehicle dynamics

$$m\dot{v}(t) = F(t) - F_1(t)$$

and

$$\dot{v}(t) = \frac{p_r(q)u(t)}{m} - \frac{c}{m}v(t)^2 \text{sign } v(t) - g \sin(d(t)),$$

where m is the mass of the vehicle and the latter two terms represent the load force $F_1(t)$ which is assumed to be proportional to the square of the velocity. The disturbance $d(t)$ is considered as the road angle.

Control tasks A gear change should occur if $\omega(t)$ reaches a *high* or *low* limit ω_{high} and ω_{low} , respectively (Fig. 1.15). According to the relation $\omega(t) = p_r(q)v(t)$ the limits correspond to certain velocities of the vehicle. The mode shifts are given by the switching sets

$$\begin{aligned} S_{q,q+1} &= \{v \in \mathbb{R} \mid v = \frac{1}{p_r(q)}\omega_{\text{high}}\}, \\ S_{q+1,q} &= \{v \in \mathbb{R} \mid v = \frac{1}{p_r(q+1)}\omega_{\text{low}}\}, \end{aligned}$$

where $S_{q,q+1}$ denotes a mode shift from mode q to $q+1$ and $S_{q+1,q}$ a mode shift from mode $q+1$ to q .

The continuous PI-control law with a compensation of the nonlinearities is given by

$$u(t) = T_P(t) + T_I(t) + \frac{c}{p_r(q)} v(t)^2 \text{sign } v(t),$$

with

$$T_P(t) = k_r(q)(v_{\text{ref}}(t) - v(t)),$$

$$\dot{T}_I(t) = \frac{k_r(q)}{T_R}(v_{\text{ref}}(t) - v(t)),$$

where T_R is the integration time constant. Every time a new value of the set point $v_{\text{ref}}(t)$ is fixed by the driver, the integrator state $T_I(t)$ is put to zero (controlled state jump).

The ride is comfortable if the acceleration is limited, which causes a restriction on the gain $k_r(q)$, and if restrictions on the derivative of the acceleration are imposed. If $k_r(q)$ takes a value out of the set

$$k_r(q) \in \{k_r(1), k_r(2), k_r(3), k_r(4)\},$$

no abrupt changes of $\ddot{v}(t)$ and $\dot{v}(t)$ occur due to a mode shift if

$$p_r(q)k_r(q) = p_r(q+1)k_r(q+1)$$

$$p_r(q)T_I(\bar{t}^-) = p_r(q+1)T_I(\bar{t}^+), \quad \text{mode change } q \rightarrow q+1,$$

$$p_r(q)T_I(\bar{t}^-) = p_r(q-1)T_I(\bar{t}^+), \quad \text{mode change } q \rightarrow q-1.$$

This is called *bumpless transfer*.

Hybrid behavior Figure 1.16 illustrates a trajectory in the state space including jumps in the integrator state at the time the gear changes. The desired velocity $v_{\text{ref}}(t)$ is set to 30 m/s and the limits ω_{high} and ω_{low} are equal to 500 rad/s or 230 rad/s, respectively. Table 1.4 contains all relevant parameter values.

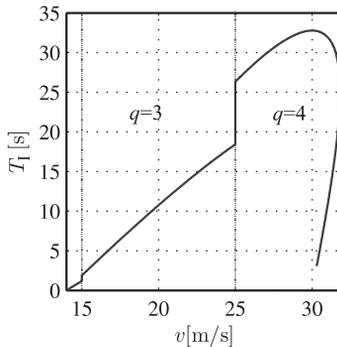


Fig. 1.16 Trajectory in the continuous state space.

Table 1.4 Parameter values of the automatic gearbox.

Parameter	Value
$p_r(1), p_r(2), p_r(3), p_r(4)$	50, 32, 20, 14
c	0.7 kg m^{-1}
m	1500 kg
g	10 m s^{-2}
$k_r(1), k_r(2), k_r(3), k_r(4),$ T_R	3.75, 5.86, 9.375, 13.39 N s 40 s
State	Value
v, T_I	$\in \mathbb{R} \text{ m s}^{-1}, \text{ Nm}$
$v(0), T_I(0)$	$14 \text{ m s}^{-1}, 0 \text{ Nm}$
q	$\in \{1, 2, 3, 4\}$
$q(0)$	2
Input	Value
u	$\in \mathbb{R} \text{ Nm}$
v_{ref}	30 m s^{-1}
ω_{high}	500 rad s^{-1}
ω_{low}	230 rad s^{-1}
Disturbance	Value
d	$\in [-\frac{\pi}{2}, \dots, \frac{\pi}{2}] \text{ rad}$

Analysis and control design methods for hybrid systems are necessary to cope with discontinuities in the continuous state evolution of the gearbox. This example will be used in the handbook in Example 3.4, where an optimal control strategy is developed. More automotive applications are investigated in Chapter 15.

1.3.3 DC-DC converter

Process description The DC-DC converter is a switched system, where the controlled switching is necessary for retaining its function of stabilizing the output voltage. The system has to be stabilized at a limit cycle rather than in an equilibrium state.

Power converters are widespread industrial devices used, for example, in variable-speed DC motor drives, computer power supply, cell phones, and cameras. The main functional principle lies in switching an electrical circuit among different structures in order to transform a constant or slowly varying DC voltage into a DC voltage that is independent of the load.

This example deals with a boost converter, whose output voltage is higher than the input voltage. The topology of a DC-DC boost converter is depicted in Fig. 1.17. The circuit consists of a load R , a capacitor C , an inductor L , a diode D , and a switch S . It has a fixed input voltage E and a variable output voltage $v(t)$. In addition, R_C and R_L represent the series resistors of the capacitor and the inductor.

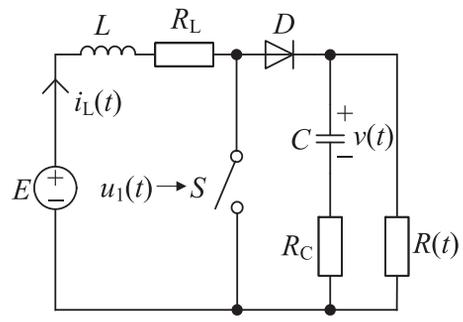


Fig. 1.17 Boost converter.

In the *on-state* the switch is closed, resulting in an increase of the inductor current $i_L(t)$. In the *off-state* the switch is open. The only path for the current is through the flyback diode, the capacitor and the load, and then the current ramps down. This situation results in transferring the energy accumulated in the inductance during the on-state into the capacitor. The process repeats cyclically, whereas the boost converter operates with the switching period T_S and the duty cycle $d_1(t)$, which corresponds to the ratio of activation duration of an *on-state* mode to the period. The duty cycle is considered as the system input $u_1(t) = d_1(t) \in [0, 1]$.

Hybrid phenomena The boost converter is a hybrid system with the three operation modes

$$q(t) \in \{1, 2, 3\}$$

summarized in Table 1.5. It is a second-order system with the continuous state variables $i_L(t)$ and $v(t)$

$$x(t) = (i_L(t) \ v(t))^T.$$

The switching scheme of the converter expressed by an automaton is shown in Fig. 1.18, where n denotes the cycle index.

If the current through the inductor never falls to zero the boost converter operates in continuous conduction mode (CCM), in which the switch and the diode are turned on and off in a cyclic and complementary manner and merely the modes $q = 1$ and

Table 1.5 Modes of the boost converter.

$q(t)$	S	$i_L(t)$
1	closed	$i_L > 0$
2	opened	$i_L > 0$
3	opened	$i_L = 0$

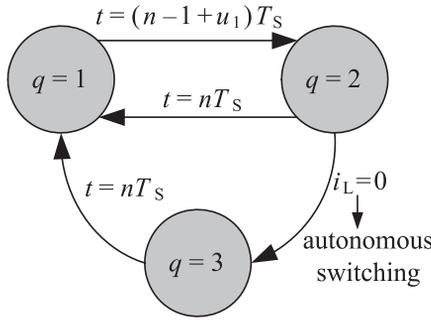


Fig. 1.18 Hybrid automaton of the boost converter.

$q = 2$ are accessible. Switching to the third mode occurs autonomously if the current falls to zero (discontinuous conduction mode (DCM)).

Dynamical model The affine state-space model of the converter

$$\dot{x}(t) = A(q)x(t) + B(q) \tag{1.2}$$

is given with the following matrices:

- $q(t) = 1$:

$$A(1) = \begin{pmatrix} -\frac{R_L}{L} & 0 \\ 0 & -\frac{1}{(R+R_C)C} \end{pmatrix},$$

$$B(1) = \begin{pmatrix} \frac{1}{L} \\ 0 \end{pmatrix} E.$$

- $q(t) = 2$:

$$A(2) = \begin{pmatrix} -\frac{1}{L} \left(R_L + \frac{R_C R}{R+R_C} \right) & -\frac{R}{(R+R_C)L} \\ \frac{R}{(R+R_C)C} & -\frac{1}{(R+R_C)C} \end{pmatrix},$$

$$B(2) = \begin{pmatrix} \frac{1}{L} \\ 0 \end{pmatrix} E.$$

- $q(t) = 3$:

$$A(3) = \begin{pmatrix} 0 & 0 \\ 0 & -\frac{1}{(R+R_C)C} \end{pmatrix},$$

$$B(3) = \begin{pmatrix} 0 \\ 0 \end{pmatrix} E.$$

Hybrid behavior Figure 1.19(a) shows the stationary behavior of $i_L(t)$ and $v(t)$ of the boost converter operating in CCM with a fixed input $u_1(t) = 0.5$ and a constant

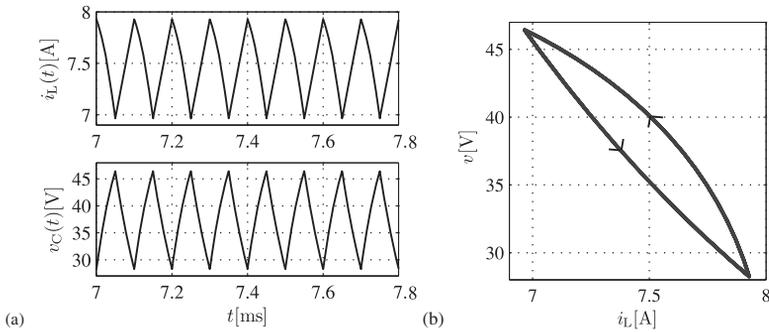


Fig. 1.19 Stationary behavior of the boost converter (a) and limit cycle (b).

disturbance $R(t) = 10 \Omega$. In Fig. 1.19(b) the stationary behavior represented by a limit cycle is shown. All parameter values of the converter are listed in Table 1.6.

Table 1.6 Values of circuit components.

Component	Value
E	20 V
L	1 mH
R_L	0.1Ω
C	$10 \mu\text{F}$
R_C	0.06Ω
T_S	0.1 ms
State	Value
i_L, v	$\in \mathbb{R} \text{ A, V}$
q	$\in \{1, 2, 3\}$
Input	Value/range
u_1	$\in [0, 1]$
Disturbance	Values/ranges
R	$\in \mathbb{R} \Omega (10 \Omega)$

Alternative models are considered in Example 5.1 where a DC-DC converter is represented as a complementarity system. One main focus on studying DC-DC converters is the controller synthesis. To stabilize the converter at a desired limit cycle new controller design methods are necessary. In Example 6.6 the design of the switching surfaces by means of an embedded map representing the converter at the switching time instances is explained. More application examples in the field of

power systems are discussed in [Chapter 13](#), where the DC-DC converter is treated in detail in [Section 13.2](#).

Bibliographical notes

There are several excellent introductions explaining the phenomena of hybrid dynamical systems, e.g. [112].

Hybrid systems are dealt with in monographs that all consider particular subclasses of hybrid systems, such as [571] focussing on complementarity systems, [342] on piecewise linear systems, [576] on quantized systems, and [401] on switched systems. Collections of papers on hybrid systems, other than the proceedings of the annual conferences on hybrid systems, can be found in [145, 224, 326, 344].

1.3 illustrates hybrid system behavior by means of a chaotic oscillator, which is described, for example, in [564, 636].

The running examples represent rather simplified descriptions of real-world applications of hybrid systems, which are developed on the basis of the references [84] for the two-tank system, [451] for the DC-DC converter, and [517] for the automatic gearbox.

controlengineers.ir

Survey of modeling, analysis, and control of hybrid systems

B. De Schutter, W. P. M. H. Heemels, J. Lunze, and C. Prieur

An overview of various modeling frameworks for hybrid systems is given followed by a comparison of the modeling power and the model complexity, which can serve as a guideline for choosing the right model for a given analysis or control problem with hybrid dynamics. Then, the main analysis and design tasks for hybrid systems are surveyed together with the methods for their solution, which will be discussed in more detail in subsequent chapters.

Chapter contents

2.1	Models for hybrid systems	page	33
2.1.1	Overview		33
2.1.2	Hybrid automata		35
2.1.3	Switched systems		35
2.1.4	Piecewise affine systems		36
2.1.5	Mixed logical dynamical systems		37
2.1.6	Complementarity systems		38
2.1.7	Discretely controlled continuous systems		39
2.1.8	Timed automata		40
2.1.9	Hybrid inclusions		40
2.2	Comparison of the models		41
2.2.1	Equivalence of model classes		41
2.2.2	Modeling power versus decisive power		41
2.3	Problems resulting from hybrid phenomena		42
2.3.1	Inadequacy of mono-disciplinary approaches		42
2.3.2	Instability of hybrid systems resulting from switching		43
2.3.3	Zeno behavior		45
2.3.4	Chattering or infinitely fast switching: sliding modes		47
2.3.5	Sensitivity and nondeterminism of the system behavior		48

2.4	Overview of solution approaches	49
2.4.1	Necessity for a novel theory on hybrid dynamical systems	49
2.4.2	Solution concepts and well-posedness	50
2.4.3	Controllability, observability, and stability	52
2.4.4	Control design	52
2.4.5	Observer design	53
2.4.6	Identification	54
2.4.7	Model checking and verification	54
2.4.8	Robust stability	54
2.4.9	Simulation	55

controlengineers.ir

2.1 Models for hybrid systems

2.1.1 Overview

As models are the ultimate tools for obtaining and dealing with knowledge, not only in engineering, but also in philosophy, biology, sociology, and economics, a search has been undertaken for appropriate mathematical models for hybrid systems. This section gives an overview of the modeling formalisms that have been elaborated in hybrid systems theory in the past.

Structure of hybrid systems Many different models have been proposed in literature, as will be seen in following chapters. These models can be distinguished with respect to the phenomena that they are able to represent in an explicit form. Consequently, these models have different fields of applications. The main idea of these models is described by the block diagram shown in Fig. 2.1, which is often used in literature as a starting point of hybrid systems modeling and analysis, although not all models use this structure in a direct way.

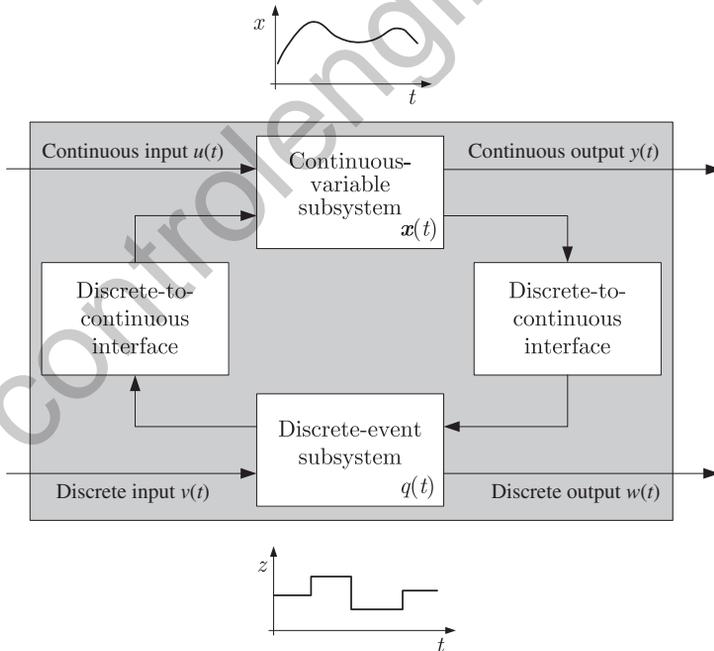


Fig. 2.1 Structure of hybrid systems.

The hybrid character of the system is reflected in this structure by the combination of a continuous-variable system shown in the upper part of the figure and a discrete-event system below. The continuous input and the continuous output are associated with the continuous subsystem and the discrete signals with the discrete subsystem. The continuous subsystem describes how the continuous state $x(t)$ of the hybrid system develops over time. The discrete subsystem characterizes the evolution of the discrete state $q(t)$.

As the signals occurring in both subsystems have different signal spaces, an interconnection between both model parts necessitates two interfaces, which are called the discrete-to-continuous interface or the continuous-to-discrete interface, respectively. The discrete-to-continuous interface associates with a discrete signal value that is generated by the discrete-event subsystem a continuous-valued signal that acts as an input of the continuous subsystem. This interface is also called an *injector*. The continuous-to-discrete interface transforms a continuous-variable signal into a signal with discrete signal space. It typically tests whether the continuous signal has exceeded a given threshold or, more generally, whether the continuous state $x(t)$ has reached a switching surface. The result is an *event*, which is described by the event name and the event time instant. Therefore, this interface is often called an *event generator* and are formed by *guards* and invariants in hybrid automata.

The models described below distinguish with respect to the way in which they represent the two kinds of subsystems shown in the figure and the interfaces.

Models for hybrid systems A whole range of possible model structures for hybrid systems has already been proposed, of which we present a partial list below:

- hybrid automata (Chapters 3);
- switched systems (Chapters 4);
- piecewise linear or piecewise affine models (Chapters 4);
- timed automata and timed or hybrid Petri nets (Section 2.1.8);
- differential automata;
- mixed logical dynamical models (Section 5.1);
- real-time temporal logics and timed communicating sequential processes;
- complementarity systems (Section 5.2);
- hybrid inclusions (Section 5.3).

This list of model classes is by no means exhaustive and we will not discuss them all here, but will only focus on the most well-known, which will also be treated in more detail in the following chapters. Additional references to other hybrid modeling formalisms not discussed here will be provided in the bibliographical notes at the end of the chapter.

The common feature of all the modeling paradigms and in fact of hybrid systems in general is the interaction of different dynamics. This also indicates that the model structure should mix two modeling formalisms. Typically, one might think of the interaction of time-driven models (governed by differential or difference equations) on one hand, and event-driven systems (described by, e.g., temporal logic, automata, finite-state machines, etc.) or logic rules on the other hand. In some way these features should be combined in one model structure. One generally accepted

manner of looking at hybrid systems is via hybrid automata, which have already been introduced in Section 1.1 and which can be seen as a cross product of finite-state machines and differential or difference equations (depending one whether a discrete-time or continuous-time formalism is used).

2.1.2 Hybrid automata

Hybrid automata result as an extension of finite-state machines by associating with each discrete state a system of differential or difference equations that describe how in this discrete state the continuous state evolves. The underlying modeling philosophy has already been explained in Section 1.2 and will be formally introduced in Chapter 3.

We would like to stress that it can be a nontrivial task to rewrite a physical model description in terms of a hybrid automaton. In particular, the definition of the guards, invariants, and reset maps (i.e. switching and re-initialization rules) can be really involved.

2.1.3 Switched systems

A quite general class of hybrid systems concerns *switched systems* given by

$$\dot{x}(t) = f_{q(t)}(x(t)), \tag{2.1}$$

where $x \in \mathbb{R}^n$ denotes the state and $q : \mathbb{R}_+ \rightarrow \{1, \dots, N\}$ is the switching signal that determines which vector field f_q with $q \in \{1, \dots, N\}$ is active at time $t \in \mathbb{R}_+$. For a fixed q , (2.1) describes a nonswitched system, which is sometimes called the subsystem of a switched system. In this context, switching means the currently active subsystem is changed to another one.

The switching can depend on time only as above, but it can also be a function of the state $x(t)$ at time t or of an external input, and it can even have memory in it.

In particular, when the switching only depends on the state variable $x(t)$ at the present time t , one speaks of *discontinuous dynamical systems* or *piecewise smooth systems*. An example is the system below, which switches between two dynamics as a result of inequalities in the state variable:

$$\dot{x}(t) = f(x(t)) = \begin{cases} f_-(x(t)), & \text{if } \phi(x(t)) < 0, \\ f_+(x(t)), & \text{if } \phi(x(t)) > 0. \end{cases} \tag{2.2}$$

The state space is separated into two parts by a hyper-surface defined by $\phi(x) = 0$ (Fig. 2.2). On one side of the surface $\mathcal{C}_+ := \{x \in \mathbb{R}^n \mid \phi(x) > 0\}$ the dynamics $\dot{x} = f_+(x)$ holds, on the opposite side $\mathcal{C}_- := \{x \in \mathbb{R}^n \mid \phi(x) < 0\}$ the dynamics $\dot{x} = f_-(x)$ is valid. Hence, one can also consider this system as a *differential equation with a discontinuous right-hand side* [237].

As the choice of whether the vector field f_- or f_+ is active at time t only depends on the state $x(t)$, no discrete state is necessary to describe such systems. In the light of the description of hybrid systems given in Chapter 1, systems of this class

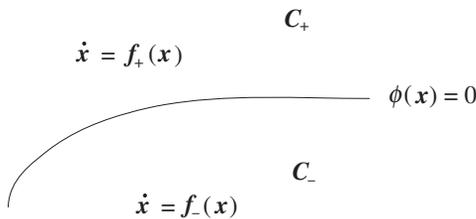


Fig. 2.2 Switching dynamics.

are not “real” hybrid system. However, they cannot be analyzed by standard methods elaborated for nonlinear systems, because the Lipschitz continuity of the vector field \mathbf{f} is lost at the switching surface and, consequently, the definition of solutions and well-posedness have to be extended with respect to nonlinear systems. Therefore, new methods for dealing with such systems have been developed in the field of hybrid dynamical systems, and q is said to be the discrete state or the operation mode of the system. The behavior of such a system will be discussed in more detail in [Chapter 4](#).

2.1.4 Piecewise affine systems

If the discontinuous dynamical system has affine dynamics in each region and the regions are polytopic, we obtain the well-studied class of *piecewise affine* (PWA) systems, which can be considered both in discrete or continuous time. Discrete-time PWA systems are described by

$$\begin{aligned} \mathbf{x}(k+1) &= \mathbf{A}_q \mathbf{x}(k) + \mathbf{B}_q \mathbf{u}(k) + \mathbf{f}_q & \text{for } \begin{bmatrix} \mathbf{x}(k) \\ \mathbf{u}(k) \end{bmatrix} \in \mathcal{C}_q, \\ \mathbf{y}(k) &= \mathbf{C}_q \mathbf{x}(k) + \mathbf{D}_q \mathbf{u}(k) + \mathbf{g}_q \end{aligned} \quad (2.3)$$

for the operation modes $q = 1, \dots, N$, where $\mathcal{C}_1, \dots, \mathcal{C}_N$ are convex polyhedra (i.e. given by a finite number of linear inequalities) in the input/state space with non-overlapping interiors. The variables $\mathbf{u}(k) \in \mathbb{R}^m$, $\mathbf{x}(k) \in \mathbb{R}^n$, and $\mathbf{y}(k) \in \mathbb{R}^l$ denote the input, state, and output, respectively, at discrete time step k with $k \in \mathbb{N}$.

PWA systems have been studied extensively as they form the “simplest” extension of linear systems that can still model many nonlinear and non-smooth processes with arbitrary accuracy and that are capable of handling some hybrid phenomena. [Chapter 4](#) gives a thorough survey of the results obtained for such systems.

Many authors also study the continuous-time variant of the above model, which is given by

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{A}_q \mathbf{x}(t) + \mathbf{B}_q \mathbf{u}(t) + \mathbf{f}_q & \text{for } \begin{bmatrix} \mathbf{x}(t) \\ \mathbf{u}(t) \end{bmatrix} \in \mathcal{C}_q, \\ \mathbf{y}(t) &= \mathbf{C}_q \mathbf{x}(t) + \mathbf{D}_q \mathbf{u}(t) + \mathbf{g}_q \end{aligned}$$

for $q = 1, \dots, N$, where $\mathcal{C}_1, \dots, \mathcal{C}_N$ are convex polyhedra, and the time t now evolves on the real line \mathbb{R} . In this chapter we will mostly focus on the discrete-time version as one can establish some relations to other well-known hybrid model

classes, such as mixed logical dynamical models and linear complementarity models (Section 5.3). In the following chapters the continuous-time PWA models will play a more dominant role.

Example 2.1 *Integrator system with saturation*

As a very simple example of a PWA model we can consider an integrator with upper saturation:

$$\begin{aligned}
 x(k+1) &= \begin{cases} x(k) + u(k) & \text{if } x(k) + u(k) \leq 1, \\ 1 & \text{if } x(k) + u(k) \geq 1, \end{cases} \\
 y(k) &= x(k).
 \end{aligned} \tag{2.4}$$

If we rewrite (2.4) as in (2.3) then we get

$$\begin{aligned}
 C_1 &= \{(x(k), u(k))^T \in \mathbb{R}^2 \mid x(k) + u(k) \leq 1\}, \\
 C_2 &= \{(x(k), u(k))^T \in \mathbb{R}^2 \mid x(k) + u(k) \geq 1\}, \\
 A_1 &= 1, \quad A_2 = 0, \quad B_1 = 1, \quad B_2 = 0, \\
 f_1 &= 0, \quad f_2 = 1, \quad C_1 = C_2 = 1, \\
 D_1 &= D_2 = 0, \quad g_1 = g_2 = 0.
 \end{aligned}$$

The system has two operation modes ($q = 1$ and $q = 2$) for which two sets of parameter matrices are described above. □

More details on PWA systems and additional results involving the control of PWA systems are presented in Chapter 4.

2.1.5 Mixed logical dynamical systems

In [62] a class of hybrid systems has been introduced in which logic, dynamics, and constraints are integrated. This resulted in the description

$$\mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{B}_1\mathbf{u}(k) + \mathbf{B}_2\boldsymbol{\delta}(k) + \mathbf{B}_3\mathbf{z}(k), \tag{2.5a}$$

$$\mathbf{y}(k) = \mathbf{C}\mathbf{x}(k) + \mathbf{D}_1\mathbf{u}(k) + \mathbf{D}_2\boldsymbol{\delta}(k) + \mathbf{D}_3\mathbf{z}(k), \tag{2.5b}$$

$$\mathbf{E}_1\mathbf{x}(k) + \mathbf{E}_2\mathbf{u}(k) + \mathbf{E}_3\boldsymbol{\delta}(k) + \mathbf{E}_4\mathbf{z}(k) \leq \mathbf{g}_5, \tag{2.5c}$$

where $\mathbf{x}(k)$ can contain both real and boolean (i.e. 1 or 0) components ($\mathbf{y}(k)$ and $\mathbf{u}(k)$ have a similar structure), and where $\mathbf{z}(k)$ and $\boldsymbol{\delta}(k)$ are respectively real-valued and boolean auxiliary variables. The inequalities (2.5c) have to be interpreted componentwise. Systems of the form (2.5) are called *mixed logical dynamical* (MLD) systems.

More detailed information on MLD systems is provided in Section 5.1.

2.1.6 Complementarity systems

Complementarity systems arise when differential equations

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{w}(t), \mathbf{u}(t)), \quad (2.6a)$$

$$\mathbf{z}(t) = \mathbf{g}(\mathbf{x}(t), \mathbf{w}(t), \mathbf{u}(t)) \quad (2.6b)$$

are coupled to “complementarity conditions”

$$0 \leq \mathbf{z}(t) \perp \mathbf{w}(t) \geq 0, \quad (2.6c)$$

where the inequalities are interpreted componentwise and \perp indicates the orthogonality between the vectors $\mathbf{z}(t)$ and $\mathbf{w}(t)$, i.e. $\mathbf{z}^\top(t)\mathbf{w}(t) = 0$. In the above description $\mathbf{x}(t)$ is the state, $\mathbf{u}(t)$ is the control input, and $\mathbf{w}(t), \mathbf{z}(t)$ are the complementarity variables. The complementarity conditions (2.6c) constitute a particular system of equalities and inequalities, which are related to the well-known relations between the constraint variables and Lagrange multipliers in the Karush–Kuhn–Tucker conditions for optimality, the voltage–current relationship of ideal diodes, the conditions between unilateral constraints and reaction forces in constrained mechanics, etc. As such, the complementarity framework includes mechanical systems with unilateral constraints, constrained optimal control problems, switched electrical circuits, piecewise linear systems, etc.

To reveal the hybrid nature of complementarity systems, observe that (2.6c) implies that $w_i(t) = 0$ or $z_i(t) = 0$ for each $i \in \{1, \dots, m\}$. As a consequence, the system (2.6) has 2^m modes. Each mode can be characterized by the active index set $J \subseteq \{1, \dots, m\}$, such that $z_i = 0$, if $i \in J$, and $w_i = 0$, if $i \in J^c$, where $J^c := \{1, \dots, n\} \setminus J$. For the mode corresponding to J the dynamics is given by the following system of differential and algebraic equations (DAEs):

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{w}(t), \mathbf{u}(t)), \quad (2.7a)$$

$$\mathbf{z}(t) = \mathbf{g}(\mathbf{x}(t), \mathbf{w}(t), \mathbf{u}(t)), \quad (2.7b)$$

$$z_i(t) = 0 \text{ if } i \in J \text{ and } w_i(t) = 0 \text{ if } i \in J^c, \quad (2.7c)$$

The evolution of system (2.6) will be governed by (2.7) for the mode corresponding to J as long as the remaining inequalities (the “invariant” in the terminology of hybrid automata) in (2.6c)

$$z_i(t) \geq 0 \text{ if } i \in J^c \text{ and } w_i(t) \geq 0 \text{ if } i \in J \quad (2.8)$$

are satisfied. Impending violation of (2.8) will trigger a mode change. As a consequence, during the evolution in time of the system several mode dynamics will be active successively and resets of the state vector might be necessary (think of constrained mechanical systems with impacts).

A particular class of complementarity systems, viz. linear complementarity systems, will be discussed in Section 5.2.

2.1.7 Discretely controlled continuous systems

In an important application of switched systems the switching signal is determined by a controller in dependence of the state x . The structure of discretely controlled continuous systems is shown in Fig. 2.3. The control input $q(t)$ is a discrete signal that prescribes the operation mode of the plant. This class of systems will be considered in detail in Section 6.5.

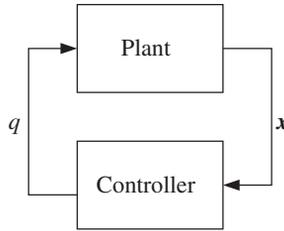


Fig. 2.3 Discretely controlled continuous system.

The difference between the switching schemes used in PWA or discretely controlled continuous systems is illustrated in Fig. 2.4. For PWA systems, the continuous state space \mathbb{R}^n is partitioned into sets \mathcal{C}_q ($q \in \mathcal{Q}$) and the q -th vector field f_q is valid as long as the state x remains in the set \mathcal{C}_q . Hence, the operation mode $q(t)$ at time t depends only on the current continuous state $x(t)$. In the figure, the trajectory $x(\cdot)$ first goes through the partition \mathcal{C}_2 and, hence, the vector field f_2 is valid. After crossing the boundary between the sets \mathcal{C}_2 and \mathcal{C}_1 , the future state trajectory is governed by the vector field f_1 .

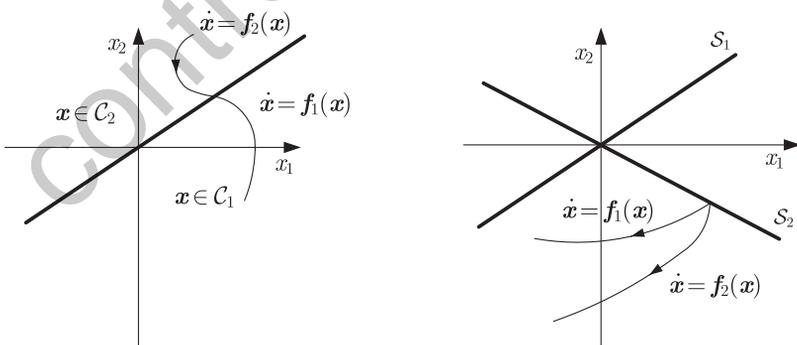


Fig. 2.4 Comparison of the switching schemes of piecewise affine and discretely controlled continuous system.

Contrary to this, discretely controlled continuous systems use a switching scheme that is usually described by switching surfaces \mathcal{S}_i , which are hyperplanes in the state space \mathbb{R}^n . If the state trajectory $x(\cdot)$ crosses such a surface at time \bar{t} , the discrete state is changed ($q \rightarrow q'$), where the successor state is determined by the switching surface touched at time \bar{t} . Whether the trajectory crosses the surface \mathcal{S}_i or not depends on the vector field that becomes active at time \bar{t} . In the figure, the new vector field f_2 moves the state back into the region from which it came. In the region “below” the surfaces \mathcal{S}_1 and \mathcal{S}_2 the state trajectory is first governed by f_1 and later by f_2 . Hence, the current discrete state q depends on the whole trajectory $x(\cdot)$ and not only on the current state $x(t)$. It is a “true” discrete state.

2.1.8 Timed automata

Timed automata are a class of hybrid automata that involve particularly simple continuous dynamics: all differential equations are of the form $\dot{x} = 1$, and all the invariants, guards, etc. involve comparison of the real-valued states with constants (e.g. $x = 1$, $x < 2$, $x \geq 0$, etc.). Clearly, timed automata are somewhat limited when it comes to modeling physical systems. They are very suited, however, for encoding timing constraints (such as “event A must take place at least 2 seconds after event B and not more than 5 seconds before event C”, etc.). For some applications, such as multimedia, internet, and audio protocol verification, this type of description is sufficient for both the dynamics of the system and the properties that we want the system to satisfy. Readers interested in the details of timed automata are referred to [13].

2.1.9 Hybrid inclusions

Hybrid inclusions form a natural extension of differential inclusions $\dot{x} \in F(x)$ in the sense that invariants, guards, and resets are added. Hybrid inclusions are given by the data of two subsets \mathcal{C} (the flow set) and \mathcal{D} (the jump set) of \mathbb{R}^n , and two set-valued mappings $F : \mathcal{C} \rightarrow \mathbb{R}^n$ and $G : \mathcal{D} \rightarrow \mathbb{R}^n$. The hybrid inclusion is then written as

$$\begin{aligned} \dot{x} &\in F(x) & \text{if } x \in \mathcal{C}, \\ x^+ &\in G(x) & \text{if } x \in \mathcal{D}. \end{aligned}$$

Clearly, this description provides compact models with a clear structure, which encompasses many hybrid phenomena. The hybrid inclusions turned out to be useful, e.g., in the general study of hybrid systems, and in the domains of networked control systems and reset control systems in particular.

2.2 Comparison of the models

2.2.1 Equivalence of model classes

It is of interest to know which type of model classes are equivalent or which model classes encompass others, as this can assist in transforming analysis and synthesis results and tools obtained for one class to another.

In [305] relationships between some classes of discrete-time hybrid models are presented, which will be summarized in [Section 5.3](#). There it will be shown that under mild assumptions discrete-time PWA systems are equivalent to some other classes of hybrid systems such as MLD systems and discrete-time linear complementarity systems. Specific analysis or design problems might be easier to solve using the formulation of one of the subclasses than that of another subclass. As a result, each system class has its own analysis and synthesis tools. As a consequence, it really depends on the problem and application at hand, which of these classes is best suited.

In the continuous-time framework, such broad equivalence relations are out of the question for the mentioned classes. However, there are relations between linear complementarity systems and other specific classes of non-smooth systems such as the “normal cone differential inclusions” and projected dynamical systems [124, 303].

Also for many simulation and verification tools it is of interest to transform model structures into others. The interchange format that is discussed in [Chapter 12](#) forms the means to transform models into certain basic formats, which can be used for various simulators and model checkers.

2.2.2 Modeling power versus decisive power

The choice of a suitable modeling framework is a trade-off between two conflicting criteria: the modeling power and the decisive power. The modeling power indicates the size of the class of systems allowing a reformulation in terms of the chosen model description. The decisive power is the ability to prove quantitative and qualitative properties of individual systems in the framework. A model structure that is too broad (like the hybrid automaton) cannot reveal specific properties of a particular element in the model class. The size of a model class is often taken too large for analysis purposes. As indicated by [92] and summarized in [Section 4.6](#), even for the easiest hybrid systems analysis and control problems are often undecidable, which means that, roughly speaking, there does not exist an algorithm that solves the problem and for which finite termination can be guaranteed. Even if the problems are decidable, then often the problems require a very high computational load to be solved (NP-complete or NP-hard, see [260] for details on these terms). Also for many classes of hybrid and timed automata the reachability problem (i.e. determining whether some of the trajectories of the system can attain a specific set of desired states) is undecidable [315].

Example 2.2 NP-hard analysis problem

As an example of an NP-hard problem, [92] considers the elementary hybrid system given by

$$\mathbf{x}(k+1) = \begin{cases} \mathbf{A}_1 \mathbf{x}(k) & \text{if } \mathbf{c}^T \mathbf{x}(k) \geq 0, \\ \mathbf{A}_2 \mathbf{x}(k) & \text{if } \mathbf{c}^T \mathbf{x}(k) < 0, \end{cases}$$

where $\mathbf{A}_1, \mathbf{A}_2$ are matrices and \mathbf{c} is a (column) vector of appropriate dimensions. The problem of deciding whether this switching system is stable is shown to be NP-hard. Loosely speaking, this means that there is no algorithm that answers the question of stability in polynomial time (as function of the size of $\mathbf{A}_1, \mathbf{A}_2$, and \mathbf{c}). \square

In conclusion, one can state that in certain cases it is useful to have some additional structure on the model class that one considers, as the hybrid automaton model is too broad for detailed analysis.

2.3 Problems resulting from hybrid phenomena

Hybrid systems are inherently nonlinear and non-smooth, and therefore many of the results available from the vast literature on linear systems and smooth nonlinear systems do not apply. As a consequence, many basic system-theoretic problems like well-posedness, stability, controllability, observability, safety, etc. and many design methods for controllers and observers have to be reconsidered within the hybrid context. Next, we will discuss several particular problems that complicate the resolution of these issues.

2.3.1 Inadequacy of mono-disciplinary approaches

Hybrid system already exist for a long time and engineers have managed to analyze and to design them. So, how did they manage to do so and what is new in hybrid systems theory?

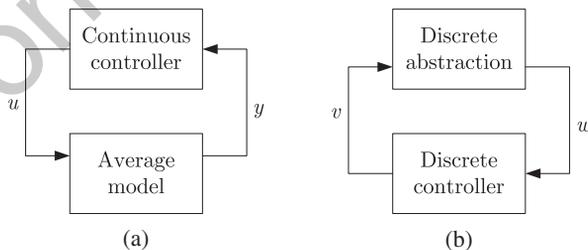


Fig. 2.5 Two ways of simplifying hybrid systems analysis and design: (a) continuous representation, (b) discrete-event representation.

Figure 2.5 shows two ways to deal with hybrid systems by avoiding the combination of continuous and discrete model parts. These approaches either abstract

from the discrete state evolution and end up with a purely continuous representation, or abstract from the continuous movement and represent the system by a discrete-event model. In both cases, either the methods elaborated in the theory of continuous systems or the theory of discrete-event systems alone are sufficient for analysis and design.

As an example of the approach presented in Fig. 2.5(a), consider the DC-DC converter described in Section 1.3.3. A discrete controller switches the operation mode in a very high frequency (usually measured in MHz), so that the voltage control problem can be solved by using an average model that describes the continuous evolution of an average state approximating the highly oscillating voltage.

On the other hand, if only the discrete-event behavior of the system is important, then the continuous state evolution can be ignored or reduced to its influence on the discrete state changes. The model that represents the discrete state transitions of a hybrid system is called a *discrete abstraction*. Methods used to obtain such abstractions are described in Section 6.3–6.5.

In both situations, the hybrid character of the system under consideration is ignored (or “abstracted away”). This may be reasonable under specific practical circumstances, because any model should not represent a system in the best possible way, but in a way suitable for solving the given task. However, the important consequence of the abstraction is the fact that typical phenomena of hybrid systems such as switching dynamics and state jumps can no longer be represented by the model and are, thus, excluded from the analysis and control design.

The main aim of hybrid systems theory is to elaborate analysis and design methods for technological systems for which both the continuous and the discrete state evolution play an important role and for which neither an abstraction from the continuous movement nor the exclusion of the discrete state transitions from the considerations is possible or suitable, and for which the interaction between both system parts thus has to be taken into account.

The following subsections describe phenomena that cannot be adequately modeled and analyzed by analysis methods purely tailored for continuous systems or for discrete-event systems. So methods for both system classes have to be combined in a suitable manner to deal with these phenomena.

2.3.2 Instability of hybrid systems resulting from switching

The fact that hybrid techniques have to be developed is evidenced by the study of stability of hybrid systems. Stability is a real “hybrid problem,” that cannot be tackled by studying, e.g., only the stability of the subsystems (except for some trivial examples). This is illustrated by the following switched system taken from [113].

Example 2.3 *Unstable hybrid system with stable operation modes*

Consider the hybrid system with two operation modes

$$\dot{x} = \begin{cases} A_1 x & \text{if } x_1 x_2 \leq 0, \\ A_2 x & \text{if } x_1 x_2 > 0, \end{cases} \quad (2.9)$$

with

$$A_1 = \begin{pmatrix} -1 & 10 \\ -100 & -1 \end{pmatrix} \quad \text{and} \quad A_2 = \begin{pmatrix} -1 & 100 \\ -10 & -1 \end{pmatrix}.$$

By inspection of the eigenvalues of A_1 and A_2 , which for both matrices are $\lambda_{1/2} \approx -1 \pm 31.62j$, one can see that both of the dynamics are stable (cf. the phase portraits in Fig. 2.6), but the switched system (2.9) is not (Fig. 2.7).

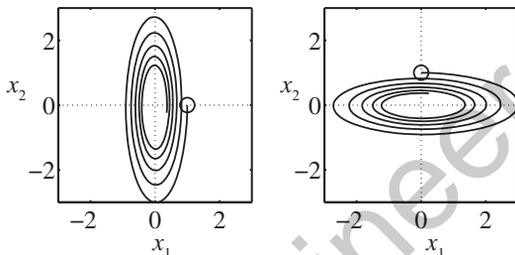


Fig. 2.6 Behavior of the stable linear submodels.

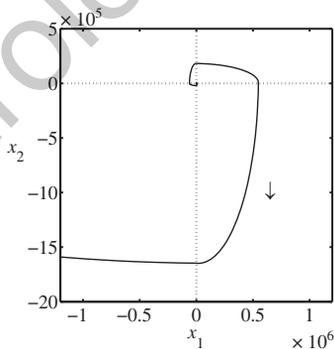


Fig. 2.7 Unstable switched system (2.9).

Indeed, the switched system activates dynamics 1 if the state lies in the second and the fourth quadrant, and dynamics 2 if the state is in the first or the third quadrant. As one can see from the trajectory shown in Fig. 2.7, the switched system is unstable. \square

Also converse examples exist where the subsystems are all unstable and the corresponding switched system is stable (e.g. consider the time-reversed version of the system above).

As a consequence, it is not sufficient to study only the stability properties of the subsystems, as the switching structure has to be taken into account as well. For many other properties (observability, controllability, etc.), this is also the case and hence, hinging on the available results for continuous and discrete-event models, a hybrid systems theory is needed.

2.3.3 Zeno behavior

The evolution of a hybrid system typically consists of smooth phases in which the discrete mode remains constant, separated by discrete events and actions. In the terminology of hybrid automata, the discrete events are often given by guards being enabled or invariants about to be violated and the discrete actions are mode switches and/or resets of the continuous part of the hybrid state variable.

Zeno behavior is the phenomenon that for a dynamical system an infinite number of events occur in a finite length time-interval.

This phenomenon is named after the ancient Greek philosopher Zeno of Elea.

Example 2.4 Bouncing ball

A famous example of a simple model of a mechanical system that exhibits Zeno behavior is the bouncing ball (Fig. 2.8).

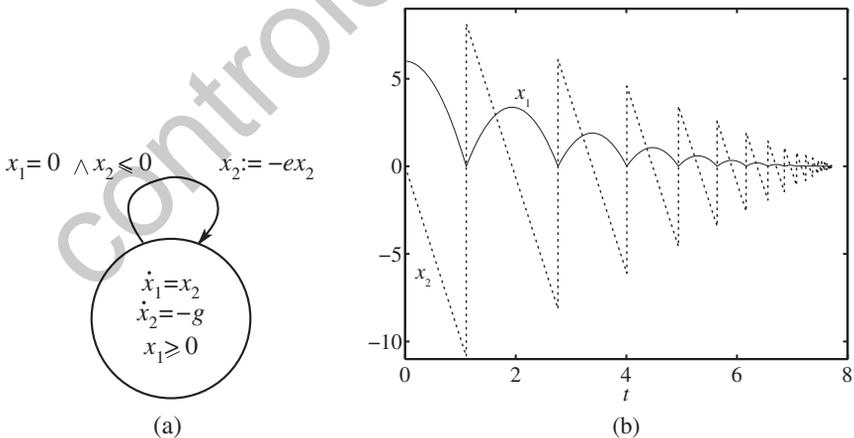


Fig. 2.8 Bouncing ball: hybrid automaton (a) and simulation (b) with $x = x_1$ and $\dot{x} = x_2$.

The height of the ball above the surface is denoted by x_1 with dynamics $\dot{x}_1 = -g$ and the constraint $x_1 \geq 0$. The velocity \dot{x}_1 of the ball will be denoted by x_2 . To complete the model we include Newton's restitution rule $x_2(\tau+) = -\alpha x_2(\tau-)$ when $x_1(\tau-) = 0$ and $x_1(\tau-) < 0$ ($0 < \alpha < 1$). The reset or event times $\{\tau_i\}_{i \in \mathbb{N}}$ can be easily computed and are given by

$$\tau_{i+1} = \tau_i + \frac{2\alpha^i x_2(0)}{g}, \quad \text{for } i \in \mathbb{N},$$

assuming that $x_1(0) = 0$ and $x_2(0) > 0$. Hence, $\{\tau_i\}_{i \in \mathbb{N}}$ has a finite limit equal to $\tau^* = 2x_2(0)/(g - g\alpha) < \infty$. So the continuous state $(x_1(t), x_2(t))$ converges to $(0, 0)$ when $t \uparrow \tau^*$. The physical interpretation is that the ball is at rest within a finite time span, but after infinitely many bounces. By extending the model (the complementarity framework presented above is quite natural for this) a continuation beyond τ^* can be defined by $(x_1(t), x_2(t)) = (0, 0)$ for $t > \tau^*$.

This is a specific example of Zeno behavior, i.e. a infinite number of events in a finite length time interval. In this case we have an infinite number of state re-initializations and the set of event times for the bouncing ball contains a so-called *right-accumulation point*. \square

This kind of Zeno behavior often prevents the existence of global solutions (i.e. defined for all times $t \in \mathbb{R}_+$) and, hence, is directly related to the well-posedness issue: the existence and uniqueness of solutions given an initial condition. In Section 5.4 it is also demonstrated that the definition of a solution trajectory can either allow for specific kind of Zenoness or not. Of course, depending on the choice of solutions, the well-posedness issue differs. Some interesting examples of this are presented in Section 5.4.

Also for the simulation of hybrid systems, Zenoness is a difficult problem. Imagine that one tries to determine a trajectory of a system like the bouncing ball by iteratively integrating over the smooth phases, estimating the event times (by detecting zero crossings), resetting the states, etc. Of course, this natural integration scheme (sometimes called "event-driven integration") would get stuck at or before the right-accumulation point as the interval lengths are decreasing to arbitrary small positive numbers. Some limiting procedure would be necessary to determine the accumulation point, which would be numerically hard to detect and to implement. Also the detection of zero crossings is crucial (as they determine the next mode for instance). As numerical simulators always produce approximation errors, the fact that hybrid systems do not generally have the property of *continuous dependence on initial conditions* (Section 2.3.5), the slightest error might lead to approximating a completely different trajectory. For certain specific classes of hybrid systems, so-called time-stepping schemes [3, 154] might be more suitable.

Not only for well-posedness or simulation issues, but also for analysis one has to be careful as certain properties of a system may only be true for trajectories that are defined on finite-length time intervals due to Zeno behavior. This might imply that the property does not hold true for all times (beyond the Zeno point). As a consequence, in case of verification of certain system properties, it is crucial which type of trajectories to include in the model of the plant (and hence, in the analysis). Definitely the mathematical behavior of the model should be "rich enough" to reflect the real plant or system behavior. What one often sees, is that, e.g., Zeno solutions are

excluded in the analysis of system's properties. In this case one has to realize that the property holds only for "non-Zeno" trajectories, while the actual system might have solutions beyond Zeno points and the system might fail in practice. One (in)famous example consisting of two tanks is given in [14]. There a controller is designed that keeps the fluid level for both tanks above a desired minimal level and this is proven to work for non-Zeno trajectories. As the (real) system has Zeno trajectories, this analysis is useless and it turns out that both tanks do get below the minimal level. Hence, this demonstrates that one has to be careful with excluding these type of phenomena, or stated differently, with the choice of solution concept used for the analysis.

The above aspects indicate that the Zeno phenomenon is a major problem in hybrid systems analysis and it is one of the challenges that the system and control theory for hybrid systems has to face. Hence, it is important to develop conditions that either exclude Zenoness or indicate when it is present. As the Zeno behavior can be seen as a modeling artifact that never can occur in reality, these conditions should be used to avoid Zeno behavior of the model.

2.3.4 Chattering or infinitely fast switching: sliding modes

For hybrid systems in general and the switched systems described in Section 2.1.3 in particular, also the presence of infinitely fast switching resulting in sliding behavior around switching surfaces can complicate the analysis (Section 5.4 for more details about sliding behavior). As Fig. 2.9 shows, chattering occurs if the vector fields f_- and f_+ that hold on either side of the surface S both point to the switching surface. Hence, the trajectory is forced to remain on S . Consequently, the surface is called a sliding surface. In practice, however, the state will not remain on this surface, but a fast switching will occur (that may overload the actuators).

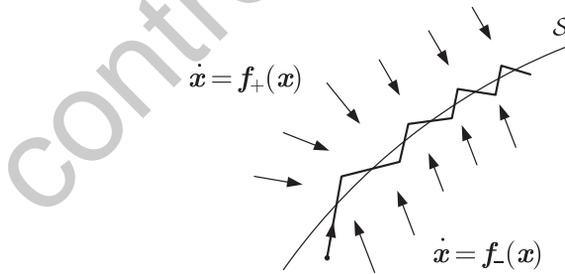


Fig. 2.9 Chattering and sliding modes.

Properties that might be true for the constituting dynamics (e.g., $\dot{x} = f_-(x)$ and $\dot{x} = f_+(x)$ in (2.2)) do not necessarily hold for the sliding modes induced by these dynamics. For instance, two stable systems might yield an unstable sliding mode

(even if there is a continuous piecewise quadratic Lyapunov function corresponding to each of the individual dynamics), as was indicated by Example Section 2.3.2 in [401] and is evidenced by the following example.

Example 2.5 Chattering of piecewise linear system

Consider the piecewise linear system (taken from [301])

$$\dot{x} = \begin{cases} A_1 x & \text{if } x_1 \geq 0 \\ A_2 x & \text{if } x_1 \leq 0 \end{cases} \quad \text{with } A_1 = \begin{pmatrix} -3 & 1 \\ -5 & 1 \end{pmatrix}, A_2 = \begin{pmatrix} -3 & -1 \\ 5 & 1 \end{pmatrix}.$$

This system allows a *continuous* piecewise quadratic Lyapunov function of the form $V(x) = x^T P_1 x$ when $x_1 \geq 0$ and $V(x) = x^T P_2 x$ when $x_1 \leq 0$ with

$$P_1 = \begin{pmatrix} 3.9140 & -2.0465 \\ -2.0465 & 1.5761 \end{pmatrix} \quad \text{and} \quad P_2 = \begin{pmatrix} 3.9140 & 2.0465 \\ 2.0465 & 1.5761 \end{pmatrix},$$

which are computed via the procedure outlined in [343]. These results prove the exponential stability of the system along “ordinary” solutions (without sliding motions).

However, the sliding mode dynamics at $x_1 = 0$ is given by $\dot{x}_2 = x_2$, which is unstable, in spite of the presence of a *continuous* piecewise quadratic Lyapunov function (satisfying $A_q^T P_q + P_q A_q < 0$ and $P_q > 0$ for $q = 1, 2$). \square

Hence, the presence of possible sliding motions has to be detected and included in a proper manner in analysis or synthesis methods.

2.3.5 Sensitivity and nondeterminism of the system behavior

The qualitative behavior of hybrid systems can be very sensitive to changes in the continuous initial state. As a consequence, the discrete behavior, which is represented by the sequence of discrete states, can be nondeterministic in the sense that for a small change in the initial continuous state x_0 it is unknown whether the discrete state sequence remains the same or changes drastically.

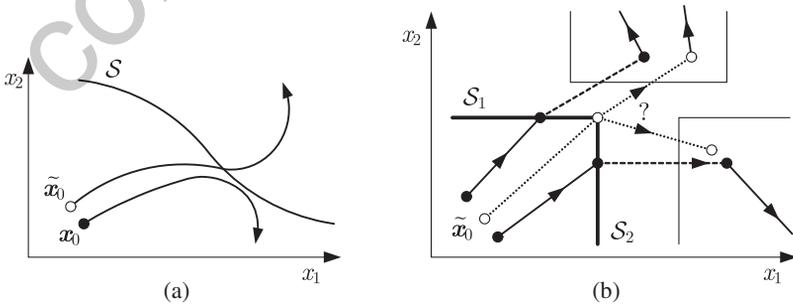


Fig. 2.10 Sensitivity of the hybrid system behavior.

The reason for this is illustrated in Fig. 2.10. Figure 2.10(a) shows two continuous state trajectories that start in nearby initial states x_0 . As one trajectory crosses the switching surface \mathcal{S} and the other does not, the continuations of the continuous state trajectories differ considerably. In particular, the hybrid system changes its discrete state in the first case, whereas it remains in the current discrete state in the second case. Note that for any given switching surface one can in general find a point at which the surface is tangential to the vector field. Consequently, the phenomenon shown here is not specific for this example, but occurs in many hybrid systems (for an example see [304]).

Another situation where this sensitivity becomes obvious is illustrated in Fig. 2.10(b), which shows state trajectories with state jumps. The jumps are invoked when the continuous state trajectory crosses one of the switching surfaces \mathcal{S}_1 or \mathcal{S}_2 . As the reset map \mathcal{R} introduced in Section 1.2 associates quite different state sets to the jumps that start at \mathcal{S}_1 or at \mathcal{S}_2 , respectively, the continuation of the continuous and the discrete state trajectories depends severely on which switching surface has been reached. Furthermore, there is an initial state x_0 for which the continuation of the behavior after the state jump cannot be unambiguously predicted (at least if the usual model uncertainties are taken into account, cf. the dotted line in the figure).

2.4 Overview of solution approaches

2.4.1 Necessity for a novel theory on hybrid dynamical systems

In this section we survey the main approaches that have been developed to deal with the new phenomena occurring in hybrid systems. The consequences for control system analysis and design are also explained.

A first look at hybrid systems raises a very basic question:

Does the class of hybrid systems necessitate a new theory?

This question is reasonable, because a mixture of continuous and discrete effects have been dealt with in control engineering for long. Think, in particular, of optimal control, where a time-optimal solution represents a “bang-bang” type control input or where inequality constraints can switch between an active and an inactive status. In control practice, gain scheduling methods or fuzzy control are used with some success and these methods are based on switching among different control laws in dependence upon the current operating conditions. Finally, programming logic controllers often use binary or multi-valued signals to switch between continuous processes.

The main difference between these methods and hybrid systems theory lies in the fact that the traditional methods mentioned above deal with specific control aspects where the co-existence of continuous and discrete dynamics can be tackled by specific analytical methods, whereas hybrid systems theory tries to elaborate methods that allow the introduction of combined continuous-discrete phenomena in a quite more general sense. The new theory starts with the hybrid phenomena explained in

Section 1.1.3 and aims at creating modeling formalisms and analysis methods that allow such phenomena to occur in arbitrary combinations.

The result of this approach is twofold. On the one hand, a very general class of hybrid systems can be dealt by the models that will be introduced in the following chapters. This class not only includes continuous models that are equipped with some discrete elements like inequality constraints, or discrete-event models that have been extended by some continuous properties like clocks. This system class is general enough to cover systems with intrinsic continuous and discrete phenomena. In particular, state jumps or vector fields with discontinuities cannot be dealt with by continuous systems theory or the methods mentioned above, which start from the basic assumption of a (Lipschitz) continuous vector field. For them, a new theory has to be elaborated.

On the other hand, as the price for this general approach, the modeling and design problems become very complex with many problems proved to be undecidable or at least NP-hard. Even if the problems are classified as “efficiently solvable” the computational complexity of real-world applications often exceeds the thresholds given by the available computing resources. In hybrid systems theory, the analytical complexity of continuous systems merge with the combinatorial complexity of discrete-event systems. Hence, the generality of the models has to be reduced systematically to end up with really applicable methods.

The difference between the classical control theory and its extensions and the more general approach of hybrid systems theory becomes obvious by the new problems explained above. Phenomena like Zeno behavior, instability occurring from stable systems, sensitivity and nondeterminism of the system behavior, etc., cannot be detected without using this general approach. Furthermore new challenges include the creation of reliable numerical methods for the reachability analysis and simulation of hybrid systems, the elaboration of verification methods for mixed continuously–discretely controlled systems or design procedures for event generators, and interfaces between continuous and discrete subsystems. These and further problems necessitate a new theory and the following chapters will show how far this theory is already developed.

2.4.2 Solution concepts and well-posedness

The new phenomena occurring in hybrid systems necessitate the definition of what a solution of a hybrid system is. Different solution concepts have been defined for the various model formats proposed, and even for a given model several solution concepts can be applied.

To illustrate the necessity of new solution concepts, remember that for continuous systems

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t)), \quad \boldsymbol{x}(0) = \boldsymbol{x}_0 \quad (2.10)$$

the classical solution is a continuously differentiable function $\boldsymbol{x} : [0, \infty) \rightarrow \mathbb{R}^n$ that satisfies the differential equation (2.10) for all times t . That is, the derivative $\dot{\boldsymbol{x}}(t)$ points into the direction of the vector field $\boldsymbol{f}(\boldsymbol{x})$ in all points $\boldsymbol{x} \in \mathbb{R}^n$. Many

existence conditions have been derived. In particular, a unique solution to (2.10) exists for all initial states x_0 if the function f is Lipschitz-continuous, which is a standard assumption in nonlinear systems theory.

For hybrid systems, the vector field f changes discontinuously as, for instance, in the scalar example

$$f(x) = \begin{cases} -a & \text{for } x > 0, \\ a & \text{for } x \leq 0, \end{cases} \quad (2.11)$$

with some $a > 0$. Then a solution in the classical sense may not exist for all initial conditions. In the example, there is no such solution for $x_0 = 0$.

New solution concepts have to release the requirement that the function $x(t)$ satisfies eqn. (2.10) for all times t . The Carathéodory solution is a solution that follows the vector field f for almost all times t but may violate the differential equation for a set of time points of measure zero. Alternatively, the Filippov solution is defined in terms of the vector field f , which is evaluated in a neighborhood of the point x and which leads to the set-valued function $\mathcal{F} : \mathbb{R}^n \rightarrow 2^{\mathbb{R}^n}$. For the example (2.11) this function is

$$\mathcal{F}(x) = \begin{cases} \{-a\} & \text{for } x > 0, \\ [-a, a] & \text{for } x = 0, \\ \{a\} & \text{for } x < 0, \end{cases}$$

which associates with $x = 0$ the interval $[-a, a]$. Then the solution of the original discontinuous differential equation is defined as a solution to the differential inclusion

$$x \in \mathcal{F}(x), \quad x(0) = x_0.$$

For the example, such a solution exists for $x_0 = 0$ and is given by $x(t) = 0$.

As a further illustration of the necessity of introducing new solution concepts, remember the discussion at the end of the section on Zeno behavior above. Studying a two-tank system with non-Zeno type of solutions might lead to the erroneous conclusion that the fluid levels in the tanks stay above specified target values, which is not the case in reality. Also the exclusion of specific solutions, e.g. by proving that Zeno behavior cannot occur in a system, might be beneficial for subsequent analysis.

In general a fundamental problem of interest is the well-posedness property of a system.

|| A system is said to be *well-posed* if a solution of the system exists and is unique given an initial condition (and possibly input signals).

The well-posedness property indicates that the system does not exhibit deadlock behavior (no solutions from certain initial conditions) and that determinism (uniqueness of solutions) is satisfied. In particular, when the hybrid models represent physical systems well-posedness is of interest. For certain other hybrid systems, the determinism may actually be too strong a requirement and one might focus only on the existence of solutions given initial conditions and inputs.

The topics of solution concepts and well-posedness will be explored in more detail in Section 5.4.

2.4.3 Controllability, observability, and stability

Notions such as controllability, stabilizability, observability, and detectability have played a central role throughout the history of modern control theory. Conceived by Kalman, the controllability concept has been studied extensively in the context of finite-dimensional linear systems, nonlinear systems, infinite-dimensional systems, hybrid systems, and behavioral systems. One may refer to, e.g., Sontag's book [600] for historical comments and references.

Outside the linear context, characterizations of global controllability have been hard to obtain. In the setting of smooth nonlinear systems, results have been obtained for local controllability, but there is no hope of obtaining general algebraic characterizations of controllability in the large. The complexity of characterizing controllability and stabilizability has been studied in [92] for some classes of hybrid systems, and the authors show that even within quite limited classes there is no algorithm to decide the controllability status of a given system. Hence, this indicates that there is no hope of finding complete conditions for general hybrid systems. The best that can be obtained seems to be characterizations for some specific classes of hybrid systems.

Controllability problems for piecewise linear systems and various related model classes have been studied in [65, 92, 160, 291, 394]. A similar story holds for observability and detectability [30, 65, 159, 183].

It has already been indicated that stability for hybrid systems is a very complex problem. For some classes of hybrid systems such as switched systems and PWA systems methods have been developed to analyze stability through various types of Lyapunov functions such as common quadratic Lyapunov functions, continuous piecewise quadratic Lyapunov functions, discontinuous piecewise quadratic Lyapunov functions, etc. These and other stability results for hybrid systems are presented in [169, 343, 401, 403, 452, 554, 676] and in Section 4.4, Section 4.5, and Section 6.2.

2.4.4 Control design

Several control methods have been developed for hybrid systems, or, more specifically, for special classes of hybrid systems.

For switched systems a wide body of literature exists on the development of stabilizing controllers using Lyapunov arguments and linear matrix inequalities (cf. [342, 452, 659] and Section 4.4, 4.5, and 6.2).

For a class of hybrid systems appearing in the context of manufacturing a control approach based on optimal control has been developed in [147, 170, 512] (Section 3.5).

Another important stream of control results for hybrid systems involves model-predictive control (MPC). MPC has originally been developed for linear systems in the pioneering work by Richalet, Cutler, and Ramaker [191, 556]. Recently it has also been extended to some classes of hybrid systems. MPC uses (on-line) optimization in combination with a prediction model and a receding horizon approach to determine control inputs that optimize the performance of the system over a given prediction

horizon subject to various operational and functional constraints on the inputs, states, and outputs of the system.

In [62] an MPC approach is presented for mixed logical dynamical (MLD) systems and thus also for PWA systems due to existing equivalence relations mentioned before. Additional results on MPC for MLD and PWA systems can be found in [63, 103, 388, 447] and Section 5.1. For related classes of hybrid systems MPC approaches have been developed in [484, 579].

2.4.5 Observer design

Observer design for hybrid systems is also a topic of interest. In many practical situations the full state is not available for feedback, while most control design methods mentioned so far are based on state feedback. As such, it is of interest to use output feedback controllers using, e.g., a “certainty equivalence principle” in which the state feedback will be based on an estimated state coming from an observer or another estimation scheme.

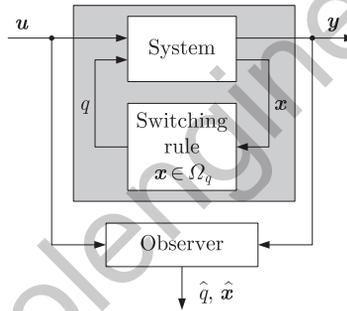


Fig. 2.11 State observation of hybrid systems.

Several interesting papers are available on observer design for hybrid systems, especially in the context of switched and piecewise linear systems. Nevertheless, the problem is still of interest as it turns out to be rather complicated, especially when the mode q of the system is not known or cannot be directly reconstructed on the basis of the measurements. This situation occurs, for example, if only the continuous input $u(t)$ and continuous output $y(t)$ are measurable and the discrete state q has to be estimated by means of this information (Fig. 2.11).

The situation becomes even more complicated when resets of the continuous state variable occur. A good starting point for investigating hybrid observer design are the following references: [9, 35, 41, 235, 350, 510, 516].

2.4.6 Identification

Models of hybrid systems typically contain parameters that cannot always be determined using a first-principles modeling approach. In such cases the parameters have to be determined based on input/output data. This process is called identification. Some results on identification of hybrid systems are described in [70, 236, 348, 559, 651] and Section 4.2. A comparison between various identification procedures is given in [347]. This comparison paper also presents interesting problems that hybrid identification approaches have to face.

2.4.7 Model checking and verification

The process of automatically analyzing the properties of systems by exploring their state space is known as *model checking*. This problem is only decidable for a few classes of hybrid systems, among which are timed automata [17, 315].

Timed automata were the first class of hybrid systems that were shown to be amenable to model checking methods [13]. Since then a number of other classes of hybrid systems with this property have been established: classes of multi-rate automata [16], classes of systems with continuous dynamics governed by constant differential inclusions [312], and classes of systems with continuous dynamics governed by linear differential equations [376]. It has also been shown that a very wide class of hybrid systems can be approximated arbitrarily closely by such “decidable” hybrid systems [545] (albeit at the cost of exponential computational complexity). For overview of the developments in this area see [17]. Additional results are presented in Chapter 9 and Section 10.3.

2.4.8 Robust stability

To indicate some of the robustness problems that can occur in hybrid systems, let us consider the following example taken from [355]:

$$x(k+1) = \begin{cases} 0, & \text{if } x(k) \leq 1, \\ 1, & \text{if } x(k) > 1. \end{cases}$$

This system is exponentially stable (actually every trajectory converges to the origin within two steps). However, the system has no robustness in the sense that the perturbed system

$$x(k+1) = \begin{cases} w(k), & \text{if } x(k) \leq 1, \\ 1 + w(k), & \text{if } x(k) > 1 \end{cases}$$

has solutions with $|x(k)| \geq 1$ no matter how small the bound ε on the disturbances $|w(k)| \leq \varepsilon$ is taken, indicating that robust stability is not present. It is well-known that for linear and smooth systems such a situation cannot arise.

In [543] it is precisely proved that, under some regularity assumptions, the stability of hybrid systems is robust with respect to sufficiently small vanishing perturbations. This fact reflects what has long been appreciated for continuous time systems [177, 390].

This “robustness for free” result is particularly important in the context of the design of hybrid stabilizers for nonlinear systems since in [543] it is claimed that any stabilizing hybrid feedback, with a discrete logic variable, meeting some basic regularity assumptions is robust, even if the logic variable does not converge to a finite set. This is a general result addressing robustness of such hybrid feedbacks; in [317, 319, 474, 540, 541, 542] robustness was established for particular hybrid feedbacks.

2.4.9 Simulation

Finally, note that in practice the most widely used technique for hybrid systems is computer simulation, via a combination of discrete-event simulation and differential algebraic equation (DAE) solvers. Some computer simulation and verification tools that are (also) used for hybrid systems are *BaSiP*, *Modelica*, *HyTech*, *KRONOS*, *Hybrid Chi*, *20-sim*, and *UPPAAL* (cf. Chapter 11). Simulation models can represent the plant with a high degree of detail, providing a close correspondence between simulated behavior and real plant behavior. This approach is, for any large system, computationally very demanding, and moreover it is difficult to understand from a simulation how the behavior depends on model parameters. This difficulty is even more pronounced in the case of large-scale hybrid systems that consist of many interacting modules. Fast simulation techniques based on variance reduction, and perturbation analysis techniques [144] have been developed in order to partially overcome these limitations.

Bibliographical notes

References that can be used as a starting point of a study of hybrid system models are for hybrid automata [115, 428], for switched systems [401], for piecewise linear or piecewise affine models [598, 599], for timed automata [13, 73], for the duration calculus [166], for timed or hybrid Petri nets [201, 324, 607], for differential automata [540, 617, 617], for mixed logical dynamic models [62], for real-time temporal logics [14, 527], for timed communicating sequential processes [202, 322], for complementarity systems [304, 572, 573], for hybrid inclusions [133, 271], for the max-min-plus scaling (MMPS) systems [577], for stochastic hybrid models [329, 533], and for Brockett’s model [120].

2.3 was first published in [113].

controlengineers.ir

Hybrid automata

S. Kowalewski, M. Garavello, H. Guéguen, G. Herberich, R. Langerak, B. Piccoli, J. W. Polderman, and C. Weise

Hybrid automata is a modeling formalism for hybrid systems that results from an extension of finite-state machines by associating with each discrete state a continuous-state model. Conditions on the continuous evolution of the system invoke discrete state transitions. A broad set of analysis methods is available for hybrid automata including methods for the reachability analysis, stability analysis, and optimal control.

Chapter contents

3.1	Definition	page	59
3.2	Abstractions in hybrid automata		62
3.2.1	Linear hybrid automata		62
3.2.2	Rectangular hybrid automata		63
3.2.3	Timed automata		63
3.3	Reachability analysis		64
3.3.1	Problem statement		64
3.3.2	Characterizing reachable space		67
3.3.3	Reachable space computation		69
3.3.4	Uncertainty		71
3.4	Stability analysis		72
3.4.1	Stability analysis using gain automata		72
3.4.2	Region stability		73
3.4.3	Structural stability		74
3.4.4	LaSalle's invariance principle		74
3.4.5	Lyapunov's indirect method		75
3.4.6	Stability analysis using formal methods		75

3.5	Optimal control	77
3.5.1	Control problem	77
3.5.2	The classical Pontryagin's Maximum Principle	78
3.5.3	Extension to a hybrid maximum principle	80

controlengineers.ir

3.1 Definition

A hybrid automaton is a transition system that is extended with continuous dynamics. It consists of locations, transitions, invariants, guards, n -dimensional continuous functions, jump functions, and synchronization labels. Various definitions exist in the literature which differ only in details. The following definition covers all of the aspects needed for the purpose of this handbook. A hybrid automaton consists of:

- a finite set of locations $\mathcal{Q}, q \in \mathcal{Q}$;
- a finite transition relation $\Theta \subseteq \mathcal{Q} \times \mathcal{Q}$

for the specification of the discrete dynamics. The locations can be seen as discrete states (also called *control modes*), in other words as the discrete part of the hybrid state space \mathcal{H} . Transitions from one control mode to the next are often called *control switches*. The continuous dynamics is described by:

- a finite and indexed set of continuous variables $V = \{x_1, x_2, \dots, x_n\}$, often written as a vector $\mathbf{x} = (x_1, \dots, x_n)$;
- a real-valued activity function $\mathbf{f} : \mathcal{Q} \times \mathbb{R}^n \rightarrow \mathbb{R}^n$, often defined by a continuous differential equation $\dot{\mathbf{x}} = d\mathbf{x}/dt = \mathbf{f}(q, \mathbf{x})$.

For each control location, the activity function defines a vector field describing the behavior of each variable x_i over time. The variables are often identified with the continuous functions or with their valuation, i.e. the values of the function in a given point. Note that in the general case the activity function can be nondeterministic, i.e. a relation instead of a function, but this is omitted for clarity in the following. The hybrid state space is the set of all possible values for the control location and the variables: $\mathcal{H} := \mathcal{Q} \times \mathbb{R}^n$. A hybrid state $s = (q, \mathbf{x})$ is a pair of a location q and a *valuation* \mathbf{x} of the continuous variables. Obviously the hybrid state space is infinite.

Definition 3.1 (Hybrid Automaton) A hybrid automaton H is a tuple

$$H = (\mathcal{Q}, \mathcal{V}, \mathbf{f}, \text{Init}, \text{Inv}, \Theta, \mathcal{G}, \mathcal{R}, \Sigma, \lambda),$$

where:

- $\mathcal{Q} = \{q_1, \dots, q_k\}$ is a finite set of discrete states (control locations);
- $\mathcal{V} = \{x_1, \dots, x_n\}$ is a finite set of continuous variables;
- $\mathbf{f} : \mathcal{Q} \times \mathbb{R}^n \rightarrow \mathbb{R}^n$ is an activity function;
- $\text{Init} \subset \mathcal{Q} \times \mathbb{R}^n$ is the set of initial states;
- $\text{Inv} : \mathcal{Q} \rightarrow 2^{\mathbb{R}^n}$ describe the invariants of the locations;
- $\Theta \subseteq \mathcal{Q} \times \mathcal{Q}$ is the transition relation;
- $\mathcal{G} : \Theta \rightarrow 2^{\mathbb{R}^n}$ is the guard condition;
- $\mathcal{R} : \Theta \rightarrow 2^{\mathbb{R}^n} \times 2^{\mathbb{R}^n}$ is the reset map;
- Σ is a finite set of synchronization labels;
- $\lambda : \Theta \rightarrow \Sigma$ is the labeling function.

The automaton H describes a set of (hybrid) states $(q, \mathbf{x}) \in \mathcal{H} = \mathcal{Q} \times \mathbb{R}^n$.

A graphical representation of a hybrid automaton has already been given in Fig. 1.9. It uses the automaton graph known as a representation of nondeterministic automata and extends it by the continuous dynamics and the interaction between the continuous and the discrete dynamics.

To model synchronization between several hybrid automata running in parallel, the definition of a hybrid automaton usually also includes:

- a finite set of synchronization labels Σ with a labeling function $\lambda : \Theta \rightarrow \Sigma$.

The behavior over time of a hybrid automaton is defined as follows. Time is passing while the system is in a specific location whereas the discrete transitions are supposed to happen instantaneously; there is no continuous evolution of the variables during a discrete transition. In order to specify the continuous evolution in a location, the activity function associates a vector field with each location. The vector field may differ for different locations. After a discrete transition, continuous evolution proceeds according to the vector field defined for the new location. The system behavior is further modeled more precisely by means of:

- a finite set of initial states $Init \subseteq \mathcal{H}$
- an invariant mapping $Inv : \mathcal{Q} \rightarrow \mathbb{R}^n$;
- a guard mapping $\mathcal{G} : \Theta \rightarrow 2^{\mathbb{R}^n}$;
- a reset mapping $\mathcal{R} : \Theta \times 2^{\mathbb{R}^n} \rightarrow 2^{\mathbb{R}^n}$.

The invariant Inv is the domain of the vector field f . It restricts the continuous evolution within a location and specifies a set of admissible valuations for the continuous variables. As soon as the invariant condition is violated—that is, the boundaries of this set are hit—the system is forced to leave the current location by taking a discrete transition.

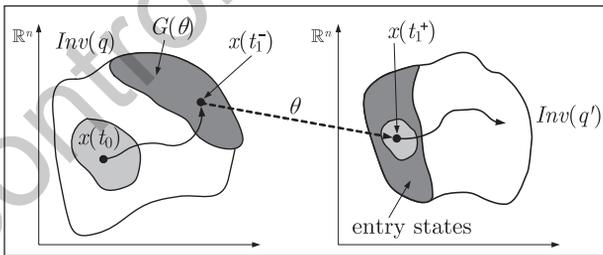


Fig. 3.1 Transition semantics of a hybrid automaton.

The discrete dynamics can be restricted using guards. A discrete transition is enabled if the guard condition is true, i.e. the valuation of the continuous variables must fulfill the guard. During the transition the continuous variables are reinitialized according to the reset mapping. If $\mathcal{R}(q, q', \mathbf{x}) = \mathbf{x}'$, then the variables are set to \mathbf{x}' after taking the transition from q to q' , if their value was \mathbf{x} before. For a given set \mathcal{B}

of valuations, the result of $\mathcal{R}(\theta)$ is the set $\{\mathbf{x}' \mid \exists \mathbf{x} \in \mathcal{B}. (\mathbf{x}, \mathbf{x}') \in \mathcal{R}(\theta)\}$. We will abuse notation and write $\mathcal{R}(\theta, B)$ for this set.

Whenever both invariant and guard condition are simultaneously true, there is a choice between continuous and discrete evolution. A transition is therefore said to be deterministic, if (1) it has a unique destination and (2) whenever this transition is possible, continuous evolution is impossible. Determinism or nondeterminism of transitions is hence a modeling issue.

The transition semantics of a hybrid automaton is illustrated by a two-dimensional example as shown in Fig. 3.1: the initial hybrid state is $h_0 = (q, \mathbf{x}(t_0)) \in \text{Init}$. In order to model admissible behavior, the continuous part of the set of initial states should be a subset of the invariant of the corresponding initial location. The system now starts evolving continuously until $\mathbf{x}(t_1^-) \in \mathcal{G}(\theta)$, where $\theta = (q, q')$, is reached. For any $\mathbf{x} \in \text{Inv}(q) \cap \mathcal{G}(\theta)$ there is a choice between further continuous evolution or discrete transition. Here, the discrete transition θ is taken at $\mathbf{x}(t_1^-)$. The state vector \mathbf{x} is then reinitialized instantaneously during the transition. In this example, \mathbf{x} is reset non-deterministically to the valuation $\mathbf{x}(t_1)$ that lies inside a specified range of possible reset values. $\mathbf{x}(t_1)$ has to be a subset of the invariant of the successor location q' . (The whole set of possible entry states is obtained by applying the reset to the guard.) Then, continuous evolution is restarting from $\mathbf{x}(t_1)$ according to the vector field of q' . (Note that from a dense-time point of view, $t_1^- \equiv t_1$ as the discrete transition is supposed to occur instantaneously. t_1^- and t_1 are only used in order to distinguish between the moment before the reset and the one after the reset.) Note that in the literature, a combination of a guard and a reset function is often used under the name of *jump* function. It is also common practice to include the functionality of guard and reset in the specification of the transition set.

Example 3.1 Hybrid automata representation of the two-tank system

We use the example of the two-tank system described in Section 1.3.1 to illustrate the modeling of hybrid systems using hybrid automata.

Remember that the two-tank system has two continuous variables describing the liquid levels. The values of continuous variables such as liquid level or temperature cannot jump. As the two-tank system is an autonomously switched system, the location graph and the invariants naturally emerge from the switching behavior of the system (as can be seen in Fig. 1.1 of the running example). Furthermore it follows that the discrete transitions between the four modes are deterministic and guards are therefore not needed. Moreover, a reset mapping of continuous variables without jump can only be an identity function and is thus negligible here. What remains to be specified are the vector fields associated with the four modes. The vector fields are given by \dot{h}_1 and \dot{h}_2 depending both on the flow $Q_{12}^{V_1}$ which depends in turn on the four modes. The vector fields (and thus the activity function) are therefore fully determined. It can easily be verified that for, e.g., location 1, the vector fields are given by

$$\dot{h}_1 = \frac{u_{P_1}(t) - Q_{12}^{V_2}(t) - Q_L^{V_{1L}}(t)}{A},$$

$$\dot{h}_2 = \frac{Q_{12}^{V_2}(t) - Q_L^{V_{2L}}(t) - Q_N^{V_3}(t)}{A} \quad \square$$

3.2 Abstractions in hybrid automata

In the previous section, we have defined a general formal model for hybrid systems. In this section we are looking at the applicability of this model to formal methods such as verification [173] or synthesis. These methods involve a search over the state space of the model. However, the state space of hybrid automata can be (and usually is) infinite (it is often even uncountable). We can nevertheless apply a computational verification or synthesis procedure that has originally been developed for finite-state machines, if we can construct a finite-state quotient transition system of the hybrid automaton. Finite-state quotients of hybrid automata fall mainly into two classes:

- property-preserving abstractions: equivalences w.r.t. a given property;
- abstractions which are sufficient for drawing conclusions about the originating model: approximations.

If a property-preserving abstraction cannot be constructed, abstractions with sufficient property preservation can be used for approximations. In the latter case, conclusions often work one-way only: if the abstraction shows that a property is fulfilled, then this is true for the original model, but if the property is not fulfilled, then nothing can be said about the original model. If in contrast we are able to construct an abstraction that preserves all properties of interest, then verifying a property for the abstraction is *equivalent* to verifying that property for the original model. If the abstraction can be effectively constructed, and the property can be checked on finite state systems, then the problem is decidable. How to find that abstraction depends on the properties one wants to preserve. These are frequently related to the temporal behavior of a system. Reasoning about this kind of properties can be done by means of temporal logics [12] such as linear temporal logic (LTL) or computation tree logic (CTL) and their timed extensions. The corresponding property-preserving abstractions are given by language-equivalence relations for linear temporal properties and bisimulations for branching temporal properties respectively [605]. Notice that finite language-equivalence quotients are coarser than bisimulations. In the following we present some suitable abstractions.

3.2.1 Linear hybrid automata

The most general abstraction of hybrid automata that is still subject to push button verification methods are the so-called linear hybrid automata. The use of the attribute “linear” has often caused misunderstandings. Note that in this context linear is not referring to the activity function but only to the bounds on the dynamics. In order to define a linear hybrid automaton formally, we use the following notions:

- linear term: $k_0 + k_1 X_1 + \dots + k_n X_n$ (k_i : integer constants, X_i : formal variables for the elements of \mathbf{x});
- linear inequality: $t_1 \circ t_2$, $\circ \in \{\leq, <, >, \geq\}$ (t_1, t_2 : linear terms);
- convex predicate: finite conjunction of linear inequalities.

A linear hybrid automaton is then defined such that the flow function f , the invariant Inv and the guard \mathcal{G} are convex predicates. The geometric representation of a convex predicate is a convex polyhedron in \mathbb{R}^n . One can imagine a *set* of continuous states to be grouped inside a polyhedron. The polyhedron for the guard includes all states for which the guard condition is true. The same holds for the polyhedron representing the invariant. Concerning the flow function, the polyhedron defines *bounds* on the derivatives of the continuous variables such that an over-approximation of the continuous dynamics (a differential inclusion as a generalization of the concept of ordinary differential equation) is specified and usually not the the exact flow function itself. In the literature one can find slightly different definitions of linear hybrid automata, e.g. using rational constants instead of integer constants. However, rational constants can easily be converted into integer constants by multiplication with a suitable constant. In the literature, there are many variants of linear hybrid automata [309] which mostly coincide with the above definition. Some versions are substantially different, e.g. [16] where a *linear hybrid system* is defined to be a system restricted to linear functions, i.e. constant flows.

3.2.2 Rectangular hybrid automata

A special case of a linear hybrid automaton is the so-called rectangular hybrid automaton: the geometric representation of a set of continuous states is here a hyper-rectangle instead of a convex polyhedron. This implies that no comparison between continuous variables is allowed in linear inequalities and therefore mutual independence of the continuous variables. The flow function f , the set of initial states $Init$, the invariant Inv , the guard \mathcal{G} , and the reset \mathcal{R} are consequently restricted to intervals. Formally, a rectangle R of dimension n is defined as the product of n intervals $I_i \subseteq \mathbb{R}$, each with rational endpoints (including $+$, $-\infty$): $R = \prod_i I_i$.

Initialized rectangular hybrid automata For initialized rectangular hybrid automata, whenever the flow interval of a continuous variable changes along a discrete transition, i.e. the successor location has a different flow interval than the originating location, then the variable has to be reinitialized. This means it is set non-deterministically to a new value within a given finite range of possibilities. It can be shown that for initialized rectangular hybrid automata, a finite language-equivalence quotient can be constructed. By this important result, reachability can be reduced to finitary time-abstract trace inclusion [309, 380] and is therefore decidable for this class of automata. However, already slight generalizations of initialized rectangular hybrid automata lead to undecidability of the reachability problem. Initialized rectangular hybrid automata therefore characterize a boundary between decidability and undecidability for reachability.

3.2.3 Timed automata

A further specialization of linear hybrid automata are timed automata [13]—a model suited for the formalization of real-time systems. Here, the continuous variables—called clocks—all evolve at a constant rate $\dot{x} = 1$. They can only be reset to zero

and are then restarted immediately. The clock constraints used to specify invariants, guards and initial states are convex predicates as for linear hybrid automata, restricted to a comparison of a maximum of two clocks.

The infinite state space of a timed automaton can be partitioned into so-called regions [13]. The equivalences for the construction of regions are based on the fact that, for two states with the same location, the runs starting from them are similar, if the clocks have the same integral part (needed to specify if a clock constraint is met) and agree on the ordering of their fractional parts (in order to decide which clock will first change its integral part).

It can be shown that this finite region graph is a timed bisimulation of the originating timed automaton. The extension of CTL by clocks is called Timed CTL (TCTL) and the region graph is TCTL-preserving. As long as only two clocks are compared in a linear inequality, TCTL-model checking is decidable. However, comparing three clocks x, y, z in a clock constraint such as $x - y < z$ already leads to undecidability of reachability.

Stopwatch automata If the continuous variables are also allowed to be stopped, i.e. their slope can be either 1 or 0, we speak of a stopwatch automaton. This is an important distinction to a timed automaton because adding one single stopwatch to a timed automaton already leads to undecidability of reachability.

For a theoretical background, [17, 315] is recommended. Note that the abstractions introduced in the sections above are all built by restricting the continuous dynamics. Another approach of abstraction is to restrict the discrete dynamics as is the case for the so-called *o-minimal hybrid systems* [376]. For these systems, a finite bisimulation can be guaranteed. However, this applies only to very restrictive classes of continuous dynamics. On the algorithmic side, a good overview is given in [73]. Algorithms for timed automata are complex, but speed up can be achieved, see, e.g., [54, 656].

3.3 Reachability analysis

3.3.1 Problem statement

The aim of reachability analysis is to consider the regions of the hybrid state space that are reachable by the hybrid automaton from the initial region. A state belongs to this reachable space if there exists a trajectory starting in one point of the initial region that eventually passes through this state. However the number of potential initial points may be infinite and the hybrid automaton is not necessarily deterministic. For example when the flow is defined by differential inclusion, then admitting an infinity of trajectories for one initial point. It is, therefore, impossible to compute all the trajectories of the automaton and set computations have to be considered.

As stated in Section 3.1, a trajectory of a hybrid automaton is a succession of continuous and discrete transitions. It is possible to define the set of continuous successors $\text{Succ}_C(\mathcal{R})$ of a set of region \mathcal{R} as the set of points that are reachable from the

states in \mathcal{R} by continuous transitions while remaining in the same location. It is also possible to define the set of discrete successors $\text{Succ}_D(\mathcal{R})$ of a set of regions \mathcal{R} as the hybrid states that are reachable from these regions by a discrete transition. If we consider a given location q and a given transition $\theta = (q, q')$ the state set reachable from the region (q, \mathcal{D}) (where $\mathcal{D} \subset \text{Inv}(q)$) is the image of the reset function of the intersection of the guard of the transition with the domain \mathcal{D} :

$$\text{Succ}_D((q, \mathcal{D}), \theta) = (q', \mathcal{R}(\theta, (\mathcal{G}(\theta) \cap \mathcal{D}))). \quad (3.1)$$

The set $\text{Succ}_D(\mathcal{R})$ is then the union for all situations, relevant with \mathcal{R} , of the union for all transitions of these sets.

From the sets of discrete and continuous successors it is possible to define a hybrid successor set of a region as the continuous successor set of its discrete successor set:

$$\text{Succ}_H(\mathcal{R}) = \text{Succ}_C(\text{Succ}_D(\mathcal{R})). \quad (3.2)$$

The reachable space from the initial region Init then appears as the smallest set of hybrid regions such that it includes the continuous successor set of the initial region and is equal to its hybrid successor set:

$$\begin{aligned} \text{Succ}_C(\text{Init}) &\subset \text{Reach}(\text{Init}), \\ \text{Succ}_H(\text{Reach}(\text{Init})) &= \text{Reach}(\text{Init}). \end{aligned} \quad (3.3)$$

Reachability analysis for hybrid systems is then based on the analysis of discrete and continuous successors sets. Relative to the nature of the \mathcal{R} functions the discrete part may be difficult. However, the main difficult point is the continuous successors set. If we consider a hybrid region (q, \mathcal{D}) , the continuous reachable space from \mathcal{D} while q remains active is specified by (3.4), where $\Phi(t)$ denotes a continuous trajectory specified by the flow that starts in one point of the domain and that is included in the invariant of the location:

$$\begin{aligned} \text{Succ}_C((q, \mathcal{D})) &= \{(q, \mathbf{x}) | \exists \mathbf{x}_0 \in \mathcal{D} \exists t_0 \geq 0 (\Phi(0) = \mathbf{x}_0 \wedge (\Phi(t_0) = \mathbf{x}) \\ &\quad \wedge (\forall t \in [0, t_0] (\Phi(t) \in \text{Inv}(q)) \wedge (\dot{\Phi}(t) \in f(q, \Phi(t)))))\}. \end{aligned} \quad (3.4)$$

This set is specified by two existential quantifiers that are to be eliminated to use it, e.g., to compute an intersection. This elimination, and especially time elimination, is the core problem in continuous reachability analysis.

Example 3.2 *Simple continuous dynamics*

Let us consider a system with a location specified by its invariant (3.5) and its flow (3.6) and let us analyze the continuous reachable space from the domain specified by (3.7):

$$\text{Inv}(q) = \{\mathbf{x} \mid (x_1 \geq 0) \wedge (x_2 \geq 0) \wedge (x_1 + x_2 \leq 2)\}, \quad (3.5)$$

$$\dot{\mathbf{x}} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \mathbf{x}, \quad (3.6)$$

$$\mathcal{X}_0 = \{\mathbf{x} \mid (x_1 \in [1, 2]) \wedge (x_2 = 0)\}. \quad (3.7)$$

From (3.6) it is possible to compute the transition matrix of the continuous system and then to characterize, for the continuous system without constraints, the set $\mathcal{T}(\mathbf{x}_0)$ of states on the trajectories starting in the initial state \mathbf{x}_0 by (3.8). From this equation it is then possible to eliminate time and get the explicit characterization of $\mathcal{T}(\mathbf{x}_0)$ given by (3.9):

$$\mathcal{T}(\mathbf{x}_0) = \left\{ \mathbf{x} \mid \exists t \geq 0, \mathbf{x} = \begin{pmatrix} \cos t & -\sin t \\ \sin t & \cos t \end{pmatrix} \mathbf{x}_0 \right\}, \quad (3.8)$$

$$\mathcal{T}(\mathbf{x}_0) = \{\mathbf{x} \mid x_1^2 + x_2^2 = x_{0,1}^2 + x_{0,2}^2\}. \quad (3.9)$$

From the analysis of $\mathcal{T}(\mathbf{x}_0)$, it is possible to determine that, for the points in \mathcal{X}_0 , if $x_{1,0} \in]\sqrt{2}, 1]$ the trajectory hits the boundary of the invariant. So the reachable space is specified by (3.10) shown in Fig. 3.2:

$$\begin{aligned} \text{Succ}_C(q, \mathcal{X}_0) = \{ & (q, \mathbf{x}) \mid (x_1^2 + x_2^2 \geq 1) \wedge (x_1 \geq 0) \wedge (x_2 \geq 0) \\ & \wedge [(x_1^2 + x_2^2 \leq 2) \vee ((x_1 + x_2 \leq 2) \wedge (x_1 \geq 1))]\}. \quad \square \end{aligned} \quad (3.10)$$

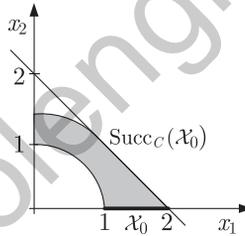


Fig. 3.2 Reachable space for Example 3.2.

This simple example illustrates the main activities that are necessary to compute the continuous reachable space:

1. solve the differential equation or inclusion;
2. eliminate time;
3. analyze the trajectories to determine the borders of the reachable domain.

This is only one step of the global hybrid reachability analysis and intersections of the obtained domains with guards and images of these intersection by “reset” functions are also to consider. It is then mandatory to have representation of the continuous domain that can be used to perform the needed operations. Generally domains such as polyhedrons, zonotopes, or ellipsoids are used because they offer compact representations and computation capacities.

Reachability analysis is generally not an objective on its own but a step in a more general analysis. Such activities are, e.g., safety analysis or controller synthesis for safety requirements. The aim of the reachability analysis is then to help to determine whether it is possible to reach a forbidden region or not. As illustrated in Fig. 3.3 the exact reachable space $\text{Reach}(Init)$ is not necessary to answer to this question and an over-approximation $\text{O_Reach}(Init)$ may be sufficient and is then considered.

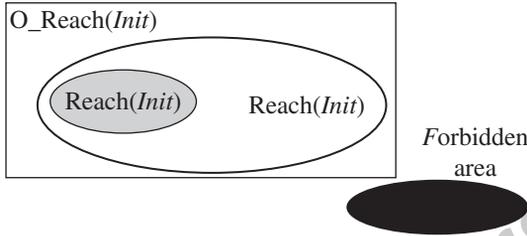


Fig. 3.3 Reachable space and over-approximation.

3.3.2 Characterizing reachable space

As stated above, reachability analysis is often used to decide whether a region is reachable or not and the exact value of the reachable space is not needed. It is, therefore, possible to search for a characterization of the reachable space that leads to the expected conclusion.

If the continuous dynamics of Example 3.2 is considered, (3.11) holds. Therefore, it is possible to state that (3.12) characterizes the reachable space from x_0 and then that a domain such as, for example $x_0^T x > x_0^T x_0$, is not reachable.

$$x_0^T x(t) = x_0^T x_0 \cos(t), \tag{3.11}$$

$$-x_0^T x_0 \leq x_0^T x \leq x_0^T x_0. \tag{3.12}$$

This example is specific, but for some classes of hybrid systems it is possible to find generic means to derive constraints. This is the case for linear systems when the matrix A that defines the dynamics have some real left eigenvectors w . For all points x of a trajectory starting in x_0 the product $w^T x$ is given by

$$w^T x = e^{\lambda t} w^T x_0. \tag{3.13}$$

The norm of this product then increases or decreases according to the sign of the associated eigenvalue λ . From considerations on the greater and smaller initial value of the product on the domain it is then possible to deduce a constraint that characterize the reachable space. For example in Fig. 3.4 if the eigenvalue associated to the left eigenvector w is negative the reachable space is characterized by $w^T x \leq a$

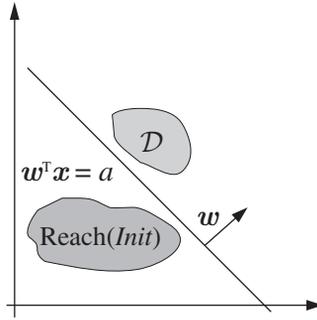


Fig. 3.4 Constraints on reachable space.

where a is the greater value of the product on $Init$, leading to the conclusion that D is not reachable.

This notion of border that separates the initial region from the forbidden one and that can not be crossed by trajectories may be generalized by the notion of barrier certificate. As shown in Fig. 3.5, the idea to prove that the domain D is not reachable from the domain $Init$ is to search for a function B such that it is negative on the domain $Init$, positive on the domain D , and vanishing on the border. Furthermore, it should have a gradient on the border, that is in the opposite direction of the flow defined by the dynamics (3.14). With some assumptions on the dynamics specified by f and the sets $Init$ and D it is possible to search for B in some given families of function, e.g. polynomial, by optimization methods:

$$\begin{aligned}
 &\forall x \in Init B(x) < 0, \\
 &\forall x \in D B(x) > 0, \\
 &\forall x \text{ s.a. } B(x) = 0 \quad \frac{\partial B(x)}{\partial x} f(x) \leq 0.
 \end{aligned} \tag{3.14}$$

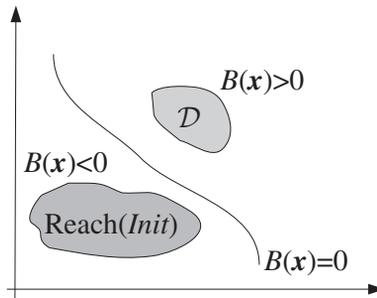


Fig. 3.5 Barrier certificate.

3.3.3 Reachable space computation

It may be useful to compute the reachable space and the computation is then based on an iterative algorithm such as Algorithm 3.1. Starting from the initial region and its continuous successor set, it computes at iteration $n + 1$ the hybrid successor set of the reachable space estimate obtained at iteration n . Actually as $\text{Succ}_H(\mathcal{A} \cup \mathcal{B}) = \text{Succ}_H(\mathcal{A}) \cup \text{Succ}_H(\mathcal{B})$, it is not necessary to compute the hybrid successor of the global estimate but only of the new part of it which has not been considered in the previous iterations. This is achieved in the proposed algorithm at step 2 where the $\text{New}(\mathcal{R}, \text{Reach}(\text{Init}))$ operator is assumed to compute the regions of \mathcal{R} that do not belong to $\text{Reach}(\text{Init})$.

Algorithm 3.1 Reachable space computation

Given: the initial region (Init)

Init: $\text{Reach}(\text{Init}) \leftarrow \text{Init}, \mathcal{R} \leftarrow \text{Succ}_C(\text{Init})$

- 1: **while** $\mathcal{R} \not\subseteq \text{Reach}(\text{Init})$ **do**
- 2: $\mathcal{R}^* \leftarrow \text{New}(\mathcal{R}, \text{Reach}(\text{Init}))$
- 3: $\text{Reach}(\text{Init}) \leftarrow \text{Reach}(\text{Init}) \cup \mathcal{R}$
- 4: $\mathcal{R} \leftarrow \text{Succ}_H(\mathcal{R}^*)$
- 5: **end while**

Result: $\text{Reach}(\text{Init})$

One critical point of this approach is that this algorithm may not converge:

|| The reachability problem for hybrid automata has been proved to be undecidable.

Hence, there does not exist any algorithm that is guaranteed to converge in every case. It may then be useful to use operators that approximate the various sets and may help to converge while guaranteeing that the given result is an over-approximation. It is also necessary to introduce a maximum number of iterations, thus computing the reachable space with this given number of discrete transitions.

As already stated, the other crucial point in this algorithm is the computation of the continuous successors that is embedded in step 2. Two main approaches may generally be used to perform this computation. The first one uses numerical integration to get a finite-time reachability computation that can be sufficient. The second one uses abstraction concepts to get an over approximation of the reachable space.

Numerical integration The principle of this approach is illustrated in Fig. 3.6 when polyhedra are chosen to represent sets in the continuous space. Starting from the initial domain (I_0), a numerical integration methods is used to compute the image (I_1) of this initial region after a given time step Δ (Fig. 3.6(a)). Then, a specific procedure is used to determine a set (P_1) that can be proved to include all trajectories

from the initial set during the first time step (Fig. 3.6(b)). Next, time steps are then considered (Fig. 3.6(c)). This can be done with the same procedure (first computation of I_n from I_{n-1} then computation of P_n from I_{n-1} and I_n) or with a new procedure that computes P_n from P_{n-1} (without explicit computation of I_n) according to considerations that take into account the computational simplicity and the effects of wrapping.

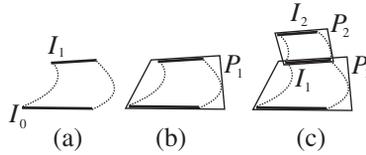


Fig. 3.6 Discrete time integration.

The important point in this approach is to use numerical methods that guarantee that the exact sets are included in the computed sets especially during the integration step. This can be achieved with set computations methods such as interval analysis or ellipsoidal computation.

The set of sets that is the result of the computation is an over-approximation of the reachable space. Its accuracy is linked to the value of the time step Δ . A trade-off has then to be found between the accuracy of the result and the number of time steps and sets to compute.

The characteristics of this approach is that its result is a finite time reachability analysis. This can be sufficient, for example when after some time all trajectories have crossed the invariant boundaries, but it may happen that some trajectories never cross the invariant boundaries, thus leading to infinite time trajectories in the same location that have to be considered.

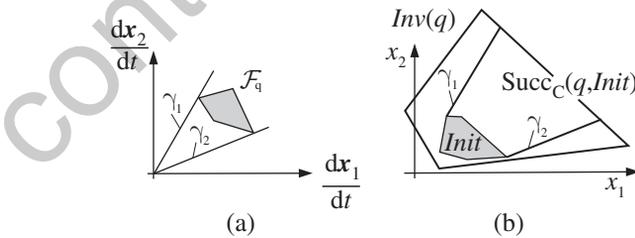


Fig. 3.7 Time elimination.

Time elimination and abstraction As illustrated in Fig. 3.7, when the dynamics of a location is specified by a linear differential inclusion it is possible to determine

maximal linear constraints that are satisfied by the flow (Fig. 3.7(a)). From these constraints it is possible to determine the reachable domain for example with convex hull operators (Fig. 3.7(a)).

When the continuous dynamics is not specified by a linear differential inclusion it is possible to build an abstraction of this dynamics (Fig. 3.8). A polyhedron F_q^* that is guaranteed to include the image of the invariant of the location by the dynamics function F_q , is computed. All trajectories of the initial systems are then also trajectories of the new one so the reachable domain of the first system is included in the one of the second system. Of course, this may lead to a very coarse approximation and it is often necessary to partition the invariant in order to associate to each element of the partition a tighter inclusion.

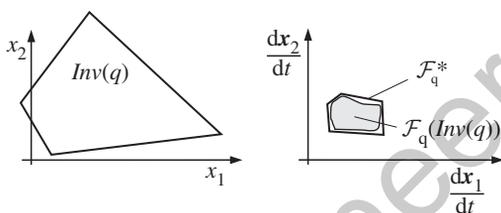


Fig. 3.8 Linear abstraction.

The accuracy of the result depends on the size of the element of the partition but also on their shape. The complexity of the abstraction step and the reachability step depends on the number of elements of the partition, so a trade-off between accuracy and complexity has to be found. For some families of systems it is possible to use structural properties, such as left eigenvectors for affine systems, to guide the partition in order to simplify the computation while keeping accuracy reasonably good.

3.3.4 Uncertainty

All the above approaches have been extended to take into account uncertainties. In most cases uncertainties are modeled by an input vector that takes its value in a bounded set. Reachability is then performed by considering that the input can vary and take any value of the set at each instant leading to an infinite number of trajectories from one initial state. The properties used to characterize or to compute the reachable space must then remain valid for all points of the input domain or, if it is a polytope, for its vertices.

3.4 Stability analysis

3.4.1 Stability analysis using gain automata

The main difficulty in the stability analysis of hybrid systems is the complicated interplay between discrete and continuous dynamics. One way to get this complication under control is by viewing a hybrid automaton as a discrete system in which the continuous dynamics has been eliminated by abstraction. Indeed, from a stability point of view all that matters is the continuous state behavior at the transition times.

For a specific class of hybrid automata this may be accomplished as described below. Consider hybrid automata where all locations share the same continuous state space and moreover in each location the zero state is an equilibrium with respect to the continuous dynamics in that location. The feature of interest is the stability of the zero state. With each pair of consecutive transitions a *gain* is associated that expresses the amplification of the norm of the continuous state from the first transition to the second. A gain automaton is then defined. The states of the gain automaton are the transition of the original hybrid automaton. Transitions in the gain automaton are pairs of consecutive transitions in the original hybrid automaton. The transitions of the gain automaton are labeled with the associated gains.

The stability analysis then amounts to the analysis of the gain automaton.

The hybrid automaton is asymptotically stable if in all cycles the product of the gains along that cycle is strictly less than one, stable if these products are less than or equal to one, and unstable if there is one cycle for which the product of the gains along that cycle exceeds one.

For general, possibly nonlinear, continuous dynamics, and general invariants and guards, the exact computation of the transition gains is a difficult problem.

In [204] the above ideas are applied to hybrid automata with linear dynamics in each location. The dimension of the continuous state space is two and the guards are lines passing through the origin. For this case all gains can be computed exactly so that for this case necessary and sufficient conditions for (asymptotic) stability of the origin are obtained.

As soon as the dimension of the state space is larger than two, the situation becomes more complicated. In [378] the same setup as in [204] is considered except that the condition on the dimension of the state space is dropped. The guards are now represented by hyperplanes. Furthermore it is assumed that in each location the continuous dynamics is asymptotically stable. Exact computation of the transition gains is no longer possible and, therefore, estimates are used. The gain estimates are based on quadratic Lyapunov functions associated with each consecutive pair of transitions. To reduce conservatism, the choice of Lyapunov functions is optimized with respect to the gains. The main result in [378] is that using the gain automaton analysis transition delays may be quantified in order to guarantee (asymptotic) stability.

3.4.2 Region stability

A region is a subset of the set of states of a hybrid automaton. An important property is whether all the trajectories of a hybrid automaton eventually end up in a certain region.

Definition 3.2 (Region stability) A hybrid automaton H is called stable w.r.t. a region ϕ if for every trajectory τ there exists a time t_0 such that from then on the trajectory is in the region:

$$\forall \tau \exists t_0 \forall t \geq t_0 : \tau(t) \in \phi.$$

The analysis of region stability makes use of the following concept:

Definition 3.3 (Sequence of snapshots) Given a hybrid automaton H and a region ϕ , a sequence of snapshots is a sequence of states such that (i) all states of the sequence lie on the same trajectory τ of H , (ii) all states are not in the region ϕ , and (iii) all pairs of consecutive states have a minimum time distance δ where δ is an arbitrary constant greater than 0.

If we restrict the regions to *interval regions*, i.e. regions that are given by $\phi = [x_{\min}, x_{\max}]$, there is a sufficient and necessary condition in terms of snapshot sequences.

States that occur just after a jump are called *entry points*; states where the flow reaches a local extremum are called *extremal points*.

Theorem 3.1 A hybrid automaton H is stable w.r.t. interval region ϕ if and only if the following conditions hold:

1. There is no infinite snapshot sequence such that no entry point or extremal point lies between two states of the sequence.
2. There is no infinite snapshot sequence such that all states are entry points.
3. There is no infinite snapshot sequence such that all states are extremal points.

These conditions can be checked by a proof system that uses a reachability checker. For proofs and details see [529, 530, 531], where it is also shown that the result can be extended to n -dimensional box regions, which are regions that can be expressed as the cartesian product of interval regions.

Theorem 3.2 A hybrid system H is stable w.r.t. an n -dimensional box region $\phi = \phi_1 \times \phi_2 \times \dots \times \phi_n$ if and only if H is stable w.r.t. each interval region ϕ_i .

3.4.3 Structural stability

A smooth dynamical system is structurally stable if it is topologically equivalent to every system which is sufficiently close to it. If a system is structurally stable, its qualitative properties are insensitive to small perturbations of the system.

This idea can be generalized to hybrid automata that are *regular*, i.e. deterministic, nonblocking, and with piecewise smooth dynamics in the locations. The precise mathematical formulation is quite involved and can be found in [590]; here we just sketch the basic idea.

Definition 3.4 (Topological equivalence) *Two regular hybrid automata without branching H and H' are called topologically equivalent if their hybrid flows are topologically equivalent.*

Next, a topology has to be defined on the space of regular hybrid automata without branching. Relative to this topology a hybrid automaton H is close to H' if they have exactly the same locations, transitions, invariants, and guards; furthermore, vector fields and reset maps of H and H' should be close.

Recall that an equilibrium point of a hybrid automaton H is a point that does not move under the hybrid flow.

Definition 3.5 (Local structural stability) *For a hybrid system H , an equilibrium x is called locally structurally stable if there exists a neighborhood of H such that every H' in this neighborhood is topologically equivalent at x .*

A Zeno state is a limit point of an execution with an infinite number of discrete transitions in a finite amount of time. A Zeno sink x is a Zeno state which locally attracts orbits, i.e. there is a neighborhood W of x such that x is the Zeno state for every execution starting in W .

In [590] the following theorem has been proved:

Theorem 3.3 *Let x be a Zeno sink. Then x is locally structurally stable.*

The fact that Zeno sinks are locally structurally stable means that it is difficult to remove them by perturbing the system.

3.4.4 LaSalle's invariance principle

A powerful method for the analysis of stability of dynamical systems is LaSalle's invariance principle, which provides conditions for an invariant set to be attracting. In [427] this result is extended to hybrid automata. Here we sketch the basic result.

Assume that a hybrid automaton H is nonblocking and deterministic, and that the set of initial states and all invariants of H are closed. Let Reach_H be the set of all reachable states of H . Let Out_H be the set of states from which continuous evolution is impossible. For a transition $e = (q, q')$ let $R(e, x)$ be the reset map of e . Let d be the Euclidean metric extended to states.

Definition 3.6 (Lyapunov function) $V(q, \mathbf{x})$ is a Lyapunov function for H if:

1. $V(q, \mathbf{x})$ is a Lyapunov function in each location q , so

$$L_f V(q, \mathbf{x}) \leq 0,$$

2. for all $(q, \mathbf{x}) \in \text{Out}_h$, $e = (q, q')$: $V(q', R(e, \mathbf{x})) \leq V(q, \mathbf{x})$.

Theorem 3.4 (LaSalle's invariance principle) Let Ω be a compact invariant set. Let $\Omega_1 = \Omega \cap \text{Out}_H^c$ and $\Omega_2 = \Omega \cap \text{Out}_H$. Assume V is a Lyapunov function for H . Let

$$\mathcal{S}_1 = \{(q, \mathbf{x}) \in \Omega_1 \mid L_f V(q, \mathbf{x}) = 0\},$$

$$\mathcal{S}_2 = \{(q, \mathbf{x}) \in \Omega_2 \mid \forall e : V(q', R(e, \mathbf{x})) = V(q, \mathbf{x})\}.$$

Let \mathcal{M} be the largest invariant subset of $\mathcal{S}_1 \cup \mathcal{S}_2$. Then for all trajectories τ starting in some $(q_0, \mathbf{x}_0) \in \Omega$:

$$\lim_{t \rightarrow |\tau|} d(\tau(t), \mathcal{M}) = 0$$

In [427] this theorem is proved, and an elaborate example of the application of LaSalle's invariance principle is given.

3.4.5 Lyapunov's indirect method

In [427] an extension of Lyapunov's indirect method to hybrid automata has been explored. The procedure involves linearizing all vector fields, guards, and reset images in the neighborhood of an equilibrium point.

For an equilibrium point $\mathbf{x}_* \in \mathcal{X}$, let

$$\mathcal{Q}_* = \{q \in \mathcal{Q} \mid (q, \mathbf{x}_*) \in \text{Reach}_H\}.$$

First, conditions are developed to ensure that the states in \mathcal{Q}_* are visited cyclically. In this case, the combined linearizations are used to calculate a number η representing the Lipschitz constant of the return map after one cycle. This Lipschitz constant can be used to estimate if this map is contracting: if $\eta < 1$, it can be concluded that the equilibrium point is locally asymptotically stable. The results are not easy to summarize; details can be found in [427]. The approach seems to have some similarity to the approach in [379] and [378].

3.4.6 Stability analysis using formal methods

In [456, 457] the stability analysis of hybrid automata is proposed using formal methods. Roughly, the idea is to associate with the given hybrid automaton a new one so that stability of the original automaton translates into a verification problem for

the associated automaton. The verification problem can then be solved using formal methods.

In [456] this idea is worked out for checking the average dwell time property, [318]. The *dwell time* is the duration that the automaton remains in a single location before switching to the next location. If this time is long enough, then the stability properties of the continuous dynamics defined for every location determine the stability property of the overall system.

Suppose that the continuous dynamics in each discrete location is stable. If the intervals during which no discrete transitions occur are sufficiently long, then overall hybrid automaton is stable. In other words, if the dwell time in each location is sufficiently long then the hybrid automaton is stable. A weaker and more general form of this statement expresses that it is sufficient that the average dwell time is small enough. The average dwell time is defined as follows.

Denote the number of discrete transitions for a given continuous state trajectory by $N(x(t))$. A hybrid automaton has average dwell time equals τ_a if there exists $N_0 > 0$ such that

$$N(x(t)) \leq N_0 + \frac{t}{\tau_a}. \quad (3.15)$$

The following theorem ([456]) formalizes this statement:

Theorem 3.5 Consider a hybrid automaton with vector fields f_p in location p , with $f_p(0) = \mathbf{0}$. Suppose that in each pair of locations p, q there exist a positive definite, radially unbounded, continuously differentiable function V_p , with $V_p(x) = 0$, and positive numbers λ_0, μ such that

$$\begin{aligned} \frac{\partial V_p}{\partial x} &\leq -\lambda_0 V_p(x) \\ V_p(x) &\leq \mu V_q(x) \end{aligned}$$

holds. Then the zero solution is globally uniformly asymptotically stable if the average dwell time τ_a satisfies the inequality

$$\tau_a > \frac{\log \mu}{2\lambda_0}.$$

To use the above result, a Lyapunov function V_p and the constants μ and λ_0 have to be found. But more importantly, the lower bound on the average dwell time needs to be checked. To this effect the original hybrid automaton is extended so that checking the lower bound in the original automaton amounts to checking an associated invariant property in the extended automaton. The obvious advantage of this approach is that invariant properties may be checked using existing tools.

An invariant property of a hybrid automaton is a condition on its continuous variables that remains true in all its reachable states. To check the average dwell time property the hybrid automaton is extended as follows. Two variables are added: a counter Q that is stepped after each switch of discrete state and a timer t . Both are initialized at 0. Q is stepped up after each discrete transition and is stepped down

every τ_a (the average dwell time) time units in between discrete transitions. The invariant property that is to be checked in the extended automaton then is whether Q does not exceed N_0 in (3.15).

Theorem 3.6 [456] *The hybrid automaton has average dwell time τ_a if and only if in the extended automaton the inequality $Q \leq N_0$ holds for all reachable states.*

3.5 Optimal control

3.5.1 Control problem

This section deals with the optimal control of hybrid systems that are described by a hybrid automaton in the time interval $[0, T]$ with the following properties:

1. the continuous state space $\mathbf{x}(t)$ belongs to \mathbb{R}^n ($n \in \mathbb{N}$);
2. the discrete state $q(t)$ belongs to a finite set \mathcal{Q} ;
3. \mathcal{U} is a subset of \mathbb{R}^m ($m \in \mathbb{N}$);
4. the input or control function \mathbf{u} belongs to \mathcal{U} , which is a subset of measurable functions from $[0, T]$ to \mathcal{U} ;
5. the vector field $\mathbf{f} : \mathbb{R}^n \times \mathcal{U} \rightarrow \mathbb{R}^n$ is smooth and depends on the discrete state $q(t)$;
6. the target set \mathcal{T} is a closed subset of \mathbb{R}^n ;
7. the switching set \mathcal{S} is a subset of $\mathcal{Q} \times \mathbb{R}^n \times \mathcal{Q} \times \mathbb{R}^n$;
8. the projected switching set $\mathcal{S}_{q,q'}$ is

$$\mathcal{S}_{q,q'} = \{(\mathbf{x}, \mathbf{x}') \in \mathbb{R}^n \times \mathbb{R}^n : \mathbf{x}' \in \mathcal{G}(q, q')\}.$$

Assuming that continuous dynamics is described by the state-space model

$$\begin{cases} \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)), & t \in [0, T], \\ \mathbf{x}(0) = \mathbf{x}_0, \end{cases} \quad (3.16)$$

with $T > 0$ and $\mathbf{x}_0 \in \mathbb{R}^n$, we want to minimize the cost functional

$$J(\mathbf{u}) = \int_0^T L(\mathbf{x}(t), \mathbf{u}(t)) dt + \psi(\mathbf{x}(T)) \quad (3.17)$$

over all $\mathbf{u} \in \mathcal{U}$, where the Lagrangian $L : \mathbb{R}^n \times \mathcal{U} \rightarrow \mathbb{R}$ and the final cost $\psi : \mathbb{R}^n \rightarrow \mathbb{R}$ are smooth functions, which depend on the discrete set $q(t)$.

If \mathbf{x} is a solution to (3.16), then we denote with $t_0 = 0, t_\nu = T$ and $t_1, \dots, t_{\nu-1}$ the switching times for \mathbf{x} .

Remark 3.1 If t_i ($i \in \{1, \dots, \nu-1\}$) is a switching time for \mathbf{x} , then $(q(t_i^-), \mathbf{x}(t_i^-), q(t_i^+), \mathbf{x}(t_i^+)) \in \mathcal{S}$ and $\mathbf{x}(t_i^+) \in \mathcal{G}_{q(t_i^-), q(t_i^+)}$.

3.5.2 The classical Pontryagin's Maximum Principle

A brief review of the main ideas behind Pontryagin's Maximum Principle is in order. We put the accent on the main issue of needle variations whose generalization to the hybrid setting is the key point to prove necessary conditions for optimality of hybrid systems.

We consider the system

$$\begin{cases} \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)), & t \in [0, T], \\ \mathbf{x}(0) = \mathbf{x}_0, \end{cases}$$

where $\mathbf{x}(t)$ belongs to \mathbb{R}^n and $\mathbf{u} \in \mathcal{U}$. Given a class \mathcal{U} , which contains the constant controls and finite concatenations of restrictions of admissible controls, one can state the optimal control problem

$$\min_{\mathbf{u} \in \mathcal{U}} \psi(\mathbf{x}(T)),$$

where $\mathcal{T} \subset \mathbb{R}^n$ is the target and, for simplicity, the Lagrangian function L is equal to 0. Such reduction can always be obtained by introducing an augmented vector field.

Assume that a bounded control $\mathbf{u}^* \in \mathcal{U}$ is candidate optimal. We can produce some variations of \mathbf{u}^* and of the corresponding trajectory \mathbf{x}^* in the following way (Fig. 3.9):

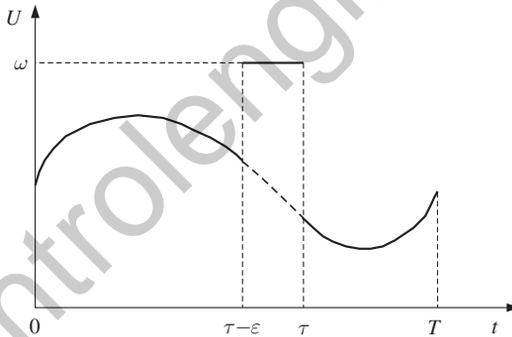


Fig. 3.9 Needle variation.

Definition 3.7 (Needle variation) Let τ be a Lebesgue point for

$$t \mapsto \mathbf{f}(\mathbf{x}^*(t), \mathbf{u}^*(t)).$$

Given $\omega \in \mathcal{U}$, define a family of inputs \mathbf{u}_ε , $\varepsilon \in [0, \tau]$, as

$$\mathbf{u}_\varepsilon(t) = \begin{cases} \mathbf{u}(t), & \text{if } t \in [0, \tau - \varepsilon], \\ \omega, & \text{if } t \in [\tau - \varepsilon, \tau], \\ \mathbf{u}(t), & \text{if } t \in [\tau, T], \end{cases} \quad (3.18)$$

and let \mathbf{x}_ε be the trajectories corresponding to \mathbf{u}_ε with $\mathbf{x}_\varepsilon(0) = \mathbf{x}_0$. The family \mathbf{x}_ε is a needle variation of $(\mathbf{x}^*, \mathbf{u}^*)$.

Given a needle variation, define, for every time $t \in [\tau, T]$, the tangent vector $\mathbf{v}(t)$ to the curve $\mathbf{x}_\varepsilon(t)$ at $\varepsilon = 0$, i.e. $\mathbf{v}(t) = (d/d\varepsilon)\mathbf{x}_\varepsilon(t)|_{\varepsilon=0}$ (Fig. 3.10). The optimality of \mathbf{u}^* implies that the inequality

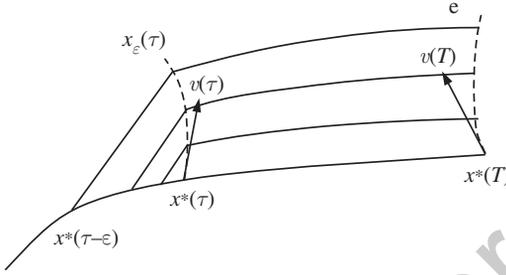


Fig. 3.10 Trajectories \mathbf{x}_ε and tangent vector \mathbf{v} .

$$0 \leq \frac{d}{d\varepsilon} \psi(\mathbf{x}_\varepsilon(T))|_{\varepsilon=0} = \nabla \psi(\mathbf{x}^*(T)) \cdot \mathbf{v}(T)$$

holds. From classical theory of ODEs, we know that \mathbf{v} solves the linear system

$$\begin{cases} \dot{\mathbf{v}}(t) = \nabla_x \mathbf{f}(\mathbf{x}^*(t), \mathbf{u}^*(t)) \cdot \mathbf{v}(t), & t \in [\tau, T], \\ \mathbf{v}(\tau) = \mathbf{f}(\mathbf{x}^*(\tau), \omega) - \mathbf{f}(\mathbf{x}^*(\tau), \mathbf{u}^*(\tau)). \end{cases}$$

Therefore, defining λ to be the solution to

$$\begin{cases} \dot{\lambda}(t) = -\lambda(t) \cdot \nabla_x \mathbf{f}(\mathbf{x}^*(t), \mathbf{u}^*(t)), & t \in [\tau, T], \\ \lambda(T) = \nabla \psi(\mathbf{x}^*(T)), \end{cases}$$

we have $(d/dt)(\lambda(t) \cdot \mathbf{v}(t)) = 0$ and so

$$0 \leq \lambda(T) \cdot \mathbf{v}(T) = \lambda(\tau) \cdot \mathbf{v}(\tau) = \lambda(\tau) \cdot (\mathbf{f}(\mathbf{x}^*(\tau), \omega) - \mathbf{f}(\mathbf{x}^*(\tau), \mathbf{u}^*(\tau))),$$

or otherwise stated, for the arbitrariness of ω ,

$$\min_{\omega \in \mathcal{U}} \lambda(\tau) \cdot \mathbf{f}(\mathbf{x}^*(\tau), \omega) = \lambda(\tau) \cdot \mathbf{f}(\mathbf{x}^*(\tau), \mathbf{u}^*(\tau)),$$

which is precisely the Hamiltonian minimization condition of Pontryagin's Maximum Principle.

The main difficulty in bringing the above reasoning to the hybrid setting is the possibility of producing the analogous of needle variations. As long as a trajectory remains in the same location, then the variation can be done in the same way.

3.5.3 Extension to a hybrid maximum principle

The hybrid maximum principle gives the necessary conditions for a trajectory x to be a solution of the minimization problem (3.16)–(3.17). The set of variations involves trajectories having the same history as the optimal candidate, i.e. having the same location switching strategy. If there is a finite number of possible switching strategies for the optimization problem, then the Maximum Principle can sometimes single out the optimal trajectory.

As explained in Section 3.5.2, the main difficulty is the extension of a needle variation after a location switching. More precisely, consider the situation in Fig. 3.11. Assume that $t_1 \in]\tau, T[$ is a switching time. The value of $v(t_{1+})$ must be chosen in such a way that the corresponding trajectory jumps are in the switching set $\mathcal{S}_{q(t_1-), q(t_1+)}$. In turn this imposes some restrictions on the corresponding values of the covectors.

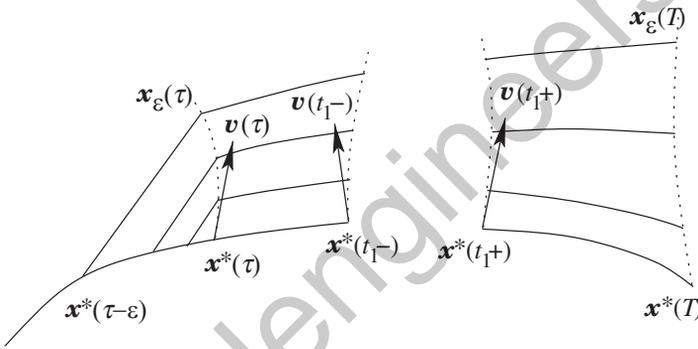


Fig. 3.11 Jump of vector v after a location switching time.

Definition 3.8 (Adjoint pair) The couple (λ, λ_0) is an adjoint pair along a trajectory x if:

- for every $i \in \{1, \dots, \nu\}$, $\lambda :]t_{i-1}, t_i[\rightarrow \mathbb{R}^n$ is an absolutely continuous function satisfying

$$\dot{\lambda}(t) = -\lambda(t) \cdot \nabla_x f(x(t), u(t)) - \lambda_0 \nabla_x L(x(t), u(t))$$

- for a.e. $t \in]t_{i-1}, t_i[$;
- $\lambda_0 \in \mathbb{R}^+$.

In order to formulate the hybrid maximum principle, the following notions have to be introduced:

- Switching condition:** The adjoint pair (λ, λ_0) along a trajectory x satisfies the switching conditions if $(-\lambda(t_i-), \lambda(t_i+)) \in \mathcal{K}_i^\perp$ for every $i \in \{1, \dots, \nu - 1\}$,

where \mathcal{K}_i^\perp denotes the polar set to the tangent cone \mathcal{K}_i to the set $\mathcal{S}_{q(t_i^-), q(t_i^+)}$ at the point $(\mathbf{x}(t_i^-), \mathbf{x}(t_i^+))$.

- *Transversality condition:* The adjoint pair $(\boldsymbol{\lambda}, \lambda_0)$ along a trajectory \mathbf{x} satisfies the transversality condition if $-\boldsymbol{\lambda}(T^-) - \lambda_0 \nabla \psi(\mathbf{x}(T)) \in \mathcal{K}_e^\perp$, where \mathcal{K}_e^\perp denotes the polar set to the tangent cone \mathcal{K}_e to the set \mathcal{T} at the point $\mathbf{x}(T)$.
- *Nontriviality condition:* The adjoint pair $(\boldsymbol{\lambda}, \lambda_0)$ along a trajectory \mathbf{x} satisfies the nontriviality condition if either $\lambda_0 \neq 0$ or $\boldsymbol{\lambda}$ is not identically zero.
- *Hamiltonian:* Consider an adjoint pair $(\boldsymbol{\lambda}, \lambda_0)$ along a trajectory \mathbf{x} . For almost every $t \in [0, T]$, define the Hamiltonian functions as

$$H(t) = \boldsymbol{\lambda}(t) \cdot \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) - \lambda_0 L(\mathbf{x}(t), \mathbf{u}(t)), \quad (3.19)$$

$$\tilde{H}(t) = \sup \{ \boldsymbol{\lambda}(t) \cdot \mathbf{f}(\mathbf{x}(t), u) - \lambda_0 L(\mathbf{x}(t), u) : u \in \mathcal{U} \}. \quad (3.20)$$

We say that the adjoint pair $(\boldsymbol{\lambda}, \lambda_0)$ satisfies the Hamiltonian maximization property if $H(t) = \tilde{H}(t)$ holds a.e. in $[0, T]$.

We state the hybrid maximum principle:

Theorem 3.7 (Hybrid maximum principle) *Let \mathbf{x} be a solution for the minimization problem (3.16)–(3.17). Then there exists an adjoint pair $(\boldsymbol{\lambda}, \lambda_0)$ along \mathbf{x} that satisfies the switching conditions, the Hamiltonian maximization, and the nontriviality condition.*

Example 3.3 Optimal control of the two-tank system

Let us consider the two-tank system introduced in Section 1.3.1 with the following parameters:

1. $h_0 = 1$;
2. $h_{\max} = 2$;
3. $A = 1$;
4. $g = 1$;
5. $c = 1$.

Two-tank system with one control input Assume that the only control is $u_2 \in [0, 1]$, i.e. $u_1 = u_3 = u_{P1} = 0$, and assume that the disturbances d_1 and d_2 are equal to 0. Consider the following optimal control problem:

$$\begin{cases} \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)), \\ \mathbf{x}(0) = \left(\frac{5}{4}, 0\right), \\ \min_{u \in \mathcal{U}} \{-h_2(T)\}, \end{cases} \quad (3.21)$$

with $\mathbf{x}(t) = (h_1(t), h_2(t))$ and

$$\mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) = \begin{pmatrix} -\sqrt{2(h_1(t) - h_2(t))} u_2(t) \\ \sqrt{2(h_1(t) - h_2(t))} u_2(t) \end{pmatrix}. \quad (3.22)$$

We show that the trajectory with control $u_2(t) \equiv 1$ satisfies the hybrid maximum principle. First of all, note that the trajectory associated to this control function is given by

$$\begin{cases} h_1(t) = \frac{5}{8} + \frac{1}{4} \left(\sqrt{\frac{5}{2}} - 2t \right)^2, \\ h_2(t) = \frac{5}{8} - \frac{1}{4} \left(\sqrt{\frac{5}{2}} - 2t \right)^2, \end{cases} \quad (3.23)$$

if $0 \leq t \leq \frac{1}{2} \sqrt{\frac{5}{2}}$, while

$$\begin{cases} h_1(t) = \frac{5}{8}, \\ h_2(t) = \frac{5}{8} \end{cases} \quad (3.24)$$

if $t \geq \frac{1}{2} \sqrt{\frac{5}{2}}$. Moreover, we have

$$q(t) = 2 \text{ if } 0 < t < \frac{1}{2\sqrt{2}} (\sqrt{5} - \sqrt{3})$$

$$q(t) = 1 \text{ if } \frac{1}{2\sqrt{2}} (\sqrt{5} - \sqrt{3}) < t < \frac{1}{2} \sqrt{\frac{5}{2}},$$

i.e. the switching time t_1 is $\frac{1}{2\sqrt{2}} (\sqrt{5} - \sqrt{3})$. Without loss of generality we put $T = \frac{1}{2} \sqrt{\frac{5}{2}}$.

If $\lambda(t) = (\lambda_1(t), \lambda_2(t))$, then the transversality condition implies

$$-\lambda(T) + (0, \lambda_0) = 0.$$

Moreover we have

$$\begin{cases} \dot{\lambda}_1(t) = \lambda_1(t) \frac{1}{\sqrt{2(h_1(t) - h_2(t))}} - \lambda_2(t) \frac{1}{\sqrt{2(h_1(t) - h_2(t))}}, \\ \dot{\lambda}_2(t) = -\lambda_1(t) \frac{1}{\sqrt{2(h_1(t) - h_2(t))}} + \lambda_2(t) \frac{1}{\sqrt{2(h_1(t) - h_2(t))}}, \end{cases}$$

and so

$$\begin{cases} \lambda_1(t) = \frac{\lambda_0}{2} \left(1 - \exp \left(\int_T^t \frac{2}{\sqrt{2(h_1(s) - h_2(s))}} ds \right) \right), \\ \lambda_2(t) = \frac{\lambda_0}{2} \left(1 + \exp \left(\int_T^t \frac{2}{\sqrt{2(h_1(s) - h_2(s))}} ds \right) \right). \end{cases}$$

Note that the nontriviality condition implies $\lambda_0 > 0$ and the function $\lambda_1(t) - \lambda_2(t) < 0$ for every t . We have

$$\begin{aligned} \tilde{H}(t) &= \sup \left\{ (\lambda_2(t) - \lambda_1(t)) \sqrt{2(h_1(t) - h_2(t))} u_2 : u_2 \in [0, 1] \right\} \\ &= (\lambda_2(t) - \lambda_1(t)) \sqrt{2(h_1(t) - h_2(t))}; \end{aligned}$$

hence the Hamiltonian maximization property implies that the control u_2 is equal to 1.

Two-tank system with two control inputs Let us consider now a tank system with control inputs $u_1 \in [0, 1]$ and $u_2 \in [0, 1]$. The disturbances d_1 and d_2 are equal to 0. Consider the following optimal control problem

$$\begin{cases} \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)), \\ \mathbf{x}(0) = \left(\frac{5}{4}, 0\right), \\ \min_{\mathbf{u} \in \mathcal{U}} \{-h_2(T)\}, \end{cases} \quad (3.25)$$

with $\mathbf{x}(t) = (h_1(t), h_2(t))$,

$$\mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) = \begin{pmatrix} -\sqrt{2(h_1(t) - 1)} u_1(t) - \sqrt{2(h_1(t) - h_2(t))} u_2(t) \\ \sqrt{2(h_1(t) - 1)} u_1(t) + \sqrt{2(h_1(t) - h_2(t))} u_2(t) \end{pmatrix},$$

if $h_1(t) > 1$ while

$$\mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) = \begin{pmatrix} -\sqrt{2(h_1(t) - h_2(t))} u_2(t) \\ \sqrt{2(h_1(t) - h_2(t))} u_2(t) \end{pmatrix}$$

if $h_1(t) \leq 1$. We show that the trajectory with controls $u_1(t) = u_2(t) \equiv 1$ satisfies the hybrid maximum principle.

Call t_1 the switching time, i.e. the time satisfying $h_1(t_1) = 1$. If $\lambda(t) = (\lambda_1(t), \lambda_2(t))$, then the transversality condition implies

$$-\lambda(T) + (0, \lambda_0) = 0.$$

Moreover we have

$$\begin{cases} \dot{\lambda}_1(t) = \left(\frac{1}{\sqrt{2(h_1(t) - 1)}} + \frac{1}{\sqrt{2(h_1(t) - h_2(t))}} \right) (\lambda_1(t) - \lambda_2(t)), \\ \dot{\lambda}_2(t) = -\frac{1}{\sqrt{2(h_1(t) - h_2(t))}} (\lambda_1(t) - \lambda_2(t)), \end{cases}$$

if $0 \leq t \leq t_1$ and

$$\begin{cases} \dot{\lambda}_1(t) = \lambda_1(t) \frac{1}{\sqrt{2(h_1(t) - h_2(t))}} - \lambda_2(t) \frac{1}{\sqrt{2(h_1(t) - h_2(t))}}, \\ \dot{\lambda}_2(t) = -\lambda_1(t) \frac{1}{\sqrt{2(h_1(t) - h_2(t))}} + \lambda_2(t) \frac{1}{\sqrt{2(h_1(t) - h_2(t))}}, \end{cases}$$

if $t \geq t_1$. Note that $\lambda_1(t) - \lambda_2(t) < 0$, $\lambda_1(t) \geq 0$ and $\lambda_2(t) > 0$ for every $t \in [t_1, T]$, since the nontriviality condition implies $\lambda_0 > 0$. If $t \in [t_1, T]$, then we have that $\tilde{H}(t)$ is equal to

$$\begin{aligned} \tilde{H}(t) &= \sup \left\{ \left(\sqrt{2(h_1(t) - h_2(t))} u_2 \right) (\lambda_2(t) - \lambda_1(t)) : u_2 \in [0, 1] \right\} \\ &= (\lambda_2(t) - \lambda_1(t)) \sqrt{2(h_1(t) - h_2(t))}; \end{aligned}$$

hence $u_2(t) = 1$ if $t \in [t_1, T]$ satisfies the hybrid maximum principle. The switching condition at $t = t_1$ implies that $\lambda_2(t_1^-) = \lambda_2(t_1^+)$ and does not give any condition on $\lambda_1(t_1 \pm)$. Therefore we can choose $\lambda_1(t_1^-) = \lambda_1(t_1^+)$. In this way, $\lambda_1(t) - \lambda_2(t) < 0$

for every $t \in [0, t_1]$. If $t \in [0, t_1]$, then $\tilde{H}(t)$ is equal to

$$\sup \left\{ \left(\sqrt{2(h_1(t) - 1)} u_1 + \sqrt{2(h_1(t) - h_2(t))} u_2 \right) (\lambda_2(t) - \lambda_1(t)) \right\},$$

where the supremum is taken over $u_1, u_2 \in [0, 1]$. Hence $u_1(t) = u_2(t)$ for every $t \in [0, t_1]$ satisfies the HMP. Note that the value of the control u_1 in the time interval $[t_1, T]$ does not affect the validity of the hybrid maximum principle. \square

Example 3.4 Automatic gearbox

Let us consider the automatic gearbox system described in Section 1.3.2 with the following parameters:

1. $r = M = c = 1$;
2. $d = 0$.

Assume that the control is $u \in [0, 1]$ and the switching sets are

$$\begin{aligned} \mathcal{S}_{1,2} &= \{(v_{1,2}, v_{1,2})\}, & \mathcal{S}_{2,3} &= \{(v_{2,3}, v_{2,3})\}, & \mathcal{S}_{3,4} &= \{(v_{3,4}, v_{3,4})\}, \\ \mathcal{S}_{2,1} &= \{(v_{2,1}, v_{2,1})\}, & \mathcal{S}_{3,2} &= \{(v_{3,2}, v_{3,2})\}, & \mathcal{S}_{4,3} &= \{(v_{4,3}, v_{4,3})\}, \end{aligned}$$

where $0 < v_{2,1} < v_{1,2}$, $0 < v_{3,2} < v_{2,3}$, and $0 < v_{4,3} < v_{3,4}$. Consider the following optimal control problem:

$$\begin{cases} \dot{x}(t) = f(x(t), u(t)), \\ x(0) = (0), \\ \min_{u \in \mathcal{U}} \{-v(T)\}, \end{cases} \quad (3.26)$$

with $x(t) = v(t)$ and

$$f(x(t), u(t)) = p(q)u(t) - v^2(t). \quad (3.27)$$

We show that the trajectory with control $u(t) \equiv 1$ satisfies the hybrid maximum principle.

First, note that the equation for the adjoint pair is

$$\begin{cases} \dot{\lambda}(t) = 2v(t)\lambda(t), \\ \lambda(T) = \lambda_0, \end{cases}$$

and so the solution is given by

$$\lambda(t) = \lambda(T)e^{\int_T^t 2v(s)ds}.$$

The nontriviality condition implies that $\lambda_0 > 0$ and so $\lambda(t) > 0$ for every $t \in [0, T]$. Moreover,

$$\begin{aligned} \tilde{H}(t) &= \sup \left\{ \lambda(T) \exp \left(\int_T^t 2v(s) ds \right) (p(q)u - v^2(t)) : u \in [0, 1] \right\} \\ &= \lambda(T) \exp \left(\int_T^t 2v(s) ds \right) (p(q) - v^2(t)); \end{aligned}$$

hence the Hamiltonian maximization allows to conclude. \square

Bibliographical notes

Hybrid automata have been originally proposed in [16]. Many extensions of hybrid automata exist, but also many other formulations of hybrid models exist, see for instance [16, 115, 271, 401, 426, 428, 571] and the references therein. For further reading, also [17, 315] are recommended. An important extension of hybrid automata, essential in the world of embedded and reactive systems, are hybrid I/O automata [430]. A practical introduction to hybrid modeling can be found in [382].

Reachability analysis The reader is referred to [288] and associated references for a general presentation of reachability analysis of hybrid automata and to [315] for decidability concerns. A complete presentation of the concept of barrier certificates and its extension to hybrid or stochastic systems may be found in the paper [536]. Use of discrete-time reachability for linear dynamics with bounded inputs is for examples solved with zonotopes [270] or with ellipsoids [371]. The basic concept of reachability analysis for linear hybrid automata is presented in [16] and abstraction methods for affine systems are presented in [396].

The computability of reachable sets for nonlinear systems is investigated in [182].

Optimal control Recently optimization problems for hybrid systems have attracted a lot of attention; see for example [115, 135, 147, 174, 175, 299, 557, 577, 674].

For classical control problems, the main tool for the construction of optimal trajectories is the Pontryagin Maximum Principle (PMP); see [5, 118, 534].

The maximum principle in the hybrid setting (HMP) was developed in [174, 523, 524, 613]. Some applications of the HMP can be found in [523, 524].

A generalization of HMP in the case of controls depending on the switching strategy was done in [259]. In this paper the authors considered variations more general than needle variations and they deduced some necessary conditions. For some applications, one can refer to [197].

Another generalization of HMP in the case of restrictions on the switching strategy by means of constraints is done in [135].

controlengineers.ir

Switched and piecewise affine systems

J. Daafouz, M. D. Di Benedetto, V. D. Blondel, G. Ferrari-Trecate, L. Hetel, M. Johansson, A. L. Juloski, S. Paoletti, G. Pola, E. De Santis, and R. Vidal

Switched systems are described by a set of continuous state-space models together with conditions that decide which model of this set is valid for the current continuous state. As an extension of the classical linear or affine state-space representations of dynamical systems, this modelling formalism has been thoroughly investigated, as this chapter shows. The identification of the model parameters, observability, and stability analysis as well as methods for stabilization and control of switched systems are surveyed. As shown in the last section, many analysis and design problems for switched systems have a high computational complexity or are even undecidable.

Chapter contents

4.1	Definition of the system class	page	89
4.2	Identification		92
4.2.1	Identification problem		92
4.2.2	Models in input/output form		93
4.2.3	Hybrid system identification problems		94
4.2.4	Data classification and region estimation		97
4.2.5	Four procedures for the identification of SARX/PWARX models		99
4.2.6	Comparison of the identification procedures		104
4.3	Observation of linear switched systems		106
4.3.1	Preliminaries and basic definitions		106
4.3.2	Observability of switched systems with arbitrary switching		108
4.3.3	Observability of switched systems with minimum dwell time		110
4.4	Stability analysis		112
4.4.1	Stability problems		112
4.4.2	Stability notions		113
4.4.3	Stability of differential inclusions		115

4.4.4	Stability analysis of switched linear systems by means of a common Lyapunov function	116
4.4.5	Lie-algebraic stability criteria	118
4.4.6	Necessary and sufficient stability criteria	119
4.4.7	Multiple Lyapunov functions	119
4.4.8	Stability of discrete-time systems	121
4.5	Stabilization and control	124
4.5.1	Stabilization problems	124
4.5.2	State-space restrictions	124
4.5.3	Time domain restrictions	126
4.5.4	Control design for arbitrary switching	128
4.5.5	Stabilization of discrete-time switched linear systems	129
4.6	Complexity issues	130
4.6.1	Decidability and NP-hardness	130
4.6.2	Switched systems and the joint spectral radius	131
4.6.3	Piecewise affine systems	134

controlengineers.ir

4.1 Definition of the system class

Switched systems represent a type of model of hybrid systems that has been studied extensively. The reason for this research activity is given by the fact that this class of systems is very close to “non-hybrid” systems and an extension of the theory of continuous systems towards hybrid systems is, therefore, rather straightforward. Nevertheless, this system class already exhibits several important phenomena of hybrid dynamical systems.

The basic representation format is the state-space model

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), q(t), \mathbf{u}(t)), \quad \mathbf{x}(0) = \mathbf{x}_0, \tag{4.1}$$

$$\mathbf{y}(t) = \mathbf{g}(\mathbf{x}(t), q(t), \mathbf{u}(t)), \tag{4.2}$$

which describes the dynamical behavior of the system for the input $\mathbf{u} \in \mathbb{R}^m$ and the operation mode $q \in \mathcal{Q}$. The vector field \mathbf{f} and the output function \mathbf{g} are assumed to be Lipschitz continuous with respect to \mathbf{x} and \mathbf{u} so that for a fixed operation mode q solutions to the state-space model exist.

For autonomous systems (systems without input) the model simplifies to

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), q(t)), \quad \mathbf{x}(0) = \mathbf{x}_0, \tag{4.3}$$

$$\mathbf{y}(t) = \mathbf{g}(\mathbf{x}(t), q(t)), \tag{4.4}$$

which is usually written as

$$\dot{\mathbf{x}}(t) = \mathbf{f}_{q(t)}(\mathbf{x}(t)), \quad \mathbf{x}(0) = \mathbf{x}_0, \tag{4.5}$$

$$\mathbf{y}(t) = \mathbf{g}_{q(t)}(\mathbf{x}(t)), \tag{4.6}$$

where the operation mode is indicated as an index of the vector field \mathbf{f} and the output function \mathbf{g} .

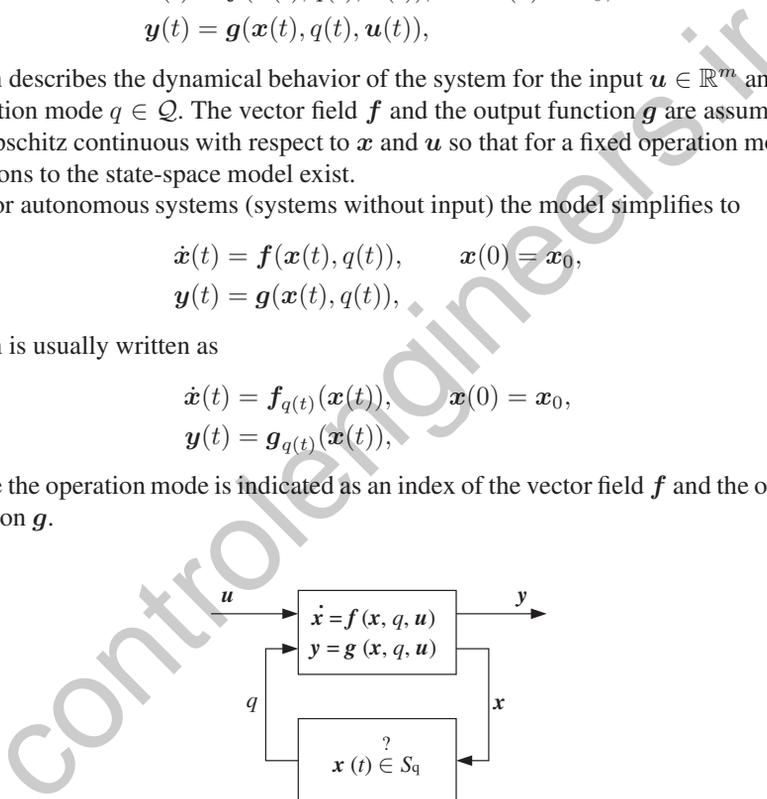


Fig. 4.1 Switched system.

There are two situations to be investigated:

- *Controlled switched system*: The switching signal $q(t)$ is an input to the system (4.2). Hence, the system has to be investigated for arbitrary switching functions $q : \mathbb{R}^+ \rightarrow \mathcal{Q}$.

- *Autonomously switched system*: The switching signal depends upon the continuous state \mathbf{x} . Then the overall system is a hybrid system with continuous input $\mathbf{u}(t)$ and output $\mathbf{y}(t)$ (Fig. 4.1), whose analysis has to take into account the rules for selecting the switching signal.

The second case, the feedback of the state \mathbf{x} towards the switching input $q(t)$, includes the continuous-to-discrete interface shown in Fig. 2.1 as an important element of hybrid systems. Usually, the switching signal changes its value if the continuous state reaches a surface \mathcal{S} in the state space \mathbb{R}^n . There may be several such surfaces $\mathcal{S}_{q'}$ $q' \in \mathcal{Q}$. If the relation

$$\mathbf{x}(t) \in \mathcal{S}_{q'}$$

is satisfied at time t , then the switching signal changes its value at this time instant towards q' . Hence, the situation that the state touches a surface determines at which time instant and towards which successor value the switching signal changes.

In a more elaborate version, the switching surface also depends upon the current mode q and may explicitly depend upon the time t . When dealing with switched systems of this kind, the switching surfaces are usually given. This distinguishes these systems from the class of discretely controlled continuous systems described in Section 6.5 where the choice of the switching surface is a problem of control design.

The terminology with respect to switched systems is not unique in literature. Besides the term “switched system” also the notion of a switching system is used, which is particularly appropriate for autonomously switched systems defined above, because these systems switch among different operation modes without being forced to do so by an external signal. Hence, these systems can be characterized by the outside observer as a switching system. However, in order to simplify the terminology, only the term “switched system” is used in this handbook.

Piecewise affine systems If the vector field \mathbf{f} and the function \mathbf{g} are affine for all values of q , the system is piecewise affine and described by the equations

$$\dot{\mathbf{x}}(t) = \mathbf{A}_{q(t)}\mathbf{x}(t) + \mathbf{B}_{q(t)}\mathbf{u}(t) + \mathbf{e}_{q(t)}, \quad \mathbf{x}(0) = \mathbf{x}_0, \quad (4.7)$$

$$\mathbf{y}(t) = \mathbf{C}_{q(t)}\mathbf{x}(t) + \mathbf{D}_{q(t)}\mathbf{u}(t) + \mathbf{f}_{q(t)}, \quad (4.8)$$

where the dependency of the model parameters upon the switching signal is usually indicated by the indices of the matrices. This system class is studied in the continuous-time version given here or in discrete time.

If autonomously switched systems are considered, the regions \mathcal{C}_q in which the q -th parameter set $(\mathbf{A}_q, \mathbf{B}_q, \dots, \mathbf{D}_q)$ hold, are usually assumed to be convex polyhedra, which can be defined by a set of linear inequalities. Then the model reads as

$$\dot{\mathbf{x}}(t) = \mathbf{A}_q\mathbf{x}(t) + \mathbf{B}_q\mathbf{u}(t) + \mathbf{e}_q, \quad \text{for } \mathbf{x}(t) \in \mathcal{C}_q, \quad (4.9)$$

$$\mathbf{y}(t) = \mathbf{C}_q\mathbf{x}(t) + \mathbf{D}_q\mathbf{u}(t) + \mathbf{f}_q. \quad (4.10)$$

The operation condition need not only depend on the current state $\mathbf{x}(t)$, but also on the input $\mathbf{u}(t)$. Then (4.9), (4.10) hold for the index q for which $(\mathbf{x}(t), \mathbf{u}(t))^T \in \mathcal{C}_q$ holds.

As the operation mode q can be determined by the relation $\mathbf{x}(t) \in \mathcal{C}_q$ for a given state $\mathbf{x}(t)$, q does not include any additional information about the current state of the system. Hence, piecewise affine systems are not “real” hybrid system, because the system state is completely described by the continuous state $\mathbf{x}(t)$. Nevertheless, the operation mode q is often referred to as the discrete state of the system.

The main reason for considering piecewise affine systems as hybrid systems is the fact that such systems are represented by differential equations with discontinuous right-hand side. When crossing a switching surface, the vector field $\mathbf{f}(\mathbf{x})$ given by the right-hand side of (4.9) changes discontinuously and, hence, does not satisfy the standing assumption of nonlinear systems theory that the vector field under investigation is Lipschitz-continuous.

Discrete-time piecewise affine systems The discrete-time version of the model (4.9), (4.10) is given by

$$\dot{\mathbf{x}}(k+1) = \tilde{\mathbf{A}}_q \mathbf{x}(k) + \tilde{\mathbf{B}}_q \mathbf{u}(k) + \tilde{\mathbf{e}}_q, \quad \text{for } \mathbf{x}(k) \in \mathcal{C}_q, \quad (4.11)$$

$$\mathbf{y}(k) = \tilde{\mathbf{C}}_q \mathbf{x}(k) + \tilde{\mathbf{D}}_q \mathbf{u}(k) + \tilde{\mathbf{f}}_q. \quad (4.12)$$

It is an important fact that a piecewise affine system (4.9), (4.10), which is equipped with a zero-order hold at the input and a sampling device at the output (or the state) can generally not be described by a discrete-time piecewise affine system (4.11), (4.12), as the following example shows.

Example 4.1 *Sampled-data piecewise affine system*

Consider the first-order system

$$\begin{aligned} \dot{x}(t) &= \lambda_1 x(t), & \text{for } x(t) \leq \bar{x}, \\ \dot{x}(t) &= \lambda_2 x(t), & \text{for } x(t) > \bar{x}. \end{aligned}$$

It is known from linear systems theory that, as long as the system does not switch its dynamics, the continuous state trajectory yields the sampled states

$$\begin{aligned} x(k+1) &= e^{\lambda_1 T} x(k), & \text{for } x(k), x(k+1) \leq \bar{x}, \\ x(k+1) &= e^{\lambda_2 T} x(k), & \text{for } x(k), x(k+1) > \bar{x}, \end{aligned}$$

where T denotes the sampling time. Hence, state sequences at the sampling moments of the linear continuous-time system is represented by a linear discrete-time model.

The linearity of the representation is lost if the system is switched between two succeeding sampling instants. To understand this phenomenon assume that the system is in the second operation mode at time $t_k = kT$ and switches to the first operation mode at time τ :

$$\begin{aligned} x(t) &> \bar{x}, & \text{for } t_k \leq t \leq \tau, \\ x(t) &\leq \bar{x}, & \text{for } \tau \leq t \leq t_{k+1}. \end{aligned}$$

Then the movement is governed by the second continuous state equation for the time interval $t_k \leq t \leq \tau$ and by the first equation for the interval $\tau \leq t \leq t_{k+1}$ with $t_{k+1} = (k+1)T$.

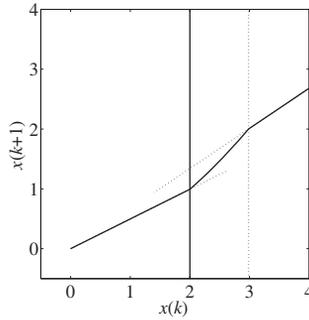


Fig. 4.2 Nonlinear state-transition function f of a sampled-data linear switched system.

Since τ depends upon $x(t_k)$, a nonlinear relation between $x(k)$ and $x(k+1)$ occurs, which for a first-order switched system can be written down explicitly:

$$x(k+1) = \left(\frac{\bar{x}}{x(k)} \right)^{-\frac{\lambda_2}{\lambda_1}} e^{\lambda_2 T} \bar{x}. \quad (4.13)$$

Figure 4.2 shows the state transition function f of the nonlinear system

$$x(k+1) = f(x(k)),$$

which represents the discrete-time version of the continuous switched system given above with $\lambda_1 = -0.4$, $\lambda_2 = -0.7$, $T = 1$ and $\bar{x} = 2$. The function f is given by the right-hand side of (4.13) for $2 \leq x(k) \leq 2.98$ where the upper bound is given by the fact that the system jumps in one sampling step over the region bound $\bar{x} = 2$. \square

As a consequence, discrete-time linear affine models are usually approximations of the sampled-data system under consideration. As the example shows, the approximation may be bad near the border of the state-space partitions whereas inside the partitions the affine model correctly describes the affine continuous system at the sampling points.

4.2 Identification

4.2.1 Identification problem

Most analysis and synthesis procedures for hybrid systems assume that a hybrid model of the process at hand is available. In some situations it is possible to obtain models starting from first principles. However, in many practical situations first-principle modeling is too complicated to apply and models have to be reconstructed

from experimental data, what is referred to as parameter identification. This route is also mandatory when first principles are not available as often happens in fields like biology or economics.

The aim of this section is to provide an overview of methods for the identification of hybrid systems. Most of the results available in the literature focus on the reconstruction of input/output descriptions of switched affine and piecewise affine systems that are introduced in Section 4.2.2. In particular, we will focus on the following identification procedures:

- Methods that are capable to reconstruct *discontinuous models*. Indeed, discontinuities play a major role in hybrid systems since they allow logic conditions to be represented by abrupt changes in the system dynamics.
- Methods that do not assume any knowledge on the shape of the regions characterizing PWA systems.

4.2.2 Models in input/output form

Instead of the state-space model (4.7), (4.8), identification methods result in an input/output model with $\mathbf{u}(k) \in \mathbb{R}^p$ and $\mathbf{y}(k) \in \mathbb{R}^q$ denoting the input and the output of the system at time $k \in \mathbb{N}$. For fixed model orders n_a and n_b , consider the regression vector

$$\mathbf{r}(k) = [\mathbf{y}(k-1)^T \dots \mathbf{y}(k-n_a)^T \mathbf{u}(k)^T \mathbf{u}(k-1)^T \dots \mathbf{u}(k-n_b)^T]^T. \quad (4.14)$$

A *switched affine autoregressive exogenous* (SARX) model is defined by the equation

$$\mathbf{y}(k) = \theta(q(k))^T \begin{bmatrix} \mathbf{r}(k) \\ 1 \end{bmatrix} + \mathbf{e}(k), \quad (4.15)$$

where $q(k) \in \{1, \dots, s\}$ is the discrete state, s is the number of modes, θ_i , $i = 1, \dots, s$, are the matrices of parameters defining each mode, and $\mathbf{e}(k) \in \mathbb{R}^q$ is a noise/error term. Note that each mode has an affine dynamics and $q(k)$ can be viewed as an additional, possibly unknown, input. In the following, the vector $\varphi(k) = \begin{pmatrix} \mathbf{r}(k) \\ 1 \end{pmatrix}$ will be called the *extended* regression vector.

Piecewise affine autoregressive exogenous (PWARX) models are identical to SARX models except that the switching mechanism is determined by a polyhedral partition of the regressor domain $\mathcal{R} \subseteq \mathbb{R}^d$, where $d = q \cdot n_a + p \cdot (n_b + 1)$. This means that for these models the discrete state $q(k)$ is given by

$$q(k) = i \quad \text{iff} \quad \mathbf{r}(k) \in \mathcal{R}_i, \quad i = 1, \dots, s, \quad (4.16)$$

where $\{\mathcal{R}_i\}_{i=1}^s$ is a partition of \mathcal{R} . Each region \mathcal{R}_i is a convex polyhedron described by

$$\mathcal{R}_i = \{\mathbf{r} \in \mathbb{R}^d : H_i \begin{bmatrix} \mathbf{r} \\ 1 \end{bmatrix} \preceq_{[i]} \mathbf{0}\}, \quad (4.17)$$

where $H_i \in \mathbb{R}^{\mu_i \times (d+1)}$, $i = 1, \dots, s$, μ_i is the number of linear inequalities defining the i th polyhedral region \mathcal{R}_i and the symbol $\preceq_{[i]}$ denotes a μ_i -dimensional vector whose elements can be the symbols \leq and $<$. In general, the shape of \mathcal{R} reflects the

physical constraints on the inputs and the outputs of the system. For instance, typical constraints on the output can be $\|\mathbf{y}(k)\|_\infty \leq y_{\max}$ or $\|\mathbf{y}(k) - \mathbf{y}(k-1)\|_\infty \leq \Delta y_{\max}$, where $\|\cdot\|_\infty$ is the infinity norm of a vector, and y_{\max} and Δy_{\max} are fixed bounds.

By introducing the piecewise affine map

$$f(\mathbf{r}) = \theta_i^T \varphi \quad \text{if } H_i \varphi \preceq_{[i]} \mathbf{0}, \quad i = 1, \dots, s, \quad (4.18)$$

with $\varphi = \begin{bmatrix} r \\ 1 \end{bmatrix}$, $r \in \mathbb{R}^d$, the model defined by (4.15), (4.16) and (4.17) can be written as

$$\mathbf{y}(k) = f(\mathbf{r}(k)) + \mathbf{e}(k). \quad (4.19)$$

The PWA map (4.18) can be discontinuous along the boundaries defined by the polyhedra (4.17), as shown in Fig. 4.3. Though, for the sake of simplicity, in the following the subscript $[i]$ will be removed from the notation $\preceq_{[i]}$, one must always take care of the definition of the regions to avoid that the PWA map is multiply defined over common boundaries of the regions \mathcal{R}_i .

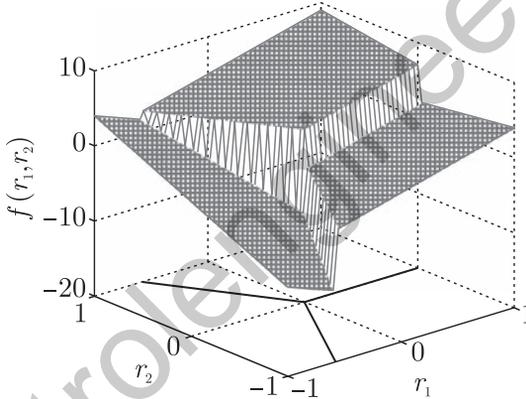


Fig. 4.3 Discontinuous PWA map of two variables with $s = 3$ regions.

4.2.3 Hybrid system identification problems

In this section, the identification problem will be stated for input/output models. For the sake of clarity, single input-single output systems (i.e. $p = q = 1$) are considered with scalar signals $y(k)$, $u(k)$, and $e(k)$, but the discussion can be straightforwardly extended to multi-input single-output systems ($p > 1$ and $q = 1$). Multi-input multi-output systems (i.e. $p > 1$ and $q > 1$) can be handled by identifying a model for each output while considering the other outputs as additional inputs. Note, however, that for PWARX models this approach may lead in general to a larger number of regions than necessary, since the overall partition is obtained by intersecting the partitions of individual models.

Identification problem for SARX models For the SARX model (4.15), the general identification problem reads as follows:

Problem 4.1 Given a collection of N input/output pairs $(y(k), u(k))$, $k = 1, \dots, N$, estimate the model orders n_a and n_b , the number of modes s , and the parameter vectors θ_i , $i = 1, \dots, s$. Moreover, estimate the discrete state $q(k)$ for $k > \max\{n_a, n_b\}$.

If the system generating the data has the structure (4.15), an exact algebraic solution to Problem 4.1 is presented in [431, 648] for the case of noiseless data (though the approach can be amended to work also with noisy data). The algorithm that is described in Section 4.2.5, only requires to fix upper bounds \bar{n}_a , \bar{n}_b , and \bar{s} on the model orders and the number of modes, respectively.

If the model orders are fixed, the problem is to fit the data to s hyperplanes. This problem is addressed in the field of data analysis, and several approaches are proposed where s is either estimated from data or fixed a priori. One way to estimate s is by solving the following problem:

Problem 4.2 Given $\delta > 0$, find the smallest number s of vectors θ_i , $i = 1, \dots, s$, and a mapping $k \mapsto q(k)$ such that

$$|y(k) - \varphi(k)^T \theta(q(k))| \leq \delta, \quad (4.20)$$

for all $k = \bar{n}, \dots, N$, where $\bar{n} = \max\{n_a, n_b\} + 1$.

Problem 4.2 consists of finding a *partition* of the system of inequalities

$$|y(k) - \varphi(k)^T \theta| \leq \delta, \quad k = \bar{n}, \dots, N, \quad (4.21)$$

into a *minimum* number of *feasible subsystems* (MIN PFS problem). The bound δ in (4.21) is not necessarily given a priori (e.g. if the noise is bounded, and the bound is known), rather it can be adjusted in order to find the desired trade-off between number of modes and accuracy. In fact, the smaller δ , the larger typically the number of modes needed to fit the data, while on the other hand, the larger δ , the worse the fit, since larger errors are allowed. The MIN PFS problem is NP-hard, and a suboptimal greedy randomized algorithm to solve it is proposed in [18].

If s is fixed, the well-known optimization approach used in linear system identification (i.e. choose the parameters of a linear model such that they minimize some prediction error norm) can be generalized to the identification of SARX models. Given a nonnegative function $\ell(\cdot)$, such as $\ell(\varepsilon) = \varepsilon^2$ or $\ell(\varepsilon) = |\varepsilon|$, the estimation of θ_i , $i = 1, \dots, s$, and of the discrete state $q(k)$ can be recast into the optimization problem:

$$\left\{ \begin{array}{l} \min_{\theta_i, \chi_{k,i}} \sum_{k=\bar{n}}^N \sum_{i=1}^s \ell(y(k) - \varphi(k)^T \theta_i) \chi_i(k) \\ s.t. \quad \sum_{i=1}^s \chi_i(k) = 1 \quad \forall k \\ \chi_i(k) \in \{0, 1\} \quad \forall k, i. \end{array} \right. \quad (4.22)$$

In (4.22), each binary variable $\chi_i(k)$ describes whether the data point $(y(k), \mathbf{r}(k))$ is associated to the i th mode, under the constraint that each data point must be associated to only one mode. The discrete state $q(k)$ can be finally reconstructed according to the rule:

$$q(k) = i \quad \text{iff} \quad \chi_i(k) = 1. \quad (4.23)$$

The optimization problem in (4.22) is a *mixed-integer* program that is computationally intractable, except for small instances. It is shown in [479] that (4.22) can be transformed into a smooth constrained optimization problem by relaxing the integer constraints, i.e. by requiring $\chi_i(k) \in [0, 1], \forall k, i$. The global optimum of the relaxed problem coincides with the global optimum of (4.22).

Identification problem for PWARX models For PWARX models defined by (4.15), (4.16), and (4.17), the general identification problem reads as follows:

Problem 4.3 Given a collection of N input/output pairs $(y(k), u(k))$, $k = 1, \dots, N$, estimate the model orders n_a and n_b , the number of modes s , the parameter vectors θ_i and the regions \mathcal{R}_i , $i = 1, \dots, s$.

All techniques specifically developed for the identification of PWARX models, assuming fixed orders n_a and n_b . The estimation of the model orders can be based on preliminary data analysis, and carried out by algebraic techniques such as [431, 648], or classical model order selection techniques [411]. Hence, in the following the orders n_a and n_b are given, and $\bar{n} = \max\{n_a, n_b\} + 1$.

The considered identification problem consists in finding the PWARX model that best matches the given data according to a specified criterion of fit. It involves the estimation of

- the number of modes s ;
- the parameters θ_i , $i = 1, \dots, s$, of the affine modes;
- the coefficients H_i , $i = 1, \dots, s$, of the hyperplanes defining the partition of the regressor set.

This issue also underlies a *classification problem* such that each data point is associated to one region, and to the corresponding mode. The simultaneous optimal estimation of all the quantities mentioned above is a very hard, computationally intractable problem. One of the main concerns is how to choose s in a sensible way. An additional difficulty is how to express efficiently the constraint that the collection $\{\mathcal{R}_i\}_{i=1}^s$ must form a partition of \mathcal{R} .

The problem becomes easy if the number of discrete states s is fixed, and the regions (4.17) are either known or fixed a priori. In that case each regression vector $\mathbf{r}(k)$ can be associated to one mode according to (4.16) and the dynamics of each mode can be reconstructed using linear identification techniques.

Most of the heuristics and suboptimal approaches that are applicable, or at least related, to the identification of PWARX models, either assume a fixed s , or adjust s iteratively (e.g. by adding one mode at a time) in order to improve the fit. A few techniques allow for the automatic estimation of s from data.

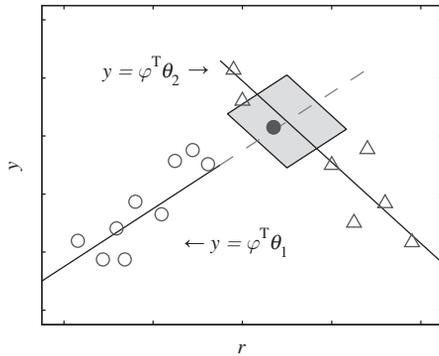


Fig. 4.4 Example showing the problem of intersecting modes. The data point denoted by the black circle could be, in principle, attributed to both modes. Wrong attribution yields two nonlinearly separable clusters of points.

4.2.4 Data classification and region estimation

A class of approaches for the identification of PWARX models, including those described in Section 4.2.5, share the idea to tackle the identification problem by firstly classifying the data and estimating the affine modes, and then estimating the partition of the regressor domain. In this section, the data classification step is discussed. Moreover, a brief overview of linear separation techniques is given, and issues related to the estimation of the partition from a finite number of points are highlighted.

Data classification Methods for the identification of PWARX models that firstly classify the data points and estimate the affine modes, and then estimate the partition of the regressor domain, split in practice the identification problem into the identification of a SARX model, followed by the shaping of the regions to the clusters of data. In this respect, such methods can be also considered as methods for the identification of SARX models, if the final region estimation step is not addressed. Vice versa, methods developed for the identification of SARX models, such as [431, 648], can be used to initialize the procedures for the identification of PWARX models.

However, in view of the subsequent step of region estimation, data classification for the identification of PWARX models needs to be carefully addressed. The main problem to deal with is represented by data points that are consistent with more than one mode, namely data points lying in the proximity of the intersection of two or more modes. Wrong attribution of these data points may lead to misclassifications when estimating the polyhedral regions.

In order to clarify this point, Fig. 4.4 shows a data set obtained from a one-dimensional PWA model with $s = 2$ discrete modes. It is assumed that the parameter vectors θ_1 and θ_2 have been previously estimated, no matter which method has been used. If each data point $(y(k), r(k))$ is associated to the mode i^* such that the prediction error is minimized, i.e. according to the rule

$$i^* = \arg \min_{i=1, \dots, s} |y(k) - \varphi(k)^T \theta_i|, \quad (4.24)$$

the point denoted by the black circle is attributed to the first mode. This yields two nonlinearly separable clusters of points. If data classification is performed solving Problem 4.2 for a given $\delta > 0$, still the black circle can be attributed to the first mode. Indeed, the gray area in Fig. 4.4 represents the region of all data points satisfying

$$|y(k) - \varphi(k)^T \theta_i| \leq \delta \quad (4.25)$$

for both $i = 1$ and $i = 2$. These data points are termed *undecidable*, because they could be in principle attributed to both modes.

The identification procedures [70, 236, 348, 482] deal with the problem of intersecting modes in different ways. For instance, an ad-hoc refinement procedure based on the certainly attributed closest neighbors is proposed in [70], weights for misclassification are introduced in [348], and clustering in a feature space is pursued in [236]. These three approaches will be described in Section 4.2.5.

Region estimation After the data classification step, providing the estimates of the discrete state $q(k) \in \{1, \dots, s\}$, it is possible to form s clusters of regression vectors as

$$\mathcal{A}_i = \{r(k) : q(k) = i\}, \quad i = 1, \dots, s. \quad (4.26)$$

The problem of region estimation consists in finding a complete polyhedral partition $\{\mathcal{R}_i\}_{i=1}^s$ of the regressor domain \mathcal{R} such that $\mathcal{A}_i \subseteq \mathcal{R}_i$ for all $i = 1, \dots, s$. The polyhedral regions (4.17) are defined by hyperplanes. Hence, the considered problem is equivalent to that of separating s sets of points by means of linear classifiers (hyperplanes). This problem can be tackled in two different ways:

1. Construct a linear classifier for each pair $(\mathcal{A}_i, \mathcal{A}_j)$, with $i \neq j$.
2. Construct a piecewise linear classifier which is able to discriminate between s classes simultaneously.

In the first approach, a separating hyperplane is constructed for each pair $(\mathcal{A}_i, \mathcal{A}_j)$, $i \neq j$. This amounts to solve $s(s-1)/2$ two-class linear separation problems. Approach 1 is computationally appealing, since it does not involve all the data simultaneously. A major drawback is that the estimated regions are not guaranteed to form a partition of the regressor domain when $d > 1$, as shown in Fig. 4.5(a). This drawback leads to PWA models that are not completely defined over the whole regressor domain.

If the presence of “holes” in the partition is not acceptable, one can resort to approach 2. *Multi-class* linear separation techniques construct s classification functions such that, at each data point, the corresponding class function is maximal. Classical two-class separation methods such as support vector machines (SVM) and robust linear programming (RLP) have been extended to this multi-class case [75, 117]. The resulting methods are called *multicategory* SVM (M-SVM) or *multicategory* RLP (M-RLP), to stress their ability of dealing with problems involving more than

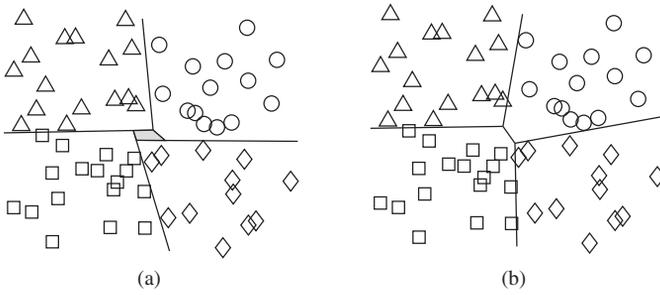


Fig. 4.5 Separation of a data set formed by four linearly separable sets. (a) Pairwise linear separation yielding a gray area that is not covered by any region. (b) Multi-class linear separation.

two classes (Fig. 4.5(b)). Multi-class problems involve all the available data, and, therefore, approach 2 is computationally more demanding than approach 1.

The previous algorithms for linear separation hinge on the idea of minimizing an error measure related to data points that are wrongly classified by a hyperplane. An alternative approach for solving point 1 is to find a hyperplane that minimizes the number of misclassified data points. Even if algorithms for solving this problem are NP-hard, several heuristics have been developed which work well in practice [18].

4.2.5 Four procedures for the identification of SARX/PWARX models

In this section, four procedures for the identification of SARX/PWARX models are briefly discussed, namely the algebraic procedure [431, 648], the clustering-based procedure [236], the Bayesian procedure [348], and the bounded-error procedure [70].

While the algebraic procedure focuses on the identification of SARX models, the other three procedures are designed for the identification of PWARX models, and are able to deal with discontinuous dynamics. The basic steps that each method performs are the estimation of the discrete state $q(k)$, and the estimation of the parameter vectors $\{\theta_i\}_{i=1}^s$.

Algebraic procedure The method proposed in [431, 648] approaches the identification of SARX models as an algebraic geometric problem. It provides a closed-form solution to the identification problem that is provably correct in the absence of noise.

The key idea behind the algebraic approach is to view the identification of multiple ARX models as the identification of a single, “lifted” ARX model that simultaneously encodes all the ARX modes and does not depend on the switching sequence. The parameters of the “lifted” ARX model can be identified through standard linear identification techniques after applying a polynomial embedding to the regression

vectors. The parameters of the original ARX modes are then given by the derivatives of this polynomial.

Assuming for simplicity that the number of modes s and the model orders n_a and n_b are known (these assumptions will be subsequently removed), the algebraic procedure works as follows. If the data are generated by model (4.15) with $e(k) = 0$ (noiseless case), each data pair $(y(k), \mathbf{r}(k))$ satisfies

$$y(k) - \theta_i^T \varphi(k) = 0, \quad (4.27)$$

for some $\theta_i, i = 1, \dots, s$. Hence, the following equality holds for all k

$$\prod_{i=1}^s (\mathbf{b}_i^T \mathbf{z}(k)) = 0, \quad (4.28)$$

where $\mathbf{b}_i = [1 \ \theta_i^T]^T$ and $\mathbf{z}(k) = [-y(k) \ \varphi(k)^T]^T$. Equation (4.28) is called the *hybrid decoupling constraint*, since it is independent of the switching sequence and the mechanism generating the transitions. In view of (4.28), the *hybrid decoupling polynomial* is defined as

$$p_s(\mathbf{z}) = \prod_{i=1}^s (\mathbf{b}_i^T \mathbf{z}) = \mathbf{h}^T \boldsymbol{\nu}_s(\mathbf{z}), \quad (4.29)$$

which is a homogeneous polynomial of degree s in $\mathbf{z} = [z_1 \ \dots \ z_K]^T, K = n_a + n_b + 3$. Note that (4.29) can be written as a linear combination of all the $M_s(K) \doteq \binom{s+K-1}{s}$ monomials of degree s in K variables. Such monomials are stacked in the vector $\boldsymbol{\nu}_s(\mathbf{z})$ according to the degree-lexicographic order. The vector $\mathbf{h} \in \mathbb{R}^{M_s(K)}$ contains the so-called *hybrid model parameters*, and encodes the parameter vectors of the s modes. Since (4.28) holds for all k , the vector \mathbf{h} can be estimated by solving the linear system (in a least-squares sense in the noisy case)

$$L_s(K) \mathbf{h} = 0, \quad (4.30)$$

where $L_s(K) = [\boldsymbol{\nu}_s(\mathbf{z}_{\bar{n}}) \ \boldsymbol{\nu}_s(\mathbf{z}_{\bar{n}+1}) \ \dots \ \boldsymbol{\nu}_s(\mathbf{z}_N)]^T$. Once \mathbf{h} has been computed, the vectors \mathbf{b}_i can be reconstructed as

$$\mathbf{b}_i = \frac{Dp_s(\mathbf{z}_{k_i})}{[1 \ 0 \ \dots \ 0] Dp_s(\mathbf{z}_{k_i})}, \quad (4.31)$$

where $Dp_s(\mathbf{z}) = \partial p_s(\mathbf{z}) / \partial \mathbf{z}$, and \mathbf{z}_{k_i} is a data point generated by the i th ARX mode, which can be chosen automatically once $p_s(\cdot)$ is known [650]. Given the \mathbf{b}_i 's (and consequently the θ_i 's), the discrete state is estimated according to the rule $q(k) = i^*$, with i^* given by (4.24). As discussed in Section 4.2.4, enhanced classification rules can be used by incorporating additional knowledge about the switching mechanism (e.g. PWARX models), when available.

The linear system (4.30) has a unique solution (requiring that the first component of \mathbf{h} is equal to 1) when the data are sufficiently exciting, and s, n_a and n_b are known exactly. If s is not known, it is shown in [650] that it can be estimated as

$$s = \arg \min \{i : \text{rank}(L_i(K)) = M_i(K) - 1\}. \quad (4.32)$$

Evaluation of (4.32) in the noisy case requires to introduce a threshold for eliminating the rank of $L_i(k)$. The algebraic procedure described above can be amended when only upper bounds \bar{s} , \bar{n}_a , or \bar{n}_b for s , n_a , or n_b , respectively, are available. In those cases, the procedure allows for the estimation of all the unknown quantities. More details can be found in [431, 648].

Clustering-based procedure The clustering-based procedure [236] exploits the fact that the PWA map (4.18) is locally linear. If the data are generated by (4.19), there likely exist groups of neighbor regression vectors belonging to the same region (and the same mode). Parameter vectors computed for these small local data sets should resemble the parameter vector of the corresponding mode. Hence, information about the modes can be obtained by clustering the local parameter vectors.

The clustering-based procedure works as follows. The positive integer c is a fixed parameter.

- *Local regression:* For $k = \bar{n}, \dots, N$, a local data set $\mathcal{C}(k)$ is built by collecting $(y(k), \mathbf{r}(k))$ and the data points (y_j, \mathbf{r}_j) corresponding to the $c - 1$ regression vectors \mathbf{r}_j that are closest (according to the Euclidean norm) to $\mathbf{r}(k)$. Local parameter vectors $\theta^{\text{LS}}(k)$ are then computed for each local data set $\mathcal{C}(k)$ by least squares. For analysis purposes, local data sets containing only data points generated by the same mode are referred to as *pure*, otherwise they are called *mixed*.

- *Construction of feature vectors:* Each data point $(y(k), \mathbf{r}(k))$ is mapped onto the feature vector

$$\xi(k) = [(\theta^{\text{LS}}(k))^T \mathbf{m}(k)^T]^T, \quad (4.33)$$

where $\mathbf{m}(k) = \frac{1}{c} \sum_{(y, \mathbf{r}) \in \mathcal{C}(k)} \mathbf{r}$ is the center of $\mathcal{C}(k)$.

- *Clustering:* Feature vectors are partitioned into s groups $\{\mathcal{F}_i\}_{i=1}^s$ by applying a “K-means”-like algorithm exploiting suitably defined confidence measures on the feature vectors. The confidence measures make it possible to reduce the influence of mixed feature vectors and poor initializations.

- *Parameter estimation:* Since the mapping of the data points onto the feature space is bijective, data points are classified into clusters $\{\mathcal{D}_i\}_{i=1}^s$ according to the rule

$$(y(k), \mathbf{r}(k)) \in \mathcal{D}_i \quad \text{iff} \quad \xi(k) \in \mathcal{F}_i. \quad (4.34)$$

A parameter vector θ_i is estimated for each data cluster \mathcal{D}_i by weighted least squares.

The clustering-based procedure requires that the model orders n_a and n_b , and the number of modes s are fixed. The parameter c , defining the cardinality of the local data sets, is the main tuning knob. In practical use, the method is expected to perform poorly if the ratio between the number of mixed and pure local data sets is high. The

number of mixed local data sets increases with c . Hence, it is desirable to keep c as small as possible. On the other hand, when the noise level is high, large values of c may be needed in order to filter out the effects of noise.

An important feature of the clustering-based procedure is its ability to distinguish modes characterized by the same parameter vector, but defined in different regions. This is possible because the feature vectors contain also information on the location of the local data sets. A modification to the clustering-based procedure is proposed in [232] to allow for the simultaneous estimation of the number of modes. The clustering-based procedure is analyzed in [233], where it is shown that optimal classification can be guaranteed under suitable assumptions in the presence of bounded noise. A software implementation of the clustering-based procedure is also available [231].

Bayesian procedure The Bayesian procedure [348] is based on the idea of exploiting the available prior knowledge about the modes and the parameters of the hybrid system. The parameter vectors θ_i are treated as random variables, and described through their probability density functions (pdfs) $p_{\theta_i}(\cdot)$. A priori knowledge on the parameters can be supplied to the procedure by choosing appropriate prior parameter pdfs. Various parameter estimates, such as expectation or maximum a posteriori probability estimate, can be easily obtained from the parameter pdfs. The data classification problem is posed as the problem of finding the data classification with the highest probability. Since this problem is combinatorial, an iterative suboptimal algorithm is derived. It is assumed that the probability density function $p_e(\cdot)$ of the additive noise term $e(k)$ is given.

Data classification and parameter estimation are carried out by sequential processing of the collected data points. In each iteration, the pdf of one of the parameter vectors is updated.

Let $p_{\theta_i}(\cdot; k)$ denote the pdf of θ_i at iteration k , when the data point $(y(k), \mathbf{r}(k))$ is considered. The conditional pdf $p((y(k), \mathbf{r}(k)) | q(k) = i)$ is given by

$$p((y(k), \mathbf{r}(k)) | q(k) = i) = \int_{\Theta_i} p((y(k), \mathbf{r}(k)) | \tilde{\theta}) p_{\theta_i}(\tilde{\theta}; k-1) d\tilde{\theta}, \quad (4.35)$$

where Θ_i is the set of possible values for θ_i , and

$$p((y(k), \mathbf{r}(k)) | \theta) = p_e(y(k) - \theta^T \varphi(k)). \quad (4.36)$$

The discrete state corresponding to $(y(k), \mathbf{r}(k))$ is estimated as $q(k) = i^*$, where

$$i^* = \arg \max_{i=1, \dots, s} p((y(k), \mathbf{r}(k)) | q(k) = i). \quad (4.37)$$

Then, the assignment of $(y(k), \mathbf{r}(k))$ to mode i^* is used to update the pdf of θ_{i^*} by using Bayes' rule

$$p_{\theta_{i^*}}(\theta; k) = \frac{p((y(k), \mathbf{r}(k)) | \theta) p_{\theta_{i^*}}(\theta; k-1)}{\int_{\Theta_{i^*}} p((y(k), \mathbf{r}(k)) | \tilde{\theta}) p_{\theta_{i^*}}(\tilde{\theta}; k-1) d\tilde{\theta}}. \quad (4.38)$$

Pdfs of the other parameter vectors remain unchanged, i.e. $p_{\theta_i}(\cdot; k) = p_{\theta_i}(\cdot; k - 1)$ for $i \neq i^*$. For the numerical implementation of the described algorithm, particle filtering algorithms are used [22]. After the parameter estimation phase, each data point is finally attributed to the mode that most likely generated it.

To estimate the regions, a modification of the standard multicategory RLP (MRLP) method [75] is proposed in [348]. If a regression vector attributed to mode i ends up in the region \mathcal{R}_j (this may happen, e.g., in the case of intersecting modes, Fig. 4.4), and the probabilities that the corresponding data point is generated by mode i and mode j are similar, misclassification should not be penalized highly. To this aim, for each data point $(y(k), \mathbf{r}(k))$ attributed to mode i , the price for misclassification into mode j is defined as

$$\nu_{i,j}(\mathbf{r}(k)) = \log \frac{p((y(k), \mathbf{r}(k)) \mid q(k) = i)}{p((y(k), \mathbf{r}(k)) \mid q(k) = j)}, \quad (4.39)$$

where $p((y(k), \mathbf{r}(k)) \mid q(k) = \ell)$ is the likelihood that $(y(k), \mathbf{r}(k))$ was generated by mode ℓ . Note that the price for misclassification is zero if the probabilities are exactly equal. Prices for misclassification are plugged into the MRLP method.

The Bayesian procedure requires that the model orders n_a and n_b , and the number of modes s are fixed. The most important tuning parameters are the prior parameter pdfs $p_{\theta_i}(\cdot; 0)$, and the pdf $p_e(\cdot)$ of the error term. In [345] the Bayesian approach has been extended to the identification of piecewise output error models.

Bounded-error procedure Inspired by ideas from set-membership identification ([453] the main feature of the bounded-error procedure [70, 503] is to impose that the error $e(k)$ in (4.19) is bounded by a given quantity $\delta > 0$ for all the samples $(y(k), \mathbf{r}(k))$ in the estimation data set, i.e.

$$|y(k) - f(\mathbf{r}(k))| \leq \delta, \quad \forall k = \bar{n}, \dots, N. \quad (4.40)$$

Hence, the bounded-error procedure fits a PWARX model satisfying (4.40) to the data, without any assumption on the system generating the data.

Since any PWARX model satisfying the bounded-error condition (4.40) is feasible, an initial guess of the number of modes s is obtained by addressing Problem 4.2. The solution of Problem 4.2, computed using the algorithms in [18] or their enhanced version proposed in [70], provides also a raw data classification. However, this classification procedure suffers from two drawbacks. The first one is related to the suboptimality of the method used to tackle Problem 4.2, implying that it is not guaranteed to yield the minimum number of modes. The second one is related to the problem of *undecidable* data points (Section 4.2.4), implying that the cardinality and the composition of the feasible subsystems may depend on the order in which they are extracted from (4.21).

To deal with the aforementioned drawbacks, an iterative *refinement* procedure is applied. The refinement procedure alternates between data reassignment and parameter update. If needed, it enables the reduction of the number of modes. For given positive thresholds α and β , modes i and j are merged if $\alpha_{i,j} < \alpha$, where

$$\alpha_{i,j} = \frac{\|\theta_i - \theta_j\|}{\min\{\|\theta_i\|, \|\theta_j\|\}}, \quad (4.41)$$

and $\|\cdot\|$ is the Euclidean norm. Mode i is discarded if the cardinality of the set \mathcal{D}_i of data points classified to mode i is less than βN . Data points that do not satisfy (4.40) are discarded as *infeasible* during the classification process, making it possible to detect outliers. In [70] parameter estimates are computed by the ℓ_∞ projection estimator, i.e.

$$\theta_i = \arg \min_{\theta} \max_{(y(k), r(k)) \in \mathcal{D}_i} |y(k) - \varphi(k)^T \theta|, \quad (4.42)$$

but any other projection estimate, such as least squares, can be used [453].

The bounded-error procedure requires that the model orders n_a and n_b are fixed. The main tuning parameter is the bound δ . As discussed in Section 4.2.3, the larger δ , the smaller the required number of modes at the price of a worse fit of the data. The optional parameters α and β , if used, also implicitly determine the final number of modes returned by the procedure. Another tuning parameter is the number c of closest neighbors used to attribute undecidable data points to modes in the refinement step.

4.2.6 Comparison of the identification procedures

The four identification procedures described in Section 4.2.5 are compared and discussed in [347, 349]. There, specific behaviors of the procedures with respect to classification accuracy, noise level, and tuning parameters are pointed out using simple one-dimensional examples. The procedures are also tested on the experimental identification of the electronic component placement process in pick-and-place machines.

From the comparison, it comes out that the algebraic procedure is well suited when the system generating the data can be accurately described as a switched affine model, and moderate noise is present. The main features of the algebraic procedure are that it can handle the cases with unknown model orders and an unknown number of modes, and it does not require any form of initialization. However, noise and/or nonlinear disturbances affecting the data may cause poor identification results. When trying to identify a PWARX model using the data classification obtained from the algebraic procedure, one must be aware that the minimum prediction error classification rule (4.24) may lead to wrong data association. In such cases, it is advisable to use one of the classification methods employed by other procedures.

The clustering-based procedure is well suited when there is no prior knowledge on the physical system, and one needs to identify a model with a prescribed structure (i.e. the number of modes and the model orders are given). Identification using the clustering-based procedure is straightforward, as only one parameter has to be tuned. However, poor results can be obtained when the model orders are overestimated, since distances in the feature space become corrupted by irrelevant information.

The Bayesian procedure is designed to take advantage of prior knowledge and physical insight into the operation modes of the system (like in the pick-and-place

machine identification [348]). Another interesting feature is the automatic computation of misclassification weights to be plugged into the linear separation techniques used for region estimation. As a major drawback, poor initialization may lead to poor identification results.

The bounded-error procedure is well suited when no prior knowledge on the physical system is available, and one needs to identify a model with a prescribed accuracy (e.g. to approximate nonlinear dynamics in each operation mode). Tuning parameters allow the model accuracy to be traded off against the model complexity, expressed in terms of the mean squared error and the number of modes, respectively. However, finding the right combination of the tuning parameters is seldom straightforward, and several attempts are often needed to get a satisfactory model.

It is stressed that mixing the features of the four procedures could still enhance their effectiveness, particularly in the following way:

- The algebraic procedure can be used to initialize the other three procedures by providing estimates of the model orders and the number of modes.
- By exploiting the idea of clustering the feature vectors, the clustering-based procedure is able to distinguish modes characterized by the same parameter vector, but defined in different regions. This is a pitfall of both the Bayesian and the bounded-error classification procedures. Since these procedures do not exploit the spatial location of the modes, data points generated by the same parameter vector in different regions are classified as a whole. This may lead to nonlinearly separable clusters. Clustering ideas contained in the clustering-based procedure can be extended to the other two procedures in order to detect and split the clusters corresponding to such situations.
- The Bayesian procedure includes the computation of misclassification weights to be plugged into the linear separation techniques used for region estimation. This feature can be extended to the clustering-based and the bounded-error procedures. In the latter case, for each data point $(y(k), \mathbf{r}(k))$ attributed to mode i , the price for misclassification into mode j could be defined as

$$\nu_{i,j}(\mathbf{r}(k)) = \lambda \log \max \left\{ 1, \frac{|y(k) - \theta_j^T \mathbf{r}(k)|}{\delta} \right\}, \quad (4.43)$$

where $\lambda > 0$ is a scale factor.

- The bounded-error procedure can be used to guess the number of modes, especially when the dynamics in each operation mode of the true system is nonlinear, and more modes than the true system are thus required to accurately approximate all the nonlinear dynamics.

4.3 Observation of linear switched systems

In many application domains, hybrid controller synthesis problems are addressed by assuming full hybrid state information, although in many realistic situations state measurements are not available. Hence, to make hybrid controller synthesis relevant, the design of hybrid state observers is of fundamental importance. A step towards a procedure for the synthesis of these observers is the analysis of observability of hybrid systems.

This section defines observability for linear switched systems as the possibility of reconstructing the hybrid state of the system from the knowledge of the output for a suitable choice of the control input. A general notion of observability is introduced for the class of linear switched systems, though this definition applies to more general classes of hybrid systems, since it involves only dynamical properties of the executions that are generated by the hybrid system. Further, a computable necessary and sufficient condition for assessing observability is proposed.

4.3.1 Preliminaries and basic definitions

Notation The symbols \mathbb{N} , \mathbb{R} , \mathbb{R}^+ , and \mathbb{R}_0^+ denote the natural, real, positive, and nonnegative real numbers, respectively. The symbol I denotes the identity matrix of appropriate dimensions. The symbol $\|\cdot\|$ denotes the Euclidean norm of a vector in the linear space \mathbb{R}^n . Given a linear subspace \mathcal{H} of \mathbb{R}^n , the symbol $\dim(\mathcal{H})$ denotes its dimension. Given a matrix $M \in \mathbb{R}^{n \times m}$, the symbol $\rho(M)$ denotes the rank of M and the symbols $\text{Im}(M)$ and $\ker(M)$ denote respectively the range and the null space of M ; given a set $\mathcal{H} \subseteq \mathbb{R}^n$ the symbol $M^{-1}(\mathcal{H})$ denotes the inverse image of \mathcal{H} through M , i.e. $M^{-1}(\mathcal{H}) = \{x \in \mathbb{R}^m \mid \exists y \in \mathcal{H} : y = Mx\}$. Given a countable set \mathcal{H} the symbol $\text{card}(\mathcal{H})$ denotes the cardinality of \mathcal{H} .

Switched systems We consider the class of linear switched systems. The hybrid state ξ is composed of two components: the discrete state q belonging to the finite set \mathcal{Q} , called the discrete state space, and the continuous state x belonging to the linear space \mathbb{R}^n . The hybrid state space of \mathcal{S} is then defined by $\mathcal{Q} \times \mathbb{R}^n$. The control input of \mathcal{S} is a function $u \in \mathcal{U}$, where \mathcal{U} denotes the class of piecewise continuous functions $u : \mathbb{R} \rightarrow \mathbb{R}^m$. The output function of \mathcal{S} belongs to the set \mathcal{Y} of piecewise continuous functions $y : \mathbb{R} \rightarrow \mathbb{R}^l$. The evolution of the continuous state x and of the output y of \mathcal{S} is determined by the linear control systems:

$$S(q) : \begin{cases} \dot{x} = A(q)x + B(q)u, \\ y = C(q)x, \end{cases} \quad (4.44)$$

whose matrices $A(q)$, $B(q)$, $C(q)$ depend on the current discrete state $q \in \mathcal{Q}$. The evolution of the discrete state of \mathcal{S} is governed by a finite-state machine (FSM), so that a transition from a state $q \in \mathcal{Q}$ to a state $p \in \mathcal{Q}$ may occur if $e = (q, p) \in \mathcal{E}$,

where $\mathcal{E} \subseteq \mathcal{Q} \times \mathcal{Q}$ is the set of (admissible) transitions in the FSM. A *linear switched system* \mathcal{S} is then specified by means of the tuple:

$$\mathcal{S} = (\mathcal{Q}, \mathbb{R}^n, \mathcal{S}, \mathcal{E}). \quad (4.45)$$

The evolution in time of linear switched systems can be defined by means of the notion of *execution*. We recall that a hybrid time basis τ is an infinite or finite sequence of sets $I_j = [t_j, t_{j+1})$, $j = 0, 1, \dots, \text{card}(\tau) - 1$, with $t_{j+1} > t_j$; let $\text{card}(\tau) = L$. If $L < \infty$, then $t_L = \infty$. Given a hybrid time basis τ , time instants t_j are called *switching times*. Let \mathcal{T} be the set of all hybrid time bases and consider a collection:

$$\chi = (\xi_0, \tau, \mathbf{u}, \xi, \mathbf{y}), \quad (4.46)$$

where $\xi_0 \in \mathcal{Q} \times \mathbb{R}^n$ is the initial hybrid state, $\tau \in \mathcal{T}$ is the hybrid time basis, $\mathbf{u} \in \mathcal{U}$ is the continuous control input, $\xi : \mathbb{R} \rightarrow \mathcal{Q} \times \mathbb{R}^n$ is the hybrid state evolution and $\mathbf{y} \in \mathcal{Y}$ is the output evolution. The function ξ is defined as:

$$\xi(t_0) = \xi_0, \quad \xi(t) = (q(t), \mathbf{x}(t)),$$

where $(q(t_{j-1}), q(t_j)) \in \mathcal{E}$ for any $j = 1, 2, \dots, L$, and at time $t \in I_j$, $q(t) = q(t_j)$, $\mathbf{x}(t)$ is the (unique) solution of the dynamical system $S(q(t_j))$, with initial time t_j , initial state $\lim_{t \rightarrow t_j^-} \mathbf{x}(t)$ and control law \mathbf{u} . The output evolution \mathbf{y} is defined for any $j = 0, 1, \dots, L - 1$ by:

$$\mathbf{y}(t) = \mathbf{C}_{q(t_j)} \mathbf{x}(t), \quad t \in [t_j, t_{j+1}).$$

A tuple χ of the form (4.46), which satisfies the conditions above, is called an *execution* of \mathcal{S} [427]. We assume the existence of a *minimum dwell time* [475] before which no transition occurs, and of a *maximum dwell time* [567] before which a transition certainly occurs.

Assumption 4.1 (Minimum and maximum dwell time) *Given the linear switched system \mathcal{S} , there exist $\delta_m \in \mathbb{R}_0^+$ and $\delta_M \in \mathbb{R}_0^+ \cup \{+\infty\}$, called respectively minimum and maximum dwell time, so that any execution $\chi = (\xi_0, \tau, \mathbf{u}, \xi, \mathbf{y})$ has to satisfy the condition*

$$\delta_m < t'_j - t_j < \delta_M, \quad \forall j = 0, 1, \dots, L - 1. \quad (4.47)$$

The existence of a minimum dwell time is a widely used assumption in the analysis of switched systems (e.g. [401, 475] and the references therein), and models the inertia of the system to react to an external (discrete) input. The existence of a maximum dwell time is related to the so-called liveness property of the system and is widely used in the context of discrete-event systems (e.g. [496]). Moreover, as shown in [569], minimum and maximum dwell times offer a method for approximating hybrid systems by means of switched systems.

4.3.2 Observability of switched systems with arbitrary switching

In this section we introduce a notion of observability for switched systems which ensures the (exact) reconstruction of the hybrid state. We start by reviewing some notions of observability introduced in the literature of hybrid systems and by showing that, for the class of switched systems, these definitions do not allow the reconstruction of the state of the system.

Survey of observability notions The observability of jump-linear systems is considered in [649], which are autonomous switched systems (i.e. $B(q) = 0, \forall q \in \mathcal{Q}$) having a minimum dwell time $\delta_m > 0$ and no maximum dwell time, i.e. $\delta_M = +\infty$. A notion of observability is proposed, which is based on the concept of indistinguishability of continuous initial states and discrete state evolutions. This notion of observability is rather strong since it considers only the free response to reconstruct the state.

Consider a switched system $\mathcal{S} = (\mathcal{Q}, \mathbb{R}^n, \mathcal{S}, \mathcal{E})$ and let $\mathcal{X}_0 \subseteq \mathbb{R}^n$ be a set of initial states such that, for any $x_0 \in \mathcal{X}_0$, all systems $S(q)$ have the same free continuous output. (The set \mathcal{X}_0 is nonempty since it contains at least the origin.) Assume that all systems $S(q)$ are observable and that there exists an input $u \in \mathcal{U}$ and $\Delta \in (0, \delta_m)$ such that

$$\int_0^{\Delta} \|y_1(s) - y_2(s)\| ds > 0, \forall q_1, q_2 \in \mathcal{Q}, q_1 \neq q_2,$$

where $y_1(t)$ and $y_2(t)$ are the outputs at time t of systems $S(q_1)$ and $S(q_2)$, respectively, starting from initial states in \mathcal{X}_0 , under the same input function u . Then, even though \mathcal{S} is not observable in the sense of [649], at time $t_j + \Delta$ the discrete state $q(t_j)$ can be determined, $\forall j = 0, 1, \dots, L$, and the continuous state $x(t)$ can be reconstructed $\forall t \in (t_j + \Delta, t'_j], \forall j = 0, 1, \dots, L$, for a suitable input function.

The forced response of the system is used in [65] where, for the class of piecewise affine (PWA) systems, incremental observability, is introduced. Informally, a PWA system is said to be incrementally observable if for any pair of continuous initial states in a given state set and for any input sequence in a given input set, the output trajectories are *sufficiently* different. In other words, incremental observability implies that different initial states always give different outputs independently of the applied input. The definition of incremental observability of [65] can be trivially extended to the class of linear switched systems. To better analyze the consequences of such a definition, consider a switched system $\mathcal{S} = (\mathcal{Q}, \mathbb{R}^n, \mathcal{S}, \mathcal{E})$ with minimum dwell time $\delta_m > 0$ and no maximum dwell time, i.e. $\delta_M = +\infty$. Assume that the systems $S(q)$ in operation modes q are controllable, $\mathbb{R}^m = \mathbb{R}^n$, and suppose that for any $q \in \mathcal{Q}$ the matrices $C(q)$ are nonsingular and are such that $\rho(C(q_1) - C(q_2)) = n, \forall q_1, q_2 \in \mathcal{Q}, q_1 \neq q_2$. In that case, for any $x \in \mathbb{R}^n \setminus \{0\}$, $C_{q_1}x \neq C_{q_2}x$. Therefore, for any pair of initial states $(q_1, x_1), (q_2, x_2) \in \mathcal{Q} \times (\mathbb{R}^n \setminus \{0\})$ and for any input function, the output functions of the switching system \mathcal{S} do not coincide, for any execution of \mathcal{S} . Hence, \mathcal{S} is incrementally observable for any set of initial states $\mathcal{Q}_0 \times \mathcal{X}_0 \subset \mathcal{Q} \times (\mathbb{R}^n \setminus \{0\})$. However, there exist input functions such that the discrete

state evolution of \mathcal{S} cannot be reconstructed. In fact, since the systems $S(q)$ are controllable, for all \mathbf{x} belonging to any subset of $\mathbb{R}^n \setminus \{0\}$ and for any $\hat{t} \in (t_0, t_0 + \delta_m)$ there exists an input function such that $\mathbf{x}(t) = 0, \forall t \geq \hat{t}$. As a consequence, it is not always possible to reconstruct the discrete state evolution, even if the state $q(0)$ were known. This shows that, for switching systems, incremental observability, based on a distinguishability property that holds for any input, does not guarantee state reconstruction.

Consider now a definition of observability based on distinguishability of initial states from the output, for a *suitable* input function. The following example shows that this notion has problems too for state reconstruction. Consider the switched system $\mathcal{S} = (\mathcal{Q}, \mathbb{R}^n, \mathcal{S}, \mathcal{E})$ with $\delta_m > 0$ and $\delta_M < +\infty$, where $\mathcal{Q} = \{q_1, q_2, q_3\}$, $\mathcal{E} = \{(q_2, q_1), (q_3, q_1), (q_1, q_1)\}$, the system $S(q_1)$ is observable, and $S(q_2) = S(q_3)$. Any pair of initial states $(q_2, \mathbf{x}_0), (q_3, \mathbf{x}_0)$, is indistinguishable, since for any input function $\mathbf{u} \in \mathcal{U}$, the same output functions are observed. However, after the first switching (which is guaranteed to happen since $\delta_M < +\infty$), the discrete state evolution is uniquely determined and the continuous state evolution can be reconstructed for any continuous input function, since $S(q_1)$ is observable.

Consequently, observability notions based on state indistinguishability do not imply state reconstructability. One of the concepts introduced in [597] based on state reconstruction is the so-called generic finite-state determinability. Generic finite-state determinability implies that *any* input/output experiment allows the determination of the state. In [36], this property was extended to hybrid systems and testable sufficient conditions were given. In this paper, following [567], we modify the notion of observability given in [597] for switched systems by focussing on state reconstruction.

Definition 4.1 (Observability of linear switched system) *A linear switched system \mathcal{S} is observable if there exist a control input $\hat{\mathbf{u}} \in \mathcal{U}$, a time $\hat{t} > 0$ and a function $\hat{\xi} : \mathcal{Y} \times \mathcal{U} \rightarrow \mathcal{Q} \times \mathbb{R}^n$ such that*

$$\hat{\xi}(\mathbf{y}|_{[t_0, t]}, \hat{\mathbf{u}}|_{[t_0, t]}) = \xi(t), \forall t \geq \hat{t}, t \neq t_j, j = 0, 1, \dots, L, \quad (4.48)$$

for any execution χ with control input $\hat{\mathbf{u}}$.

By specializing Definition 4.1 to linear systems, the classical observability notion is recovered. Note that the reconstruction of the current hybrid state is required at every time $t \geq \hat{t}$ with $t \neq t_j$. Time instants t_j are ruled out as it is for observable linear systems, where the current state may be reconstructed only at every time strictly greater than the initial time. However, observability for linear systems is defined independently from the control function, while here we assume that a suitable control law can be chosen. The two definitions coincide for linear systems but not for linear switched systems. In fact, if the observability property were required for *any* input function, then any linear switched system would never be observable [31, 567]. However, we will show in the next section that if a switched system is observable in the sense of Definition 4.1, then it is observable for “almost all” input functions.

4.3.3 Observability of switched systems with minimum dwell time

In this section we characterize observability of switched systems with no maximum dwell time $\delta_M = +\infty$, which occur in many applications. We start by focussing on the reconstruction of the discrete component of the hybrid state *only*. By specializing Definition 4.1, we have:

Definition 4.2 (Location observability) *A linear switched system S is location observable if there exist a control input $\hat{\mathbf{u}} \in \mathcal{U}$, a time $\hat{t} > 0$ and a function $\hat{q}: \mathcal{Y} \times \mathcal{U} \rightarrow \mathcal{Q}$ such that*

$$\hat{q}(\mathbf{y}|_{[t_0, t]}, \hat{\mathbf{u}}|_{[t_0, t]}) = q(t), \forall t \geq \hat{t}, t \neq t_j, j = 0, 1, \dots, L-1, \quad (4.49)$$

for any execution χ with control input $\hat{\mathbf{u}}$.

A linear switched system S is said to be *location observable for a control input $\hat{\mathbf{u}} \in \mathcal{U}$* if there exists a function $\hat{q}: \mathcal{Y} \times \mathcal{U} \rightarrow \mathcal{Q}$ such that condition (4.49) is satisfied. For later use, given $q, p \in \mathcal{Q}$, define the following augmented linear system $S(q, p)$:

$$\dot{\mathbf{z}} = \mathbf{A}(q, p)\mathbf{z} + \mathbf{B}(q, p)\mathbf{u}, \quad \mathbf{y}(q, p) = \mathbf{C}(q, p)\mathbf{z}, \quad (4.50)$$

where

$$\mathbf{A}(q, p) = \begin{pmatrix} \mathbf{A}(q) & \mathbf{O} \\ \mathbf{O} & \mathbf{A}(p) \end{pmatrix}, \quad \mathbf{B}(q, p) = \begin{pmatrix} \mathbf{B}(q) \\ \mathbf{B}(p) \end{pmatrix}, \quad \mathbf{C}(q, p) = (\mathbf{C}(q) - \mathbf{C}(p)).$$

Let $\mathcal{V}(q, p) \subseteq \mathbb{R}^{2n}$ be the maximal controlled invariant subspace [47] for system $S(q, p)$ contained in $\ker(\mathbf{C}(q, p))$, i.e. the maximal subspace $\mathcal{F} \subseteq \mathbb{R}^{2n}$ satisfying the following sets inclusions:

$$\mathbf{A}(q, p)\mathcal{F} \subseteq \mathcal{F} + \mathfrak{S}(\mathbf{B}(q, p)), \quad \mathcal{F} \subseteq \ker(\mathbf{C}(q, p)). \quad (4.51)$$

Define $\hat{\mathcal{Q}} = \{(q, p) \in \mathcal{Q} \times \mathcal{Q} : q \neq p\}$ and consider the set

$$\mathcal{U}^* = \left\{ \mathbf{u} \in \mathcal{U} : \mathbf{u} \neq \tilde{\mathbf{u}}, \text{ a.e., } \forall \tilde{\mathbf{u}} \in \tilde{\mathcal{U}} \right\},$$

where

$$\begin{aligned} \tilde{\mathcal{U}} &= \bigcup_{(q, p) \in \hat{\mathcal{Q}}} \mathcal{U}(q, p), \\ \mathcal{U}(q, p) &= \left\{ \mathbf{u} \in \mathcal{U} : \mathbf{u}(t) = \mathbf{K}(q, p)\mathbf{z}(t) + \mathbf{v}(t), \right. \\ &\quad \left. t \geq \hat{t} \text{ for some } \hat{t} \in \mathbb{R} \right\}, \end{aligned} \quad (4.52)$$

the gain $\mathbf{K}(q, p)$ is such that

$$(\mathbf{A}(q, p) + \mathbf{B}(q, p)\mathbf{K}(q, p))\mathcal{V}(q, p) \subseteq \mathcal{V}(q, p),$$

$\mathbf{v}(t) \in \mathbf{B}(q, p)^{-1}(\mathcal{V}(q, p))$, $\forall t \geq \hat{t}$ and $\mathbf{z}(t)$ is the state of system $S(q, p)$ at time t , under control \mathbf{u} with $\mathbf{z}(\hat{t}) \in \mathcal{V}(q, p)$. The set \mathcal{U}^* is composed of the control inputs \mathbf{u} such that after a finite time \hat{t} the output $\mathbf{y}(q, p)$ of $S(q, p)$ with any initial state $\mathbf{x}_0 \in \mathbb{R}^{2n}$ and the control input \mathbf{u} is not identically zero for any choice of $(q, p) \in \hat{\mathcal{Q}}$. We will show that control inputs in \mathcal{U}^* ensure the reconstruction of the discrete state. The following result identifies conditions for the nonemptiness of \mathcal{U}^* :

Lemma 4.1 *Given a linear switched system \mathcal{S} , the set \mathcal{U}^* is nonempty if*

$$\forall (q, p) \in \hat{\mathcal{Q}}, \exists k \in \mathbb{N}, k < 2n : \mathbf{C}(q)\mathbf{A}(q)^k\mathbf{B}(q) \neq \mathbf{C}(p)\mathbf{A}(p)^k\mathbf{B}(p). \quad (4.53)$$

The proof of the above result requires some technicalities and is, therefore, not given here. We now have all the ingredients for characterizing the location observability of switched systems.

Theorem 4.1 *A linear switched system \mathcal{S} is location observable if and only if condition (4.53) holds.*

Proof (Necessity) Suppose by contradiction that $\exists (q, p) \in \hat{\mathcal{Q}}$ such that condition (4.53) is not satisfied and consider any $\mathbf{u} \in \mathcal{U}$ and any executions $\chi_1 = ((q, 0), \tau, \mathbf{u}, \xi_1, y_1)$ and $\chi_2 = ((p, 0), \tau, \mathbf{u}, \xi_2, y_2)$ with $\tau = \{I_0\}$ and $I_0 = [0, \infty)$. It is readily seen that $y_1 = y_2$ and, therefore, the discrete state cannot be reconstructed.

(Sufficiency) By Lemma 4.1, condition (4.53) implies that $\mathcal{U}^* \neq \emptyset$; choose any $\mathbf{u} \in \mathcal{U}^*$ and consider any execution $\chi = (\xi_0, \tau, \mathbf{u}, \mathbf{x}, \mathbf{y})$. Consider any $j < L$ and let $\xi(t) = (q, \mathbf{x}(t)), t \in [t_j, t_{j+1})$. Given any $p \in \mathcal{Q}$, denote by $\mathbf{y}(q, p)(t, t_j, z, \mathbf{u}|_{[t_j, t)})$ the output evolution at time t of system $S(q, p)$ with initial state $z \in \mathbb{R}^{2n}$ at initial time t_j and control law $\mathbf{u}|_{[t_j, t)}$. Since $\mathbf{u} \in \mathcal{U}^*$ then for any $\varepsilon > 0$, for any $q \neq p$ and for any $w \in \mathbb{R}^n$ there exists a time $t \in (t_j, t_j + \varepsilon)$ such that $\mathbf{y}(q, p)(t, t_j, (\mathbf{x}(t_j) \ w)', \mathbf{u}) \neq 0$. This implies that $\mathbf{y}(t) \neq \mathbf{y}'(t)$, where $\mathbf{y}'(t)$ is the output associated with the execution $(\xi'_0, \tau, \mathbf{u}, \xi', \mathbf{y}')$ with $\xi'(t) = (p, \mathbf{x}'(t)), t \in [t_j, t_{j+1})$. Hence, the discrete state can be reconstructed for any $t \in (t_j, t_{j+1})$, and the statement follows. \square

It is seen from the above result that *if a linear switching system \mathcal{S} is location observable then it is location observable for any input function $\mathbf{u} \in \mathcal{U}^*$* . If the set of control inputs is the set $\mathcal{C}^\infty(\mathbb{R}^m)$ of smooth functions $\mathbf{u} : \mathbb{R} \rightarrow \mathbb{R}^m$ (instead of the set \mathcal{U} of piecewise continuous functions), then \mathcal{U}^* contains all and nothing but the control inputs which ensure location observability. Definition 4.1 implies the following corollary:

Corollary 4.1 *A linear switched system is observable if and only if it is location observable and $S(q)$ is observable for any $q \in \mathcal{Q}$.*

The intuitive algorithm for the reconstruction of the (current) hybrid state of an observable linear switched system \mathcal{S} , processes the output $\mathbf{y} \in \mathcal{Y}$ and the input $\mathbf{u} \in \mathcal{U}^*$. It first reconstructs the current discrete state, by looking for the unique $q \in \mathcal{Q}$ such that

$$\mathbf{Y}^{(n)}(t) \in \mathfrak{S}(\mathcal{O}(q)) + \mathcal{F}(q)\mathbf{u}(t), \quad (4.54)$$

where $\mathbf{Y}^{(n)}(t) = (\mathbf{y}(t)' \ \dot{\mathbf{y}}(t)' \ \dots \ \mathbf{y}^{(n-1)}(t)')'$, $\mathcal{O}(q)$ is the observability matrix associated with $S(q)$ and

$$\mathcal{F}(q) = \begin{pmatrix} C(q) & O & \cdots & O \\ C(q)A(q) & C(q)B(q) & \cdots & O \\ \cdots & \cdots & \cdots & O \\ C(q)A(q)^n & C(q)A(q)^{n-1}B(q) & \cdots & C(q)B(q) \end{pmatrix}.$$

Then, on the basis of the knowledge of q , it reconstructs the current continuous state $\mathbf{x}(t)$, by computing:

$$\{\mathbf{x}(t)\} = \mathcal{O}(q)^{-1} \left(Y^{(n)}(t) - \mathcal{F}(q)\mathbf{u}(t) \right). \quad (4.55)$$

Note that if the switched system \mathcal{S} is location observable and $\mathbf{u} \in \mathcal{U}^*$, Theorem 4.1 guarantees that discrete state q is unique. Moreover if the switched system $\mathcal{S}(q)$ is observable then the state $\mathbf{x}(t)$ is unique.

The combination of (4.54) and (4.55) is a hybrid observer. However, such an observer requires an infinite precision in the computation of the vector $Y^{(n)}(t)$. Further work will identify appropriate conditions on linear switching systems, for the existence and design of hybrid observers.

4.4 Stability analysis

4.4.1 Stability problems

This section presents stability criteria for switched systems. As Example 2.3 has shown, the stability of the system in every operation mode q is neither necessary nor sufficient for the stability of the switched system. Hence, new stability criteria have to be derived for hybrid systems.

The section focuses on continuous-time switched linear systems

$$\dot{\mathbf{x}}(t) = \mathbf{A}_{q(t)}\mathbf{x}(t), \quad \mathbf{x}(0) = \mathbf{x}_0, \quad (4.56)$$

for which the most important stability notions are available. The particularities of the discrete-time case will be summarized in Section 4.4.8. Time domain restrictions, such as dwell time conditions, will be addressed in Section 4.5 dedicated to stabilization and control.

There are two important stability analysis problems, which refer to the two situations introduced in Section 4.1 (Fig. 4.6):

- *Stability of controlled switched system:* Under what condition is the switched system stable for arbitrary switching, i.e. for all switching functions $q : \mathbb{R}^+ \rightarrow \mathcal{Q}$?
- *Stability of autonomous switched system:* Under what condition is a switched system stable for a given switching rule?

This section focuses on the first problem, which is relevant for all systems that underlie an unknown discrete input changing the operation mode of the system in an arbitrary way. The following systems belong to this class:

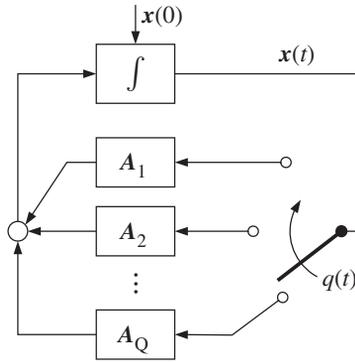


Fig. 4.6 Stability of piecewise affine systems.

- *Multi-controller schemes:* If several controllers are designed and implemented for the same plant and one of them is selected at any time with respect to the current disturbance or operating conditions, then the overall system is a switched system with arbitrary switching function. The stability of the overall system has to be ensured for all switching functions.
- *Networked control systems:* If open communication networks are used for the information links between the controller and the plant, package drop-out may occur at arbitrary time instants. Then the system virtually acts with non-uniform sampling intervals, which in turn leads to different discrete-time models. The stability of the system has to be ensured for arbitrary switching.

The stability problem for autonomous switched systems is investigated in [Section 4.5](#), which deals with the problem to restrict the switching function such that the overall system is stable.

4.4.2 Stability notions

First, some fundamental concepts from stability theory are recalled. Intuitively, stability is a system property that corresponds to returning to its *equilibrium* position when it is punctually disturbed. Let's recall the classical stability definition for a nonlinear time-invariant autonomous system

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t)), \quad (4.57)$$

with $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^n$ a locally Lipschitz function.

Definition 4.3 (Stability) *The equilibrium point of the system (4.57) is*

- *stable if $\forall \epsilon > 0 \exists \delta = \delta(\epsilon) > 0$ such that*

$$\|\mathbf{x}(0) - \mathbf{x}^*\| < \delta \Rightarrow \|\mathbf{x}(t) - \mathbf{x}^*\| < \epsilon, \quad \forall t \geq 0;$$

- asymptotically stable if \mathbf{x}^* is stable and δ may be taken such that

$$\|\mathbf{x}(0) - \mathbf{x}^*\| < \delta \Rightarrow \lim_{t \rightarrow \infty} \mathbf{x}(t) = \mathbf{x}^*;$$

- exponentially stable if there exist three positive real scalars c , K , and λ such that

$$\|\mathbf{x}(t) - \mathbf{x}^*\| \leq K \|\mathbf{x}(0) - \mathbf{x}^*\| e^{-\lambda t}, \forall \|\mathbf{x}(0) - \mathbf{x}^*\| < c;$$

- globally asymptotically stable if \mathbf{x}^* is stable and $\forall \mathbf{x}(0) \in \mathbb{R}^n$

$$\lim_{t \rightarrow \infty} \mathbf{x}(t) = \mathbf{x}^*.$$

Very often, for simplicity reasons, the term *stable system* is used to describe a system that has a stable equilibrium point.

The concept of stability leads to the Lyapunov stability theory, which establishes the fact that a system whose trajectories are attracted toward an asymptotically stable equilibrium point is progressively losing its energy, in a monotone fashion. Lyapunov generalized the energy notion by using a function $V(\mathbf{x})$ which depends on the system state. This function is usually a norm.

Theorem 4.2 *Considering the nonlinear system (4.57) with an isolated equilibrium point $(\mathbf{x}^* = \mathbf{0} \in \Omega \subset \mathbb{R}^n)$. If there exist a locally Lipschitz function $V : \mathbb{R}^n \rightarrow \mathbb{R}$ that has continuous partial derivatives and two \mathcal{K} functions¹ α and β such that*

$$\alpha(\|\mathbf{x}\|) \leq V(\mathbf{x}) \leq \beta(\|\mathbf{x}\|), \forall \mathbf{x} \in \Omega \subset \mathbb{R}^n,$$

the origin $\mathbf{x} = \mathbf{0}$ of system (4.57) is

- stable if

$$\frac{dV(\mathbf{x})}{dt} \leq 0, \forall \mathbf{x} \in \Omega, \mathbf{x} \neq \mathbf{0};$$

- asymptotically stable if there exists a \mathcal{K} function φ such that

$$\frac{dV(\mathbf{x})}{dt} \leq -\varphi(\|\mathbf{x}\|), \forall \mathbf{x} \in \Omega, \mathbf{x} \neq \mathbf{0};$$

- exponentially stable if there exist four positive constant scalars $\bar{\alpha}, \bar{\beta}, \gamma, p$ such that

$$\alpha(\|\mathbf{x}\|) = \bar{\alpha} \|\mathbf{x}\|^p, \beta(\|\mathbf{x}\|) = \bar{\beta} \|\mathbf{x}\|^p, \varphi(\|\mathbf{x}\|) = \gamma \|\mathbf{x}\|.$$

These local properties are globally valid if \mathcal{K}_∞ functions are used in Theorem 4.2 instead of \mathcal{K} functions. A function $V(\mathbf{x})$ that verifies the properties given in the in Theorem 4.2 is called a *Lyapunov function* for the system.

¹ A function $\varphi : [0, a) \rightarrow [0, \infty)$ is a \mathcal{K} function, if it is strictly decreasing and $\varphi(0) = 0$. It is a \mathcal{K}_∞ function if $a = \infty$ and $\lim_{t \rightarrow \infty} \varphi(t) = \infty$.

Extensions of this theorem for the case of non-autonomous systems (a smooth nonlinear system with explicit dependence upon time) is given in [359]. The main new notion is uniformity, i.e. the stability properties are re-defined in order to guarantee that these properties are not dependent on the initial time t_0 . In the case of switched linear systems, which is the main subject of this section, we are mainly concerned with the following definition:

Definition 4.4 (Stability of switched systems) *The switched system*

$$\dot{\mathbf{x}} = \mathbf{f}_{q(t)}(\mathbf{x}(t)), \quad \mathbf{x}(0) = \mathbf{x}_0 \quad (4.58)$$

is called locally uniformly exponentially stable if there exist positive constants M , c , and μ such that for any switching signal q the solution of (4.58) with $\|\mathbf{x}(0)\| \leq M$ satisfies

$$\|\mathbf{x}(t)\| \leq ce^{-\mu t} \|\mathbf{x}(0)\|, \quad \forall t \geq 0.$$

The term “uniform” is used here to describe uniformity with respect to switching signals. If there exist positive constants c and μ such that the definition holds for any switching signal q and any initial condition $\mathbf{x}(0)$, then the switched system is called globally uniformly exponentially stable. Similarly, one can also define the property of uniform asymptotic stability as local or global. For switched linear systems, uniform exponential stability is equivalent to the weaker property of asymptotic stability for any switching signal [461]. From now on, for simplicity reasons, we will use the term “stable” if we design globally uniformly exponentially stable switched systems.

4.4.3 Stability of differential inclusions

The stability of switched systems under arbitrary switching signals is closely related to a well-known stability problem discussed in the literature for ordinary differential equation with discontinuous right side member, and more precisely for the case of differential inclusions.

Consider the linear differential inclusion described by

$$\dot{\mathbf{x}} \in \mathcal{F}(\mathbf{x}) = \{\mathbf{y} : \mathbf{y} = \mathbf{A}\mathbf{x}, \mathbf{A} \in \mathcal{A}\}, \quad (4.59)$$

where \mathcal{A} is a compact set. A switched linear system under the form

$$\dot{\mathbf{x}}(t) = \mathbf{A}_{q(t)}\mathbf{x}(t),$$

with $\mathbf{A}_{q(t)} \in \{\mathbf{A}_1, \mathbf{A}_2, \dots, \mathbf{A}_N\}$, $\forall q(t) \in \mathcal{Q}$, can be considered as a differential inclusion (4.59) with $\mathcal{A} = \{\mathbf{A}_1, \mathbf{A}_2, \dots, \mathbf{A}_N\}$.

Stability analysis of the linear differential inclusion (4.59) is connected to the analysis of its convex hull $\text{co}\mathcal{A}$ of the set \mathcal{A} .

Theorem 4.3 [461] *The inclusion (4.59) is stable if and only if the convex differential inclusion*

$$\dot{x} \in \{y : y = Ax, A \in \text{co}A\} \quad (4.60)$$

is stable.

When the set A is a convex polyhedron, an algebraic criteria can be deduced:

Theorem 4.4 *For the stability of the origin $x = 0$ of the convex linear differential inclusion*

$$\dot{x} \in \mathcal{F}(x) = \{y : y = Ax, A \in \text{co}\{A_1, \dots, A_M\}\} \quad (4.61)$$

it is necessary and sufficient that there exists a number $m > n$, a rank n matrix \mathcal{L} and M row diagonal negatives matrices ($m \times m$)

$$\Gamma_s = \left(\gamma_{ij}^{(s)} \right)_{i,j=1}^m, \quad \forall s = 1, \dots, M,$$

with

$$\gamma_{ii}^{(s)} + \sum_{i \neq j} \left| \gamma_{ij}^{(s)} \right| < 0, \quad \forall i = 1, \dots, m, s = 1, \dots, M,$$

such that the relation

$$A_s^T \mathcal{L} = \mathcal{L} \Gamma_s^T, \quad \forall s = 1, \dots, M$$

is verified.

This stability condition is closely related with the auxiliary stable differential inclusion

$$\dot{z} \in G(z) = \{y : y = \Lambda z, \Lambda \in \text{co}\{\Lambda_1, \dots, \Lambda_N\}\} \quad (4.62)$$

in the augmented space \mathbb{R}^m whose solutions contain the trajectories of the original inclusion. The \mathcal{L} matrix, with $z = \mathcal{L}^T x$, represents the associated transformation matrix. The proof is based on the existence of a quasi-quadratic Lyapunov function $V(x) = x^T P(x)x$.

4.4.4 Stability analysis of switched linear systems by means of a common Lyapunov function

The two theorems given in the last paragraph can be directly applied to switched linear systems. The main idea is the construction of a *common Lyapunov function* for all subsystems.

Consider the system

$$\dot{\mathbf{x}}(t) = \mathbf{A}_{q(t)} \mathbf{x}(t), \quad q(t) \in \mathcal{Q} = \{1, 2, \dots, Q\}, \quad (4.63)$$

which is subject to an arbitrary switching function $q(t)$. For all operation modes q , the system has the equilibrium $\mathbf{x} = \mathbf{0}$. The asymptotic stability of this equilibrium can be proved by constructing a Lyapunov function $V(\mathbf{x})$ that possesses the property

$$\frac{dV(\mathbf{x})}{dt} < 0$$

along the trajectory $\mathbf{x}(\cdot)$ of the system (4.63) as long as $\mathbf{x} \neq \mathbf{0}$ holds.

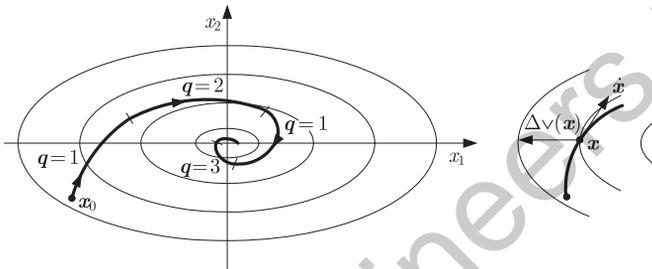


Fig. 4.7 Common Lyapunov function of a switching linear system.

The main idea of using a common Lyapunov function is illustrated in Fig. 4.7. The ellipsoidal lines show the states \mathbf{x} with $V(\mathbf{x}) = \text{const}$. Along the state trajectory $\mathbf{x}(\cdot)$ shown the value of $V(\cdot)$ decreases and, hence, the state asymptotically approaches the equilibrium $\mathbf{x} = \mathbf{0}$ in the center of the figure. To construct a common Lyapunov function a function $V(\cdot)$ has to be found such that $V(\cdot)$ decreases along all state trajectories that can be obtained by starting in any initial state $\mathbf{x}_0 \in \mathbb{R}^n$ and by considering all possible switching functions $q(t)$.

If a quadratic Lyapunov function $V(\mathbf{x}) = \mathbf{x}^T \mathbf{P} \mathbf{x}$ is used, a sufficient stability condition can be expressed as a linear matrix inequality (LMI).

Theorem 4.5 Consider the system (4.61). If there exists a matrix \mathbf{P} , $\mathbf{P} = \mathbf{P}^T > 0$ as a solution of the LMIs

$$\mathbf{A}_i^T \mathbf{P} + \mathbf{P} \mathbf{A}_i < 0, \quad \forall i = 1, \dots, Q, \quad (4.64)$$

then the quadratic function $V(\mathbf{x}) = \mathbf{x}^T \mathbf{P} \mathbf{x}$ is a Lyapunov function for the system (4.61), and the origin $\mathbf{x} = 0$ is stable.

When a common quadratic Lyapunov function exists, we may say that the system is *quadratically stable* and the term *quadratic stability* is used. This implies that there

exists a scalar ϵ such that

$$\frac{dV(\mathbf{x})}{dt} < -\epsilon \|\mathbf{x}\|.$$

4.4.5 Lie-algebraic stability criteria

This paragraph extends the idea of using a common Lyapunov function and provides further stability criteria for switched linear systems (4.63). The system dynamics is described by the matrix set $\mathcal{A} = \{\mathbf{A}_1, \mathbf{A}_2, \dots, \mathbf{A}_N\}$. The Lie algebra $\mathfrak{g} = \text{Lie}\{\mathbf{A}_i : i \in \mathcal{Q}\}$ corresponds to the set of matrices $\mathbf{A}_i, \forall i \in \mathcal{Q}$ and all the iterative commutators obtained using the Lie operator,

$$[\mathbf{A}_i, \mathbf{A}_j] = \mathbf{A}_i \mathbf{A}_j - \mathbf{A}_j \mathbf{A}_i, \forall i, j \in \mathcal{Q}.$$

Several algebraic stability criteria in relation with this Lie algebra have been presented in the literature.

If all the state matrices $\mathbf{A}_i, \forall i \in \mathcal{Q}$ are pairwise commutative, i.e. if the Lie operator $[\mathbf{A}_i, \mathbf{A}_j]$ is zero for all the pairs $\mathbf{A}_i, \mathbf{A}_j, i, j \in \mathcal{Q}$, the switched system (4.63) is asymptotically stable [4, 483]. Moreover, if the Lie algebra \mathfrak{g} is nilpotent, then the system is asymptotically stable. In addition, it can be shown that if the $\mathbf{A}_i, \forall i \in \mathcal{Q}$ matrices simultaneously accept an upper / lower triangulation, then there exists a common quadratic Lyapunov function.

Theorem 4.6 [473] *Consider the switched linear system (4.63). If all the $\mathbf{A}_i, i \in \mathcal{Q}$ matrices are Hurwitz stable and if there exists an invertible matrix $\mathbf{T} \in \mathbb{R}^{n \times n}$ such that all the matrices*

$$\mathbf{A}_i = \mathbf{T}^{-1} \mathbf{A}_i \mathbf{T}, \forall i \in \mathcal{Q}$$

are upper (or lower) triangular, then there exists a common quadratic Lyapunov function

$$V(\mathbf{x}) = \mathbf{x}^T \mathbf{P} \mathbf{x}$$

for the family of systems $\{\dot{\mathbf{x}} = \mathbf{A}_i \mathbf{x}, \forall i \in \mathcal{Q}\}$ and the switched system (4.63) is stable.

The previous results can be generalized for complex transformation matrices $\mathbf{T} \in \mathbb{C}^{n \times n}$ and a sufficient condition for a simultaneous upper triangulation of a matrix set in terms of solvable Lie algebra can be obtained. If

$$\mathfrak{g} = \text{Lie}\{\mathbf{A}_i : \forall i \in \mathcal{Q}\}$$

is a solvable Lie algebra, then the system family

$$\{\dot{\mathbf{x}} = \mathbf{A}_i \mathbf{x}, \forall i \in \mathcal{Q}\}$$

simultaneously accepts an upper/lower triangulation and the switched linear system (4.63) is stable.

When all the matrices of the set $\{A_i, \forall i \in Q\}$ are pairwise commutative, or they generate a nilpotent Lie algebra \mathfrak{g} , they also generate a solvable Lie algebra. However, this represents only a sufficient condition for simultaneous triangulation. The approach is important from a theoretical point of view since it establishes the connection between the Lie algebra based stability conditions and the simultaneous triangulation criteria [401, 405, 473]. However, all of these criteria represent sufficient conditions for the existence of a common Lyapunov function, which implies some conservatism.

4.4.6 Necessary and sufficient stability criteria

In order to reduce the conservatism of the previous results, necessary and sufficient conditions for the existence of a common quadratic Lyapunov function can be obtained for a pair of second-order systems. Consider the convex envelope described by the matrices $A_1, A_2 \in \mathbb{R}^{n \times n}$:

$$\text{co}\{A_1, A_2\} \triangleq \{\lambda A_1 + (1 - \lambda)A_2 : \lambda \in [0, 1]\}.$$

The set of systems

$$\{\dot{x} = A_1 x, \dot{x} = A_2 x\}, \quad A_1, A_2 \in \mathbb{R}^{2 \times 2}$$

has a common quadratic Lyapunov function if and only if all the matrices of the two convex envelopes $\text{co}\{A_1, A_2\}$ and $\text{co}\{A_1, A_2^{-1}\}$ are Hurwitz stables.

The extension to the general case is very difficult. One can show that for symmetric matrices

$$A_i = A_i^T, \quad \forall i \in Q$$

and normal systems

$$A_i \cdot A_i^T = A_i^T \cdot A_i, \quad \forall i \in Q,$$

a necessary and sufficient condition for both the existence of a common quadratic Lyapunov function and the stability is that all the subsystems are Hurwitz stable. More details on these particular cases can be found in [360, 585, 679].

4.4.7 Multiple Lyapunov functions

The existence of a common quadratic Lyapunov function is only a sufficient stability condition, not a necessary one. We can show analytically that there exist switched linear systems that are asymptotically stable for which no common quadratic Lyapunov function exists [205]. This means that looking for such a function may be too conservative. This motivates the search for other types of Lyapunov functions.

In the literature, we can find several types of Lyapunov functions that may be classified, in a generic framework, under the name of *multiple Lyapunov functions*. The multiple Lyapunov functions describe a family of functions of the form

$$V(x) = x^T P(q, x)x,$$

where the Lyapunov matrix may depend on the state vector and/or the switching law. The concatenation of these functions forms one common, non-quadratic, Lyapunov function.

Piecewise linear Lyapunov functions In the context of switched systems, one can notice that it is necessary and sufficient to have a quasi-quadratic Lyapunov function $V(\mathbf{x}) = \mathbf{x}^T \mathbf{P}(\mathbf{x})\mathbf{x}$, whose Lyapunov matrix is varying according to the system state. Based on these ideas, the concept of *piecewise linear Lyapunov functions* has been derived. This concept denotes a set of functions which when placed side by side results in an overall non-quadratic common Lyapunov function.

The first approach for deriving such function was to approximate the level functions of the quasi-quadratic Lyapunov function $V(\mathbf{x}) = \mathbf{x}^T \mathbf{P}(\mathbf{x})\mathbf{x}$ by a piecewise linear Lyapunov function:

$$V_m(\mathbf{x}) = \max_{1 \leq i \leq m} |\langle l_i, \mathbf{x} \rangle|. \quad (4.65)$$

The operator \langle, \rangle denotes the classical scalar product and the elements $l_i \in \mathbb{R}^n$, $i = 1, \dots, m$, represent constant vectors called generator vectors. For a sufficiently large number m of generator vectors, it is possible to show that the existence of such a function is necessary and sufficient for the stability. Few methods exist for verifying the existence of such functions. The difficulty lies in the a priori specification of generator vectors l_i . The existence of such a piecewise linear Lyapunov function may be verified by analyzing the spectrum of all the matrices in the convex hull generated by \mathbf{A}_i , $\forall i \in \mathcal{Q}$ [461, 491, 670].

Multiple Lyapunov-like functions Another approach is based on the concept of Lyapunov-like functions. These are families of piecewise continuous and piecewise differentiable functions that concatenated together produce a single non-traditional Lyapunov function. The discontinuous structure of switched systems suggest the use of discontinuous Lyapunov functions. A family of *Lyapunov-like functions* $\{V_i(\mathbf{x}) = \mathbf{x}^T \mathbf{P}_i \mathbf{x}, i \in \mathcal{Q}\}$ such that each vector field $\mathbf{A}_i \mathbf{x}$, $i \in \mathcal{Q}$ has its own Lyapunov function is used (Fig. 4.8). The particularity of Lyapunov-like functions is that the decay of the function is required only when the system is active. A multiple Lyapunov-like function satisfies the following conditions:

- $V_i(\mathbf{x}) = \mathbf{x}^T \mathbf{P}_i \mathbf{x}$ is positive definite $\forall \mathbf{x} \neq 0$ and $V_i(\mathbf{0}) = 0$;
- the derivative of each $V_i(\mathbf{x}) = \mathbf{x}^T \mathbf{P}_i \mathbf{x}$ function satisfies the relation

$$\dot{V}_i(\mathbf{x}) = \frac{\partial V_i}{\partial \mathbf{x}} \mathbf{A}_i \mathbf{x}(t) \leq 0 \quad \forall i \in \mathcal{Q} \quad (4.66)$$

when the i -th subsystem is active.

The stability results developed in this context are based on the decay of the Lyapunov function at any successive instants for which a subsystem is switched-in.

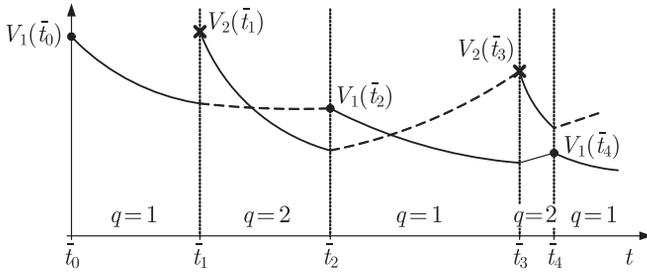


Fig. 4.8 Multiple Lyapunov functions.

Theorem 4.7 [511] Consider a family of Lyapunov-like functions V_q , each associated with a vector field $\mathbf{A}_q \mathbf{x}$. For $i < j$, let $t_i < t_j$ be the switching times for which $q(t_i) = q(t_j)$. If there exists a $\gamma > 0$ such that

$$V_{q(t_j)}(\mathbf{x}(t_{j+1})) - V_{q(t_i)}(\mathbf{x}(t_{i+1})) \leq -\gamma \|\mathbf{x}(t_{i+1})\|^2,$$

then the switched system is stable

A more general result, assuming a so-called *weak Lyapunov function*, can be obtained [676]. In this case the condition (4.66) is replaced by

$$V_i(\mathbf{x}(t)) \leq \alpha(V_i(\mathbf{x}(t_j))), \quad \forall t \in [t_j, t_{j+1}],$$

where $[t_j, t_{j+1}]$ is the time interval for which a subsystem i is active, t_j is any switching instant for which the system i is activated and $\alpha: \mathbb{R}^+ \cup \{0\} \rightarrow \mathbb{R}^+ \cup \{0\}$ is a continuous function that satisfies $\alpha(0) = 0$. This allows defining Lyapunov functions that may occasionally increase, but their growth is bounded.

The results are difficult to apply to the case of arbitrary switching sequences, because the theorem requires the system trajectory to be known at least at the switching instants. Another problem is that no analytical or numerical method for the construction of such a Lyapunov-like function is given. These problems can be solved for particular cases, for example if the switching sequence is a priori known or when the switching law depends on a partition of the state space as it is the case for piecewise affine systems. For an extension to nonlinear systems [113, 206].

4.4.8 Stability of discrete-time systems

In the discrete-time case, the continuous time approach for stability analysis of switched linear systems can be extended but some particularities should be mentioned. The most important particularity is related to necessary and sufficient

conditions for stability analysis or for the existence of multiple Lyapunov functions. While in the continuous time setting only specific cases lead to necessary and sufficient conditions, as those presented in Section 4.4.6, some general problems admit a more interesting solution in the discrete-time case.

Necessary and sufficient conditions for stability analysis In discrete time, the concept of the *joint spectral radius* gives a necessary and sufficient condition for the stability of switched linear systems. The joint spectral radius refers to the maximal growing rate which may be obtained using long products of matrices from a given set and is discussed in Section 4.6. When the admissible switching sequences are known, a necessary and sufficient condition can be obtained using quadratic Lyapunov functions.

Theorem 4.8 [393] *The discrete-time switched linear system is uniformly exponentially stable if and only if there exist an indexed family of symmetric positive definite matrices $\mathbf{X}_j \in \mathbb{R}^{n \times n}$ and $m > 0$ such that*

$$\mathbf{A}_{i_m}^T \mathbf{X}_{(i_0, \dots, i_M)} \mathbf{A}_{i_m} < \mathbf{X}_{(i_0, \dots, i_{M-1})}, \quad \forall (i_0, \dots, i_M) \in \mathcal{N}(m),$$

where $\mathcal{N}(m)$ is the set of admissible switching sequences of length m .

Necessary and sufficient conditions for the existence of multiple Lyapunov functions In contrast to the continuous-time case, it can be shown in discrete-time that there is equivalence between different kind of quadratic Lyapunov functions.

Definition 4.5 (Parameter-dependent quadratic stability) *We say that a switched linear discrete-time system is parameter-dependent quadratically stable (PD-quadratically stable) if there exist three positive constants α_0 , α_1 , α_2 and a Lyapunov function*

$$V(\mathbf{x}, q) = \mathbf{x}^T \mathbf{P}(q) \mathbf{x}, \quad (4.67)$$

such that

$$\alpha_1 \|\mathbf{x}\|^2 \leq V(\mathbf{x}, q) \leq \alpha_2 \|\mathbf{x}\|^2,$$

and whose difference along the system's solutions is negative definite decreasing, that is, for every $\mathbf{x}(0) \in \mathbb{R}^n$, every $\{q(k)\}_{k \in \mathcal{N}} \subset \mathcal{Q}$, and every $k \in \mathcal{N}$, we have

$$V(\mathbf{x}(k+1), q(k+1)) - V(\mathbf{x}(k), q(k)) \leq -\alpha_0 \|\mathbf{x}(k)\|^2.$$

Definition 4.6 (Parameter and time-dependent quadratic stability) *We say that a switched linear discrete-time system is parameter- and time-dependent quadratically stable (PTD-quadratically stable) if there exist three positive constants α_0 , α_1 , α_2 and a Lyapunov function*

$$V(k, \mathbf{x}, q) = \mathbf{x}^T \mathbf{P}(k, q) \mathbf{x}, \quad (4.68)$$

such that

$$\alpha_1 \|\mathbf{x}\|^2 \leq V(k, \mathbf{x}, q) \leq \alpha_2 \|\mathbf{x}\|^2, \quad (4.69)$$

and for every $\mathbf{x}(0) \in \mathbb{R}^n$, every $\{q(k)\}_{k \in \mathcal{N}} \subset \mathcal{Q}$, and every $k \in \mathcal{N}$, we have

$$V(k+1, \mathbf{x}(k+1), q(k+1)) - V(k, \mathbf{x}(k), q(k)) \leq -\alpha_0 \|\mathbf{x}(k)\|^2.$$

Definition 4.7 (Quadratic stability) We say that a switched linear discrete-time system is poly-quadratically stable if a Lyapunov function V can be found with $P(\cdot)$ linear with respect to q .

Theorem 4.9 [442] A switched linear discrete-time system is PTD-quadratically stable if and only if it is PD-quadratically stable if and only if it is poly-quadratically stable.

The previous theorem deserves some comments. The most classical and viable way of studying the uniform stability is to check for the existence of a quadratic Lyapunov function. It is known from the literature that letting the Lyapunov function depend on the time-varying dynamics improves the chance that a quadratic Lyapunov function exists. Theorem 4.9 proves that the dependence on the dynamics can be actually assumed to be linear, with no prejudice on the effectiveness of the method. Moreover, no gain in the sensibility is obtained if we allow the Lyapunov function to depend on the time as well. However, Lyapunov quadratic stability still is a strictly stronger notion than uniform asymptotic stability.

Tractable necessary and sufficient conditions of the existence of a multiple quadratic Lyapunov function can be obtained in discrete-time framework. These conditions ensure that the switched linear discrete-time system is PD-quadratically stable and are expressed in terms of linear matrix inequalities [193].

Theorem 4.10 The following statements are equivalent:

1. There exists a poly-quadratic Lyapunov function

$$V(k, \mathbf{x}(k), q(k)) = \mathbf{x}^T(k) \mathbf{P}_{q(k)} \mathbf{x}(k),$$

strictly decreasing along the system trajectories $\forall q \in \mathcal{Q}$.

2. There exist N symmetric matrices $\mathbf{P}_i = \mathbf{P}_i^T > 0, \forall i \in \mathcal{Q}$, satisfying the LMIs:

$$\begin{pmatrix} \mathbf{P}_i & \mathbf{A}_i^T \mathbf{P}_j \\ \mathbf{P}_j \mathbf{A}_i & \mathbf{P}_j \end{pmatrix} > 0, \forall (i, j) \in \mathcal{Q} \times \mathcal{Q}. \quad (4.70)$$

3. There exist N symmetric matrices $\mathbf{S}_i = \mathbf{S}_i^T > 0$, and N matrices $\mathbf{G}_i \forall i \in \mathcal{Q}$, satisfying the LMIs:

$$\begin{pmatrix} \mathbf{G}_i + \mathbf{G}_i^T - \mathbf{S}_i & \mathbf{G}_i^T \mathbf{A}_i^T \\ \mathbf{A}_i \mathbf{G}_i & \mathbf{S}_j \end{pmatrix} > 0, \forall (i, j) \in \mathcal{Q} \times \mathcal{Q}. \quad (4.71)$$

4.5 Stabilization and control

4.5.1 Stabilization problems

Three stabilization problems are discussed in this section:

Problem A: Design of a stabilizing switching law by state space restrictions, when the switching law is a control signal.

Problem B: Design of a stabilizing switching law by time domain restrictions, when it is possible to control only the dwell time of each mode, and the switching law is arbitrary.

Problem C: Design of a control law for arbitrary switching signals.

The search for a stabilizing switching law (Problems A and B) is motivated by the following question: what restrictions should we consider for the switching law in order to guarantee the stability of the switched system? The problem of stabilization of switched systems with arbitrary switchings signals is closely related to robust stabilization problems.

4.5.2 State-space restrictions

The design of a stabilizing switching law may be obtained by dividing the state space into several regions such that the obtained piecewise linear system is stable. The problem is trivial when at least one of the A_i matrices is Hurwitz stable: the stable system is always activated. A more delicate stabilization problem occurs when all the A_i , $i \in I$ matrices are unstable. A necessary condition for the stabilization by switching law has been proposed in [612]:

Theorem 4.11 *If there exists a stabilizing switching sequence, then there exists a subsystem $\dot{x}(t) = A_i x(t)$, $i \in Q$ such that at least one of the eigenvalues of $A_i + A_i^T$ is negative.*

Also, the results of [660] are based on the fact that the trajectory of any convex combination $\{A_1 x, A_2 x\}$ can be approximated by fast switchings among the two subsystems. The first result is given as follows:

Theorem 4.12 [230] *Consider a pair of unstable systems with the associated matrices $\{A_1, A_2\}$ ($M = 2$). If there exists a stable convex combination, i.e. if there exists a scalar $\alpha \in (0, 1)$ such that the matrix $A_{eq} = \alpha A_1 + (1 - \alpha) A_2$ has all of its eigenvalues in the left side of the complex plane, there exists a switching sequence such that the system $\dot{x}(t) = A_{q(t)} x(t)$, $q(t) \in \{1, 2\}$ is stable.*

There are classes of systems for which no stable convex combination exists, yet a stabilizing switching law may be obtained. Necessary and sufficient conditions exist

for the particular case of second order systems where the existence of a stabilizing switching law may be verified by analyzing the vectors fields [673]. Moreover, the results presented here are based on the existence of a stable convex combination A_{eq} . However, in general, finding a stable convex combination is a NP-hard problem [87, 591]. In some particular cases, the construction of the matrix A_{eq} is possible as shown in the following example.

Example 4.2 *Stabilization by switching*

We consider the continuous-time switched system

$$\frac{dx}{dt} = A_q x(t), \quad q \in \{1, 2\},$$

where

$$A_1 = \begin{pmatrix} 0.1 & 0.3 \\ 0.6 & -0.2 \end{pmatrix}, \quad A_2 = \begin{pmatrix} -0.13 & -0.16 \\ -0.33 & 0.03 \end{pmatrix}.$$

For this both A_1 and A_2 have positive eigenvalues, that is each subsystem is unstable. The phase portraits of each subsystem are given in Fig. 4.9.

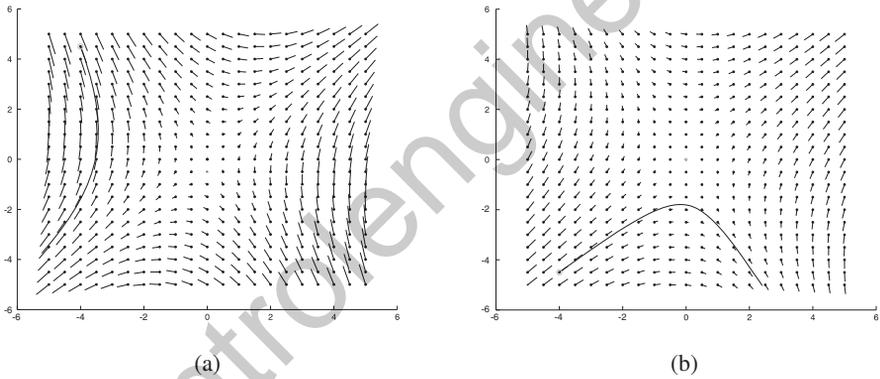


Fig. 4.9 Phase portraits for each subsystem and examples of evolution from arbitrary initial positions. (a) $q = 1$, (b) $q = 2$.

We can remark that both $A_1^T + A_1$ and $A_2^T + A_2$ have a negative eigenvalue, therefore the necessary stabilizability condition is verified.

In order to find a stabilizing switching sequence we look for a stable convex combination $A_{\text{eq}}(\alpha) = \alpha A_1 + (1 - \alpha) A_2$, $\alpha \in (0, 1)$. The $A_{\text{eq}}(\alpha)$ matrix can be found by using a tight gridding of the domain of variation of α and by verifying the eigenvalues of $A_{\text{eq}}(\alpha)$ for all the values of the grid. Using a discretization step of 0.01 we can notice that $A_{\text{eq}}(\alpha)$ is stable for $\alpha \in [0.29, 0.42]$. For $\alpha = 0.35$ we obtain

$$A_{\text{eq}} = \begin{pmatrix} -0.052 & -0.003 \\ -0.006 & -0.05 \end{pmatrix}.$$

for which the positive definite matrix

$$P = \begin{pmatrix} 2.3706 & 0.0079 \\ 0.0079 & 2.3652 \end{pmatrix}$$

verifies the linear matrix inequality

$$A_{\text{eq}}^T P + P A_{\text{eq}} < 0.$$

This condition can be expressed as :

$$\alpha(A_1^T P + P A_1) + (1 - \alpha)(A_2^T P + P A_2) < 0,$$

which is the same as

$$\alpha \cdot x^T (A_1^T P + P A_1) x + (1 - \alpha) \cdot x^T (A_2^T P + P A_2) x < 0,$$

$\forall x \in \mathbb{R}^n \setminus \{0\}$. The terms $x^T (A_1^T P + P A_1) x$ and $x^T (A_2^T P + P A_2) x$ are weighted by positive coefficients ($\alpha \in (0, 1)$). This means that, for any value of the state vector, at least one of the terms should be negative, i.e. $x^T (A_1^T P + P A_1) x < 0$ or $x^T (A_2^T P + P A_2) x < 0$. This implies that the state space is covered by the conic regions:

$$\mathcal{X}_i = \left\{ x \in \mathbb{R}^n : x^T (A_i^T P + P A_i) x < 0 \right\}, \quad i = 1, 2.$$

Namely,

$$\mathcal{X}_1 = \left\{ x \in \mathbb{R}^n : x^T \begin{pmatrix} 0.48 & 2.13 \\ 2.13 & -0.94 \end{pmatrix} x < 0 \right\}$$

and

$$\mathcal{X}_2 = \left\{ x \in \mathbb{R}^n : x^T \begin{pmatrix} -0.64 & -1.18 \\ -1.18 & 0.15 \end{pmatrix} x < 0 \right\},$$

such that $V(x) := x^T P x$ is strictly decreasing in the region \mathcal{X}_1 for any solution of $\dot{x} = A_1 x$ and in the region \mathcal{X}_2 for the solutions of $\dot{x} = A_2 x$. Then the system is stabilized using the switching law

$$q(t) = \arg \min_{i \in \{1, 2\}} (A_i^T P + P A_i).$$

(See Fig. 4.10) \square

4.5.3 Time domain restrictions

The concept of *dwell time* is of first importance for the switched system stabilization. The dwell time represents the time interval between two instances of switch.

The basic idea is very simple. Consider that all the subsystems $\dot{x} = A_q x$, with $q \in \mathcal{Q}$, are asymptotically stable. It is natural to think that the switched system is exponentially stable provided that the dwell time is sufficiently large to allow each subsystem to reach the steady state. The problem is to compute the minimum time τ_D between two successive switch instants that ensures the stability of the switched system [475, 683]. Let $\Phi_q(t, \tau)$ denote the fundamental matrix of the q -th subsystem $\dot{x} = A_q x$, $q \in \mathcal{Q}$. Since all the subsystems are asymptotically stable, one can find two scalars $\mu > 0$ and $\lambda_0 > 0$ such that

$$\|\Phi_q(t, \tau)\| \leq \mu e^{-\lambda_0(t-\tau)}, \quad t \geq \tau \geq 0, \forall q \in \mathcal{Q}.$$

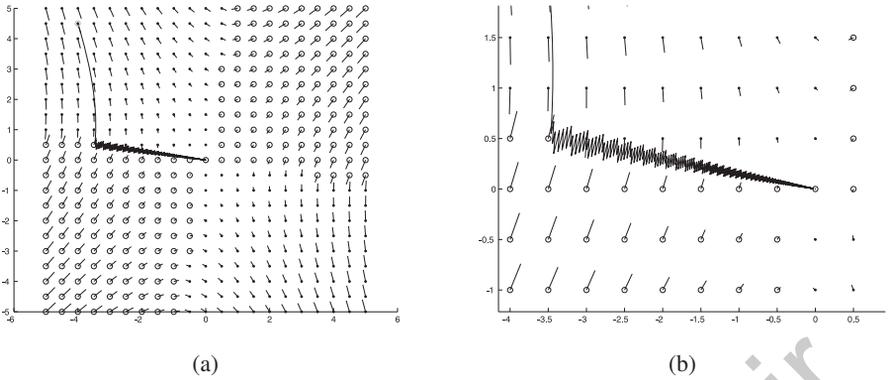


Fig. 4.10 (a) Phase portraits for the obtained piecewise linear system and an example of evolution from an arbitrary initial position. The points in the state space where $q = 1, 2$ are marked by a dot and a circle, respectively. (b) Zoom on the sliding mode from (a).

The scalar λ_0 may be interpreted as a common decay rate for the family of subsystems. Actually the scalars μ and λ_0 may be computed, $\mu \triangleq \max_{q \in \mathcal{Q}} \mu_q$ and $\lambda_0 \triangleq \max_{q \in \mathcal{Q}} \lambda_q$, where μ_q and λ_q are the constants that define the convergence of each subsystem $\dot{x} = A_q x$, $\forall q \in \mathcal{Q}$. Consider t_1, t_2, \dots, t_k , the switching instants in the time interval (τ, t) , such that $t_i - t_{i-1} \geq \tau_D$. Then the value of the state vector at a given instant of time t is given by

$$x(t) = \Phi_{q(t_k)}(t, t_k) \Phi_{q(t_{k-1})}(t_k, t_{k-1}) \cdots \Phi_{q(t_1)}(t_2, t_1) \Phi_{q(t_1)}(t_1, \tau) x(\tau).$$

The transition matrices between two successive switching times satisfy the relation:

$$\|\Phi_{q(t_{l-1})}(t_l, t_{l-1})\| \leq \mu e^{-\lambda_0(t_l - t_{l-1})} \leq \mu e^{-\lambda_0 \tau_D}, \quad \forall l \in \{2, 3, \dots, k\}.$$

Then the system is asymptotically stable if $\mu e^{-\lambda_0 \tau_D} \leq 1$. This condition may be satisfied for

$$\tau_D \geq \frac{\log \mu}{\lambda_0 - \lambda} \quad (4.72)$$

$\forall \lambda \in (0, \lambda_0)$.

Theorem 4.13 [475, 683] Consider the switched linear system

$$\dot{x} = A_q x, \quad q \in \mathcal{S}(\mathcal{Q}, \tau_D),$$

where all the subsystems $\dot{x} = A_i x$, $\forall i \in \mathcal{Q}$ are asymptotically stable with the stability margin λ_0 . Here $\mathcal{S}(\mathcal{Q}, \tau_D)$ denotes the set of switching sequences for which the dwell time is greater than τ_D . For any scalar $\lambda \in (0, \lambda_0)$, the system is stable with the stability margin λ if the minimum dwell time τ_D satisfies the condition (4.72).

A more general result is given in [318] with the concept of *average dwell time*, τ_{moy} . The idea is that the system is asymptotically stable if in average the switching intervals are less than τ_{moy} . This allow the system, occasionally, to switch “faster” than the rate corresponding to the average dwell time τ_{moy} .

4.5.4 Control design for arbitrary switching

When the switching signal is arbitrary, the switched system stabilization problem is closely related to the stabilization of uncertain systems if the switching signal is unknown. When the switching signal is arbitrary but available in real time, the problem is related to the well-known gain scheduling problem studied in the context of a linear system with time-varying parameters (LPV).

Consider the following polytopic system :

$$\dot{\mathbf{x}}(t) = \mathbf{A}(\lambda(t))\mathbf{x}(t) + \mathbf{B}(\lambda(t))\mathbf{u}(t), \quad (4.73)$$

where

$$\mathbf{A}(\lambda(t)) = \sum_{i=1}^N \lambda_i(t)\mathbf{A}_i, \quad \mathbf{B}(\lambda(t)) = \sum_{i=1}^N \lambda_i(t)\mathbf{B}_i$$

and

$$\lambda(t) \in \Lambda = \left\{ \lambda = [\lambda_1 \dots \lambda_N] \in \mathbb{R}^N : \lambda_i \geq 0, \sum_{i=1}^N \lambda_i = 1 \right\}.$$

The switched linear system

$$\dot{\mathbf{x}}(t) = \mathbf{A}_{q(t)}\mathbf{x}(t) + \mathbf{B}_{q(t)}\mathbf{u}(t), \quad q(t) \in \mathcal{Q} \quad (4.74)$$

can be expressed as a particular case of a polytopic system with λ_i parameters restricted to two discrete values $\lambda_i \in \{0, 1\}$, such that $\lambda_i = 1$ when $q(t) = i$. There exist a large variety of approaches for the polytopic system stabilization. For example, when the switching law is not known, a design method for a LMI static state feedback has been proposed in [109]. The result is based on the existence of a common quadratic Lyapunov function for the closed-loop system. It requires the existence of a symmetric positive definite matrix S and of a matrix Y that verify the LMI

$$S\mathbf{A}_i^T + \mathbf{A}_i S + \mathbf{B}_i Y + Y^T \mathbf{B}_i < 0, \quad \forall i \in \mathcal{Q}.$$

In this case, the common quadratic Lyapunov is given with the Lyapunov matrix $P = S^{-1}$. The state feedback is described by

$$\mathbf{u}(t) = \mathbf{K}\mathbf{x}(t),$$

with $\mathbf{K} = Y S^{-1}$.

If the switching signal is available in real time then we can look for a *switched state feedback*:

$$\mathbf{u}(t) = \mathbf{K}_{q(t)}\mathbf{x}(t). \quad (4.75)$$

The solution depends on the existence of several matrices Y_i and the previous LMI is replaced by

$$SA_i^T + A_iS + B_iY_i + Y_i^T B_i < 0, \forall i \in Q. \quad (4.76)$$

The state feedback is given by the equation (4.75) with $K_i = Y_i S^{-1}$, $\forall i \in Q$.

More efficient stabilization criteria can be derived when a priori knowledge about the switching function is used. A notable result in this context is the work of Johansson [342] for switched systems with state space dependent switching laws. The author uses multiple Lyapunov-like functions in order to obtain a switched state feedback for piecewise linear systems. The controller is derived via optimal control methods.

4.5.5 Stabilization of discrete-time switched linear systems

As mentioned for stability analysis, many approaches followed for stabilization in continuous time can be extended in the discrete-time case. The one related to the dwell time notion is particularly interesting since it leads in the case of discrete-time switched linear systems to an interesting result.

Theorem 4.14 [262] Consider the system $\mathbf{x}(k+1) = A_{q(k)}\mathbf{x}(k)$, $q(k) \in Q$. For a given $\tau_D \geq 1$ if there exists a set of positive definite matrices $\{P_1, \dots, P_Q\}$ such that

$$A_q^T P_q A_q - P_q < 0, \forall q \in Q$$

and

$$A_q^{T\tau_D} P_{q'} A_q^{\tau_D} - P_{q'} < 0, \forall q, q' \in Q, q \neq q'$$

then the system is stable for a dwell time greater than τ_D .

In this theorem, posing $\tau_D = 1$, one recovers the LMI stability condition proposed in [194]. Note that in the continuous-time framework, multiple Lyapunov function cannot be used for arbitrary switching laws. The main difference with the discrete-time case states in the fact that arbitrary switching means switching with a minimum dwell time $\tau_D = 1$ and that's why multiple Lyapunov functions can be used in discrete time. In this case, the design of switched control laws reduces to the design of feedback gains such that the closed-loop switched system is uniformly exponentially stable.

Theorem 4.15 [194] The switched system

$$\mathbf{x}(k+1) = A_q\mathbf{x}(k) + B_q\mathbf{u}(k), \forall q \in Q$$

can be stabilized using the switched state feedback $\mathbf{u} = \mathbf{K}_q \mathbf{x}$ if there exist positive definite matrices \mathbf{S}_i , \mathbf{R}_i , and \mathbf{G}_i , $\forall i \in \mathcal{Q}$, such that the LMIs

$$\begin{pmatrix} \mathbf{G}_q + \mathbf{G}_q^\top - \mathbf{S}_q & (\mathbf{A}_q^\top \mathbf{G}_q + \mathbf{B}_q \mathbf{R}_q)^\top \\ \mathbf{A}_q^\top \mathbf{G}_q + \mathbf{B}_q \mathbf{R}_q & \mathbf{S}_{q'} \end{pmatrix} > 0, \forall (q, q') \in \mathcal{Q} \times \mathcal{Q} \quad (4.77)$$

are satisfied $\forall q, q' \in \mathcal{Q}$. The state feedback is given with $\mathbf{K}_q = \mathbf{R}_q \mathbf{G}_q^{-1}$ and the Lyapunov matrices are $\mathbf{P}_q = \mathbf{S}_q^{-1}$.

4.6 Complexity issues

This section gives a survey of computational complexity results for switched and piecewise affine systems. There is a large literature on these topics and we will not try to be exhaustive. In particular, we have chosen to concentrate our attention to discrete-time systems (as opposed to continuous-time systems), and to analysis problems rather than design problems. System design is computationally more difficult than system analysis and so the computational complexity results that we describe for system analysis have natural counterparts for system design. For more extensive lists of references and a more detailed survey (including continuous-time systems and system design) we refer the reader to the survey papers [91] and [586].

4.6.1 Decidability and NP-hardness

Some of the problems that we describe are *algorithmically undecidable* (or, simply, *undecidable*), others are *NP-hard*. In this subsection, we briefly describe the meaning of these two terms. Interested readers are referred to [506] for more details.

A problem is algorithmically undecidable if there exists no algorithm that always halts with the correct answer for all instances of the problem. One may wonder whether this definition depends on the choice of the computation model. According to the Church–Turing thesis, all reasonable models of digital computation lead to the same class of algorithmically decidable problems. This thesis is supported by the fact that all reasonable models that have been proposed and studied lead indeed to the same class of decidable problems. Proving a problem to be algorithmically undecidable is thus a strong result. It implies in particular that the problem cannot be solved with, say, a *MATLAB* routine or a *C++* program that have access to unbounded memory and have finite but unlimited computation time.

There are many decidable problems of practical interest for which no polynomial-time algorithm is known, despite intensive research efforts. Many of these problems belong to a class known as NP (Nondeterministic Polynomial time), which includes the class P of problems for which polynomial-time algorithms are possible. A decision problem is said to belong to NP if every positive instance to the problem has a “certificate” of being a positive instance whose validity can be verified

with a polynomial amount of computation. Some problems in NP turn out to be a hardest problems within the class NP in the sense that every problem in NP can be reduced to them in polynomial time. Such problems are said to be *NP-complete*. If we had a polynomial-time algorithm for some NP-complete problem, then all problems in NP could also be solved in polynomial time and it would then follow that NP is the same as the class P. Whether this is the case or not is a major open problem in theoretical computer science. It is widely believed that $P \neq NP$ and if this is indeed true, then none of the NP-complete problems are polynomial-time solvable.

By now, NP-completeness has been established for thousands of problems. If a problem A is at least as hard as some NP-complete problem B we say that A is *NP-hard*. NP-hardness of a problem therefore means that it is about as difficult as any NP-complete problem, and this is often interpreted as an indication of inherent intractability. Assuming that the conjecture $P \neq NP$ is true, an NP-hardness result eliminates the possibility of algorithms that run in polynomial time and that are correct for all problem instances.

4.6.2 Switched systems and the joint spectral radius

A discrete-time switching linear system generates trajectories of points

$$x(k+1) = A(k)x(k), \quad x_0 \in \mathbb{R}^n, \quad (4.78)$$

with the matrices $A(k)$ taken in some *uncertainty set* $\mathcal{A} \subset \mathbb{R}^{n \times n}$. These systems can also be thought of as discrete-time linear inclusion

$$x(k+1) \in \mathcal{A}x(k), \quad x_0 \in \mathbb{R}^n.$$

The worst-case growth rate of the trajectories over all possible switching sequences can be characterized by a *joint spectral radius*.

The *joint spectral radius* of the set of matrices \mathcal{A} is the smallest value $\rho \geq 0$ such that for every trajectory there is some constant C for which

$$\|x(k)\| \leq C\rho^k,$$

for all k . This optimal ρ provides valuable information about the stability of switching linear system.

|| The trajectories of the system all converge to the origin if and only if $\rho < 1$.

The joint spectral radius can thus be associated with the stability properties of time-varying linear systems in the worst case over all possible time variations. The joint spectral radius can also be defined as the measure of the maximal asymptotic growth rate that can be obtained by forming long products of matrices. The stability condition $\rho < 1$ is thus equivalent to the condition that all infinite products of matrices taken in \mathcal{A} converge to zero.

The joint spectral radius can also be defined with matrix norms. Let $\|\cdot\|$ be a matrix norm. The *spectral radius* of the matrix A is defined by

$$\rho(A) = \lim_{k \rightarrow +\infty} \|A^k\|^{1/k}. \quad (4.79)$$

The spectral radius of a matrix does not depend on the chosen matrix norm and it is also equal to $\rho(A) = \max\{|\lambda| : \lambda \text{ is an eigenvalue of } A\}$. The definition (4.79) easily extends to *sets* of matrices:

$$\rho(\mathcal{M}) = \limsup_{k \rightarrow \infty} \max\{\|A\|^{1/k} : A \text{ is a product of length } k \text{ of matrices in } \mathcal{A}\}. \quad (4.80)$$

A definition analogous to the one given in (4.80) is possible by replacing the norm appearing there by a spectral radius. The quantity defined in this way is the *generalized spectral radius* introduced in [200]. In [77], the joint and generalized spectral radii of finite (or bounded) sets of matrices are proved to be equal (see also [653] for an elementary proof), which reinforces the status of the joint spectral radius as a legitimate generalization of the spectral radius of a single matrix.

Computation of the joint spectral radius Questions related to the computability of the joint spectral radius and to the existence of efficient approximation algorithms have been posed more than a decade ago [377, 634]. Despite the interest generated by these questions, there are only few matrix sets for which exact computation algorithms are available. The list includes the case where \mathcal{A} contains only symmetric matrices, only triangular matrices of identical orientation, two symmetric matrices $\mathcal{A} = \{A, A^T\}$ (in this case the joint spectral radius is given by the singular value of A), or nonnegative matrices that have row or column uncertainties.

As it has clearly emerged over the last decade, the joint spectral radius of general sets of matrices is quite difficult to compute and to approximate. In fact, even for the case of two matrices of dimension 47×47 that have nonnegative rational entries, the problem of checking the inequality $\rho \leq 1$ is algorithmically undecidable [88, 90].

|| It is still unknown whether the problem $\rho < 1$ is algorithmically decidable.

Thus, the stability problem of switched systems is very complex and it is still not known whether one can find an algorithm that checks whether all infinite products of matrices taken from a given set converge to zero. On the other hand, the related problem of determining whether the set of all matrix products is *bounded* is known to be undecidable. From a practical viewpoint, this result means that the structure of the matrices occurring in an application has to be utilized to find a stability test that can be accomplished with finite effort. Another way of stability analysis is to use approximations, which, however, yield either necessary or sufficient stability conditions, as will be shown below.

The problem of finding an algorithm to decide $\rho < 1$ is related to the so-called finiteness property for sets of matrices. A set of matrices is said to have the *finiteness*

property if its joint spectral radius is given by an average spectral radius of some finite product of matrices in the set. Stability is decidable for all sets of matrices that satisfy the finiteness property and this property was conjectured to hold for all sets of matrices in [377]. This conjecture has since then been proved to be false (see [108] and [94]), but the counterexamples that have been provided so far are all non-constructive. In particular, it is yet unknown if a counterexample is possible with matrices that have only rational entries. In the recent contribution [351] it is shown that the finiteness property holds for all sets of matrices with nonnegative entries if and only if it holds for all sets of matrices with *binary* entries.

Approximations of the joint spectral radius In recent years most research efforts have concentrated on finding reasonable approximations for the joint spectral radius (see, e.g., [89, 509, 544]). In principle, the joint spectral radius can be approximated to any desired accuracy by computing converging sequences bounds. The following upper and lower bounds, proved in [377],

$$\max_{q \in \{1, \dots, Q\}^k} \rho(\mathbf{A}_q)^{1/k} \leq \rho(\mathbf{A}_1, \dots, \mathbf{A}_Q) \leq \max_{q \in \{1, \dots, Q\}^k} \|\mathbf{A}_q\|^{1/k} \quad (4.81)$$

can be evaluated for increasing values of k and lead to arbitrary accurate approximations of ρ (see, e.g., [199] or [282]).

However, approximations that are directly based on these inequalities are expensive to compute. In [433], the exponential number of products that appear in the direct and naive computation of the bounds in (4.81) is reduced by avoiding duplicate computation of cyclic permutations. The total number of products to consider remains exponential, however. Thus, approximations of arbitrary degree of accuracy can be computed, but at a price that may be prohibitive.

There are in fact intrinsic limitations for the rate at which the joint spectral radius can be approximated. Let us say that the value ξ approximates the value ρ with *relative accuracy* $\mu \in [0, 1]$ (or $100 \mu \%$), if $\mu \xi \leq \rho \leq \xi$. It is known that, unless $P=NP$, there is no algorithm that can compute the joint spectral radius of two matrices with relative accuracy $1 - \epsilon$ in time polynomial in the size of the matrices and in $1/\epsilon$.

|| For any *fixed* desired accuracy, there exists a polynomial-time algorithm that computes the joint spectral radius of the matrices with that accuracy.

It is proved in [89] that the joint spectral radius of m matrices of size n can be approximated with relative accuracy $1/\sqrt{m}$ by computing the spectral radius of a single matrix whose size is less than n^2 . This procedure can be applied in a recursive way and in general a relative accuracy of $(1/\sqrt{m})^{1/k}$ can be obtained by computing the spectral radius of a single matrix of size less than n^{2k} . As an illustration, the spectral radius of two matrices of size n can be computed with an accuracy of 70% by computing the spectral radius of a single matrix of size n^2 , and an accuracy of 95% can be obtained for the joint spectral radius of three matrices by computing the spectral radius of a single matrix of size n^{11} .

4.6.3 Piecewise affine systems

We start with a partition of \mathbb{R}^n into finitely many disjoint subsets $\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_m$. We assume that different affine systems are associated with each subset, and the overall system is described by

$$\mathbf{x}(k+1) = \mathbf{A}_i \mathbf{x}(k) + \mathbf{b}_i, \quad \text{if} \quad \mathbf{x}(k) \in \mathcal{C}_i. \quad (4.82)$$

When the subsets \mathcal{C}_i are defined in terms of a finite number of linear inequalities, the systems of the form (4.82) are the *piecewise affine* systems introduced in Section 4.1. When the vectors \mathbf{b}_i are equal to zero, the system is said to be piecewise linear.

Stability and boundedness The stability and boundedness analysis poses again unsolvable problems:

|| The problems of determining whether all trajectories of a given piecewise affine system go to the origin, or whether all trajectories remain bounded, are both undecidable [93].

Undecidability persists even if the state space has dimension 2. The proof provided in [93] makes use of 2-counter machines, a model of computation equivalent to Turing machines and relies on the fact that the problem of determining whether a given 2-counter machine halts for *every* initial configuration of the machine is undecidable.

Piecewise affine systems do not need to be continuous, the mapping $\mathbf{x}(k+1) = \mathbf{f}(\mathbf{x}(k))$ resulting from a piecewise affine system may exhibit discontinuities at the boundaries between partitions. If the function \mathbf{f} is piecewise affine but *continuous*, the problems related to stability are decidable in dimension 1, undecidable in dimension 3 [93], and open in dimension 2.

For piecewise affine systems there is a trade-off between the number of partitions and the dimension of the state space. Problems are already undecidable in state-space dimension 2 but what about the number of partitions? The systems (4.83) with two partitions and with linear updates are of the form

$$\mathbf{x}(k+1) = \begin{cases} \mathbf{A}_+ \mathbf{x}(k) & \text{when } \mathbf{c}^T \mathbf{x}(k) \geq 0, \\ \mathbf{A}_- \mathbf{x}(k) & \text{when } \mathbf{c}^T \mathbf{x}(k) < 0. \end{cases} \quad (4.83)$$

These are systems of the form $\mathbf{x}(k+1) = \mathbf{A}(k)\mathbf{x}(k)$ where each $\mathbf{A}(k)$ belongs to the set $\{\mathbf{A}_-, \mathbf{A}_+\}$. This is a situation similar to the one considered in the previous section, except that now the sequence of matrices is not arbitrary, but is determined by the state sequence. Still, this does not help in reducing complexity as we now argue using an example taken from [92].

Example 4.3 *Switched system with undecidable stability property*

Consider a system described by a state vector $(v(k), y(k), z(k))$, where $v(k)$ and $y(k)$ are scalars and $z(k)$ is a vector in R^n , and the dynamics is of the form

$$\begin{pmatrix} v(k+1) \\ y(k+1) \\ z(k+1) \end{pmatrix} = \begin{pmatrix} 1/4 & 0 & 0 \\ -1/4 & 1/2 & 0 \\ 0 & 0 & \mathbf{A}_+ \end{pmatrix} \begin{pmatrix} v(k) \\ y(k) \\ z(k) \end{pmatrix} \text{ if } y(k) \geq 0,$$

and

$$\begin{pmatrix} v(k+1) \\ y(k+1) \\ z(k+1) \end{pmatrix} = \begin{pmatrix} 1/4 & 0 & 0 \\ 1/4 & 1/2 & 0 \\ 0 & 0 & \mathbf{A}_- \end{pmatrix} \begin{pmatrix} v(k) \\ y(k) \\ z(k) \end{pmatrix} \text{ if } y(k) < 0.$$

This system consists of two linear systems, each of which is enabled in one of two half-spaces, as determined by the sign of $y(k)$. Given that y_0 can be any real number, it can be verified that the sequence $\text{sign}(y(k))$ is completely arbitrary, which then implies that the matrices \mathbf{A}_- and \mathbf{A}_+ can be multiplied in an arbitrary order. Clearly, all trajectories converge to zero if and only if $z(k)$ always converges to zero and so this problem is at least as hard as the corresponding question in which arbitrary products were allowed. If the latter problem is ever shown to be undecidable, the same will be automatically true for the problem on two partitions considered here. On the other hand, since the boundedness of all products of \mathbf{A}_- and \mathbf{A}_+ is undecidable, it follows that the uniform boundedness of the trajectories of the system (4.83), for all initial states in a bounded set, is also undecidable. \square

Turing machine simulation Let us fix a scalar function $\sigma : \mathbb{R} \mapsto \mathbb{R}$, and consider systems of the form

$$x(k+1) = \sigma(\mathbf{A}x(k)), \tag{4.84}$$

with the function σ applied componentwise. When the scalar function σ is piecewise affine, then the resulting dynamical system is also piecewise affine. It is easy to show that such systems with $\sigma(x) = \max(0, x)$, which we call saturated linear systems, are capable of simulating Turing machines [587, 588]. Such simulations are done by suitably encoding the transition rules of the Turing machine in the matrix \mathbf{A} , while the tape contents and the machine's internal state are encoded on some of the states of the saturated linear system. Thus,

|| as computational devices, linear saturated systems are as powerful as Turing machines.

From this it follows that the problem of determining if the trajectory emanating from a given state ever reaches the origin is undecidable.

By using a universal Turing machine,² one can in fact prove that, for saturated linear systems, the problem of determining if the trajectory emanating from a given

² A universal Turing machine is a Turing machine M that receives as input the description of another machine M' and a string s , and then simulates the operation of machine M' on input s .

state ever reaches the origin is undecidable for some *particular* matrix A . More concretely, there exists a particular integer matrix A (of size approximately equal to 1000×1000) for which there exists no algorithm that takes an initial state x_0 (with rational components) as input and decides whether the state will eventually hit the origin [587]. The problem of finding the smallest possible such matrix is a difficult open problem that is related to the smallest possible size of universal Turing machines [558].

From these results we deduce that piecewise linear systems can simulate Turing machines and that it is undecidable for piecewise linear systems to determine if the trajectory emanating from a given state ever reaches the origin. Using a different encoding it is possible to show that simulation of Turing machines is already possible by using piecewise affine systems in state dimension two [469], but not in dimension one [361]. Similar to saturated linear systems, one can use a universal Turing machine to prove that there exists a *particular* piecewise linear system in R^3 (with the state space being partitioned into fewer than 800 subsets) for which the problem of determining if the trajectory emanating from a given state ever reaches the origin is undecidable [361].

Control systems Let us now introduce a control variable and consider systems of the form

$$x(k+1) = A_i x(k) + B u(k), \quad \text{if} \quad x(k) \in C_i. \quad (4.85)$$

When $B = O$, these systems are equivalent to autonomous piecewise linear systems and so the problem of determining whether a given initial state can be driven to the origin is undecidable. This result is also obtained in [92], with a different proof technique that has the advantage that it can be used to derive sharper estimates of the decidability limit (see also [625] for a similar proof).

There is an obvious trade-off in piecewise linear systems between the state space dimension n and the number of subsets m . When there is only one subset, or when the state dimension is equal to one, most properties are easy to check. On the other hand, the problem of determining whether a given initial state can be driven to the origin is undecidable as soon as $nm \geq 44$ (with n the state dimension and m the number of partitions). In particular, this problem is undecidable for piecewise linear systems of state dimension 22 and with as few as two subsets.

Bibliographical notes

A survey of different switched system classes and the associated problems is given in [206, 401, 611] and [586]. More details on differential inclusions can be found in the references [25, 109, 205, 403, 437, 461]. The phenomena occurring in sampled-data switched systems are illustrated in [323].

Identification of switched systems The identification methods considered here have been successfully exploited in several real applications, such as the identification of the electronic

component placement process in pick-and-place machines [70, 346, 348], the modeling of a current transformer [236], traction control [106], and motion segmentation in computer vision [652]. However, we highlight that other effective techniques for the identification of hybrid models both in input/output and state-space forms are available and defer the interested reader to the paper [505] for a tutorial. Generalizations of the bounded-error procedure to the case of multi-input multi-output models are discussed in [70, 503]. A software implementation of the bounded-error procedure is also available [504].

Observability and state observation Observability has been extensively studied both in the continuous and in the discrete domains and various researchers investigated observability of hybrid systems. In particular, Sontag in [597] defined a number of observability concepts and analyzed their relations for polynomial systems. More recently, incremental observability was introduced in [65] for the class of piecewise affine systems. Incremental observability implies that different initial states always give different outputs independently of the applied input. A characterization of observability and the definition of a hybrid observer for the class of autonomous piecewise affine systems can be found in [183]. In [214] the observability of autonomous hybrid systems was analyzed by using abstraction techniques. In [36], the notion of generic final-state determinability proposed in [597] was extended to hybrid systems and sufficient conditions were given for linear hybrid systems. The work in [649] considered autonomously switched systems and proposed a definition of observability based on the concept of indistinguishability of continuous initial states and discrete state evolutions from the outputs in free evolution. In [31, 567] the observability of switched systems (with control) was investigated. Critical observability for safety-critical switched systems was introduced in [568], where a set of "critical" states must be reconstructed immediately since they correspond to hazards that may yield catastrophic events.

The problem of observer design was studied, for example, in [65]. For the stochastic case, [98] investigates the stability and performance of switched Kalman filters.

Stability The joint spectral radius has proved useful not only for switched and hybrid systems, but also for a number of other applications, including "asynchronous" [635] or "desynchronised" [365] systems, wavelets [181], iterated function systems, capacity of codes [95, 460], numerical solutions to ordinary differential equations [289], sensor networks [352], combinatorics on words, autoregressive models, Markov chains, etc.

The stability analysis of switched systems is mainly based on the use of common Lyapunov functions as presented, for example, in [360, 585, 679] and of multiple Lyapunov functions [461, 491, 670]. The discrete-time version of such results is given in [393]. Input-to-state stability is likewise considered in terms of Lyapunov functions in, for example, [320].

controlengineers.ir

Further switched systems

A. Bemporad, M. K. Çamlıbel, W. P. M. H. Heemels, A. J. van der Schaft, J. M. Schumacher, and B. De Schutter

Mixed logical dynamical systems and linear complementarity systems are representations of switched systems, which under the conditions described here are equivalent to the model used in Chapter 4. They are particularly useful for model-predictive control. The equivalences of several hybrid system models show that different models, which are suitable for specific analysis and design problems and have been investigated in detail, cover the same class of hybrid systems. The analysis of the well-posedness of the models leads to conditions on the model equations under which a unique solution exists.

Chapter contents

5.1	Model-predictive control of hybrid systems	page 141
5.1.1	Linear model-predictive control	141
5.1.2	Discrete hybrid automata	143
5.1.3	Mixed logical dynamical systems	145
5.1.4	Hybrid model-predictive control	148
5.2	Complementarity systems	151
5.2.1	Modeling aim	151
5.2.2	Definition	152
5.2.3	Examples	153
5.2.4	Preliminaries	159
5.2.5	Existence and uniqueness of solutions	161
5.3	Equivalence of piecewise affine systems, mixed logical dynamical systems, and linear complementarity systems	167
5.3.1	Summary of the five classes of hybrid models	167
5.3.2	Systems equivalence	169

5.4	Solution concepts and well-posedness	173
5.4.1	Problem statement	173
5.4.2	Model classes	174
5.4.3	Solution concepts	175
5.4.4	Well-posedness notions	178
5.4.5	Comparison of some solution concepts	189
5.4.6	Zenoness	191

controlengineers.ir

5.1 Model-predictive control of hybrid systems

Model-predictive control (MPC) is a widely used technology in industry for control design of highly complex multivariable processes. The idea behind MPC is to start with a model of the open-loop process that explains the dynamical relations among system's variables (command inputs, internal states, and measured outputs). Then, constraint specifications on system variables are added, such as input limitations (typically due to actuator saturation) and desired ranges where states and outputs should remain. Desired performance specifications complete the control problem setup and are expressed through different weights on tracking errors and actuator efforts (as in classical linear quadratic regulation). At each sampling time, an open-loop optimal control problem based on the given model, constraints, weights, and with initial condition set at the current (measured or estimated) state, is repeatedly solved through numerical optimization. The result of the optimization is an optimal sequence of future control moves. Only the first sample of such a sequence is actually applied to the process; the remaining moves are discarded. At the next time step, a new optimal control problem based on new measurements is solved over a shifted prediction horizon.

After quickly reviewing the basics of MPC based on linear models, in this section we introduce two hybrid model classes useful for MPC design, discrete hybrid automata (DHA) and mixed logical dynamical systems, and review the main ideas of hybrid MPC.

This section is based on the paper [58] for reviewing the basics of model-predictive control (MPC), and on [632] for DHA and MLD models used in MPC of hybrid systems.

5.1.1 Linear model-predictive control

The simplest MPC algorithm is based on the linear discrete-time prediction model

$$\mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k) \quad (5.1)$$

of the open-loop process, where $\mathbf{x}(k) \in \mathbb{R}^n$ is the state vector at time k , and $\mathbf{u}(k) \in \mathbb{R}^m$ is the vector of manipulated variables to be determined by the controller, and on the solution of the finite-time-optimal control problem

$$\min_U \mathbf{x}_N^T \mathbf{P} \mathbf{x}_N + \sum_{k=0}^{N-1} \mathbf{x}^T(k) \mathbf{Q} \mathbf{x}(k) + \mathbf{u}^T(k) \mathbf{R} \mathbf{u}(k), \quad (5.2a)$$

$$\text{s.t. } \mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k), \quad k = 0, \dots, N-1, \quad (5.2b)$$

$$\mathbf{x}_0 = \mathbf{x}(k), \quad (5.2c)$$

$$\mathbf{u}_{\min} \leq \mathbf{u}(k) \leq \mathbf{u}_{\max}, \quad k = 0, \dots, N-1, \quad (5.2d)$$

$$\mathbf{y}_{\min} \leq \mathbf{C}\mathbf{x}(k) \leq \mathbf{y}_{\max}, \quad k = 1, \dots, N, \quad (5.2e)$$

where N is the prediction horizon, $\mathbf{U} \triangleq [\mathbf{u}^T(0) \dots \mathbf{u}^T(N-1)]^T \in \mathbb{R}^{Nm}$ is the sequence of manipulated variables to be optimized, $\mathbf{Q} = \mathbf{Q}^T \geq 0$, $\mathbf{R} = \mathbf{R}^T > 0$, and

$P = P^T \geq 0$ are weight matrices of appropriate dimensions defining the performance index, $\mathbf{u}_{\min}, \mathbf{u}_{\max} \in \mathbb{R}^m$, $\mathbf{y}_{\min}, \mathbf{y}_{\max} \in \mathbb{R}^p$, $\mathbf{C} \in \mathbb{R}^{p \times n}$ define constraints on input and state variables, respectively, and “ \leq ” denotes component-wise inequalities. By substituting $\mathbf{x}(k) = \mathbf{A}^k \mathbf{x}(0) + \sum_{j=0}^{k-1} \mathbf{A}^j \mathbf{B} \mathbf{u}(k-1-j)$, (5.2) can be recast as the quadratic programming (QP) problem

$$\mathbf{U}^*(\mathbf{x}(k)) \triangleq \arg \min_{\mathbf{U}} \frac{1}{2} \mathbf{U}^T \mathbf{H} \mathbf{U} + \mathbf{x}^T(k) \mathbf{C}^T \mathbf{U} + \frac{1}{2} \mathbf{x}^T(k) \mathbf{Y} \mathbf{x}(k), \quad (5.3a)$$

$$\text{s.t. } \mathbf{G} \mathbf{U} \leq \mathbf{W} + \mathbf{S} \mathbf{x}(k), \quad (5.3b)$$

where $\mathbf{U}^*(\mathbf{x}(k)) = [\mathbf{u}^{T^*(0)}(\mathbf{x}(k)) \dots \mathbf{u}^{T^*(N-1)}(\mathbf{x}(k))]^T$ is the optimal solution, $\mathbf{H} = \mathbf{H}^T > 0$ and $\mathbf{C}, \mathbf{Y}, \mathbf{G}, \mathbf{W}, \mathbf{S}$ are matrices of appropriate dimensions [57, 67, 69]. Note that \mathbf{Y} is not needed to compute $\mathbf{U}^*(\mathbf{x}(k))$, as it only affects the optimal value of (5.3a).

The MPC control algorithm is based on the following iterations: at time k , measure or estimate the current state $\mathbf{x}(k)$, solve the QP problem (5.3) to get the optimal sequence of future input moves $\mathbf{U}^*(\mathbf{x}(k))$, apply

$$\mathbf{u}(k) = \mathbf{u}_0^*(\mathbf{x}(k)) \quad (5.4)$$

to the process, discard the remaining optimal moves, and repeat the procedure again at time $k+1$.

In the absence of constraints (5.2d)–(5.2e), for $N \rightarrow \infty$ (or, equivalently, for $N < \infty$ and by choosing \mathbf{P} as the solution of the algebraic Riccati equation associated with matrices (\mathbf{A}, \mathbf{B}) and weights (\mathbf{Q}, \mathbf{R})), the MPC control law (5.3)–(5.4) coincides with the linear quadratic regulator (LQR) [67]. From a design viewpoint, the MPC setup (5.2) can therefore be thought of as a way of bringing the LQR methodology to systems with constraints.

The basic MPC setup (5.2) can be extended in many ways. In particular in tracking problems usually one has to make a certain output vector $\mathbf{y}(k) = \mathbf{C} \mathbf{x}(k) \in \mathbb{R}^p$ track a reference signal $\mathbf{r}(k) \in \mathbb{R}^p$ under constraints (5.2d)–(5.2e). In order to do so, the cost function (5.2a) is replaced by

$$\sum_{k=0}^{N-1} (\mathbf{y}(k) - \mathbf{r}(k))^T \mathbf{Q}_y (\mathbf{y}(k) - \mathbf{r}(k)) + \Delta \mathbf{u}^T(k) \mathbf{R} \Delta \mathbf{u}(k), \quad (5.5)$$

where $\mathbf{Q}_y = \mathbf{Q}_y^T \geq 0 \in \mathbb{R}^{p \times p}$ is a matrix of output weights, and the increments of command variables $\Delta \mathbf{u}(k) \triangleq \mathbf{u}(k) - \mathbf{u}(k-1)$ are the new optimization variables, possibly further constrained by $\Delta \mathbf{u}_{\min} \leq \Delta \mathbf{u}(k) \leq \Delta \mathbf{u}_{\max}$. In the above tracking setup vector $[\mathbf{x}^T(k) \mathbf{r}^T(k) \mathbf{u}^T(k-1)]^T$ replaces $\mathbf{x}(k)$ in (5.3b) and the control law (5.4) becomes $\mathbf{u}(k) = \mathbf{u}(k-1) + \Delta \mathbf{u}_0^*(\mathbf{x}(k), \mathbf{r}(k), \mathbf{u}(k-1))$.

The standard way of computing the linear MPC control action, which is implemented in most commercial MPC packages, is to solve the QP problem (5.3) on-line at each time k (for example in the *MPC Toolbox for MATLAB* [69]).

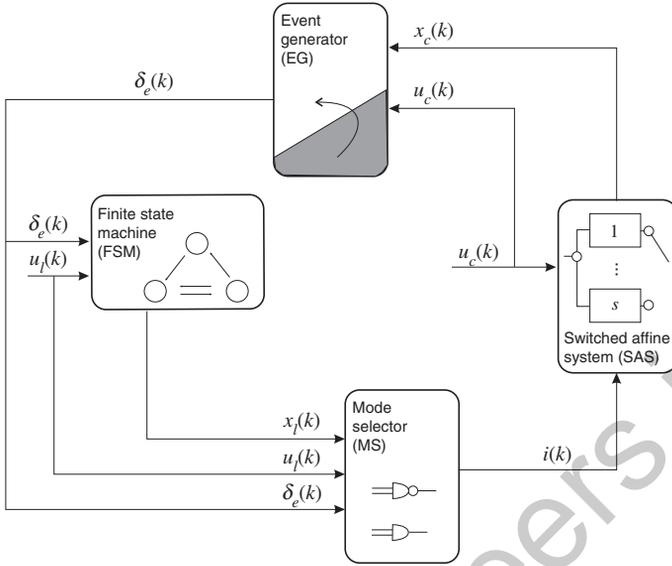


Fig. 5.1 Discrete hybrid automaton (DHA) as the connection of a finite-state machine (FSM) and a switched affine system (SAS), through a mode selector (MS) and an event generator (EG). The output signals are omitted for clarity.

Besides MPC schemes based on linear prediction models, several formulations of MPC based on general smooth nonlinear prediction models (as well as on uncertain linear models) exist. Most of them rely on nonlinear optimization methods for generic nonlinear functions/constraints to compute the control actions, and are therefore more rarely deployed in practical applications.

MPC based on *hybrid* dynamical models has emerged as a very promising approach to handle switching linear dynamics, on/off inputs, logic states, as well as logic constraints on input and state variables [62]. Here below we review a modeling framework for hybrid systems that is tailored to the synthesis of MPC controllers.

5.1.2 Discrete hybrid automata

Discrete hybrid automata (DHA) [632] are the interconnection of a finite-state machine and a switched linear dynamical system through a mode selector and an event generator (Fig. 5.1).

In the following we will use the fact that any discrete variable $\alpha \in \{\alpha_1, \dots, \alpha_j\}$, admits a Boolean encoding $a \in \{0, 1\}^{d(j)}$, where $d(j)$ is the number of bits used to represent $\alpha_1, \dots, \alpha_j$. From now on we will refer to either the variable or its encoding with the same name.

Switched affine system (SAS) A switched affine system is a collection of linear affine systems:

$$\mathbf{x}_c(k+1) = \mathbf{A}_{i(k)}\mathbf{x}_c(k) + \mathbf{B}_{i(k)}\mathbf{u}_c(k) + \mathbf{f}_{i(k)}, \quad (5.6a)$$

$$\mathbf{y}_c(k) = \mathbf{C}_{i(k)}\mathbf{x}_c(k) + \mathbf{D}_{i(k)}\mathbf{u}_c(k) + \mathbf{g}_{i(k)}, \quad (5.6b)$$

where $k \in \mathbb{Z}^+$ is the time indicator, $\mathbf{x}_c \in \mathcal{X}_c \subseteq \mathbb{R}^{n_c}$ is the continuous state vector, $\mathbf{u}_c \in \mathcal{U}_c \subseteq \mathbb{R}^{m_c}$ is the exogenous continuous input vector, $\mathbf{y}_c \in \mathcal{Y}_c \subseteq \mathbb{R}^{p_c}$ is the continuous output vector, $\{\mathbf{A}_i, \mathbf{B}_i, \mathbf{f}_i, \mathbf{C}_i, \mathbf{D}_i, \mathbf{g}_i\}_{i \in \mathcal{Q}}$ is a collection of matrices of opportune dimensions, and the mode $i(k) \in \mathcal{Q} \triangleq \{1, \dots, s\}$ is an input signal that chooses the affine state update dynamics. An SAS can be rewritten as the combination of linear terms and *if-then-else* rules: the state-update equation (5.6a) is equivalent to

$$\mathbf{z}_1(k) = \begin{cases} \mathbf{A}_1\mathbf{x}_c(k) + \mathbf{B}_1\mathbf{u}_c(k) + \mathbf{f}_1, & \text{if } (i(k) = 1), \\ \mathbf{0}, & \text{otherwise,} \end{cases} \quad (5.7a)$$

$$\vdots$$

$$\mathbf{z}_s(k) = \begin{cases} \mathbf{A}_s\mathbf{x}_c(k) + \mathbf{B}_s\mathbf{u}_c(k) + \mathbf{f}_s, & \text{if } (i(k) = s), \\ \mathbf{0}, & \text{otherwise,} \end{cases} \quad (5.7b)$$

$$\mathbf{x}_c(k+1) = \sum_{i=1}^s \mathbf{z}_i(k), \quad (5.7c)$$

where $\mathbf{z}_i(k) \in \mathbb{R}^{n_c}$, $i = 1, \dots, s$, and (5.6b) admits a similar transformation.

Event generator (EG) An event generator is a mathematical object that generates a logic signal according to the satisfaction of a linear affine constraint

$$\delta_e(k) = \mathbf{f}_H(\mathbf{x}_c(k), \mathbf{u}_c(k), k), \quad (5.8)$$

where $\mathbf{f}_H : \mathcal{X}_c \times \mathcal{U}_c \times \mathbb{Z}_{\geq 0} \rightarrow \mathcal{D} \subseteq \{0, 1\}^{n_e}$ is a vector of descriptive functions of a linear hyperplane, and $\mathbb{Z}_{\geq 0} \triangleq \{0, 1, \dots\}$ is the set of nonnegative integers. In particular *threshold events* are modeled as $[\delta_e^i(k) = 1] \leftrightarrow [\mathbf{a}^T \mathbf{x}_c(k) + \mathbf{b}^T \mathbf{u}_c(k) \leq c]$, where the superscript i denotes the i -th component of a vector. *Time events* can be also modeled as: $[\delta_e^i(k) = 1] \leftrightarrow [t(k) \geq t_0]$, where $t(k+1) = t(k) + T_s$ denotes time, T_s is the sampling time, and t_0 is a given time.

Finite state machine (FSM) A finite-state machine (or automaton) is a discrete dynamical process that evolves according to a logic state update function:

$$\mathbf{x}_\ell(k+1) = \mathbf{f}_B(\mathbf{x}_\ell(k), \mathbf{u}_\ell(k), \delta_e(k)), \quad (5.9a)$$

where $\mathbf{x}_\ell \in \mathcal{X}_\ell \subseteq \{0, 1\}^{n_\ell}$ is the Boolean state, $\mathbf{u}_\ell \in \mathcal{U}_\ell \subseteq \{0, 1\}^{m_\ell}$ is the exogenous Boolean input, $\delta_e(k)$ is the endogenous input coming from the EG, and $\mathbf{f}_B : \mathcal{X}_\ell \times \mathcal{U}_\ell \times \mathcal{D} \rightarrow \mathcal{X}_\ell$ is a deterministic logic function. (Here we will only refer to synchronous finite-state machines, where the transitions may happen only at sampling times. The adjective “synchronous” will be omitted for brevity.) An FSM can be conveniently represented using an oriented graph. An FSM may also have an associated Boolean output

$$\mathbf{y}_\ell(k) = \mathbf{g}_B(\mathbf{x}_\ell(k), \mathbf{u}_\ell(k), \delta_e(k)), \quad (5.9b)$$

where $\mathbf{y}_\ell \in \mathcal{Y}_\ell \subseteq \{0, 1\}^{p_\ell}$ and $\mathbf{g}_\ell : \mathcal{X}_c \times \mathcal{U}_c \times D \rightarrow \mathcal{Y}_\ell$.

Mode selector (MS) The logic state $\mathbf{x}_\ell(k)$, the Boolean inputs $\mathbf{u}_\ell(k)$, and the events $\delta_e(k)$ select the dynamical mode $i(k)$ of the SAS through a Boolean function $f_M : \mathcal{X}_\ell \times \mathcal{U}_\ell \times \mathcal{D} \rightarrow \mathcal{Q}$, which is therefore called a *mode selector*. The output of this function

$$i(k) = f_M(\mathbf{x}_\ell(k), \mathbf{u}_\ell(k), \delta_e(k)) \quad (5.10)$$

is called the *active mode*. We say that a *mode switch* occurs at step k if $i(k) \neq i(k-1)$. Note that, in contrast to continuous-time hybrid models, where switches can occur at any time, in our discrete-time setting a mode switch can only occur at sampling instants.

DHA are related to *hybrid automata* (HA) [15], the main difference is in the time model: DHA admit time in the natural numbers, while in HA the time is a real number. Moreover, DHA models do not allow instantaneous transitions, and are deterministic, as opposed to HA where any enabled transition may occur in zero time. This has two consequences: (i) DHA do not admit live-locks (infinite switches in zero time), (ii) DHA do not admit Zeno behaviors (infinite switches in finite time). Finally, in DHA models, guards, reset maps, and continuous dynamics are limited to linear affine functions. Moreover, contrarily to HA, in DHA the continuous dynamics is not a property of the state of the automaton but is selected by the mode selector (MS) according also to discrete inputs and events. For equivalence results between linear hybrid automata and continuous-time piecewise affine systems see [136]. Reset maps in DHA can be dealt with as described in [632].

5.1.3 Mixed logical dynamical systems

This section describes how to transform a DHA into an equivalent hybrid model described by linear mixed-integer equations and inequalities, by generalizing several results that have already appeared in the literature [62, 331, 455, 553, 664]. The hybrid systems modeling language *HYSDEL* introduced in [632] and also described in Chapter 10 was developed to describe DHA and to automatically operate the transformations.

Logical functions Boolean functions can be equivalently expressed by inequalities [165].

In order to introduce our notation, we recall here some basic definitions of Boolean algebra. A variable X is a *Boolean variable* if $X \in \{0, 1\}$. A *Boolean expression* is inductively defined (for the sake of simplicity, we will neglect precedence) by the grammar

$$\begin{aligned} \phi ::= & X | \neg\phi_1 | \phi_1 \vee \phi_2 | \phi_1 \oplus \phi_2 | \phi_1 \wedge \phi_2 | \\ & \phi_1 \leftarrow \phi_2 | \phi_1 \rightarrow \phi_2 | \phi_1 \leftrightarrow \phi_2 | (\phi_1), \end{aligned} \quad (5.11)$$

where X is a Boolean variable, and the logic operators \neg (not), \vee (or), \wedge (and), \leftarrow (implied by), \rightarrow (implies), \leftrightarrow (iff) have the usual semantics. A Boolean expression is in *conjunctive normal form* (CNF) or *product of sums* if it can be written according to the following grammar:

$$\phi ::= \psi | \phi \wedge \psi, \tag{5.12}$$

$$\psi ::= \psi_1 \vee \psi_2 | \neg X | X, \tag{5.13}$$

where ψ are called the *terms of the product*, and X are the *terms of the sum* ψ . A CNF is minimal if it has the minimum number of terms of product and each term has the minimum number of terms of sum. Every Boolean expression can be rewritten as a minimal CNF.

A Boolean expression f will be also called a *Boolean function* when is used to define a literal X_n as a function of X_1, \dots, X_{n-1} :

$$X_n = f(X_1, X_2, \dots, X_{n-1}). \tag{5.14}$$

In general, we can define relations among Boolean variables X_1, \dots, X_n through a *Boolean formula*

$$F(X_1, \dots, X_n) = 1, \tag{5.15}$$

where $X_i \in \{0, 1\}$, $i = 1, \dots, n$. Note that each Boolean function is also a Boolean formula, but not vice versa. Boolean formulas can be equivalently translated into a set of integer linear inequalities. For instance, $X_1 \vee X_2 = 1$ is equivalent to $X_1 + X_2 \geq 1$ [664]. The translation can be performed either using an *symbolical* method or a *geometrical* method (see details in [632]). In particular, the symbolical method consists of first converting (5.14) or (5.15) into its CNF

$$\bigwedge_{j=1}^m \left(\bigvee_{i \in P_j} X_i \bigvee_{i \in N_j} \neg X_i \right),$$

with $N_j, P_j \subseteq \{1, \dots, n\} \forall j = 1, \dots, m$. Then, the corresponding set of integer linear inequalities is

$$\begin{cases} 1 \leq \sum_{i \in P_1} X_i + \sum_{i \in N_1} (1 - X_i), \\ \vdots \\ 1 \leq \sum_{i \in P_m} X_i + \sum_{i \in N_m} (1 - X_i). \end{cases} \tag{5.16}$$

Continuous-logic interfaces By using the so-called “big-M” technique, events of the form (5.8) can be equivalently expressed as

$$f_{\text{H}}^i(x_c(k), u_c(k), k) \leq M^i(1 - \delta_e^i), \tag{5.17a}$$

$$f_{\text{H}}^i(x_c(k), u_c(k), k) > m^i \delta_e^i, \quad i = 1, \dots, n_e, \tag{5.17b}$$

where M^i , m^i are upper and lower bounds, respectively, on $f_{\text{H}}^i(x_c(k), u_c(k), k)$. As we will point out in Section 5.19e, sometimes, from a computational point of view, it may be convenient to have a system of inequalities without strict inequalities. In this case we will follow the common practice [664] of replacing the strict inequality (5.17b) by

$$f_{\text{H}}^i(x_c(k), u_c(k), k) \geq \epsilon + (m^i - \epsilon)\delta_e^i, \quad (5.17c)$$

where ϵ is a small positive scalar, e.g. the machine precision, although the equivalence does not hold for $0 < f_{\text{H}}^i(x_c(k), u_c(k), k) < \epsilon$, as the numbers in the interval $(0, \epsilon)$ cannot be represented in a computer.

The most common *logic to continuous* interface is the if-then-else construct

$$\text{IF } \delta \text{ THEN } z = a_1^{\text{T}}x + b_1^{\text{T}}u + f_1 \text{ ELSE } z = a_2^{\text{T}}x + b_2^{\text{T}}u + f_2, \quad (5.18)$$

which can be translated into [66]

$$(m_2 - M_1)\delta + z \leq a_2x + b_2u + f_2, \quad (5.19a)$$

$$(m_1 - M_2)\delta - z \leq -a_2x - b_2u - f_2, \quad (5.19b)$$

$$(m_1 - M_2)(1 - \delta) + z \leq a_1x + b_1u + f_1, \quad (5.19c)$$

$$(m_2 - M_1)(1 - \delta) - z \leq -a_1x - b_1u - f_1, \quad (5.19d)$$

where M_i , m_i are upper and lower bounds on $a_i x + b_i u + f_i$, $i = 1, 2$, $\delta \in \{0, 1\}$, $z \in \mathbb{R}$, $x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$. Note that when a_2, b_2, f_2 are zero, (5.18)–(5.19) coincide with the product $z = \delta \cdot (ax + bu + f)$ described in [664].

Continuous dynamics As already mentioned, we will deal with dynamics described by linear affine difference equations

$$x_c(k+1) = \sum_{i=1}^s z_i(k). \quad (5.20)$$

Mixed logical dynamical systems In [62] the authors proposed discrete-time hybrid systems denoted as mixed logical dynamical (MLD) systems. An MLD system is described by the following relations:

$$\mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{B}_1\mathbf{u}(k) + \mathbf{B}_2\delta(k) + \mathbf{B}_3\mathbf{z}(k) + \mathbf{B}_5, \quad (5.21a)$$

$$\mathbf{y}(k) = \mathbf{C}\mathbf{x}(k) + \mathbf{D}_1\mathbf{u}(k) + \mathbf{D}_2\delta(k) + \mathbf{D}_3\mathbf{z}(k) + \mathbf{D}_5, \quad (5.21b)$$

$$\mathbf{E}_2\delta(k) + \mathbf{E}_3\mathbf{z}(k) \leq \mathbf{E}_1\mathbf{u}(k) + \mathbf{E}_4\mathbf{x}(k) + \mathbf{E}_5, \quad (5.21c)$$

$$\tilde{\mathbf{E}}_2\delta(k) + \tilde{\mathbf{E}}_3\mathbf{z}(k) < \tilde{\mathbf{E}}_1\mathbf{u}(k) + \tilde{\mathbf{E}}_4\mathbf{x}(k) + \tilde{\mathbf{E}}_5. \quad (5.21d)$$

where $\mathbf{x} \in \mathbb{R}^{n_c} \times \{0, 1\}^{n_\ell}$ is a vector of continuous and binary states, $\mathbf{u} \in \mathbb{R}^{m_c} \times \{0, 1\}^{m_\ell}$ are the inputs, $\mathbf{y} \in \mathbb{R}^{p_c} \times \{0, 1\}^{p_\ell}$ the outputs, $\delta \in \{0, 1\}^{r_\ell}$, $\mathbf{z} \in \mathbb{R}^{r_c}$ represent auxiliary binary and continuous variables, respectively, and \mathbf{A} , \mathbf{B}_1 , \mathbf{B}_2 , \mathbf{B}_3 , \mathbf{C} , \mathbf{D}_1 , \mathbf{D}_2 , \mathbf{D}_3 , $\mathbf{E}_1, \dots, \mathbf{E}_5$, and $\tilde{\mathbf{E}}_1, \dots, \tilde{\mathbf{E}}_5$ are matrices of suitable

dimensions. Given the current state $\mathbf{x}(k)$ and input $\mathbf{u}(k)$, the time-evolution of (5.21) is determined by solving $\delta(k)$ and $\mathbf{z}(k)$ from (5.21c)–(5.21d), and then updating $\mathbf{x}^T(k)$ and $\mathbf{y}(k)$ from (5.21a)–(5.21b). Since the problems of synthesis and analysis of MLD models are tackled using optimization techniques, we have replaced strict inequalities as in (5.17b) by non-strict inequalities as in (5.17c). (One may also explicitly include in (5.21) strict inequalities, as well as equalities.) A formal definition of well-posedness for MLD systems and a test to assess the well-posedness have been presented in [62].

For equivalence results between MLD systems and PWA systems, see Section 5.3.

5.1.4 Hybrid model-predictive control

MPC based on hybrid dynamical models has emerged in recent years as a very promising approach to operate switching linear dynamics, on/off inputs, and logic states, subject to combinations of linear and logical constraints on input and state variables [62]. Hybrid dynamics are often so complex that a satisfactory feedback controller cannot be synthesized by using analytical tools, and heuristic design procedures usually require trial and error sessions and extensive testing, and are time consuming, costly, and often inadequate to deal with the complexity of the hybrid control problem properly.

As for the linear MPC case, hybrid MPC design is a systematic approach to meet performance and constraint specifications in spite of the aforementioned switching among different linear dynamics, logical state transitions, and more complex logical constraints on system's variables. The approach consists of modeling the switching open-loop process and constraints as a discrete hybrid automaton using the language *HYSDEL* [632], and then automatically transforming the model into the MLD form (5.21).

The associated finite-horizon optimal control problem based on quadratic costs takes the form (5.3) with

$$\mathbf{U} = [\mathbf{u}^T(0) \dots \mathbf{u}^T(N-1) \delta^T(0) \dots \delta^T(N-1) \mathbf{z}^T(0) \dots \mathbf{z}^T(N-1)]^T,$$

subject to the further restriction that some of the components of \mathbf{U} must be either 0 or 1. The problem is therefore a mixed-integer quadratic programming (MIQP) problem, for which both commercial [198, 334] and public domain solvers (such as the one in [61]) are available. When infinity norms $\|\mathbf{Q}\mathbf{x}(k)\|_\infty$, $\|\mathbf{R}\mathbf{u}(k)\|_\infty$, $\|\mathbf{P}\mathbf{x}(k)\|_\infty$ are used in (5.2a) in place of quadratic costs, the optimization problem becomes a mixed-integer linear programming (MILP) problem [57, 63], which can be also handled by efficient public domain solvers such as [434], as well as by commercial solvers [198, 334].

Unfortunately MIPs are NP-complete problems. However, the state of the art in solving MIP problems is growing constantly, and problems of relatively large size can be solved quite efficiently. While MIP problems can always be solved to the global optimum, closed-loop stability properties can be guaranteed as long as the

optimum value in (5.3) decreases at each time step. Usually, MIP solvers provide good feasible solutions within a relatively short time compared to the total time required to find and certify the global optimum. In the worst-case the complexity of optimally computing the control action $u(k)$ on-line at each time k depends exponentially on the number of integer variables [553]. In principle, this limits the scope of application of the proposed method to relatively slow systems, since the sampling time should be large enough for real-time implementation to allow the worst-case computation.

In general, an MIP solver provides the solution after solving a sequence of relaxed standard linear (or quadratic) problems (LP, QP). A potential drawback of MIP is (1) the need for converting the discrete/logic part of the hybrid problem into mixed-integer inequalities, therefore losing most of the original discrete structure, and (2) the fact that its efficiency mainly relies upon the tightness of the continuous LP/QP relaxations. Such drawbacks are not suffered by techniques for solving constraint satisfaction problems (CSP), i.e. the problem of determining whether a set of constraints over discrete variables can be satisfied. Under the class of CSP solvers we mention constraint logic programming (CLP) [439] and satisfiability (SAT) solvers [287], the latter specialized for the satisfiability of Boolean formulas. The approach of [60] combines MIP and CSP techniques in a co-operative way. In particular, convex programming for optimization over real variables, and SAT solvers for determining the satisfiability of Boolean formulas (or logic constraints), are combined in a single branch and bound solver.

Another approach for reducing the complexity of on-line computations is to look for suboptimal solutions. For instance in [337] the authors propose to suitably constrain the mode sequence over the prediction horizon, so that on-line optimization is solved more quickly. Although closed-loop stability is still guaranteed by this approach, clearly in general the overall tracking performance of the feedback loop gets deteriorated.

In the last decade, *explicit model-predictive control* has been proposed as a way to completely get rid of the need of on-line solvers (see [11] for a survey on explicit MPC).

For linear MPC, to get rid of on-line QP an approach to evaluate the MPC law (5.4) was proposed in [67]. Rather than solving the QP problem (5.3) on-line for the current vector $x(k)$, the idea is to solve (5.3) off-line for all vectors x within a given range and make the dependence of u on x *explicit* (rather than implicitly defined by the optimization procedure (5.3)). The key idea is to treat (5.3) as a *multi-parametric* quadratic programming problem, where $x(k)$ is the vector of parameters. It turns out that the optimizer $U^* : \mathbb{R}^n \rightarrow \mathbb{R}^{mN_u}$ is a piecewise affine and continuous function, and consequently the MPC controller defined by (5.4) can be represented explicitly as

$$u(x) = \begin{cases} F_1 x + g_1 & \text{if } H_1 x \leq k_1 \\ \vdots & \vdots \\ F_M x + g_M & \text{if } H_M x \leq k_M. \end{cases} \quad (5.22)$$

It turns also out that the set of states \mathcal{X}^* for which problem (5.3) admits a solution is a polyhedron, and that the optimum value in (5.3) is a piecewise quadratic, convex, and continuous function of $\mathbf{x}(k)$. The controller structure (5.22) is simply a look-up table of linear gains $(\mathbf{F}_i, \mathbf{g}_i)$, where the i -th gain is selected according to the set of linear inequalities $\mathbf{H}_i \mathbf{x} \leq \mathbf{k}_i$ that the state vector satisfies. Hence, the evaluation of the MPC controller (5.4), once put in the form (5.22), can be carried out by a very simple piece of control code. In the most naive implementation, the number of operations depends linearly in the worst case on the number M of partitions, or even logarithmically if the partitions are properly stored [630].

An alternative way of solving MIP problems on-line is to extend explicit MPC ideas to the hybrid case. For hybrid MPC problems based on infinity norms, [63] showed that an equivalent piecewise affine explicit reformulation—possibly discontinuous, due to binary variables—can be obtained through off-line multiparametric mixed-integer linear programming techniques.

Thanks to the possibility of converting hybrid models (such as those designed through *HYSDEL*) to an equivalent piecewise affine (PWA) form [56], an explicit hybrid MPC approach dealing with quadratic costs was proposed in [105], based on dynamical programming (DP) iterations. Multiparametric quadratic programs (mpQP) are solved at each iteration, and quadratic value functions are compared to possibly eliminate regions that are proved to never be optimal. A different approach still exploiting the PWA structure of the hybrid model was proposed in [446], where all possible switching sequences are enumerated, an mpQP is solved for each sequence, and quadratic costs are compared on-line to determine the optimal input (in this respect, one could define the approach semi-explicit). To overcome the problem of enumerating all switching sequences and storing all the corresponding mpQP solutions, backwards reachability analysis is exploited in [10] (and implemented in the *Hybrid Toolbox*). A procedure to post-process the mpQP solutions and eliminate all polyhedra (and their associated control gains) that never provide the lowest cost was suggested in [10]. Typically the DP approach provides simpler explicit solutions when long horizons N are chosen, but on the contrary tends to subdivide the state space in a larger number of polyhedra than the enumeration approach for short horizons.

For closed-loop convergence results of hybrid MPC the reader is referred to [62, 138, 386, 387, 388] and to the PhD thesis [385]. Extensions of hybrid MPC to stochastic hybrid systems was proposed in [59], and to event-based continuous-time hybrid systems in [71].

The *Hybrid Toolbox for MATLAB* [57] provides a nice development environment for hybrid and explicit MPC design. Hybrid dynamical systems described in *HYSDEL* are automatically converted to *MATLAB* MLD and PWA objects. MLD and PWA objects can be validated in open-loop simulation, either from the command line or through their corresponding *Simulink* blocks. Hybrid MPC controllers based on MILP/MIQP optimization can be designed and simulated, either from the command line or in *Simulink*, and can be converted to their explicit form for deployment. Several demos are available in the *Hybrid Toolbox* distribution. The toolbox can be freely downloaded from <http://www.dii.unisi.it/hybrid/toolbox>.

Similar functionalities are also included in the *Multi Parametric Toolbox* [375]. The reader is referred to [Chapter 10](#) for a more detailed description of these tools.

In conclusion, hybrid MPC control can deal with very complex specifications in terms of models and constraints by using mixed-integer programming solvers. Explicit versions of hybrid MPC are possible, but still limited to small systems with few binary variables. Examples of applications of hybrid MPC to industrial control problems arising in the automotive domain are reported in [Chapter 15](#).

5.2 Complementarity systems

5.2.1 Modeling aim

In many areas, especially in the domain of physical systems or in economic applications, continuous-time hybrid systems usually arise in specific forms. The continuous-time dynamics corresponding to the different modes, as well as their location invariants and guards, are often closely related. Indeed, in many cases the dynamics corresponding to the different modes all share a part that can be called the *core dynamics* of the system.

The theory of complementarity hybrid systems, as originally put forward in [572, 573], aims at providing a compact representation of many of such systems. It combines location invariants and guards in the form of *complementarity conditions* such as $0 \leq z \perp w \geq 0$, where z and w are equal-dimensional vectors, and the inequalities hold componentwise. It is not without reason that many hybrid systems can be formulated in this manner, since complementarity conditions are closely related with variational and optimal formulations, which are known to be underlying many systems in physics and economical applications. Furthermore, it can be shown that, roughly speaking, all piecewise-linear characteristics can be modeled by complementarity conditions.

In addition to the rather broad applicability of complementarity modeling there are two other important advantages of complementarity models. First, complementarity models often provide a very compact description of hybrid systems, especially in comparison with hybrid automata. Furthermore, the complementarity model usually remains to the physics of the system, and physical system properties (such as passivity) are naturally reflected in the representation. Secondly, complementarity modeling offers powerful methods for analysis. Using the well-developed theory of the linear complementarity problem (LCP) from optimization theory [189] one may prove strong results concerning well-posedness (existence and uniqueness of solutions), stability and controllability. Also the theory of the LCP offers a wealth of computational methods, e.g. for the efficient computation of the next location at an event time. We refer to, e.g., [149, 300] for a detailed description of these results, especially for linear complementarity systems.

In this section we will mainly concentrate on indicating the modeling power of complementarity hybrid systems by discussing a list of appealing examples from

different application areas, including the running examples in this handbook. Furthermore, we will briefly sketch some of the main results which have been obtained on the well-posedness, stability, controllability, and stabilizability of linear complementarity systems.

5.2.2 Definition

Complementarity systems can be constructed as follows [572, 573]. Start from a nonlinear input/output system, with k inputs and k outputs:

$$\frac{d\mathbf{x}}{dt}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)), \quad (5.23a)$$

$$\mathbf{y}(t) = \mathbf{h}(\mathbf{x}(t), \mathbf{u}(t)), \quad (5.23b)$$

where $\mathbf{x}(t)$ is an n -dimensional state variable, $\mathbf{u}(t) \in \mathbb{R}^k$ is the input vector and $\mathbf{y}(t) \in \mathbb{R}^k$ is the output vector. To the system (5.23a)–(5.23b), add the relation

$$0 \leq \mathbf{y}(t) \perp \mathbf{u}(t) \geq 0. \quad (5.23c)$$

A relation of the form (5.23c) are called a *complementarity relation* in mathematical programming; whence the name *complementarity systems* for dynamical systems of the form (5.23). Note that (5.23c) is equivalent to the componentwise requirement that, for each $i = 1, \dots, k$, the following holds: $y_i(t) \geq 0$, $u_i(t) \geq 0$, and at least one of these two inequalities is satisfied with equality.

In view of the particular role of the input and output variables in the formulation of complementarity systems, the notations \mathbf{y} and \mathbf{u} are sometimes replaced by \mathbf{w} and \mathbf{z} , to steer away from the interpretation of the input as a control and the output as an observation and also to be in line with notational conventions in mathematical programming. In addition, the formulation in (5.23) can be made more general by allowing the functions \mathbf{f} and \mathbf{h} to depend directly on time.

Implicit in (5.23c) is the choice of an “active index set” $\alpha(t) \subset \{1, \dots, k\}$ which is such that $y_i(t) = 0$ for $i \in \alpha(t)$ and $u_i(t) = 0$ for $i \notin \alpha(t)$. Any such index set is said to represent a *mode* of operation. In a fixed mode, the system above behaves as the dynamical system described by the differential equation (5.23a) and the algebraic relations (5.23b) together with the equalities that follow from the choice of the active index set in (5.23c). A change of mode occurs when continuation within a given mode would violate the nonnegativity constraints associated with this mode. The description format of complementarity systems is such that the nonsmoothness is made canonical, and specific properties, therefore, must be expressible in terms of the functions $\mathbf{f}(\cdot, \cdot)$ and $\mathbf{h}(\cdot, \cdot)$ occurring in (5.23a) and (5.23b), and possibly in terms of an initial condition.

A subclass of particular interest arises when the functions \mathbf{f} and \mathbf{h} in (5.23) are required to be linear; the resulting *linear complementarity systems* [304] are described by relations of the form

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t), \quad (5.24a)$$

$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t), \quad (5.24b)$$

$$0 \leq \mathbf{y}(t) \perp \mathbf{u}(t) \geq 0, \quad (5.24c)$$

where \mathbf{A} , \mathbf{B} , \mathbf{C} , and \mathbf{D} are linear mappings. In some applications it is natural to allow an external input (forcing term) in a complementarity system. The equations (5.23a) and (5.23b) are then replaced by equations of the form

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{v}(t)), \quad (5.25a)$$

$$\mathbf{y}(t) = \mathbf{h}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{v}(t)), \quad (5.25b)$$

where $\mathbf{v}(t)$ denotes the forcing term; the equation (5.23c) is unchanged. In linear complementarity systems we require that the forcing term also enters linearly, so that the system (5.24) is replaced by

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) + \mathbf{E}\mathbf{v}(t), \quad (5.26a)$$

$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t) + \mathbf{F}\mathbf{v}(t), \quad (5.26b)$$

$$0 \leq \mathbf{y}(t) \perp \mathbf{u}(t) \geq 0, \quad (5.26c)$$

where \mathbf{A} , \mathbf{B} , \mathbf{C} , \mathbf{D} , \mathbf{E} , and \mathbf{F} are linear mappings. Another useful generalization of (5.23) is obtained when the complementarity relation (5.23c) is replaced by the relation

$$\mathcal{C} \ni \mathbf{y}(t) \perp \mathbf{u}(t) \in \mathcal{C}^*, \quad (5.27)$$

where \mathcal{C} is a cone in \mathbb{R}^k and \mathcal{C}^* is the dual cone defined by

$$\mathcal{C}^* = \{\mathbf{u} \mid \langle \mathbf{y}, \mathbf{u} \rangle \geq 0 \text{ for all } \mathbf{y} \in \mathcal{C}\}.$$

In particular, this format allows the incorporation of both equality and inequality constraints; a typical choice of the cone \mathcal{C} is $\mathbb{R}_+^{k_1} \times \{0\}$, which implies $\mathcal{C}^* = \mathbb{R}_+^{k_1} \times \mathbb{R}^{k_2}$.

5.2.3 Examples

This section shows by means of several examples how complementarity relations can be obtained when modeling physical systems of different nature.

Example 5.1 DC-DC converter

The DC-DC converter shown in Fig. 1.17 consists of an inductor L , a capacitor C , resistances R_L , R_C , and a resistance load R , together with a diode D and an ideal switch S . The diode is modeled as an ideal diode, and its constitutive relation can be succinctly expressed by the complementarity condition

$$0 \leq v_D \perp i_D \geq 0, \quad (5.28)$$

with i_D , respectively v_D , the current through the diode and the voltage across the diode. Furthermore, the constitutive relation of the ideal switch S can be simply expressed as

$$v_S \perp i_S, \quad (5.29)$$

with i_S, v_S the current through the switch and the voltage across the switch.

Taking as continuous state variables the voltage v_C across the capacitor and the current i_L through the inductor, we obtain the following dynamical equations of the DC-DC converter:

$$\frac{d}{dt} \begin{bmatrix} v_C \\ i_L \end{bmatrix} = \begin{bmatrix} -\frac{1}{C(R+R_C)} & 0 \\ 0 & -\frac{R_L}{L} \end{bmatrix} \begin{bmatrix} v_C \\ i_L \end{bmatrix} + \begin{bmatrix} \frac{R}{C(R+R_C)} & 0 \\ 0 & \frac{1}{L} \end{bmatrix} \begin{bmatrix} i_D \\ v_S \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{L} \end{bmatrix} E, \quad (5.30a)$$

$$\begin{bmatrix} v_D \\ i_S \end{bmatrix} = \begin{bmatrix} \frac{R}{R+R_C} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_C \\ i_L \end{bmatrix} + \begin{bmatrix} \frac{RR_C}{R+R_C} & 1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} i_D \\ v_S \end{bmatrix}. \quad (5.30b)$$

Here, E is the voltage of the input source.

Defining the *switching function* π as

$$\pi(t) = \begin{cases} -1 & \text{if the switch } S \text{ is off at time } t, \\ 1 & \text{if the switch } S \text{ is on at time } t, \end{cases}$$

and the cones

$$C_{-1} = \mathbb{R}_+ \times \{0\} \quad C_1 = \mathbb{R}_+ \times \mathbb{R},$$

one can represent the relations (5.28) and (5.29) as

$$C_{\pi(t)} \ni \begin{bmatrix} v_D \\ i_S \end{bmatrix} \perp \begin{bmatrix} \dot{i}_D \\ v_S \end{bmatrix} \in C_{\pi(t)}^*. \quad (5.30c)$$

Systems of the form (5.30) are called *switched cone complementarity systems*. As shown in [157, 158, 307], this class provides a compact representation of any type of power converters.

Note that certain properties of the DC-DC converter are immediately obtained as a direct consequence of the complementarity modeling. For example, it is readily verified that the total energy $H(v_C, i_L) := \frac{1}{2}Cv_C^2 + \frac{1}{2}Li_L^2$ stored in the circuit satisfies

$$\frac{d}{dt}H = -R_L i_L^2 - R_C i_C^2 - RI^2 + EI_E,$$

where I denotes the current through the resistive load R , and I_E is the current through the voltage source with voltage E . Hence *passivity* of the obtained model is directly established. (In fact, this becomes even more transparent by writing (5.30) into a port-Hamiltonian form, thus obtaining a *port-Hamiltonian complementarity system*.) \square

Example 5.2 Two-tank system

The discrete states of the two-tank system, introduced in Section 1.3.1 are determined by inequalities involving the continuous states and by external switches. The system can be modeled in a switched complementarity framework. The main issues are: (i) modeling of the mode-dependent flow through the valve V_1 , and (ii) modeling of the opening and closing of the valves.

Consider first item (i). The flow equations can be described in terms of the positive-part operator which is defined, for $x \in \mathbb{R}$, by

$$x^+ = \max(x, 0).$$

Indeed, we can write

$$Q_{12}^{V_1}(t) = u_1(t)T((h_1(t) - h_0)^+ - (h_2(t) - h_0)^+), \quad (5.31)$$

where $T(x)$ is the Torricelli characteristic

$$T(x) = c \operatorname{sgn}(x) \sqrt{2g|x|}, \quad (5.32)$$

which may be considered to be a smooth function even though its derivative at $x = 0$ is infinity. The positive-part operator in its turn can be described in terms of a complementary characteristic, since the relations

$$w = x^+, \quad z = x^- = (-x)^+$$

are equivalent to

$$x = w - z, \quad 0 \leq w \perp z \leq 0.$$

Therefore, the relation (5.31) can alternatively be formulated as

$$Q_{12}^{V_1}(t) = u_1(t)T(w_1(t) - w_2(t)), \quad (5.33)$$

together with the complementarity relations

$$0 \leq w_1(t) \perp z_1(t) \geq 0, \quad (5.34)$$

$$0 \leq w_2(t) \perp z_2(t) \geq 0, \quad (5.35)$$

and the algebraic relations

$$w_1(t) = h_1(t) - h_0 + z_1(t), \quad (5.36)$$

$$w_2(t) = h_2(t) - h_0 + z_2(t). \quad (5.37)$$

The Torricelli characteristic (5.32) could be replaced by a nonsmooth function; for instance a relay characteristic might be an alternative. Complementarity modeling is then still possible by using the techniques described below in the discussion of relay systems.

The switching of the valves can be modeled as in (5.31) by means of a multiplicative factor, but an alternative is to use the setting of switched cone complementarity systems as proposed in [157]. One then introduces to the flow variable Q_{12} another variable λ_{12} , which relates to pressure drop across the valve V_1 . The modeling (5.31) (or equivalently (5.33)) is then replaced by

$$Q_{12}^{V_1}(t) = T(w_1(t) - w_2(t)) + \lambda_{12}(t), \quad (5.38)$$

together with the complementarity relation

$$\mathcal{C}_{u_1}^* \ni \lambda_{12}(t) \perp Q_{12}(t) \in \mathcal{C}_{u_1}(t), \quad (5.39)$$

where \mathcal{C} is the cone-valued function defined by

$$\mathcal{C}_0 = \{0\}, \quad \mathcal{C}_1 = \mathbb{R}.$$

This models an on/off switch. One could also describe switches that admit flow in one direction only by making use of the cones \mathbb{R}_+ and \mathbb{R}_- . \square

Example 5.3 *Relay systems*

Consider a dynamical system of the form

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), u(t)), \quad (5.40)$$

together with the following nonsmooth relation that specifies the dependence of the input variable $u(t)$ on the state $\mathbf{x}(t)$:

$$u(t) = 1 \text{ if } h(\mathbf{x}(t)) > 0, \quad (5.41a)$$

$$0 \leq u(t) \leq 1 \text{ if } h(\mathbf{x}(t)) = 0, \quad (5.41b)$$

$$u(t) = 0 \text{ if } h(\mathbf{x}(t)) < 0. \quad (5.41c)$$

Such a system is called a *differential equation with discontinuous right-hand side* or a *relay system*. The latter terminology is derived from the fact that, if one introduces an output variable $y(t) = h(\mathbf{x}(t))$, the relation between $u(t)$ and $y(t)$ given by (5.41) is known as a *relay characteristic*. The system can of course also be viewed as a hybrid system with three different modes.

Relay systems can be modeled as cone complementarity systems. For this purpose, introduce two input variables, say $v(t)$ and $z(t)$, in addition to $u(t)$. Corresponding to the three input variables $u(t)$, $v(t)$, and $z(t)$, introduce three output variables $p(t)$, $q(t)$, and $r(t)$, which are defined in terms of the state and input variables by

$$\begin{aligned} p(t) &= z(t), \\ q(t) &= z(t) + h(\mathbf{x}(t)), \\ r(t) &= u(t) + v(t) - 1. \end{aligned}$$

A cone complementarity system is formed by taking the equation (5.40) together with the following cone complementarity relations:

$$\begin{aligned} 0 &\leq p(t) \perp u(t) \geq 0, \\ 0 &\leq q(t) \perp v(t) \geq 0, \\ 0 &= r(t) \perp z(t) \in \mathbb{R}. \end{aligned}$$

This is of the form (5.27) with $\mathcal{C} = \mathbb{R}_+^2 \times \{0\}$. The third relation is equivalent to the requirement that $u(t) + v(t) = 1$ for all t . As a consequence, the number of modes implied by the first two relations is reduced from four to three, since the mode corresponding to $u(t) = 0$ and $v(t) = 0$ is eliminated. The three remaining modes can be described as follows:

1. $u(t) = 0, p(t) \geq 0, q(t) = 0, v(t) \geq 0$. The relations $p(t) \geq 0$ and $q(t) = 0$ imply that in this case we must have $h(\mathbf{x}(t)) \leq 0$.
2. $u(t) \geq 0, p(t) = 0, q(t) = 0, v(t) \geq 0$. The relations $p(t) = 0$ and $q(t) = 0$ imply that $h(\mathbf{x}(t)) = 0$. From the relations $u(t) \geq 0, v(t) \geq 0$, and $u(t) + v(t) = 1$ it follows that $0 \leq u(t) \leq 1$.
3. $u(t) \geq 0, p(t) = 0, q(t) \geq 0, v(t) = 0$. The relations $p(t) \geq 0$ and $q(t) = 0$ imply that in this case we must have $h(\mathbf{x}(t)) \geq 0$. From $v(t) = 0$ and $u(t) + v(t) = 1$ it follows that $u(t) = 1$.

It is seen that the cone complementarity system describes the same dynamics as the relay system. In this specific case, where we have a single relay characteristic, the reformulation in cone complementarity form may appear to be artificial and perhaps even awkward. However, the cone complementarity form simplifies substantially the description of multi-regime dynamics in more complicated situations. This is demonstrated below. \square

Example 5.4 *Filippov systems*

Consider the dynamics

$$\dot{\mathbf{x}}(t) = \sum_{i=1}^k \lambda_i(t) f_i(\mathbf{x}(t)), \quad \mathbf{w}(t) = H(\mathbf{x}(t)), \quad (5.42a)$$

where H is a smooth mapping from \mathbb{R}^n to \mathbb{R}^k , together with the linear constraint

$$\sum_{i=1}^k \lambda_i(t) = 1 \quad (5.42b)$$

and the complementarity conditions

$$0 \leq \boldsymbol{\lambda}(t) \perp \mathbf{w}(t) - a(t)\mathbb{1} \geq 0, \quad (5.42c)$$

where $a(t)$ is an additional unknown, and where $\mathbb{1}$ denotes the k -vector whose entries are all equal to 1. To explain the meaning of these equations, consider a time t at which the state vector $\mathbf{x}(t)$ is located in the open set \mathcal{H}_i defined by

$$\mathcal{H}_i = \{\mathbf{x} \mid H_i(\mathbf{x}) < H_j(\mathbf{x}), \text{ for all } j \neq i\}. \quad (5.43)$$

To ensure that the i -th component of $\mathbf{w}(t) - a(t)\mathbb{1}$ is nonnegative, we must have $a(t) \leq w_i(t) = H_i(\mathbf{x}(t))$. If the strict inequality $a(t) < w_i(t)$ was valid, then all components of the vector $\mathbf{w}(t) - a(t)\mathbb{1}$ would be strictly positive, which by the complementarity condition would imply that all coefficients of the vector $\boldsymbol{\lambda}$ would be zero. This would contradict the constraint (5.42b). It follows that, for $\mathbf{x}(t) \in \mathcal{H}_i$, we must have $a(t) = w_i(t)$. Since $w_j(t) - a(t) > 0$ for $j \neq i$, the complementarity conditions implies then that $\lambda_j = 0$ for $j \neq i$, and from the constraint (5.42b) it follows that $\lambda_i = 1$. Therefore, we find that for all $i = 1, \dots, k$,

$$\dot{\mathbf{x}}(t) = f_i(\mathbf{x}(t)), \quad \text{if } \mathbf{x}(t) \in \mathcal{H}_i. \quad (5.44)$$

In this way we see that the equations (5.42) describe a multi-regime system with state-dependent switching. Moreover, the equations define a convex relaxation on the boundaries between these regions. Systems of this type have been studied extensively [237].

To write the system in the cone complementarity form (5.23a)–(5.23b)–(5.27), define

$$\mathcal{C} = \mathbb{R}_+^k \times \{0\}, \quad \mathbf{u} = \begin{bmatrix} \boldsymbol{\lambda} \\ a \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} H(\mathbf{x}) - a\mathbb{1} \\ \mathbb{1}^T \boldsymbol{\lambda} - 1 \end{bmatrix}. \quad \square \quad (5.45)$$

Example 5.5 *A Leontiev economy*

A model for a continuous-time Leontiev economy may be constructed as follows. Let $x_i(t)$, $u_i(t)$, and $v_i(t)$ respectively denote the inventory, production rate, and net exogenous demand associated with commodity i at time t . Furthermore, let q_{ij} denote the amount of commodity i required for the production of one unit of commodity j . A balance equation for the evolution of the inventory may then be written in the form

$$\dot{\mathbf{x}}(t) = (\mathbf{I} - \mathbf{Q})\mathbf{u}(t) - \mathbf{v}(t), \quad (5.46a)$$

where \mathbf{Q} is the matrix formed from the elements q_{ij} . It is natural to impose that inventory should be nonnegative, but this is by no means sufficient to determine a solution uniquely.

However, if we furthermore impose that the economy is efficient in the sense that it produces the lowest amounts of commodities that are sufficient to meet demand, then commodities are not produced when there is still a positive inventory, and are otherwise produced in just sufficient amounts to prevent inventory from becoming negative. In other words, the complementarity relation

$$0 \leq \mathbf{x}(t) \perp \mathbf{u}(t) \geq 0 \quad (5.46b)$$

must hold for all t . The system (5.46) is in the form of the forced linear complementarity system (5.26) with $\mathbf{A} = \mathbf{O}$, $\mathbf{B} = \mathbf{I} - \mathbf{Q}$, $\mathbf{C} = \mathbf{I}$, $\mathbf{D} = \mathbf{O}$, $\mathbf{E} = -\mathbf{I}$, and $\mathbf{F} = \mathbf{O}$. \square

Example 5.6 A user-resource model

Many models for network usage can be described in terms of users who have access to several resources. For instance, users may be origin-destination pairs in a traffic network model, and in this case resources are the links between crossings. In the context of production planning, users may be products and resources may be machines. The use of a given resource generates a certain cost for the user, for instance in terms of incurred delay; this cost depends in general on the load that is placed on the resource by all users. A typical purpose of modeling is to describe the behavior of users in determining their demand for services from the resources available to them.

To set up a general model in mathematical terms, suppose that we have p users and m resources. Introduce the following quantities:

- $l_{ij}(t)$ = load per unit of time placed by user i on resource j at time t ;
- $q_{ij}(t)$ = cost incurred at time t by user i when applying to resource j ;
- $d_i(t)$ = total demand of user i at time t ;
- $a_i(t)$ = cost accepted by user i at time t .

The above quantities are summarized in a *load matrix* $\mathbf{L}(t) \in \mathbb{R}_+^{p \times m}$ (load is taken to be nonnegative), a *cost matrix* $\mathbf{Q}(t) \in \mathbb{R}^{p \times m}$, a *demand vector* $\mathbf{d}(t) \in \mathbb{R}^p$, and an *accepted cost vector* $\mathbf{a}(t) \in \mathbb{R}^p$. Moreover we introduce a *state vector* $\mathbf{x}(t) \in \mathbb{R}^n$ in terms of which the dynamics of the system is described, and which moreover determines the cost matrix:

$$\frac{d\mathbf{x}(t)}{dt} = f(\mathbf{x}(t), \mathbf{L}(t)), \quad (5.47a)$$

$$\mathbf{Q}(t) = h(\mathbf{x}(t), \mathbf{L}(t)). \quad (5.47b)$$

To describe the behavior of users, we assume that the Wardrop principle holds at every time instant t . In other words, given a demand level, each user distributes its load over resources in such a way that all resources that are used generate the same cost (this is the accepted cost), and there is no resource that is not used and that would generate a lesser cost. This behavioral principle, together with the nonnegativity of the load, can be expressed in matrix terms by

$$0 \leq \mathbf{L}(t) \perp \mathbf{Q}(t) - \mathbf{a}(t) \cdot \mathbf{1}^T \geq 0, \quad (5.47c)$$

where the “perp” relation is understood in the sense of the inner product $\langle A, B \rangle = \text{tr}(\mathbf{A}^T \mathbf{B})$ for $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{p \times m}$. To close the model, we furthermore need the accounting relation

$$\mathbf{L}(t) \mathbf{1} = \mathbf{d}(t) \quad (5.47d)$$

as well as a “constitutive relation” between the demand and the accepted cost, which we take to be of the form

$$R(\mathbf{d}, \mathbf{a}) = 0, \quad (5.47e)$$

where R is a mapping from $\mathbb{R}^p \times \mathbb{R}^p$ to \mathbb{R}^p . The system (5.47) can be rendered as a cone complementarity system (5.23a) – (5.23b) – (5.27) by means of the identifications

$$\begin{aligned} \mathbf{u} &= (\mathbf{L}, \mathbf{a}), \\ \mathbf{y} &= (h(\mathbf{x}, \mathbf{L}) - \mathbf{a} \cdot \mathbb{1}^T, R(\mathbf{L}\mathbb{1}, \mathbf{a})), \\ \mathcal{C} &= \mathbb{R}_+^{p \times m} \times \{0\} \subset \mathbb{R}^{p \times (m+1)}. \end{aligned} \quad (5.48)$$

As a specific case, consider a situation where the resources consist of m noninteracting queues and the state variables are the queue lengths. Ignoring the situations in which buffers are empty or full (which in fact could be naturally modeled in a complementarity framework), we write simple queue dynamics

$$\frac{dx_j}{dt}(t) = (\mathbb{1}^T \mathbf{L})_j - c_j \quad (5.49)$$

where c_j is a constant that represents the processing speed of queue j . A possible expression for cost is

$$Q_{ij} = k_j x_j + m_{ij},$$

where k_j is a proportionality constant, and the constants m_{ij} represent a fixed cost that may be user-specific. Finally assume that demand is constant, say, $\mathbf{d}(t) = \mathbf{d}_0$ irrespective of the actual cost $\mathbf{a}(t)$. We then arrive at the following dynamical model:

$$\dot{\mathbf{x}}(t) = \mathbf{L}^T(t) \mathbb{1} - \mathbf{c}, \quad (5.50a)$$

$$\mathbf{Q}(t) = \mathbb{1} \cdot (\mathbf{K} \mathbf{x}(t))^T + \mathbf{M}, \quad (5.50b)$$

$$\mathbf{L}(t) \mathbb{1} = \mathbf{d}_0, \quad (5.50c)$$

$$0 \leq \mathbf{L}(t) \perp \mathbf{Q}(t) - \mathbf{a}(t) \cdot \mathbb{1}^T \geq 0. \quad (5.50d)$$

This is a linear (actually affine) cone complementarity system. The constant terms can be treated as external inputs, analogously to (5.26). \square

5.2.4 Preliminaries

For the sake of completeness, we review the linear complementarity problem of mathematical programming and the notion of passivity of systems theory.

Linear complementarity problem Given an m -vector \mathbf{q} and an $m \times m$ matrix \mathbf{M} , the linear complementarity problem $\text{LCP}(\mathbf{q}, \mathbf{M})$ is to find an m -vector \mathbf{z} such that

$$\mathbf{z} \geq 0, \quad (5.51a)$$

$$\mathbf{q} + \mathbf{M}\mathbf{z} \geq 0, \quad (5.51b)$$

$$\mathbf{z}^T (\mathbf{q} + \mathbf{M}\mathbf{z}) = 0. \quad (5.51c)$$

If such a vector \mathbf{z} exists, we say that \mathbf{z} solves (is a solution of) $\text{LCP}(\mathbf{q}, \mathbf{M})$. We say that the $\text{LCP}(\mathbf{q}, \mathbf{M})$ is feasible if there exists \mathbf{z} satisfying (5.51a) and (5.51b).

We define the sets

$$\text{LCP} - \text{Im}(\mathbf{M}) := \{\mathbf{q} \in \mathbb{R}^m \mid \text{LCP}(\mathbf{q}, \mathbf{M}) \text{ admits a solution}\} \quad (5.52)$$

and

$$\text{LCP} - \text{ker}(\mathbf{M}) := \{\mathbf{z} \in \mathbb{R}^m \mid \mathbf{z} \text{ solves } \text{LCP}(0, \mathbf{M})\}. \quad (5.53)$$

The LCP is a well-studied subject of mathematical programming. An excellent survey of the topic can be found in the book [189]. For the sake of completeness, we quote the following two theorems. The first one can be considered as the *fundamental theorem* of LCP theory. It states necessary and sufficient conditions for the unique solvability of the LCP for all vectors \mathbf{q} .

Theorem 5.1 [189] *The LCP(\mathbf{q}, \mathbf{M}) has a unique solution for all \mathbf{q} if and only if all the principal minors of the matrix \mathbf{M} are positive.*

Matrices with the above-mentioned property are known as P-matrices. It is well-known that every positive definite matrix is in this class. Besides positive definite matrices, the nonnegative definite matrices will appear in the LCP context in the sequel. If the \mathbf{M} matrix is nonnegative definite then the LCP does not necessarily have solutions for all vectors \mathbf{q} . For example, the LCP($\mathbf{q}, 0$) admits solutions only if $\mathbf{q} \geq 0$.

The following theorem characterizes the conditions under which an LCP with a nonnegative definite matrix \mathbf{M} has solutions:

Theorem 5.2 [189] *Let \mathbf{M} be a nonnegative definite matrix. Then,*

$$\text{LCP} - \text{Im}(\mathbf{M}) = (\text{LCP} - \text{ker}(\mathbf{M}))^*. \quad (5.54)$$

Linear passive systems Having roots in circuit theory, passivity is a concept that has always played a central role in systems theory. A system is passive if for any time interval the difference between the stored energy at the end of the interval and at the beginning is less than or equal to the supplied energy during the interval.

Definition 5.1 (Passive system) [663] *A linear system $\Sigma(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D})$ given by*

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{z}(t), \quad (5.55a)$$

$$\mathbf{w}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{z}(t) \quad (5.55b)$$

is called passive if there exists a nonnegative function $V : \mathbb{R}^n \rightarrow \mathbb{R}_+$ such that for all $t_0 \leq t_1$ and all trajectories (z, x, w) of the system (5.55) the following inequality holds:

$$V(\mathbf{x}(t_0)) + \int_{t_0}^{t_1} \mathbf{z}^T(t)\mathbf{w}(t) dt \geq V(\mathbf{x}(t_1)). \quad (5.56)$$

If it exists the function V is called a storage function.

Passivity property can be characterized in terms of the state space representation or the transfer matrix of the system as follows.

Proposition 5.1 Consider the following statements:

1. The system $\Sigma(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D})$ is passive.
2. The linear matrix inequalities

$$\mathbf{K} = \mathbf{K}^T \geq 0 \quad \text{and} \quad \begin{bmatrix} \mathbf{A}^T \mathbf{K} + \mathbf{K} \mathbf{A} & \mathbf{K} \mathbf{B} - \mathbf{C}^T \\ \mathbf{B}^T \mathbf{K} - \mathbf{C} & -(\mathbf{D} + \mathbf{D}^T) \end{bmatrix} \leq 0 \quad (5.57)$$

have a solution \mathbf{K} .

3. The function $V(x) = \frac{1}{2} x^T \mathbf{K} x$ defines a storage function.
4. The transfer matrix $\mathbf{G}(s) = \mathbf{D} + \mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1} \mathbf{B}$ is positive real, i.e., $u^*[\mathbf{G}(\lambda) + \mathbf{G}^*(\lambda)]u \geq 0$ for all complex vectors u and all complex numbers λ such that $\text{Re}(\lambda) > 0$ and λ is not an eigenvalue of \mathbf{A} .
5. The triple $(\mathbf{A}, \mathbf{B}, \mathbf{C})$ is minimal.
6. The pair (\mathbf{C}, \mathbf{A}) is observable.
7. The matrix \mathbf{K} is positive definite.

The following implications hold:

- (i). $1 \Leftrightarrow 2 \Leftrightarrow 3$.
- (ii). $2 \Rightarrow 4$.
- (iii). $4 \text{ and } 5 \Rightarrow 2$.
- (iv). $2 \text{ and } 6 \Rightarrow 7$.

5.2.5 Existence and uniqueness of solutions

Consider the system

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{z}(t) + \mathbf{E}\mathbf{u}(t), \quad (5.58a)$$

$$\mathbf{w}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{z}(t) + \mathbf{F}\mathbf{u}(t), \quad (5.58b)$$

$$0 \leq \mathbf{z}(t) \perp \mathbf{w}(t) \geq 0, \quad (5.58c)$$

where the state \mathbf{x} takes values from \mathbb{R}^n , the input \mathbf{u} from \mathbb{R}^k , the complementarity variables (\mathbf{z}, \mathbf{w}) from \mathbb{R}^{m+m} . We call these systems *linear complementarity systems* and denote (5.58) by $\text{LCS}(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}, \mathbf{E}, \mathbf{F})$. When the sextuple $(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}, \mathbf{E}, \mathbf{F})$ is clear from the context, we use only LCS .

We say that a triple $(\mathbf{z}, \mathbf{x}, \mathbf{w})$, where \mathbf{x} is absolutely continuous and (\mathbf{z}, \mathbf{w}) is locally integrable:

- is a *Carathéodory solution* of (5.58) for the initial state \mathbf{x}_0 and the input \mathbf{u} if $\mathbf{x}(0) = \mathbf{x}_0$ and (5.58a) is satisfied for almost all $t \geq 0$ and (5.58b)–(5.58c) are satisfied for all $t \geq 0$;
- is a *forward solution* of (5.58) for the initial state \mathbf{x}_0 and the input \mathbf{u} if $(\mathbf{z}, \mathbf{x}, \mathbf{w})$ is a solution and for each $\bar{t} \geq 0$ there exist an index set $\alpha(\bar{t}) \subseteq \{1, 2, \dots, m\}$, and a positive number $\varepsilon_{\bar{t}}$ such that

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{z}(t) + \mathbf{E}\mathbf{u}(t), \quad (5.59a)$$

$$\mathbf{w}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{z}(t) + \mathbf{F}\mathbf{u}(t), \quad (5.59b)$$

$$\mathbf{z}_{\alpha(\bar{t})}(t) \geq 0 \quad \mathbf{w}_{\alpha(\bar{t})}(t) = 0, \quad (5.59c)$$

$$\mathbf{z}_{\alpha^c(\bar{t})}(t) = 0 \quad \mathbf{w}_{\alpha^c(\bar{t})}(t) \geq 0 \quad (5.59d)$$

holds for all $t \in (\bar{t}, \bar{t} + \varepsilon)$. Here α^c denotes the complement of the set α in $\{1, 2, \dots, m\}$.

Throughout the chapter, we will be mainly interested in Bohl-type inputs. A function $f : \mathbb{R}_+ \rightarrow \mathbb{R}^p$ is said to be a *Bohl function* if $f(t) = \mathbf{Z} \exp(\mathbf{X}t)\mathbf{Y}$ holds for all $t \geq 0$ and for some matrices \mathbf{X} , \mathbf{Y} , and \mathbf{Z} with appropriate sizes.

Existence and uniqueness of forward solutions The following theorem provides sufficient conditions for the existence and uniqueness of forward solutions:

Theorem 5.3 [153] *Let $\mathbf{G}(s) = \mathbf{D} + \mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}$. Suppose that*

- *for all $\pi \subseteq \{1, 2, \dots, m\}$, $\mathbf{G}_{\pi\pi}(s)$ is invertible as a rational matrix and $s^{-1}\mathbf{G}^{-1}_{\pi\pi}(s)$ is proper; and*
- *$\mathbf{G}(\sigma)$ is a P -matrix for all sufficiently large real numbers σ .*

Then, the following statements are equivalent:

1. *There exists a forward solution of the LCS (5.58) for the initial state \mathbf{x}_0 and the Bohl input \mathbf{u} .*
2. *$\mathbf{C}\mathbf{x}_0 + \mathbf{F}\mathbf{u}(0) \in \text{LCP} - \text{Im}(\mathbf{D})$.*

Moreover, if a forward solution exists it is unique.

Existence and uniqueness of Carathéodory solutions Theorem 5.3 presents conditions for the existence and uniqueness of forward solutions. However, the uniqueness of Carathéodory solutions is not guaranteed by those conditions in general as illustrated by the following example.

Example 5.7 *Complementarity system with multiple solutions*

The LCS $(\mathbf{A}, \mathbf{B}, \mathbf{C}, 0, 0, 0)$ with

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 1 & 3 & 0 \\ 0 & 1 & 3 \\ 3 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

has multiple Carathéodory solutions for the initial state $\mathbf{x}_0 = \text{col}(0, 0, 0, 1)$ [78, 584]. Note that $\mathbf{C}\mathbf{B}$ is a P -matrix and hence all the conditions of Theorem 5.3 are satisfied. Consequently, there exists a unique forward solution. \square

The following theorem provides conditions for uniqueness of Carathéodory solutions. It follows from the standard existence and uniqueness results of ordinary differential equations with Lipschitzian right-hand sides.

Theorem 5.4 Suppose that D is a P -matrix. Then, the following statements are equivalent:

1. There exists a Carathéodory solution of the LCS (5.58) for any initial state x_0 and any locally integrable input u .
2. There exists a forward solution of the LCS (5.58) for any initial state x_0 and any locally integrable input u .

Moreover, if (z^i, x^i, w^i) $i = 1, 2$ are solutions with the initial state x_0 , and the input u , then $(z^1, x^1, w^1) = (z^2, x^2, w^2)$.

The P -matrix condition of this theorem is somewhat restrictive. It turns out that passivity of the underlying linear system is sufficient in order to guarantee uniqueness of Carathéodory solutions as stated next.

Theorem 5.5 [161] Suppose that the system $\Sigma(A, B, C, D)$ is passive and the LMIs (5.57) have a positive definite solution. Then, the following statements are equivalent for a given positive real number T , an initial state x_0 , and an input u :

1. There exists a Carathéodory solution of the LCS (5.58) for the initial state x_0 and the Bohl input u .
2. There exists a forward solution of the LCS (5.58) for the initial state x_0 and the Bohl input u .
3. The relations

$$F u(t) \in (\text{LCP} - \ker(D))^* + \text{Im } C, \quad \text{for all } t \geq 0, \quad (5.60a)$$

$$C x_0 + F u(0) \in (\text{LCP} - \ker(D))^* \quad (5.60b)$$

hold.

Moreover, if (z^i, x^i, w^i) $i = 1, 2$ are solutions with the initial state x_0 , and the input u , then the relations

1. $x^1 - x^2 = 0$;
2. $z^1 - z^2 \in \ker \begin{bmatrix} B \\ D + D^T \end{bmatrix}$;
3. $w^1 - w^2 \in D \ker \begin{bmatrix} B \\ D + D^T \end{bmatrix}$

hold.

Zeno phenomena Consider the input-free LCS

$$\dot{x}(t) = Ax(t) + Bz(t), \quad (5.61a)$$

$$w(t) = Cx(t) + Dz(t), \quad (5.61b)$$

$$0 \leq z(t) \perp w(t) \geq 0. \quad (5.61c)$$

Let (z, x, w) be a Carathéodory solution of the LCS (5.61). Define the index sets

$$\alpha(t) = \{i \mid z_i(t) > 0 = w_i(t)\}, \quad (5.62)$$

$$\beta(t) = \{i \mid z_i(t) = 0 = w_i(t)\}, \quad (5.63)$$

$$\gamma(t) = \{i \mid z_i(t) = 0 < w_i(t)\}, \quad (5.64)$$

for $t \geq 0$. We say that a time instant $t^* > 0$ is:

- a *nonswitching time instant* with respect to the solution (z, x, w) if there exist a positive real number ε and index sets $(\alpha_*, \beta_*, \gamma_*)$ such that $(\alpha(t), \beta(t), \gamma(t)) = (\alpha_*, \beta_*, \gamma_*)$ for all $t \in (t^* - \varepsilon, t^*) \cup (t^*, t^* + \varepsilon)$;
- a *switching time instant* if it is not a nonswitching time instant.

Let Γ be the set of all switching time instants with respect to the solution (z, x, w) . We say that the solution (z, x, w) is

- *left-Zeno free* if the set Γ has no left accumulation points, i.e. for each $t \geq 0$ there exists a positive real number ε such that $\Gamma \cap (t, t + \varepsilon) = \emptyset$;
- *right-Zeno free* if the set Γ has no right accumulation points, i.e. for each $t > 0$ there exists a positive real number ε such that $\Gamma \cap (t - \varepsilon, t) = \emptyset$;
- *Zeno free* if it is both left- and right-Zeno free.

Four theorems that provide sufficient conditions that exclude certain types of Zeno behavior are in order. The first one rules out both left and right Zeno behavior under a restrictive condition:

Theorem 5.6 [583] *Suppose that D is a P -matrix. Then, all solutions of the LCS (5.61) are Zeno free.*

The second rules out only left-Zenoness under a less restrictive condition, namely the passivity assumption:

Theorem 5.7 [306] *Suppose that the system $\Sigma(A, B, C, D)$ is passive and $\text{col}(B, D + D^T)$ is of full column rank. Then, all solutions of the LCS (5.61) are left-Zeno free.*

The third result rules out Zeno behavior in case the underlying system is passive and D matrix satisfies certain conditions:

Theorem 5.8 [149, 152] *Suppose that the system $\Sigma(A, B, C, D)$ is passive and $\text{col}(B, D + D^T)$ is of full column rank. If there exists an index set $\alpha \subseteq \{1, 2, \dots, m\}$ such that*

- $D_{\alpha\alpha}$ is positive definite;
- $D_{\alpha\alpha^c} = O$ and $D_{\alpha^c\alpha} = O$; and
- $D_{\alpha^c\alpha^c}$ is skew-symmetric.

Then all solutions of the LCS (5.61) are Zeno free.

The final result relaxes the passivity requirement:

Theorem 5.9 [162] *Suppose that $m = 1$, $D = O$, and $CB > 0$. Then, all solutions of the LCS (5.61) are Zero free.*

Stability To study stability, we introduce a stronger version of passivity:

Definition 5.2 (Strictly passive system) *The system $\Sigma(A, B, C, D)$ is called strictly passive, if the matrix inequalities*

$$K = K^T > 0 \text{ and } \begin{bmatrix} A^T K + K A + \varepsilon K & K B - C^T \\ B^T K - C & -(D + D^T) \end{bmatrix} \leq 0 \quad (5.65)$$

have a solution K for some $\varepsilon > 0$.

Lyapunov stability of linear complementarity systems is established by the following theorem under the passivity assumption:

Theorem 5.10 [155] *Consider the LCS (5.61). Suppose that the linear system $\Sigma(A, B, C, D)$ is strictly passive. Then the LCS (5.61) is globally exponentially stable. In case $\Sigma(A, B, C, D)$ is passive only, then the system is Lyapunov stable.*

In general, obtaining necessary and sufficient conditions for stability is a hard task. Only in the planar case, one can provide such conditions as stated in the next theorem:

Theorem 5.11 [156] *Consider the LCS (5.61) with $m = 1$, $n = 2$, and (C, A) is an observable pair. The following statements hold:*

1. *Suppose that $D \succ 0$. The LCS (5.61) is asymptotically stable if and only if*
 - (a) *neither A nor $A - BD^{-1}C$ has a real nonnegative eigenvalue, and*
 - (b) *if both A and $A - BD^{-1}C$ have nonreal eigenvalues then $\sigma_1/\omega_1 + \sigma_2/\omega_2 < 0$ where $\sigma_1 \pm i\omega_1$ ($\omega_1 > 0$) are the eigenvalues of A and $\sigma_2 \pm i\omega_2$ ($\omega_2 > 0$) are the eigenvalues of $A - BD^{-1}C$.*
2. *Suppose that $D > 0$. The LCS (5.61) has a nonconstant periodic solution if and only if both A and $A - BD^{-1}C$ have nonreal eigenvalues, and $\sigma_1/\omega_1 + \sigma_2/\omega_2 = 0$ where $\sigma_1 \pm i\omega_1$ are the eigenvalues of A and $\sigma_2 \pm i\omega_2$ are the eigenvalues of $A - BD^{-1}C$. Moreover, if there is one periodic solution, then all other solutions are also periodic. And, $\pi/\omega_1 + \pi/\omega_2$ is the period of any solution.*
3. *Suppose that $D = O$ and $CB > 0$. The LCS (5.61) is asymptotically stable if and only if A has no real nonnegative eigenvalue and $[I - B(CB)^{-1}C]A$ has a real negative eigenvalue (note that one eigenvalue is already zero).*

Controllability and stabilizability Let $(z^{x_0, u}, x^{x_0, u}, w^{x_0, u})$ denote the solution of the LCS (5.58) for the initial state x_0 and the input u . We say that the LCS (5.58) is

- *controllable* if for any pair of states $(x_0, x_f) \in \mathbb{R}^{n+n}$ there exists a locally integrable input u such that the trajectory $x^{x_0, u}$ satisfies $x^{x_0, u}(T) = x_f$ for some $T > 0$;
- *stabilizable* if for any initial state x_0 there exists a locally integrable input u such that $\lim_{t \uparrow \infty} x^{x_0, u} = 0$.

The following theorem presents algebraic necessary and sufficient conditions for the controllability of an LCS:

Theorem 5.12 [150] *Suppose that D is a P -matrix and the transfer matrix $F + C(sI - A)^{-1}E$ is invertible as a rational matrix. Then, the LCS (5.58) is controllable if, and only if, the following two conditions hold:*

1. The pair $(A, [B \ E])$ is controllable.
2. The system of inequalities

$$\eta \geq 0, \quad (5.66a)$$

$$[\xi^T \ \eta^T] \begin{bmatrix} A - \lambda I & E \\ C & F \end{bmatrix} = 0, \quad (5.66b)$$

$$[\xi^T \ \eta^T] \begin{bmatrix} B \\ D \end{bmatrix} \leq 0 \quad (5.66c)$$

admits no solution $\lambda \in \mathbb{R}$ and $0 \neq (\xi, \eta) \in \mathbb{R}^{n+m}$.

It turns out that stabilizability can also be characterized in the same way:

Theorem 5.13 *Suppose that D is a P -matrix and the transfer matrix $F + C(sI - A)^{-1}E$ is invertible as a rational matrix. Then, the LCS (5.58) is stabilizable if, and only if, the following two conditions hold:*

1. The pair $(A, [B \ E])$ is stabilizable.
2. The system of inequalities

$$\eta \geq 0, \quad (5.67a)$$

$$[\xi^T \ \eta^T] \begin{bmatrix} A - \lambda I & E \\ C & F \end{bmatrix} = 0, \quad (5.67b)$$

$$[\xi^T \ \eta^T] \begin{bmatrix} B \\ D \end{bmatrix} \leq 0 \quad (5.67c)$$

admits no solution $0 \leq \lambda \in \mathbb{R}$ and $0 \neq (\xi, \eta) \in \mathbb{R}^{n+m}$.

5.3 Equivalence of piecewise affine systems, mixed logical dynamical systems, and linear complementarity systems

In this section we discuss equivalences among five classes of discrete-time hybrid systems, viz. mixed logical dynamical (MLD) systems, linear complementarity (LC) systems, extended linear complementarity (ELC) systems, piecewise affine (PWA) systems, and max-min-plus-scaling (MMPS) systems. Some of the equivalences can be established under (rather mild) additional assumptions. These results are of paramount importance for transferring theoretical properties and tools from one class to another, with the consequence that for the study of a particular hybrid system that belongs to any of these classes, one can choose the most convenient hybrid modeling framework. The proofs of all the equivalence results reported in this section can be found in [305].

5.3.1 Summary of the five classes of hybrid models

In the previous chapters of this handbook it has already been indicated that, as tractable methods to analyze general hybrid systems are not available, several authors have focussed on special subclasses of hybrid dynamical systems for which analysis and/or control design techniques are currently being developed. Some examples of such subclasses are: linear complementarity (LC) systems, mixed logical dynamical (MLD) systems, first-order linear hybrid systems with saturation, and piecewise affine (PWA) systems. Each subclass has its own advantages over the others. For instance, stability criteria were proposed for PWA systems (Section 4.4), control and verification techniques for MLD hybrid models (Section 5.1), and conditions of existence and uniqueness of solution trajectories (well-posedness) for LC systems (Section 5.4).

In this section we will show that several of such subclasses of hybrid systems are equivalent when considered in their discrete-time formulation. Some of the equivalences are obtained under additional assumptions related to well-posedness and boundedness of input, state, output, or auxiliary variables. These results allow to transfer all the above analysis and synthesis tools to any of the equivalent subclasses of hybrid systems.

First we briefly recapitulate the five classes of hybrid systems considered in this section. The variables $\mathbf{u}(k) \in \mathbb{R}^m$, $\mathbf{x}(k) \in \mathbb{R}^n$ and $\mathbf{y}(k) \in \mathbb{R}^l$ denote the input, state and output, respectively, at time k .

Piecewise affine (PWA) systems PWA systems are described by

$$\begin{aligned}
 \mathbf{x}(k+1) &= \mathbf{A}_i \mathbf{x}(k) + \mathbf{B}_i \mathbf{u}(k) + \mathbf{f}_i \\
 \mathbf{y}(k) &= \mathbf{C}_i \mathbf{x}(k) + \mathbf{D}_i \mathbf{u}(k) + \mathbf{g}_i
 \end{aligned}
 \quad \text{for } \begin{bmatrix} \mathbf{x}(k) \\ \mathbf{u}(k) \end{bmatrix} \in \Omega_i, \quad (5.68)$$

where Ω_i are convex polyhedra (i.e. given by a finite number of linear inequalities) in the input/state space. PWA systems form the “simplest” extension of linear systems that can still model nonlinear and non-smooth processes with arbitrary accuracy and are capable of handling hybrid phenomena.

Mixed logical dynamical (MLD) systems As introduced in Section 5.1.3, an integration of logic, dynamics, and constraints results in the description

$$\mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{B}_1\mathbf{u}(k) + \mathbf{B}_2\boldsymbol{\delta}(k) + \mathbf{B}_3\mathbf{z}(k), \quad (5.69)$$

$$\mathbf{y}(k) = \mathbf{C}\mathbf{x}(k) + \mathbf{D}_1\mathbf{u}(k) + \mathbf{D}_2\boldsymbol{\delta}(k) + \mathbf{D}_3\mathbf{z}(k), \quad (5.70)$$

$$\mathbf{E}_2\boldsymbol{\delta}(k) + \mathbf{E}_3\mathbf{z}(k) \leq \mathbf{E}_1\mathbf{u}(k) + \mathbf{E}_4\mathbf{x}(k) + \mathbf{E}_5, \quad (5.71)$$

where $\mathbf{x}(k) = [\mathbf{x}_c^T(k) \ \mathbf{x}_b^T(k)]^T$ with $\mathbf{x}_c(k) \in \mathbb{R}^{n_c}$ and $\mathbf{x}_b(k) \in \{0, 1\}^{n_b}$. $\mathbf{z}(k) \in \mathbb{R}^{r_c}$ and $\boldsymbol{\delta}(k) \in \{0, 1\}^{r_b}$ are auxiliary variables. The inequalities (5.71) have to be interpreted componentwise.

Remark 5.1 It is assumed that for all $\mathbf{x}(k)$ with $\mathbf{x}_b(k) \in \{0, 1\}^{n_b}$, all $\mathbf{u}(k)$ with $\mathbf{u}_b(k) \in \{0, 1\}^{m_b}$, all $\mathbf{z}(k) \in \mathbb{R}^{r_c}$ and all $\boldsymbol{\delta}(k) \in \{0, 1\}^{r_b}$ satisfying (5.71) it holds that $\mathbf{x}(k+1)$ and $\mathbf{y}(k)$ determined from (5.69)–(5.70) are such that $\mathbf{x}_b(k+1) \in \{0, 1\}^{n_b}$ and $\mathbf{y}_b(k) \in \{0, 1\}^{l_b}$. This is without loss of generality, as we can take binary components of states and outputs (if any) to be auxiliary variables as well (see the proof of Proposition 1 of [65]). Indeed, if, for instance, $\mathbf{y}_b(k) \in \{0, 1\}^{l_b}$ is not directly implied by the (in)equalities, we introduce an additional binary vector variable $\boldsymbol{\delta}_y(k) \in \{0, 1\}^{l_b}$ and the inequalities

$$\begin{aligned} [\mathbf{C}\mathbf{x}(k) + \mathbf{D}_1\mathbf{u}(k) + \mathbf{D}_2\boldsymbol{\delta}(k) + \mathbf{D}_3\mathbf{z}(k)]_b - \boldsymbol{\delta}_y(k) &\leq 0, \\ [-\mathbf{C}\mathbf{x}(k) - \mathbf{D}_1\mathbf{u}(k) - \mathbf{D}_2\boldsymbol{\delta}(k) - \mathbf{D}_3\mathbf{z}(k)]_b + \boldsymbol{\delta}_y(k) &\leq 0, \end{aligned}$$

which sets $\boldsymbol{\delta}_y(k)$ equal to $\mathbf{y}_b(k)$. The notation $[\]_b$ is used to select the rows of the expression (5.70) that correspond to the binary part of $\mathbf{y}(k)$. Hence, $\mathbf{y}_b(k) = \boldsymbol{\delta}_y(k) \in \{0, 1\}^{l_b}$. Similarly, we can deal with $\mathbf{u}_b(k)$ and $\mathbf{x}_b(k+1)$.

Linear complementarity (LC) systems In discrete time these systems are given by the equations

$$\mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{B}_1\mathbf{u}(k) + \mathbf{B}_2\mathbf{w}(k), \quad (5.72)$$

$$\mathbf{y}(k) = \mathbf{C}\mathbf{x}(k) + \mathbf{D}_1\mathbf{u}(k) + \mathbf{D}_2\mathbf{w}(k), \quad (5.73)$$

$$\mathbf{v}(k) = \mathbf{E}_1\mathbf{x}(k) + \mathbf{E}_2\mathbf{u}(k) + \mathbf{E}_3\mathbf{w}(k) + \mathbf{g}_4, \quad (5.74)$$

$$\mathbf{0} \leq \mathbf{v}(k) \perp \mathbf{w}(k) \geq \mathbf{0}, \quad (5.75)$$

with $\mathbf{v}(k), \mathbf{w}(k) \in \mathbb{R}^s$ and where \perp denotes the orthogonality of vectors (i.e. $\mathbf{v}(k) \perp \mathbf{w}(k)$ means that $\mathbf{v}^T(k)\mathbf{w}(k) = 0$). We call $\mathbf{v}(k)$ and $\mathbf{w}(k)$ the complementarity variables.

Extended linear complementarity (ELC) systems Several types of hybrid systems can be modeled as extended linear complementarity (ELC) systems:

$$\mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{B}_1\mathbf{u}(k) + \mathbf{B}_2\mathbf{d}(k), \quad (5.76)$$

$$\mathbf{y}(k) = \mathbf{C}\mathbf{x}(k) + \mathbf{D}_1\mathbf{u}(k) + \mathbf{D}_2\mathbf{d}(k), \quad (5.77)$$

$$\mathbf{E}_1\mathbf{x}(k) + \mathbf{E}_2\mathbf{u}(k) + \mathbf{E}_3\mathbf{d}(k) \leq \mathbf{g}_4, \quad (5.78)$$

$$\sum_{i=1}^p \prod_{j \in \phi_i} (\mathbf{g}_4 - \mathbf{E}_1\mathbf{x}(k) - \mathbf{E}_2\mathbf{u}(k) - \mathbf{E}_3\mathbf{d}(k))_j = 0, \quad (5.79)$$

where $\mathbf{d}(k) \in \mathbb{R}^r$ is an auxiliary variable. Condition (5.79) is equivalent to

$$\prod_{j \in \phi_i} (\mathbf{g}_4 - \mathbf{E}_1 \mathbf{x}(k) - \mathbf{E}_2 \mathbf{u}(k) - \mathbf{E}_3 \mathbf{d}(k))_j = 0 \quad \text{for each } i \in \{1, 2, \dots, p\}, \quad (5.80)$$

due to the inequality conditions (5.78). This implies that (5.78)–(5.79) can be considered as a system of linear inequalities (i.e. (5.78)), where there are p groups of linear inequalities (one group for each index set ϕ_i) such that in each group at least one inequality should hold with equality.

Max-min-plus-scaling (MMPS) systems In [578] a class of discrete-event systems has been introduced that can be modeled using the operations maximization, minimization, addition, and scalar multiplication. Expressions that are built using these operations are called max-min-plus-scaling (MMPS) expressions.

Definition 5.3 (Max-min-plus-scaling expression) A max-min-plus-scaling expression f of the variables x_1, \dots, x_n is defined by the grammar

$$f := x_i | \alpha | \max(f_k, f_l) | \min(f_k, f_l) | f_k + f_l | \beta f_k, \quad (5.81)$$

with $i \in \{1, 2, \dots, n\}$, $\alpha, \beta \in \mathbb{R}$, and where f_k, f_l are again MMPS expressions. (The symbol $|$ stands for OR and the definition is recursive.)

An MMPS expression is, for example

$$5x_1 - 3x_2 + 7 + \max(\min(2x_1, -8x_2), x_2 - 3x_3).$$

Consider now systems that can be described by

$$\mathbf{x}(k+1) = \mathcal{M}_x(\mathbf{x}(k), \mathbf{u}(k), \mathbf{d}(k)), \quad (5.82)$$

$$\mathbf{y}(k) = \mathcal{M}_y(\mathbf{x}(k), \mathbf{u}(k), \mathbf{d}(k)), \quad (5.83)$$

$$\mathcal{M}_c(\mathbf{x}(k), \mathbf{u}(k), \mathbf{d}(k)) \leq \mathbf{c}, \quad (5.84)$$

where $\mathcal{M}_x, \mathcal{M}_y$, and \mathcal{M}_c are MMPS expressions in terms of the components of $\mathbf{x}(k)$, the input $\mathbf{u}(k)$, and the auxiliary variables $\mathbf{d}(k)$, which are all real-valued. Such systems will be called MMPS systems.

5.3.2 Systems equivalence

In this section we prove that MLD, LC, ELC, PWA and MMPS systems are equivalent (although in some cases additional assumptions are required). The relations between the models are depicted in Fig. 5.2.

MLD and LC systems

Proposition 5.2 Every MLD system can be written as an LC system.

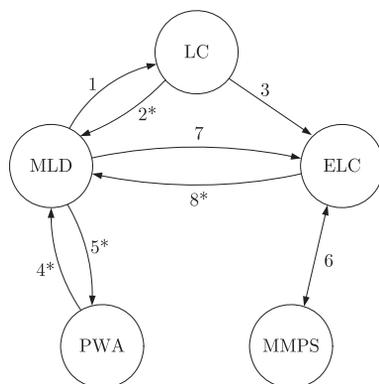


Fig. 5.2 Graphical representation of the links among the classes of hybrid systems considered in this paper. An arrow going from class A to class B means that A is a subset of B. The number next to each arrow corresponds to the proposition that states this relation. Moreover, arrows with a star (*) require conditions to establish the indicated inclusion.

As mentioned before, all proofs of the equivalence results presented here can be found in [305].

Proposition 5.3 *Every LC system can be written as an MLD system, provided that the variables $w(k)$ and $v(k)$ are (componentwise) bounded.*

Proposition 5.3 assumes that upper bounds on w and v are known. This hypothesis is not restrictive in practice, as these quantities are related to continuous inputs and states of the system, which are usually bounded for physical reasons.

LC and ELC systems

Proposition 5.4 *Every LC system can be written as an ELC system.*

PWA and MLD systems

A PWA system of the form (5.68) is called well-posed, if (5.68) is uniquely solvable in $x(k+1)$ and $y(k)$, once $x(k)$ and $u(k)$ are specified. The following proposition has been stated in [65] and is an easy extension of the corresponding result in [62] for piecewise linear (PWL) systems (i.e. PWA systems with $f_i = g_i = 0$):

Proposition 5.5 *Every well-posed PWA system can be rewritten as an MLD system assuming that the set of feasible states and inputs is bounded.*

Remark 5.2 As MLD models only allow for nonstrict inequalities in (5.71), in rewriting a discontinuous PWA system as an MLD model strict inequalities like $x(k) < 0$ (where assume here for the sake of simplicity of the exposition and without

loss of generality that $x(k)$ is a scalar) must be approximated by $x(k) \leq -\varepsilon$ for some $\varepsilon > 0$ (typically the machine precision), with the assumption that $-\varepsilon < x(k) < 0$ cannot occur due to the finite number of bits used for representing real numbers (no problem exists when the PWA system is continuous, where the strict inequality can be equivalently rewritten as nonstrict, or $\varepsilon = 0$). See [62] for more details and 5.8 for an example. From a strictly theoretical point of view, the inclusion stated in Proposition 5.5 is therefore not exact for discontinuous PWA systems, and the same clearly holds for an LC, ELC or MMPS reformulation of a discontinuous PWA system when the route via MLD is taken. One way to circumvent such an inexactness is to allow part of the inequalities in (5.71) to be strict. On the other hand, from a numerical point of view this issue is not relevant. The equivalence of LC and MLD systems (cf. Proposition 5.2 and 5.3) implies that all continuous PWA can be exactly written as LC systems as well. A similar result for continuous PWA systems can be derived from [217].

The MLD system (5.69) is called completely well-posed, if $\mathbf{x}(k+1)$, $\mathbf{y}(k)$, $\delta(k)$ and $\mathbf{z}(k)$ are uniquely defined in their domain, once $\mathbf{x}(k)$ and $\mathbf{u}(k)$ are assigned [62]. The reverse statement of Proposition 5.5 has been established in [65] under the condition that the MLD system is completely well-posed:

Proposition 5.6 *A completely well-posed MLD system can be rewritten as a PWA system.*

Constructive procedures for converting MLD systems into PWA form were provided in [55, 56] (and implemented in the Hybrid Toolbox [57], see Chapter 10) and in [265]. Equivalences between PWA systems and other hybrid model classes have been also investigated in [136], where the authors examine a relationship existing among linear hybrid automata (LHA) and piecewise affine (PWA) systems, showing in a constructive way that a LHA can be equivalently represented as a continuous-time PWA system.

MMPS and ELC systems

Proposition 5.7 *The classes of MMPS and ELC systems coincide.*

MLD and ELC systems

Proposition 5.8 *Every MLD system can be rewritten as an ELC system.*

Remark 5.3 Note that the condition $\delta_i(k) \in \{0, 1\}$ is also equivalent to the MMPS constraint $\max(-\delta_i(k), \delta_i(k) - 1) = 0$ or $\min(\delta_i(k), 1 - \delta_i(k)) = 0$.

Proposition 5.9 *Every ELC system can be written as an MLD system, provided that the quantity $\mathbf{g}_4 - \mathbf{E}_1 \mathbf{x}(k) - \mathbf{E}_2 \mathbf{u}(k) - \mathbf{E}_3 \mathbf{d}(k)$ is (componentwise) bounded.*

Note that (just as for Proposition 5.3) the boundedness hypothesis in Proposition 5.9 is not restrictive in practice, since the inputs and states of the system are usually bounded for physical reasons.

Example 5.8 *Equivalent hybrid systems*

To demonstrate the equivalences proven above, we consider the example [62]

$$x(k+1) = \begin{cases} 0.8x(k) + u(k), & \text{if } x(k) \geq 0, \\ -0.8x(k) + u(k), & \text{if } x(k) < 0, \end{cases} \quad (5.85)$$

with $m \leq x(k) \leq M$. In [62] it is shown that (5.85) can be written as

$$\begin{aligned} x(k+1) &= -0.8x(k) + u(k) + 1.6z(k), \\ -m\delta(k) &\leq x(k) - m, & x(k) &\leq (M + \varepsilon)\delta(k) - \varepsilon, \\ z(k) &\leq M\delta(k), & z(k) &\geq m\delta(k), \\ z(k) &\leq x(k) - m(1 - \delta(k)), & z(k) &\geq x(k) - M(1 - \delta(k)), \end{aligned} \quad (5.86)$$

and the condition $\delta(k) \in \{0, 1\}$. Note that the strict inequality $x(k) < 0$ has been replaced by $x(k) \leq -\varepsilon$, where $\varepsilon > 0$ is a small number (typically the machine precision). In view of Remark 5.2 observe that $\varepsilon = 0$ results in a mathematically exact MLD model. In this case the model is well-posed, but not completely well-posed as $x(k) = 0$ allows both $\delta(k) = 0$ and $\delta(k) = 1$. (An MLD model is called well-posed, if $x(k+1)$ and $y(k)$ are uniquely determined, once $x(k)$ and $u(k)$ are given. Note that there are no requirements on $\delta(k)$ and $z(k)$.)

One can verify that (5.85) can be rewritten as the MMPS model

$$x(k+1) = -0.8x(k) + 1.6 \max(0, x(k)) + u(k), \quad (5.87)$$

as the LC formulation

$$x(k+1) = -0.8x(k) + u(k) + 1.6z(k), \quad (5.88)$$

$$0 \leq w(k) = -x(k) + z(k) \perp z(k) \geq 0, \quad (5.89)$$

and as the ELC representation

$$x(k+1) = -0.8x(k) + u(k) + 1.6d(k),$$

$$-d(k) \leq 0,$$

$$x(k) - d(k) \leq 0,$$

$$0 = (x(k) - d(k))(-d(k)).$$

While the MLD representation (5.86) requires bounds on $x(k)$, $u(k)$ to be specified (although such bounds can be arbitrarily large), the PWA, MMPS, LC, and ELC expressions do not require such a specification.

Note that we only need one max-operator in (5.87) and one complementarity pair in (5.88)–(5.89). If we would transform the MLD system (5.86) into, e.g., the LC model as indicated by the equivalence proof, this would require nine complementarity pairs. Hence, it is clear that the proofs only show the conceptual equivalence, but do not result in the most compact models. \square

Outlook In this section we have discussed the equivalence of five classes of discrete-time hybrid systems: MLD, LC, ELC, PWA, and MMPS systems. For some of the transformations additional conditions like boundedness of the state and input variables or well-posedness had to be made. These results allow one to transfer

properties and tools from one class to another. So for the study of a particular hybrid system that belongs to any of these classes, one can choose the most convenient modeling framework.

In the continuous-time framework, which is the natural habitat for most of the applications for LC systems, such broad equivalence relations are out of the question. There are relations though of LC systems to other specific classes of nonsmooth systems such as specific differential inclusions based on the normal cones of convex analysis and so-called projected dynamical systems. The reader may consult [124, 303] for these relationships.

5.4 Solution concepts and well-posedness

This section considers the fundamental system-theoretic property of well-posedness for hybrid dynamical systems. We intend to provide an overview on the available results on existence and uniqueness of solutions for given initial conditions in the context of various description formats for hybrid systems and their corresponding solution concepts.

5.4.1 Problem statement

On an abstract level, scientific modeling may be defined as the process of finding common descriptions for groups of observed phenomena. Often, several description forms are possible.

Example 5.9 *Flying ball*

To take an example from not very recent technology, suppose we want to describe the flight of iron balls fired from a cannon. One description can be obtained by noting that such balls approximately follow parabolas, which may be parametrized in terms of firing angle, cannon ball weight, and amount of gun powder used. Another possible description characterizes the trajectories of the cannon balls as solutions of certain differential equations. The latter description may be viewed as being fairly *indirect*; after all it represents trajectories only as solutions to some problem, rather than expressing directly what the trajectories are, as the first description form does. On the other hand, the description by means of differential equations is applicable to a wider range of phenomena, and one may, therefore, feel that it represents a deeper insight. Besides, interconnection (composition) becomes much easier since it is in general much easier to write down equations than to determine the solutions of the interconnected system. □

There are many examples in science where, as above, an implicit description (that is, a description in terms of a mathematical problem to be solved) is useful and possibly more powerful than explicit descriptions. Whenever an implicit description is used, however, one has to show that the description is a “good” one in the sense that the stated problem has a well-defined solution. This is essentially the issue of well-posedness.

Many different description formats have been proposed in recent years for hybrid systems. Some proposed forms are quite direct, others lead to rather indirect descriptions. The direct forms have advantages from the point of view of *analysis*, but the indirect forms are often preferable from the perspective of *modeling* (specification); examples will be seen below. The more indirect a description form is, the harder it becomes to show that solutions are well-defined. This section intends to provide a survey on the available results on existence and uniqueness of solutions for given initial conditions in the context of the description formats for hybrid systems as considered in this handbook.

5.4.2 Model classes

This section summarizes the models of hybrid systems that will be investigated later with respect to the existence and well-posedness of a solution.

Hybrid automata Hybrid automata were already defined in [Section 1.2](#) and [Section 2.1](#) and we refer to the formal definition of this model class based on the 8-tuple $H = (\mathcal{Q}, \mathcal{X}, \mathbf{f}, \text{Init}, \text{Inv}, \mathcal{E}, \mathcal{G}, \mathcal{R})$ given there.

Differential equations with discontinuous right-hand sides During the past decades, extensive studies have been made of *differential equations with discontinuous right-hand sides* (cf. in particular [237] and [639, 640]). For a typical example, consider the following specification:

$$\dot{x} = \begin{cases} f_1(x), & \text{when } h(x) > 0, \\ f_2(x), & \text{when } h(x) < 0, \end{cases} \quad (5.90)$$

where h is a real-valued function. A system of this form can be looked at either as a discontinuous dynamical system or as a hybrid system of a particular form. The specification above is obviously incomplete since no statement is made about the situation in which $h(x) = 0$. One way to arrive at a solution concept is to adopt a suitable *relaxation*. Specifically, Filippov [237] proposed rewriting the equations in a *convex relaxation* (5.90) as

$$\dot{x} \in \mathcal{F}(x), \quad (5.91)$$

where the set-valued function $\mathcal{F}(x)$ is defined by

$$F(x) = \begin{cases} \{f_1(x)\}, & \text{when } h(x) > 0, \\ \{f_2(x)\}, & \text{when } h(x) < 0, \\ \{y \mid \exists a \in [0, 1] \text{ s. t. } y = af_1(x) + (1-a)f_2(x)\}, & \text{when } h(x) = 0, \end{cases} \quad (5.92)$$

where it is assumed (for simplicity) that f_1 and f_2 are given as continuous functions defined on $\{x \mid h(x) \geq 0\}$ and $\{x \mid h(x) \leq 0\}$, respectively.

The discontinuous dynamical system has now been reformulated as a *differential inclusion*, and so solution concepts and well-posedness results can be applied that

have been developed for systems of this type [26]. Other methods to obtain differential inclusions are proposed by Utkin (“control equivalent definition”) and Aizerman and Pyatnitskii (Sect. 5.4.4). In case the vector fields $f_i(x)$ are linear (i.e. of the form $A_i x$ for some matrix A_i) and the switching surface is given by a linear function h , then the system (5.90) is called a *piecewise linear system*. These systems will receive special attention below.

Hybrid inclusions A conceptually simple model, but still powerful to model many classes of interest, was developed recently in [133, 271, 273]. It extends the differential inclusion (5.91) by restricting its “flow region” to a set \mathcal{C} and including resets of the state variable in the “jump set” \mathcal{D} . As such, the model consists of the data of two subsets \mathcal{C} and \mathcal{D} of \mathbb{R}^n , and two set-valued mappings \mathcal{F} and \mathcal{G} , from \mathcal{C} , respectively from \mathcal{D} , to \mathbb{R}^n . The hybrid system is written as

$$\dot{x} \in \mathcal{F}(x) \text{ if } x \in \mathcal{C}, \tag{5.93a}$$

$$x^+ \in \mathcal{G}(x) \text{ if } x \in \mathcal{D}, \tag{5.93b}$$

The state variable is now given by $x(t) \in \mathbb{R}^n$ for time $t \in \mathbb{R}$, but some parts of the state vector are also allowed to take only integer values.

Complementarity systems Complementarity systems have been discussed already in detail in Section 5.2. The reader is referred to that section for an exposition on this class of hybrid systems.

5.4.3 Solution concepts

A description format for a class of dynamical systems only specifies a collection of trajectories if one provides a notion of solution. Actually the term “solution” already more or less suggests an implicit description format; in computer science terms, one may also say that a definition should be given of what is understood by a *run* (or an *execution*) of a system description. Formally speaking, description formats are a matter of syntax: they specify what is a well-formed expression. The notion of solution provides semantics: to each well-formed expression it associates a collection of functions of time. In the presentation of description formats above, the syntactic and semantic aspects have not been strictly separated, for reasons of readability. Here we review in a more formal way solution concepts for several of the description formats that were introduced.

Solution concepts for hybrid automata We will use the (autonomous) hybrid automata formulation as in Section 1.2 and Section 2.1 based on the 8-tuple $H = (\mathcal{Q}, \mathcal{X}, \mathbf{f}, \text{Init}, \text{Inv}, \mathcal{E}, \mathcal{G}, \mathcal{R})$. To formalize the solution concept based on this model syntax, we will use the following definitions.

Definition 5.4 (Hybrid time trajectory) [427] *A hybrid time trajectory $\tau = \{I_i\}_{i=0}^N$ is a finite ($N < \infty$) or infinite ($N = \infty$) sequence of intervals of the real line, such that:*

- $I_i = [\tau_i, \tau'_i]$ with $\tau_i \leq \tau'_i = \tau_{i+1}$ for $0 \leq i < N$;
- if $N < \infty$, either $I_N = [\tau_N, \tau'_N]$ or $I_N = [\tau_N, \tau'_N]$ with $\tau_N \leq \tau'_N \leq \infty$.

A hybrid time trajectory does not allow left accumulation points. Indeed, the event times set $\mathcal{E} := \{0\} \cup \{\frac{1}{n} \mid n \in \mathbb{N}\}$ and the corresponding sequence of intervals cannot be rewritten in terms of a hybrid time trajectory. Hence, the above definition excludes implicitly specific Zeno behavior and that this concept has a “preferred direction of time.” This is caused by the fact that it assumes that the set of event times is well-ordered by the usual order of the reals, but not necessarily by the reverse order; in other words, event times may accumulate to the right, but not to the left. (An ordered set \mathcal{S} is said to be well-ordered if each nonempty subset of \mathcal{S} has a least element.) This lack of symmetry with respect to time can be removed by allowing the set of event times \mathcal{E} to be of a more general type. Similar asymmetries in time are also the case for the solutions of hybrid inclusions and the forward solutions of complementarity systems as discussed below. Interestingly, Filippov solutions for discontinuous dynamical systems do have a more symmetric notion of time, which guarantees that time-reversed solutions remain to be solutions of the time-reversed system. This property is lost for the executions of hybrid automata, solutions to hybrid inclusions and forward solutions to complementarity systems (see also [532] for a further discussion).

We say that the hybrid time trajectory $\tau = \{I_i\}_{i=0}^N$ is a prefix of $\tau' = \{J_i\}_{i=0}^M$ and write $\tau \leq \tau'$, if they are identical or τ is finite, $M \geq N$, $I_i = J_i$ for $i = 0, 1, \dots, N - 1$, and $I_N \subseteq J_N$. In case τ is a prefix of τ' and they are not identical, τ is a *strict prefix* of τ' .

Definition 5.5 (Execution) An execution χ of a hybrid automaton is a collection $\chi = (\tau, \lambda, \xi)$ with:

- $\tau = \{I_i\}_{i=0}^N$ a hybrid time trajectory;
- $\lambda = \{\lambda_i\}_{i=0}^N$ with $\lambda_i : I_i \rightarrow \mathcal{Q}$; and
- $\xi = \{\xi_i\}_{i=0}^N$ with $\xi_i : I_i \rightarrow \mathcal{X}$ satisfying
 - *initial condition* $(\lambda(\tau_0), x(\tau_0)) \in \text{Init}$;
 - *continuous evolution for all i :*
 - λ_i is constant, i.e., $\lambda_i(t) = \lambda_i(\tau_i)$ for all $t \in I_i$;
 - ξ_i is the solution to the differential equation $\dot{\xi} = f(\lambda_i(t), \xi(t))$ on the interval I_i with initial condition $\xi_i(\tau_i)$ at τ_i ;
 - for all $t \in [\tau_i, \tau'_i]$ it holds that $\xi_i(t) \in \text{Inv}(\lambda_i(t))$;
 - *discrete evolution for all i , $e = (\lambda_i(\tau'_i), \lambda_{i+1}(\tau_{i+1})) \in \mathcal{E}$, $\xi(\tau'_i) \in \mathcal{G}(e)$ and $(\xi_i(\tau'_i), \xi_{i+1}(\tau_{i+1})) \in \mathcal{R}(e)$.*

Solution concepts for differential equations with discontinuous right-hand side

As we have seen above, some differential equations with discontinuous right-hand side can be considered from the perspective of differential inclusions. The standard solution concept for differential inclusions is the following. A vector function $x(t)$ defined on an interval $[a, b]$ is said to be a *solution* of the differential inclusion

$\dot{x} \in \mathcal{F}(x)$, where $\mathcal{F}(\cdot)$ is a set-valued function, if $x(\cdot)$ is absolutely continuous and satisfies $\dot{x}(t) \in \mathcal{F}(x(t))$ for almost all $t \in [a, b]$. The requirement of absolute continuity guarantees the existence of the derivative almost everywhere. One may note that the solution concept for differential inclusions does not have a preferred direction of time, as opposed to the notion of an execution for hybrid automata.

Solution concepts for hybrid inclusions For the hybrid inclusions (5.93) a solution concept (cf. [133, 271, 273]) is used that shows similarities with the one adopted for the hybrid automata. It is based upon the notion of a hybrid time domain, which is tightly connected to hybrid time trajectory as in Definition 5.4, because the hybrid time trajectory includes the “event counter j ” into the hybrid time domain.

Definition 5.6 (Hybrid time domain) A compact hybrid time domain is a set $\mathcal{D} \subset \mathbb{R}_{\geq 0} \times \mathbb{N}$ given by :

$$\mathcal{D} = \bigcup_{j=0}^{J-1} [t_j, t_{j+1}] \times \{j\},$$

where $J \in \mathbb{N}$ and $0 = t_0 \leq t_1 \cdots \leq t_J$. A hybrid time domain is a set $\mathcal{D} \subset \mathbb{R}_{\geq 0} \times \mathbb{N}$ such that, for each $(T, J) \in \mathcal{D}$, $\mathcal{D} \cap ([0, T] \times \{0, \dots, J\})$ is a compact hybrid time domain.

Also the hybrid time domains have a “preferred direction of time” as left accumulations of the reset times $\{t_j\}$ are not allowed.

Definition 5.7 (Hybrid trajectory) A hybrid trajectory is a pair $(\text{dom } x, x)$ consisting of hybrid time domain $\text{dom } x$ and a function x defined on $\text{dom } x$ that is locally absolutely continuous in t on $(\text{dom } x) \cap (\mathbb{R}_{\geq 0} \times \{j\})$ for each $j \in \mathbb{N}$.

Now we are ready to formally introduce a solution to (5.93).

Definition 5.8 (Hybrid arc) A hybrid trajectory $x : \text{dom } x \rightarrow \mathbb{R}^n$ is a solution sometimes called a hybrid arc to (5.93) if:

1. for all $j \in \mathbb{N}$ and for almost all $t \in I_j := \text{dom } x \cap (\mathbb{R}_{\geq 0} \times \{j\})$, we have $x(t, j) \in \mathcal{C}$ and $\dot{x}(t, j) \in \mathcal{F}(x(t, j))$;
2. for all $(t, j) \in \text{dom } x$ such that $(t, j+1) \in \text{dom } x$, we have $x(t, j) \in \mathcal{D}$ and $x(t, j+1) \in \mathcal{G}(x(t, j))$.

Solution concept for complementarity systems Section 5.2 introduced the concepts of Carathéodory and forward solutions for complementarity systems. These two notions are only valid for absolute continuous solutions implying that the x -part of the solutions cannot jump across events (mode switches). For (linear) complementarity systems of a higher index such as mechanical systems with unilateral constraints that induce impacts, this requirement is too strong and one has to add jump rules that connect continuous states before and after an event has taken place. Under

suitable conditions (specifically, in the case of linear complementarity systems and in the case of Hamiltonian complementarity systems), a general jump rule may be given [302, 304, 572].

5.4.4 Well-posedness notions

In the context of systems of differential equations, the term well-posedness roughly means that there is a nice relation between trajectories and initial conditions (or, more generally, boundary conditions). There are various ways in which this idea can be made more precise, so the meaning of the term may in fact be adapted to the particular problem class at hand. Typically it is required that solutions exist and are unique for any given initial condition. Both for the existence and for the uniqueness statement, one has to specify a function class in which solutions are considered. The function class used for existence may be the same as the one used for uniqueness, or they may be different; for instance, one might prove that solutions exist in some function class and that uniqueness holds in a larger function class. In the latter situation one is able to show specific properties (the ones satisfied by the smaller function class) of solution trajectories in the larger class. In case one is dealing with a system description that includes equality and/or inequality constraints, it may be reasonable to limit the set of initial conditions to a suitably chosen set of “feasible” or “consistent” initial conditions.

If solutions exist and are unique, a given system description defines a mapping from the set of initial conditions to trajectory set. In the theory of smooth dynamical systems, it is usually taken as part of the definition of well-posedness that this mapping is continuous with respect to suitably chosen topologies. In the case of non-smooth and hybrid dynamical systems, it frequently happens that there are certain boundaries in the continuous state space separating regions of initial conditions that generate widely different trajectories. Therefore, continuous dependence of solutions on initial conditions (at least in the sense of the topologies that are commonly used for smooth dynamical systems) may be a requirement too strong for hybrid systems. See, for instance, the mechanical example in [304] consisting of two carts connected by a hook and a spring, where the motion of the first cart is constrained by a block. This simple example illustrates the discontinuous dependence on initial conditions nicely.

One may also distinguish between various notions of well-posedness on the basis of the time interval that is involved. For instance, in the context of hybrid automata, one may say that a given automaton is *nonblocking* [427] if for each initial condition either at least one transition is enabled or an a smooth evolution according to the dynamics of one of the modes is possible on an interval of positive length. If the continuation is unique (the automaton is *deterministic* [427]), one may then say that the automaton is *initially well-posed*. This definition allows a situation in which a transition from location 1 to location 2 is immediately followed by a transition back to location 1 and so on in an infinite loop, so that $\tau'_i = \tau_i$ for all i in the hybrid time trajectory corresponding to this execution indicating that this solution does not make progress in the continuous time direction t (live-lock). A stronger

notion is obtained by requiring that a solution exists at least on an interval $[0, \varepsilon]$ with $\varepsilon > 0$; system descriptions for which such solutions exist and are unique are called *locally well-posed*. In computer science terminology, such systems “allow time to progress.” Finally, if solutions exist and are unique on the whole half-line $[0, \infty)$, then one speaks of *global well-posedness*. Local and global well-posedness can be seen to be asymmetric in their consideration of time in the sense that it considers “continuous” time t to be dominant over the “discrete time” j (in the terminology of hybrid time domains). For “physical” hybrid systems this asymmetry is useful as we are interested in the actual progress of real time t and less interested in the number of events. Initial well-posedness is from this point of view more symmetric.

Well-posedness of hybrid automata Necessary and sufficient conditions for well-posedness of hybrid automata have been stated in [427]. Basically these conditions mean that transitions with non-trivial reset relations are enabled whenever continuous evolution is impossible (this property is called *nonblocking*) and that discrete transitions must be forced by the continuous flow exiting the invariant set, no two discrete transitions can be enabled simultaneously, and no point x can be mapped onto two different points $x' \neq x''$ by the reset relation $R(q, q')$ - this property is called *determinism*. We will formally state the results of [427] after introducing some necessary concepts and definitions.

An execution $\chi = (\tau, \lambda, \xi)$ as defined in Definition 5.5 is called *finite*, if τ is a finite sequence ending with a closed interval; *infinite*, if τ is an infinite sequence or if $\sum_i (\tau'_i - \tau_i) = \infty$; and *maximal* if it is not a strict prefix of any other execution of the hybrid automaton. We denote the set of all maximal and infinite executions of the automaton with initial state $(q_0, x_0) \in \text{Init}$ by $\mathcal{H}_{(q_0, x_0)}^M$ and $\mathcal{H}_{(q_0, x_0)}^\infty$, respectively.

Definition 5.9 (Nonblocking automaton) A hybrid automaton is called nonblocking if $\mathcal{H}_{(q_0, x_0)}^\infty$ is nonempty for all $(q_0, x_0) \in \text{Init}$. It is called deterministic if $\mathcal{H}_{(q_0, x_0)}^M$ contains at most one element for all $(q_0, x_0) \in \text{Init}$.

These well-posedness concepts are similar to what we called *initial* well-posedness as they do not say anything about live-lock or the continuation beyond accumulation points of event times.

To simplify the characterization of nonblocking and deterministic automata, the following assumption has been introduced in [427]:

Assumption 5.1 The vector field $f(q, \cdot)$ is globally Lipschitz continuous for all $q \in \mathcal{Q}$. The edge (q, q') is contained in \mathcal{E} if and only if $\mathcal{G}(q, q') \neq \emptyset$ and $x \in \mathcal{G}(q, q')$ if and only if there is an $x' \in \mathcal{X}$ such that $(x, x') \in \mathcal{R}(q, q')$.

The first part of the assumption is standard to guarantee global existence and uniqueness of solutions within each location given a continuous initial state. The latter part is without loss of generality as can easily be seen [427].

A state (\hat{q}, \hat{x}) is called *reachable* if there exists a finite execution (τ, λ, ξ) with $\tau = \{[\tau_i, \tau'_i]\}_{i=0}^N$ and $(\lambda_N(\tau'_N), \xi_N(\tau'_N)) = (\hat{q}, \hat{x})$. The set $\text{Reach} \subseteq \mathcal{Q} \times \mathcal{X}$ denotes the collection of reachable states of the automaton.

The set of states from which continuous evolution is impossible is defined as

$$Out = \{(q_0, \mathbf{x}_0) \in \mathcal{Q} \times \mathcal{X} \mid \forall \varepsilon > 0 \exists t \in [0, \varepsilon) \mathbf{x}_{q_0, \mathbf{x}_0}(t) \notin Inv(q_0)\},$$

in which $\mathbf{x}_{q_0, \mathbf{x}_0}(\cdot)$ denotes the unique solution to $\dot{\mathbf{x}} = \mathbf{f}(q_0, \mathbf{x})$ with $\mathbf{x}(0) = \mathbf{x}_0$.

Theorem 5.14 [427] *Let Assumption 5.1 be satisfied.*

1. *A hybrid automaton is nonblocking if, for all $(q, \mathbf{x}) \in Reach \cup Out$, there exists $(q, q') \in \mathcal{E}$ with $\mathbf{x} \in \mathcal{G}(q, q')$. In case the automaton is deterministic, this condition is also necessary.*
2. *A hybrid automaton is deterministic if and only if for all $(q, \mathbf{x}) \in Reach$*
 - *if $\mathbf{x} \in \mathcal{G}(q, q')$ for some $(q, q') \in \mathcal{E}$, then $(q, \mathbf{x}) \in Out$;*
 - *if $(q, q') \in \mathcal{E}$ and $(q, q'') \in \mathcal{E}$ with $q' \neq q''$, then $\mathbf{x} \notin \mathcal{G}(q, q') \cap \mathcal{G}(q, q'')$; and*
 - *if $(q, q') \in \mathcal{E}$ and $\mathbf{x} \in \mathcal{G}(q, q')$, then there is at most one $\mathbf{x}' \in \mathcal{X}$ with $(\mathbf{x}, \mathbf{x}') \in \mathcal{R}(q, q')$.*

As a consequence of the broad class of systems covered by the results in this section, the conditions are rather implicit in the sense that for a particular example the conditions cannot be verified by direct calculations (i.e. are not in an algorithmic form). Especially, if the model description itself is implicit (e.g. piecewise affine systems or complementarity models) these results are only a start of the well-posedness analysis as the hybrid automaton model and the corresponding sets *Reach* and *Out* have to be determined first. In the next sections, we will present results that can be checked by direct computations.

The extension of the initial well-posedness results for hybrid automata to local or global existence of executions are awkward as Zeno behavior is hard to characterize or exclude, and continuation beyond Zeno times is not easy to show. This is one of the motivation to derive conditions that guarantee the existence or absence of Zeno behavior (see, e.g., [19, 159, 272, 341, 532, 583, 619, 680]) To guarantee continuation beyond Zeno times the hybrid model is sometimes extended or modified by using, e.g., relaxations [341]. As another example of an extension, consider the the bouncing ball model (Section 2.3.3) in which “global solutions” defined for all t in $[0, \infty)$ can be obtained by adding the “constrained mode” $\dot{x}_1 = \dot{x}_2 = 0$. Note that in case of complementarity modelling of the bouncing ball by $\dot{x}_1 = -g + w$, $0 \leq w \perp x_1 \geq 0$ (completed with the elastic reset map), where w represents the constraint force exerted by the ground on the ball, this constrained mode with $x_1 = 0$ follows naturally. For complementarity systems, but also for differential equations with discontinuous right-hand sides such as piecewise affine systems or other switched systems, one has the advantage that the location or mode can be described as a function of the continuous state. Of course, in this case one is able to define an evolution beyond the Zeno time by proving that the (left-)limit of the continuous state exists at the Zeno point (e.g. show for the bouncing ball as in Section 2.3.3 that $\lim_{t \uparrow \tau^*} x_1(t) = 0$

and $\lim_{t \uparrow \tau^*} x_2(t) = 0$, and that from $(0, 0)^\top$ continuation in the constrained mode is clearly possible). Continuation from this limit follows then from initial or local existence.

Well-posedness of piecewise linear systems A problem of considerable importance is to find necessary and sufficient conditions for well-posedness of piecewise linear systems

$$\dot{x} = \begin{cases} A_1 x, & \text{when } x \in C_1, \\ A_2 x, & \text{when } x \in C_2, \\ \vdots \\ A_r x, & \text{when } x \in C_r, \end{cases} \quad (5.94)$$

where C_i are certain subsets of \mathbb{R}^n having the property that

$$\begin{aligned} C_1 \cup C_2 \cup \dots \cup C_r &= \mathbb{R}^n \\ \text{int } C_i \cap \text{int } C_j &= \emptyset, \quad i \neq j. \end{aligned} \quad (5.95)$$

This situation may naturally arise from modeling, as well as from the application of a switching linear feedback scheme (with different feedback laws corresponding to the subsets C_i). Of course, even more general cases may be considered, or, instead, extra conditions may be imposed on the subsets C_i . Note that the first condition in (5.95) is a necessary (but not sufficient) condition for existence of solutions for all initial conditions and the second one is necessary (but again not sufficient) for uniqueness (unless the vector fields are equal on the overlapping parts of the regions C_i).

A particular case of the above problem, which has been investigated in depth, is the *bimodal* linear case

$$\dot{x} = \begin{cases} A_1 x, & \text{when } Cx \geq 0, \\ A_2 x, & \text{when } Cx \leq 0, \end{cases} \quad (5.96)$$

under the additional assumption that both pairs (C, A_1) and (C, A_2) are *observable*.

The solution concept that will be employed is the *extended Carathéodory solution*, which is a function $x : [t_0, t_1] \rightarrow \mathbb{R}^n$, which is absolutely continuous on $[t_0, t_1]$, satisfies

$$x(t) = x(t_0) + \int_{t_0}^t f(x(\tau)) d\tau, \quad (5.97)$$

where $f(x)$ is the (discontinuous) vector field given by the right-hand side of (5.96), and there are no left-accumulation points of event times on $[t_0, t_1]$.

Note that Filippov solutions involving sliding modes are not extended Carathéodory solutions. Moreover, note that if $f(x)$ is *continuous* then necessarily there exists a K such that $A_1 = A_2 + KC$, and f is automatically Lipschitz continuous, implying local uniqueness of solutions by classical results on ordinary differential equations.

Before stating the main result we introduce some notation. First we define the $n \times n$ observability matrices corresponding to (C, A_1) , respectively (C, A_2) :

$$W_1 := \begin{pmatrix} C \\ CA_1 \\ \vdots \\ CA_1^{n-1} \end{pmatrix}, \quad W_2 := \begin{pmatrix} C \\ CA_2 \\ \vdots \\ CA_2^{n-1} \end{pmatrix} \tag{5.98}$$

(by assumption they both have rank n). Furthermore we define the following subsets of the state space \mathbb{R}^n :

$$\begin{aligned} S_i^+ &= \{x \in \mathbb{R}^n \mid W_i x \succeq 0\} \\ S_i^- &= \{x \in \mathbb{R}^n \mid W_i x \preceq 0\} \end{aligned} \quad i = 1, 2, \tag{5.99}$$

where \succeq denotes *lexicographic* ordering, that is $x = 0$ or $x \succeq 0$ if the first component of x that is nonzero is positive. Furthermore, $x \preceq 0$ iff $-x \succeq 0$. Then the following result from [335] can be stated:

Theorem 5.15 *The bimodal linear system (5.96) is well-posed if and only if one of the following equivalent conditions are satisfied:*

- (a) $S_1^+ \cup S_2^- = \mathbb{R}^n$;
- (b) $S_1^+ \cap S_2^- = \{0\}$;
- (c) $W_2 W_1^{-1}$ is a lower-triangular matrix with positive diagonal elements.

Possible extensions to noninvertible observability matrices, the situation of more than two modes, as well as to modification of the sets $Cx \geq 0$, $Cx \leq 0$, are discussed in [335, 336].

Complementarity systems Several well-posedness results were already presented in Section 5.2. These results focussed on Carathéodory and forward solutions that applied to absolutely continuous trajectories only. However, in various application domains of complementarity systems the restriction to continuous trajectory is too stringent. This is the case in the context of unilaterally constrained mechanical systems (cf. [122, 123, 302, 413, 462]) in which impacts cause discontinuities in the velocities of the impacting bodies. In this section we will provide a result that applies to linear complementarity systems of the form

$$\dot{x}(t) = Ax(t) + Bu(t), \tag{5.100a}$$

$$y(t) = Cx(t) + Du(t), \tag{5.100b}$$

$$0 \leq u(t) \perp y(t) \geq 0, \tag{5.100c}$$

in which impacts are allowed. Before doing so, we will present a result for a class of nonsmooth dynamical systems consisting of linear saturation systems and linear relay systems, which are based on “complementarity reasoning,” see [149, 151].

Linear saturation and linear relay systems As is well-known [217], piecewise linear relations may be described in terms of the linear complementarity problem. In the circuits and systems community (cf. [395, 641]) the complementarity formulation has already been used for *static* piecewise linear systems; this subsection may be viewed as an extension of the cited work in the sense that we consider *dynamic* systems. For the sake of simplicity, we will focus on a specific type of piecewise linear systems, namely linear saturation systems, i.e. linear systems coupled to saturation characteristics. They are of the form

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t), \quad (5.101a)$$

$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t), \quad (5.101b)$$

$$(\mathbf{u}(t), \mathbf{y}(t)) \in \text{saturation}_i, \quad (5.101c)$$

where $\mathbf{x}(t) \in \mathbb{R}^n$, $\mathbf{u}(t) \in \mathbb{R}^m$, $\mathbf{y}(t) \in \mathbb{R}^m$, \mathbf{A} , \mathbf{B} , \mathbf{C} and \mathbf{D} are matrices of appropriate sizes, and saturation_i is the curve depicted in Fig. 5.3 with $e_2^i - e_1^i > 0$ and $f_1^i \geq f_2^i$. We denote the overall system (5.101) by $\text{SAT}(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D})$. Note that relay characteristics can be obtained from saturation characteristics by setting $f_1^i = f_2^i$.

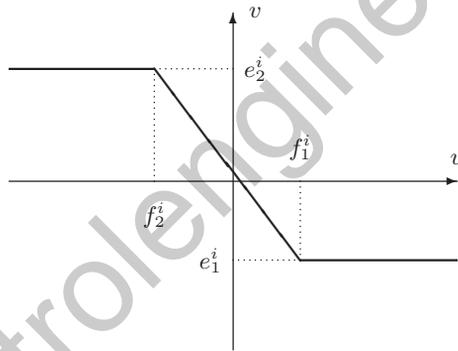


Fig. 5.3 Saturation characteristic.

One may argue that the saturation characteristic is a Lipschitz continuous function (provided that $f_1^i - f_2^i > 0$) and hence the existence and uniqueness of the solutions follow from the theory of ordinary differential equations. The following example shows that this is not correct in general if the feedthrough term D is nonzero:

Example 5.10 Linear saturation system

Consider the single-input single-output system

$$\dot{x} = u, \quad (5.102)$$

$$y = x - 2u, \quad (5.103)$$

where u and y restricted by a saturation characteristic with $e_1 = -f_1 = -e_2 = f_2 = 1/2$ as shown in Fig. 5.3. Let the periodic function $\tilde{u} : \mathbb{R}_+ \rightarrow \mathbb{R}$ be defined by

$$\tilde{u}(t) = \begin{cases} 1/2, & \text{if } 0 \leq t < 1, \\ -1/2, & \text{if } 1 \leq t < 3, \\ 1/2, & \text{if } 3 \leq t < 4, \end{cases}$$

and $\tilde{u}(t - 4) = \tilde{u}(t)$ whenever $t \geq 4$. By using this function define $\tilde{x} : \mathbb{R}_+ \rightarrow \mathbb{R}$ as

$$\tilde{x}(t) = \int_0^t \tilde{u}(s) \, ds,$$

and $\tilde{y} : \mathbb{R}_+ \rightarrow \mathbb{R}$ as

$$\tilde{y} = \tilde{x} - 2\tilde{u}.$$

It can be verified that $(-\tilde{u}, -\tilde{x}, -\tilde{y})$, $(0, 0, 0)$, and $(\tilde{u}, \tilde{x}, \tilde{y})$ are all solutions of SAT(0, 1, 1, -2) with the zero initial state. \square

As illustrated in the example, the Lipschitz continuity argument does not work in general when $f_1^i > f_2^i$. Also in the case, where $f_1^i = f_2^i$ this reasoning does not apply. The following theorem gives a sufficient condition for the well-posedness of linear systems with saturation characteristics. Recall that a P -matrix is a matrix with all its principal minors being positive.

Theorem 5.16 [149, 151] Consider SAT(A, B, C, D). Let $R = \text{diag}(e_2^i - e_1^i)$ and $S = \text{diag}(f_2^i - f_1^i)$. Suppose that $G(\sigma)R - S$ is a P -matrix for all sufficiently large $\sigma \in \mathbb{R}$, where

$$G(\sigma) = C(\sigma I - A)^{-1}B + D.$$

Then, there exists a unique forward solution of SAT(A, B, C, D) for all initial states.

Linear complementarity systems with jumps Up to this point, the results on well-posedness for complementarity systems concerned solutions of which the x -part is continuous. As mentioned before, for applications such as constrained mechanical systems (e.g. the bouncing ball) discontinuities in the state variables are required. For linear complementarity systems as in (5.100) a distributional framework was used to obtain an extension of the forward solution concept (see [304] for details). The work [304] presented also sufficient conditions for local well-posedness. In case of one complementarity pair, these conditions are also sufficient for global well-posedness.

Consider the LCS(A, B, C, D) as in (5.100) with Markov parameters $H^0 = D$ and $H^i = CA^{i-1}B$, $i = 1, 2, \dots$ and define the leading row and column indices by

$$\rho_j := \inf\{i \in \mathbb{N} \mid H_{j\bullet}^i \neq 0\}, \quad \eta_j := \inf\{i \in \mathbb{N} \mid H_{\bullet j}^i \neq 0\},$$

where $j \in \{1, \dots, k\}$ and $\inf \emptyset := \infty$. The leading row coefficient matrix \mathcal{M} and leading column coefficient matrix \mathcal{N} are then given for finite leading row and column indices by

$$\mathcal{M} := \begin{pmatrix} H_{1\bullet}^{\rho_1} \\ \vdots \\ H_{k\bullet}^{\rho_k} \end{pmatrix} \text{ and } \mathcal{N} := (H_{\bullet 1}^{\eta_1} \dots H_{\bullet k}^{\eta_k}).$$

Theorem 5.17 [304] *If the leading column coefficient matrix \mathcal{N} and the leading row coefficient matrix \mathcal{M} are both defined and P-matrices, then $LCS(A, B, C, D)$ has a unique local forward solution (with jumps) on an interval of the form $[0, \varepsilon)$ for some $\varepsilon > 0$. Moreover, live-lock (an infinite number of events at one time instant) does not occur.*

Differential equations with discontinuous right-hand sides Differential equations of the form

$$\dot{x}(t) = f(t, x(t)) \tag{5.104}$$

with f being piecewise continuous in a domain \mathcal{G} and with the set \mathcal{M} of discontinuity points having measure zero, received quite some attention in the literature. Major roles have been played in this context by Filippov [237] and Utkin [640]. An example of such a system with two “modes” was given in (5.90). As mentioned in Subsection 5.4.2, solution concepts have been defined by replacing the basic differential equation (5.104) by a differential inclusion of the form

$$\dot{x}(t) \in \mathcal{F}(t, x(t)), \tag{5.105}$$

where \mathcal{F} is constructed from f . The solution concept is then inherited from the realm of differential inclusions [26].

Definition 5.10 (Solution of differential inclusion) *The function $x : \Omega \rightarrow \mathbb{R}^n$ is called a solution of the differential inclusion (5.105) if x is absolutely continuous on the time-interval Ω and satisfies $\dot{x}(t) \in \mathcal{F}(t, x(t))$ for almost all $t \in \Omega$.*

There are several ways to transform f into \mathcal{F} and we will restrict ourselves to the two most famous ones and briefly discuss an alternative transformation proposed by Aizerman and Pyatnitskii [6]. For further details see [237].

In the convex definition [237], as already briefly mentioned in Section 5.4.2 the set $\mathcal{F}_\alpha(t, x)$ is taken to be the smallest convex closed set containing all the limit values of the function $f(\bar{t}, \bar{x})$ for $\bar{x} \rightarrow x, \bar{t} = t$ and $(\bar{t}, \bar{x}) \notin M$.

The control equivalent definition proposed by Utkin [640] (see also page 54 in [237]) applies to equations of the form

$$\dot{x}(t) = f(t, x(t), u_1(t, x), \dots, u_r(t, x)), \tag{5.106}$$

where \mathbf{f} is continuous in its arguments, but $u_i(t, \mathbf{x})$ is a scalar-valued function being discontinuous only on a smooth surface \mathcal{S}_i given by $\phi_i(\mathbf{x}) = 0$. We define the sets $U_i(t, \mathbf{x})$ as $\{u_i(t, \mathbf{x})\}$ when $\mathbf{x} \notin \mathcal{S}_i$ and in case $\mathbf{x} \in \mathcal{S}_i$ by the closed interval with end-points $u_i^-(t, \mathbf{x})$ and $u_i^+(t, \mathbf{x})$. The values $u_i^-(t, \mathbf{x})$ and $u_i^+(t, \mathbf{x})$ are the limiting values of the function u_i on both sides of the surface \mathcal{S}_i which we assume to exist. The differential equation (5.106) is replaced by (5.105) with $\mathcal{F}_b(t, \mathbf{x}) = \mathbf{f}(t, \mathbf{x}, U_1(t, \mathbf{x}), \dots, U_r(t, \mathbf{x}))$.

Remark 5.4 In case $\mathcal{F}_c(t, \mathbf{x})$ is chosen as the smallest convex closed set containing $\mathcal{F}_b(t, \mathbf{x})$, then the general definition of Aizerman and Pyatnitskii [6] is obtained. In case \mathbf{f} is affine in u_1, \dots, u_r and the surfaces $\mathcal{S}_1, \dots, \mathcal{S}_r$ are all different and at the point of intersection the normal vectors are linearly independent, all the before mentioned definitions coincide, i.e. $\mathcal{F}_a = \mathcal{F}_b = \mathcal{F}_c$.

The well-posedness results of the differential equation (5.104) or (5.106) can now be based on the theory available for differential inclusions (cf. [26, 237] and the references therein). A set-valued function \mathcal{F} is called *upper semicontinuous* at p_0 , if for all $\varepsilon > 0$ there is a $\delta > 0$ such that $\mathcal{F}(p + \delta\mathbb{B}) \subseteq \mathcal{F}(p_0) + \varepsilon\mathbb{B}$, where \mathbb{B} denotes the unit ball. \mathcal{F} is called upper semicontinuous on a set \mathcal{D} , if \mathcal{F} is upper semicontinuous in each point of the set \mathcal{D} .

Definition 5.11 (Basic condition) We say that the set-valued map $\mathcal{F}(t, \mathbf{x})$ satisfies the basic conditions, if:

- for all $(t, \mathbf{x}) \in \mathcal{G}$ the set $\mathcal{F}(t, \mathbf{x})$ is nonempty, bounded, closed, and convex
- \mathcal{F} is upper semicontinuous in t, \mathbf{x} .

The following result is described on page 77 of the monograph [237].

Theorem 5.18 (Theorems 2.7.1 and 2.7.2 in [237]) If $\mathcal{F}(t, \mathbf{x})$ satisfies the basic conditions in the domain \mathcal{G} , then for any point $(t_0, \mathbf{x}_0) \in \mathcal{G}$ there exists a solution of the problem

$$\dot{\mathbf{x}}(t) \in \mathcal{F}(t, \mathbf{x}(t)), \quad \mathbf{x}(t_0) = \mathbf{x}_0. \quad (5.107)$$

If the basic conditions are satisfied in a closed and bounded domain \mathcal{G} , then each solution can be continued on both sides up to the boundary of the domain \mathcal{G} .

In combination with the following result Theorem 5.18 proves the existence of solutions for the differential inclusions related to \mathcal{F}_a , \mathcal{F}_b , and \mathcal{F}_c :

Theorem 5.19 (Page 67 in [237]) The sets $\mathcal{F}_a(t, \mathbf{x})$, $\mathcal{F}_b(t, \mathbf{x})$ and $\mathcal{F}_c(t, \mathbf{x})$ are nonempty, bounded, and closed. $\mathcal{F}_a(t, \mathbf{x})$ and $\mathcal{F}_c(t, \mathbf{x})$ are also convex. \mathcal{F}_a is upper semicontinuous in \mathbf{x} , and \mathcal{F}_b and \mathcal{F}_c are upper semicontinuous in t, \mathbf{x} .

Theorem 5.18 and 5.19 together show the existence of solutions when Filippov's convex definition is used under the condition that f is time-invariant. In case f is not time-invariant, additional assumptions are needed to arrive at \mathcal{F} being upper semicontinuous in t as well (cf. page 68 in [237]). For the definition of Aizerman and Pyatnitskii (i.e. using \mathcal{F}_c) existence of solutions is guaranteed. In case $\mathcal{F}_b(t, \mathbf{x})$ is convex for all relevant (t, \mathbf{x}) (e.g. if the conditions mentioned in Remark 5.4 are satisfied), then existence follows as well. If the convexity assumption is not satisfied, the existence result still holds if upper semicontinuity is replaced by continuity (cf. page 79 in [237]). In fact, the two major cases studied in Chapter 3 of [26] are related to these two situations: (i) the values of \mathcal{F} are compact and convex and \mathcal{F} is upper semicontinuous; and (ii) the values of \mathcal{F} are compact, but not necessarily convex and \mathcal{F} is continuous.

Now we will discuss the issue of uniqueness. Right uniqueness (in the Filippov sense) holds for the differential equation (5.104) at the point (t_0, \mathbf{x}_0) , if there exists $t_1 > t_0$ such that each two solutions of this equation satisfying the initial condition $\mathbf{x}(t_0) = \mathbf{x}_0$ coincide on the interval $[t_0, t_1]$ or on the interval on which they are both defined. Right uniqueness holds for a domain \mathcal{D} if from each point $(t_0, \mathbf{x}_0) \in \mathcal{D}$ right uniqueness holds.

Not too many uniqueness results are available in the literature. The most useful result given in [237] is related to the following situation. Let the domain $G \subset \mathbb{R}^n$ be separated by a smooth surface \mathcal{S} into domains G^- and G^+ . Let f and $\partial f / \partial x_i$ be continuous in the domains G^- and G^+ up to the boundary such that $f^-(t, \mathbf{x})$ and $f^+(t, \mathbf{x})$ denote the limit values of the function f at (t, \mathbf{x}) , $\mathbf{x} \in \mathcal{S}$ from the regions G^- and G^+ , respectively. We define $h(t, \mathbf{x}) = f^+(t, \mathbf{x}) - f^-(t, \mathbf{x})$ as the discontinuity vector over the surface \mathcal{S} . Moreover, let $n(\mathbf{x})$ be the normal vector to \mathcal{S} at point \mathbf{x} directed from G^- to G^+ .

Theorem 5.20 Consider the differential equation (5.104) with f as above. Let \mathcal{S} be a twice continuously differentiable surface and suppose that the function h is continuously differentiable. If for each $t \in (a, b)$ and each point $\mathbf{x} \in \mathcal{S}$ at least one of the inequalities $n(\mathbf{x})^T f^-(t, \mathbf{x}) > 0$ or $n(\mathbf{x})^T f^+(t, \mathbf{x}) < 0$ (possibly different inequalities for different \mathbf{x} and t) is fulfilled, then right uniqueness holds for (5.104) in the domain \mathcal{G} for $t \in (a, b)$ in the sense of Filippov.

The criterion above clearly holds for general nonlinear systems, but needs to be verified on a point-by-point basis. Alternatively, the results on complementarity systems, piecewise affine systems or linear saturation systems are more straightforward to check as it requires, for instance, the computation of the determinants of all principal minors of the transfer function of the underlying linear system, or determine the signs of the leading Markov parameters. However, the latter theory applies to specific classes of hybrid systems and uses a *different* solution concept. Hence, uniqueness is not proved in the Filippov sense, but in a forward sense.

Hybrid inclusions Adding reset maps and restricting the “flow region” for the above differential inclusions (5.105) leads to the hybrid inclusions (5.93). In [271] the following basic conditions are adopted:

- \mathcal{C} and \mathcal{D} are closed sets;
- \mathcal{F} is outer semicontinuous in the sense that for all $\mathbf{x} \in \mathbb{R}^n$ and all sequences $\{\mathbf{x}_i\}$ with $\mathbf{x}_i \rightarrow \mathbf{x}$, $y_i \in \mathcal{F}(\mathbf{x}_i)$ such that $y_i \rightarrow y$, it holds that $y \in \mathcal{F}(\mathbf{x})$;
- \mathcal{F} is locally bounded (i.e. for any compact set $K \subseteq \mathbb{R}^n$ there exists $m > 0$ such that for all $\mathbf{x} \in K$ it holds that $\mathcal{F}(\mathbf{x}) \subseteq m\mathbb{B}$, where \mathbb{B} is the unit ball) and $\mathcal{F}(\mathbf{x})$ is nonempty and convex for all $\mathbf{x} \in \mathcal{C}$;
- \mathcal{G} is outer semicontinuous and $\mathcal{G}(\mathbf{x})$ is nonempty for all $\mathbf{x} \in \mathcal{D}$.

Note that in the case of locally bounded set-valued mappings with closed values, outer semicontinuity agrees with upper semicontinuity. In general this is not true [271].

Based on these basic conditions, Proposition 2.4 in [271] states existence results for these hybrid inclusions. Actually, [271] follows here closely the lines of [27], where a similar result was obtained for so-called *impulse differential inclusions*. To formulate the existence result we will use the following concepts. The tangent cone $T_{\mathcal{C}}(\mathbf{x})$ to a set \mathcal{C} at the point $\mathbf{x} \in \mathcal{C}$ consists of all $\mathbf{v} \in \mathbb{R}^n$ for which there exist real numbers $\alpha_i > 0$ with $\alpha_i \rightarrow 0$ and vectors $\mathbf{v}_i \rightarrow \mathbf{v}$ such that for $i = 1, 2, \dots$ $\mathbf{x} + \alpha_i \mathbf{v}_i \in \mathcal{C}$. A solution (in the form of a hybrid arc as defined in Definition 5.8) to (5.93) is called maximal if it cannot be extended and is called complete if its domain is unbounded (either in the “ j ” and/or “ t ” directions). The notions of maximal and complete solutions are similar in nature as maximal and infinite executions of hybrid automata in Section 5.4.4.

Theorem 5.21 Consider the system (5.93) with the above basic conditions are fulfilled. If $\mathbf{x}_0 \in \mathcal{D}$ or $\mathbf{x}_0 \in \mathcal{C}$ and for some neighborhood \mathcal{U} of \mathbf{x}_0 it holds that

$$\mathbf{x}' \in \mathcal{U} \cap \mathcal{C} \text{ implies that } T_{\mathcal{C}}(\mathbf{x}') \cap \mathcal{F}(\mathbf{x}') \neq \emptyset, \quad (5.108)$$

then there exists a hybrid arc \mathbf{x} of the hybrid inclusion with $\mathbf{x}(0, 0) = \mathbf{x}_0$ and $\text{dom } \mathbf{x} \neq (0, 0)$. If (5.108) holds for any $\mathbf{x}_0 \in \mathcal{C} \setminus \mathcal{D}$, then for any maximal solution \mathbf{x} at least one of the following is true:

- (i) \mathbf{x} is complete;
- (ii) \mathbf{x} eventually leaves every compact subset of \mathbb{R}^n ;
- (iii) for some $(T, J) \in \text{dom } \mathbf{x}$ with $(T, J) \neq (0, 0)$ we have $\mathbf{x}(T, J) \notin \mathcal{C} \cup \mathcal{D}$.

Case (iii) does not occur if additionally we have for all $\mathbf{x}_0 \in \mathcal{D}$ that $\mathcal{G}(\mathbf{x}_0) \subseteq \mathcal{C} \cup \mathcal{D}$.

This result can be considered as an initial well-posedness result. In particular, it does not give any guarantees that a solution can be defined on a domain containing

some (t, J) with $t > 0$ (as live-lock is not excluded) nor for $t \rightarrow \infty$ (due to finite escape times or right-accumulations of reset times). Uniqueness of trajectories is not considered in the context of hybrid inclusions. Note that statement (ii) above expresses a kind of “finite escape time” condition, which is similar as in Theorem 5.18 for differential inclusions only.

5.4.5 Comparison of some solution concepts

The difference between Filippov and forward and extended Carathéodory solutions will be discussed in the context of the class of systems for which all these concepts apply. In particular, we will study the plant

$$\dot{x}(t) = Ax(t) + Bu(t); y(t) = Cx(t), \quad (5.109)$$

in a closed loop with the relay feedback

$$u(t) = -\text{sgn}(y(t)). \quad (5.110)$$

Note that, in the context of Theorem 5.16, we are dealing with the situation in which $R = 2I$ and $S = 0$. Note also that $\mathcal{F}_a = \mathcal{F}_b = \mathcal{F}_c$ for such linear relay systems and that the corresponding solution concepts coincide and will therefore be referred to as “Filippov solutions” from now on.

Example 5.11 *Difference between Filippov and forward concepts*

The difference between the forward solutions and Filippov solution is related to Zeno behavior and is nicely demonstrated by an example constructed by Filippov (page 116 in [237]), which is given by

$$\dot{x}_1 = -u_1 + 2u_2, \quad (5.111a)$$

$$\dot{x}_2 = -2u_1 - u_2, \quad (5.111b)$$

$$y_1 = x_1, \quad (5.111c)$$

$$y_2 = x_2, \quad (5.111d)$$

$$u_1 = -\text{sgn}(y_1), \quad (5.111e)$$

$$u_2 = -\text{sgn}(y_2). \quad (5.111f)$$

This system has, besides the zero solution (which is both a Filippov and a forward solution), an infinite number of other trajectories (being Filippov, but not forward solutions) starting from the origin. The nonzero solutions (Fig. 5.4) leave the origin due to left-accumulations of the relay switching times and are Filippov solutions, but are not forward solutions. However, this example does not satisfy the conditions for uniqueness given in Theorem 5.16 in Section 5.4.4. Hence, it is not clear if the conditions in Section 5.4.4 are sufficient for Filippov uniqueness as well. □

The latter problem mentioned in the example is studied in [532] for the case where (5.109) is a single-input-single-output (SISO) system. Theorem 5.16 states that the positivity of the leading Markov parameter H^ρ with $H^i = CA^{i-1}B$, $i = 1, 2, \dots$ and $\rho = \min\{i \mid H^i \neq 0\}$ implies existence uniqueness in forward sense. What about uniqueness in Filippov’s sense?

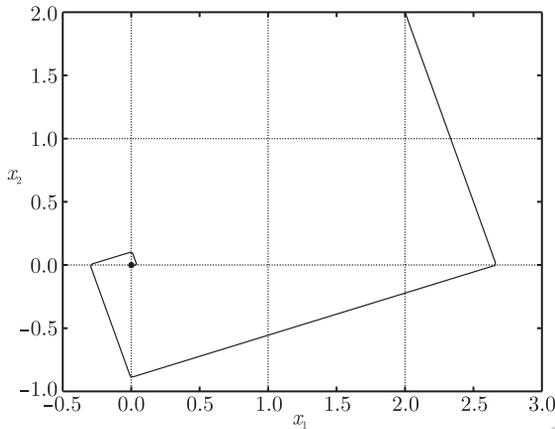


Fig. 5.4 Trajectory in the phase plane of (5.111).

Theorem 5.22 [532] *Consider the system (5.109)–(5.110). The following statements hold for the relative degree ρ being 1 or 2:*

$\rho = 1$: *The system (5.109)–(5.110) has a unique Filippov solution for all initial conditions if and only if the leading Markov parameter \mathbf{H}^p is positive.*

$\rho = 2$: *The system (5.109)–(5.110) has a unique Filippov solution for initial condition $\mathbf{x}(0) = 0$ if and only if the leading Markov parameter \mathbf{H}^p is positive.*

Moreover, in the case $\mathbf{H}^1 = \mathbf{CB} > 0$, Filippov solutions do not have left-accumulations of relay switching times.

Interestingly, the above theorem presents conditions that exclude particular types of Zeno behavior.

Up to this point, one might hope that the positivity of the leading Markov parameter is also sufficient for Filippov uniqueness for higher relative degrees. However, in [532] a counter-example is presented of the form (5.109)–(5.110), with (5.109) being a triple integrator.

Example 5.12 *Linear relay example*

This relay system has one forward solution (being identically zero) starting in the origin (as expected, as the leading Markov parameter is positive), but has infinitely many Filippov solutions of which one is the zero solution and the other starts with a left-accumulation point of relay switching times [532]. This example can also be considered in the light of the piecewise linear systems (5.96) considered in Section 5.4.4, which are of the form

$$\begin{cases} \text{mode 1 : } \dot{\mathbf{x}} = \mathbf{A}_1\mathbf{x}, & \text{if } \mathbf{y} = \mathbf{C}\mathbf{x} \geq 0, \\ \text{mode 2 : } \dot{\mathbf{x}} = \mathbf{A}_2\mathbf{x}, & \text{if } \mathbf{y} = \mathbf{C}\mathbf{x} \leq 0, \end{cases} \quad (5.112)$$

with

$$\mathbf{A}_1 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \quad \mathbf{A}_2 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \quad \mathbf{C} = (1 \ 0 \ 0 \ 0). \quad (5.113)$$

An extended Carathéodory solution concept (5.97) for this type of systems and necessary and sufficient conditions for existence and uniqueness are presented in [335] (see Theorem 5.15). As this solution concept does not allow for sliding modes and left-accumulation points of event times, the above system does not have any extended Carathéodory solution starting from the initial state $(0, 0, 0, 1)^T$ as can easily be seen (cf. also Theorem 5.15).

In summary, the triple integrator connected to a (negative) relay forms a nice comparison between the three mentioned solution concepts; for the system (5.112) with (5.113) and $\mathbf{x}_0 = (0, 0, 0, 1)^T$, there exist [532]

- no extended Carathéodory solution;
- one forward solution; and
- infinitely many Filippov solutions. \square

For specific applications in discontinuous feedback control the Filippov solution concept allows trajectories, which are not practically relevant for the stabilization problem at hand. So-called Euler (or sampling) solutions seem to be more appropriate in this context [176, 178]. Also in this case the discontinuous dynamical system is replaced by a differential inclusion with the difference that a particular choice of the controller is made at the switching surface. This choice determines which trajectories are actually Euler solutions by forming the limits of certain numerical integration routines (cf. [176, 178] for more details).

In Section 2.4.2 of [237] some further results can be found on uniqueness. The most general result in [237] for uniqueness in the setting of Filippov's convex definition uses the exclusion of left-accumulation points as one of the conditions to prove uniqueness. Unfortunately, it is not clear how such assumptions should be verified. As a consequence, Theorem 5.22 is quite useful. In Section 2.4.2 of [237] one can also find some results on continuous dependence of solutions on initial data. See also the recent survey of [188] on discontinuous dynamical systems.

5.4.6 Zenoness

The above examples, and also the discussion in Chapter 2, indicate that the Zeno phenomenon in all its forms complicates simulation and many analysis and design problems, including the well-posedness issue. Depending on which type of Zenoness is allowed in the solution concept, the answer to the well-posedness problem differs. So, conditions stating the existence or absence of certain variants of Zenoness are of interest. Such conditions are generally hard to come by, but some rather recent works provide some interesting insights in this difficult problem. The reader might want to consult [19, 159, 272, 341, 532, 583, 619, 680] and the references therein. Some of

these works also indicate possibilities on how to proceed (define solutions) beyond Zeno points.

Bibliographical notes

Introductions to model-predictive control are given in [58, 140, 432, 470, 555].

A tutorial overview on the mathematical aspects of discontinuous differential equations is given in [188]. Impulsive systems are considered in the monograph [292].

controlengineers.ir

Hybrid systems: quantization and abstraction

J. Lunze, A. Bicchi, T. Moor, L. Palopoli, B. Picasso, J. Raisch, and A. Schild

Several control and supervision problems for hybrid systems are posed in terms of abstract information. If the reduction of the measurement resolution leads to quantized signals, the problem to stabilize a continuous or hybrid systems by quantized feedback has to be solved. For process supervision with abstract design specifications, it is reasonable to reduce the complexity of analysis and design tasks by using abstract models that ignore the continuous movement of the hybrid systems. This chapter shows how abstract models like automata or embedded maps can be set up and used for the diagnosis and supervisory control of hybrid systems.

Chapter contents

6.1	Quantization and model abstraction	page 195
6.2	Stabilization of quantized systems	196
6.2.1	Systems with quantized feedback	196
6.2.2	Issues on the stabilization problem for quantized systems	199
6.2.3	Control under communication constraints	203
6.2.4	Control with quantized sensors and actuators	210
6.2.5	Channel sharing for systems with input constraints	212
6.3	Discrete-event modeling and diagnosis	213
6.3.1	Diagnostic problem	213
6.3.2	Hybrid model of quantized systems	214
6.3.3	Properties of quantized systems	215
6.3.4	Discrete-event modeling of quantized systems	218
6.3.5	Diagnosis of quantized systems	220
6.3.6	Example: Diagnosis of the air path of a diesel engine	221
6.4	Abstraction-based supervisory control design	223
6.4.1	Motivation of abstraction-based design	223
6.4.2	Control problem	224

6.4.3	Behaviors	225
6.4.4	Supervisory control	226
6.4.5	Abstraction-based supervisory control	228
6.4.6	Extensions	230
6.5	Control design by means of embedded maps	231
6.5.1	Stabilization problem for discretely controlled continuous systems	231
6.5.2	Modeling the behavior of discretely controlled continuous systems	234
6.5.3	Stabilization of periodic stationary solutions	238
6.5.4	Application: Event generator design of a boost converter	244

controlengineers.ir

6.1 Quantization and model abstraction

The subject of this chapter is a class of hybrid systems that have to be analyzed and controlled in terms of their symbolic dynamics. Only the sequence of symbolic inputs and outputs associated with the hybrid state trajectory is accessible for measurement and, hence, provides the on-line information to be used in fault diagnosis and feedback control.

The way of dealing with abstract measurement and modeling information is explained by considering quantized systems that consist of a continuous-variable system whose input, state, and output are accessible only through quantizers. The hybrid character of such systems becomes obvious from the fact that internally the system behavior is described by the continuous state $x(t)$ whereas the outside observer only sees the quantized version $[x(t)]$ of the state and of the input and output, which jumps between discrete (symbolic) values.

The chapter is divided into four parts. (Section 6.2) shows important properties of quantized systems and how such systems can be stabilized by feeding back the quantized state or output information. This section investigates the consequences of quantization for feedback control.

The other three sections concern the use of abstract models representing the system behavior in terms of the symbolic information. The main modeling idea of these sections is depicted in Fig. 6.1. Analysis and control design concern the discrete-event behavior of the hybrid system, which is symbolized by the upper right box. The usual way of finding these data is first to determine the hybrid behavior consisting of continuous and discrete trajectories and second to ignore the continuous part to get the discrete-event behavior. This way follows the arrows from the lower left to the lower right block and then to the upper right block in the figure.

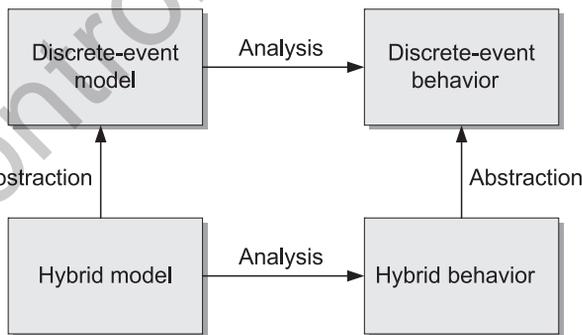


Fig. 6.1 Abstraction-based modeling of hybrid systems.

Abstraction-based modeling uses the alternative way, which is shown by the arrows from the lower to the upper left blocks and then to the upper right block. Here

the abstraction step is carried out in the modeling step, resulting in a purely discrete-event model of the hybrid system. Then, the analysis only concerns the discrete-event dynamics of the hybrid system.

In (Section 6.3) it is shown that both modeling and analysis methods yield the same result only for a very specific class of systems. As an important consequence in general, the discrete-event model cannot be a precise, but only a complete representation of the hybrid system. This model includes sufficient information for fault diagnosis and for supervisory control design as shown in (Section 6.4). In (Section 6.5) the embedded map is introduced as another model that concerns only the time instances at which the hybrid system changes its discrete state.

The term of a discrete-event model is used in this chapter synonymously to a symbolic model. Such models refer to symbolic, in particular quantized, information. In the discrete-time setting adopted in Sections Section 6.2 and Section 6.3 the time instant at which the model should represent the hybrid system is described by a clock. In contrast to this, in the modeling approaches described in Sections Section 6.4 and Section 6.5 the time instant at which the model should describe the symbolic dynamics of the hybrid system is given by the moments at which the input, state, or output signals cross a quantization border or, more generally, some surface in the signal space.

6.2 Stabilization of quantized systems

6.2.1 Systems with quantized feedback

In this section, we consider the stabilization of quantized control systems, which are a particular class of hybrid systems. Even though some of the results shown below refer to continuous-time or nonlinear systems, our main focus is on discrete-time strictly proper quantized linear systems described by the following model:

$$\begin{cases} \mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k), \\ z(k) = \mathbf{C}\mathbf{x}(k), \\ \mathbf{u}(k) = \phi_{\mathbf{u}}(\mathbf{w}(k), k), \\ \mathbf{y}(k) = \phi_{\mathbf{y}}(z(k), k), \\ \mathbf{x} \in \mathbb{R}^n, \mathbf{w} \in \mathbb{R}^m, \mathbf{z} \in \mathbb{R}^p, \\ \phi_{\mathbf{u}} : \mathbb{R}^m \times \mathbb{N} \rightarrow \mathcal{U} \subseteq \mathbb{R}^m, \\ \phi_{\mathbf{y}} : \mathbb{R}^p \times \mathbb{N} \rightarrow \mathcal{Y} \subseteq \mathbb{R}^p, \end{cases} \quad (6.1)$$

where \mathcal{U} and/or \mathcal{Y} are typically discrete sets (Fig. 6.2). Here, $\mathbf{w}(k)$ and $z(k)$ are continuous variables. The input of the system is $\mathbf{w}(k)$, and the output is $\mathbf{y}(k)$. $\phi_{\mathbf{u}}$ and $\phi_{\mathbf{y}}$ are the input and output quantization maps, respectively. We will refer to quantization maps as *static* maps if they are not explicitly depending on time k and as *dynamical* maps otherwise. If $\mathbf{C} = \mathbf{I}$, then $\phi_{\mathbf{y}}$ is referred to as a state quantizer, and denoted by $\phi_{\mathbf{x}}$. The quantized system is the cascade connection of the input quantizer $\phi_{\mathbf{u}}$, of the linear system $\Sigma(\mathbf{A}, \mathbf{B}, \mathbf{C})$ and of the output quantizer $\phi_{\mathbf{y}}$.

In this framework, a controller is a dynamical system mapping the system outputs into inputs for the system. Depending on the problem, the input and/or the output

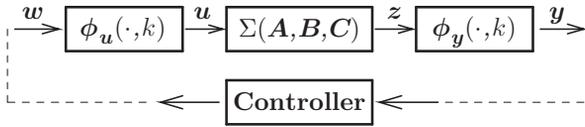


Fig. 6.2 Quantized control loop.

quantizers may be assigned or be a design objective. Accordingly, the quantizers may be considered as part of the plant (in which case, the control law would map $y(\cdot)$ into $w(\cdot)$), or as part of the controller mapping $z(\cdot)$ into $u(\cdot)$.

The hybrid nature of the model is caused by the co-existence of continuous state variables with discrete input and output variables. In particular, when the controller is regarded as a system that maps sequences of outputs in a finite set \mathcal{Y} into sequences of inputs in a finite set \mathcal{U} , the mixed logical dynamical nature of the overall system clearly appears.

Motivation for introducing quantizers The presence of quantization is traditionally believed to play adversely on control performance. However, its introduction has profound practical and technological motivations deeply rooted in the following two classes of problems.

Case 1: Control under communication constraints Consider the control of a system whose input and output variables take values in continuous spaces (e.g. $u \in \mathbb{R}^m$ and $z \in \mathbb{R}^p$), but where the flow of information between the various components of the loop is subject to communication constraints. In this case, the input and output variables have to be quantized and encoded into discrete-valued variables, which are suitable for the transmission. In this framework, the quantizers ϕ_u and ϕ_y as well as the discrete sets \mathcal{U} and \mathcal{Y} are design parameters to be chosen to accomplish the desired goals and to satisfy the communication constraints. Motivated by the spreading of technological applications involving complex and distributed networked systems, this has been one of the most active research areas on quantized control, and also appears to be the predominant trend for the near future.

Case 2: Control with discrete sensors and actuators Digital controllers interact with the environment by means of analog-to-digital converters or digital-to-analog converters that have a finite resolution, which may even be very coarse (e.g. think of a stepper motor or a low-resolution camera). In these cases, the input and/or output variables are inherently quantized. As a result, the set of control actions \mathcal{U} in the system (6.1) and the output quantizer ϕ_y modeling the sensors are fixed, while ϕ_u can be chosen as part of the controller. Prominent issues are in this case to sort out the achievable control goals and to synthesize a controller for maximizing performance.

These two classes of problems offer a good classification of the main avenues taken by the researchers that in the past years have coped with the problems of

quantized control (see [481] and some of the most significant references listed in the bibliography). The complexity of these problems requires a multi-disciplinary approach merging the traditionally separate expertise of the control, communication, and theoretic computer science communities.

Table 6.1 and 6.2 provide a list of the problems faced in this section, along with a pointer to the part of the section where each problem is analyzed and to the related reference. The meaning of the symbols in the tables are as follows: \vee = or; \wedge = and; D = discrete-time; C = continuous-time; (nl) = nonlinear; \mathcal{X} = state quantization; \mathcal{Y} = output quantization; $\hat{\mathcal{X}}$ = quantization of the state estimation \hat{x} ; \mathcal{E} = quantization of the innovation $Cx - C\hat{x}$; \mathcal{U} = input quantization; Dyn = dynamical quantizer; St = static quantizer; (h) = topic only hinted to in the chapter.

Table 6.1 Quantizers to be designed.

Reference	Time	Type	Quantization		In this handbook
[119, 401]	D \vee C	MIMO	$\mathcal{X} \vee \mathcal{Y} \vee \mathcal{U}$	Dyn	Sec. 6.2.3, Part 1
[402, 515]	C	MIMO (nl)	\mathcal{X}	Dyn	Sec. 6.2.3, Part 1 (h)
[615]	D	MIMO	$\mathcal{X} \vee \mathcal{E}$	Dyn	Sec. 6.2.3, Part 1 (h)
[480]	D	MIMO	$\hat{\mathcal{X}}$	Dyn	Sec. 6.2.3, Part 1 (h)
[408]	D	MIMO	\mathcal{X}	Dyn	Sec. 6.2.3, Part 1 (h)
[219]	D \vee C	SISO	$\mathcal{U} \vee \mathcal{E}$	St	Sec. 6.2.3, Part 2
[255]	D	MIMO	$\mathcal{U} \vee \mathcal{Y}$	St	Sec. 6.2.3, Part 2
[338]	C	SISO	\mathcal{X}	St	Sec. 6.2.3, Part 2 (h)
[163]	C	SISO (nl)	\mathcal{U}	St	Sec. 6.2.3, Part 2 (h)
[132]	C	MIMO	\mathcal{X}	St	Sec. 6.2.3, Part 2 (h)
[208, 225, 226]	D	SISO	$\mathcal{U} \vee \mathcal{X}$	St \vee Dyn	Sec. 6.2.3, Part 3
[32]	C	SISO	\mathcal{U}	St	Sec. 6.2.3, Part 3 (h)

Table 6.2 Assigned quantization.

Reference	Time	Type	Quantization	In this handbook
[520]	D	SISO	$\mathcal{U} \wedge (\mathcal{X} \vee \mathcal{Y})$	Sec. 6.2.4, Part 1
[519]	D	MIMO	\mathcal{U}	Sec. 6.2.4, Part 1 (h)
[207]	D	MIMO	\mathcal{X}	Sec. 6.2.4, Part 1 (h)
[521, 546]	D	SISO	\mathcal{U}	Sec. 6.2.4, Part 2 (h)
[610]	D	MIMO	\mathcal{U}	Sec. 6.2.4, Part 2 (h)
[522]	D	SISO	\mathcal{U}	Sec. 6.2.5

The rest of this section is organized as follows: in (Section 6.2.2) the main issues on the stabilization problem for quantized systems are introduced and illustrated through simple examples. The overview on the literature and some of the most

significant stabilization techniques are illustrated in the subsequent sections. (Section 6.2.3) deals with the control under communication constraints, (Section 6.2.4) is on the control with discrete sensors and/or actuators, whereas in (Section 6.2.5) a problem where the two frameworks are combined is presented.

6.2.2 Issues on the stabilization problem for quantized systems

The input and output quantizers Instrumental to a description of the main issues related to stabilization of quantized linear systems is a precise mathematical characterization of input and output quantizers. The input quantizer ϕ_u is supposed to take values in a *quantized set* $\mathcal{U} \subset \mathbb{R}^m$:

Definition 6.1 (Quantized set) A set $\mathcal{U} \subset \mathbb{R}^m$ is said to be *quantized* iff it is a closed set and $\mathcal{U} \setminus \{0\}$ is made of isolated points. If also 0 is an isolated point, then \mathcal{U} is said to be *strictly quantized*.

A strictly quantized set \mathcal{U} is such that any bounded subset of \mathbb{R}^m contains only a finite number of elements of \mathcal{U} (i.e. it is a *locally finite set*).

Example 6.1 Quantized sets

1. Any finite set $\mathcal{U} \subset \mathbb{R}^m$ is a strictly quantized set.
2. $\mathcal{U} = \mathbb{Z}^m \subset \mathbb{R}^m$ is a strictly and *uniformly* quantized set.
3. $\mathcal{U} = \{0\} \cup \{\pm u_0 \theta^h \mid h \in \mathbb{Z}\} \subset \mathbb{R}$, for some $u_0 > 0$ and $\theta > 1$, is a *logarithmically* quantized set of parameter θ . If $h \in \mathbb{N}$, then \mathcal{U} is a strictly and logarithmically quantized set. \square

A special type of static input quantizer is the so-called *nearest-neighbor* quantizer: let $\mathcal{U} \subset \mathbb{R}^m$ be a quantized set, $\phi_u : \mathbb{R}^m \rightarrow \mathcal{U}$ is said to be a nearest neighbor quantizer iff $\forall w \in \mathbb{R}^m$, $\phi_u(w)$ is an element of \mathcal{U} minimizing the Euclidean distance from w . The nearest-neighbor quantizer is well-defined because \mathcal{U} is a closed set.

As for the output quantizer ϕ_y , $\forall k \in \mathbb{N}$, let $\{\mathcal{C}_{y,k}\}_{y \in \mathcal{Y}}$ be the output-space partition induced by $\phi_y(\cdot, k)$ (that is, $\mathcal{C}_{y,k} := \{z \in \mathbb{R}^p \mid \phi_y(z, k) = y\}$). We assume that $\forall k \in \mathbb{N}$, any bounded subset of \mathbb{R}^p intersects only a finite number of elements of the output-space partition (i.e. $\{\mathcal{C}_{y,k}\}_{y \in \mathcal{Y}}$ is a *locally finite* partition).

Practical stabilization A first important issue related to the stabilization of system (6.1) emerges when the control law $u(\cdot)$ takes values in a strictly quantized set (either because \mathcal{U} is strictly quantized or as a consequence of the output quantization). In this case, the Lyapunov stabilization of $x = 0$ is not possible [207] for open-loop unstable systems. Indeed, if $0 \in \mathcal{U}$ is an isolated point and $0 \in \mathbb{R}^n$ is a stable equilibrium for the closed-loop system, then $u(\cdot)$ must be zero whenever the state of the system is in a suitable neighborhood of the equilibrium. Therefore,

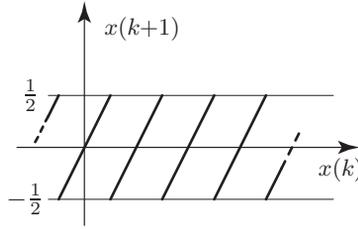


Fig. 6.3 Closed-loop dynamics in case 1 of Example 6.2 ($\alpha = 2$).

in such a neighborhood, the closed-loop dynamics coincides with the unstable open-loop dynamics. This simple consideration inspires a weaker notion of stability for quantized systems: the so-called *practical stability*.

Example 6.2 Practical stability of quantized feedback systems

Consider a scalar system

$$\begin{cases} \mathbf{x}(k+1) = \alpha \mathbf{x}(k) + \mathbf{u}(k), \\ \mathbf{y}(k) = \mathbf{x}(k), \\ |\alpha| > 1, \\ \mathbf{u} \in \mathcal{U} \subset \mathbb{R}, \end{cases} \tag{6.2}$$

where \mathcal{U} is an assigned and strictly quantized set. Let the control law be $\mathbf{u}(\mathbf{x}) = \phi_u(\mathbf{w}(\mathbf{x}))$, where ϕ_u is a nearest-neighbor quantizer and $\mathbf{w}(\mathbf{x}) = -\alpha \mathbf{x}$ holds. That is, $\mathbf{u}(\mathbf{x})$ is a quantized version of the classical deadbeat controller. The corresponding closed-loop dynamics is

$$\mathbf{x}(k+1) = \phi_e(-\alpha \mathbf{x}(k)),$$

where $\phi_e(\mathbf{w}) := \phi_u(\mathbf{w}) - \mathbf{w}$ is the *quantization error*. Let us analyze this dynamics for three different strictly quantized sets:

1. If $\mathcal{U} = \mathbb{Z}$, then $\forall \mathbf{x}(k) \in \mathbb{R}, |\mathbf{x}(k+1)| \leq 1/2$ (Fig. 6.3). Namely, for $k \geq 1$, all the trajectories are confined inside the interval $\Omega_1 := [-1/2, 1/2]$.

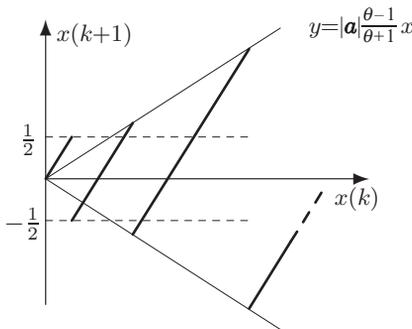


Fig. 6.4 Closed-loop dynamics in case 2 of Example 6.2 ($\alpha = 2$ and $\theta = 7/3$).

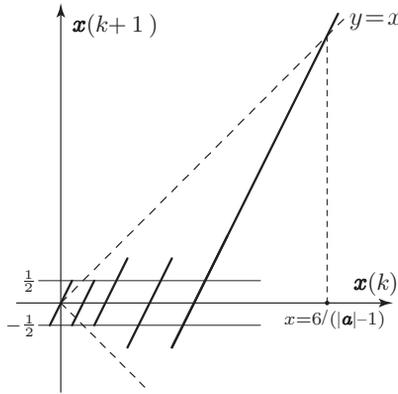


Fig. 6.5 Closed-loop dynamics in case 3 of Example 6.2 ($a = 2$).

2. If \mathcal{U} is a strictly and logarithmically quantized set of parameter θ , with $u_0 = 1$ and θ such that $|a|(\theta-1)/(\theta+1) < 1$, then: for $|x(k)| \leq 1/2$, it holds that $|x(k+1)| \leq 1/2$; whereas for $|x(k)| > 1/2$, $|x(k+1)| \leq |a| \frac{\theta-1}{\theta+1} |x(k)|$ (Fig. 6.4). Thus, for $\Omega_1 := [-1/2, 1/2]$, it holds that $\forall x(k) \in \Omega_1, x(k+1) \in \Omega_1$ and $\forall x_0 \in \mathbb{R}, \exists k \in \mathbb{N}$ such that $x(k) \in \Omega_1$: also in this case all the trajectories eventually enter a neighborhood Ω_1 of the equilibrium and remain confined therein.
3. If $\mathcal{U} = \{0\} \cup \{\pm 1, \pm 2, \pm 4, \pm 6\}$ and $a < 3$, then: for $|x(k)| \leq 1/2$, it holds that $|x(k+1)| \leq 1/2$; for $1/2 < |x(k)| < 6/(|a|-1)$, $|x(k+1)| < |x(k)|$ and for $|x(k)| \geq 6/(|a|-1)$, $|x(k+1)| \geq |x(k)|$ (Fig. 6.5). Thus, with $\Omega_1 := [-1/2, 1/2]$ and $\Omega_0 := [-\chi_0/2, \chi_0/2]$ (for any $\chi_0 \in [0, 12/(|a|-1)]$), it holds that $\forall x(k) \in \Omega_1, x(k+1) \in \Omega_1$ and $\forall x_0 \in \Omega_0, \exists k \in \mathbb{N}$ such that $x(k) \in \Omega_1$. That is, all the trajectories starting from Ω_0 eventually enter a neighborhood Ω_1 of the equilibrium and remain confined therein. \square

As shown in the example, although $x = 0$ is an unstable equilibrium for the closed-loop dynamics, it is possible to make the trajectories nondivergent (in the finite control set case, this is possible only starting from a sufficiently small neighborhood Ω_0 of the equilibrium), and, even better, to make them convergent to a bounded neighborhood Ω_1 of the equilibrium. This is the closest behavior to stability that one can obtain by the strictly quantized control of an unstable linear system and it is an example of the typical behavior of the so-called *practically stable* systems.

The following definition, stated for discrete-time and time-invariant systems, subsumes many notions of practical stability introduced for quantized systems in the literature:

Definition 6.2 (Practical stability) Consider a discrete-time time-invariant system

$$x(k+1) = f(x(k)). \quad (6.3)$$

A set $\Omega \subseteq \mathbb{R}^n$ is said to be invariant iff $\forall x(k) \in \Omega, x(k+1) \in \Omega$. Let $\Omega_1 \subset \Omega_0 \subseteq \mathbb{R}^n$ be such that Ω_1 is a bounded neighborhood of 0. The

system (6.3) is said to be (Ω_0, Ω_1) -stable iff both Ω_0 and Ω_1 are invariant and $\forall \mathbf{x}_0 \in \Omega_0, \exists \bar{k} \in \mathbb{N}$ such that $\forall k \geq \bar{k}, \mathbf{x}(k) \in \Omega_1$.

Lyapunov stabilization In a different situation in which \mathcal{U} is not a strictly quantized set, we can overcome the limitations described above and achieve asymptotic Lyapunov stabilization. For instance, let $\mathbf{u}(\mathbf{x})$ be a quantized controller for system (6.1) ensuring that the closed-loop dynamics is (Ω_0, Ω_1) -stable with $\Omega_1 = \delta\Omega_0$, for some $0 < \delta < 1$. Assume that, as soon as the trajectory enters Ω_1 , the control set \mathcal{U} (as well as the control law) can be scaled so that the invariance of Ω_1 is still guaranteed but convergence to a smaller Ω_2 is ensured. For instance, within Ω_1 , the control law is defined by $\mathbf{u}^{(1)}(\mathbf{x}) := \delta\mathbf{u}(\mathbf{x}/\delta)$. Such a law is (Ω_1, Ω_2) -stabilizing with $\Omega_2 = \delta\Omega_1$. The iteration of this control policy results in a signal $\mathbf{u}(k)$ taking values in a quantized set having 0 as an accumulation point and it provably achieves closed-loop stability of the system. Such a technique is known in the literature as the “zooming-in” [119] or “nesting” technique [226], according to the way it is implemented.

Stabilization through a “zooming-in” technique can be usefully studied in terms of small-gain theorems [404].

The usefulness of small-gain conditions for the stabilization of quantized systems is shown in the following example.

Example 6.3 *Small-gain condition for the stability of quantized feedback systems*

Consider system (6.2), where \mathcal{U} is a logarithmically quantized set of parameter θ (the value of u_0 is not relevant). Let the control law be $\mathbf{u}(\mathbf{x}) = \phi_{\mathbf{u}}(\mathbf{w}(\mathbf{x}))$, where $\phi_{\mathbf{u}}$ is a nearest-neighbor quantizer and $\mathbf{w}(\mathbf{x}) = \mathbf{K}\mathbf{x}$, $\mathbf{K} \in \mathbb{R}$ such that $|\mathbf{a} + \mathbf{K}| < 1$ holds and, consequently, $\mathbf{w}(\mathbf{x})$ is the quantized version of a stabilizing control law. The closed-loop dynamics is

$$\mathbf{x}(k+1) = (\mathbf{a} + \mathbf{K})\mathbf{x}(k) + \phi_e(\mathbf{K}\mathbf{x}(k)).$$

This is the feedback interconnection of the asymptotically stable linear system $\Sigma(\mathbf{a} + \mathbf{K}, \mathbf{1}, \mathbf{K})$ with the nonlinearity ϕ_e representing the quantization error.

If \mathbf{K} is such that $\forall \mathbf{x}(k) \neq 0, |\mathbf{x}(k+1)| < |\mathbf{x}(k)|$, then $\mathbf{x} = 0$ is a globally asymptotically stable equilibrium for the closed-loop system. Simple geometric arguments (Fig. 6.6) allow one to prove that a tight condition on \mathbf{K} ensuring that this property holds is

$$|\mathbf{a} + \mathbf{K}| + \frac{\theta - 1}{\theta + 1} |\mathbf{K}| < 1. \tag{6.4}$$

With $\gamma_e(\theta) := \frac{\theta-1}{\theta+1}$ and $\gamma_s(\mathbf{K}) := \frac{|\mathbf{K}|}{1-|\mathbf{a}+\mathbf{K}|}$, under the assumption that $|\mathbf{a} + \mathbf{K}| < 1$, condition (6.4) is equivalent to

$$\gamma_s(\mathbf{K}) \cdot \gamma_e(\theta) < 1. \tag{6.5}$$

This inequality is a *small-gain* condition [359]. Indeed, $\gamma_s(\mathbf{K})$ is the ℓ_2 -gain (or, equivalently, the H_∞ -norm) of the linear system $\Sigma(\mathbf{a} + \mathbf{K}, \mathbf{1}, \mathbf{K})$, whereas $\gamma_e(\theta)$ is the ℓ_2 -gain of the nonlinearity ϕ_e .

The set of gains \mathbf{K} ensuring that the small-gain condition is satisfied is strictly contained in the set of the stabilizing gains for the linear system $\Sigma(\mathbf{a}, \mathbf{1}, \mathbf{1})$ (see condition (6.4)). If θ

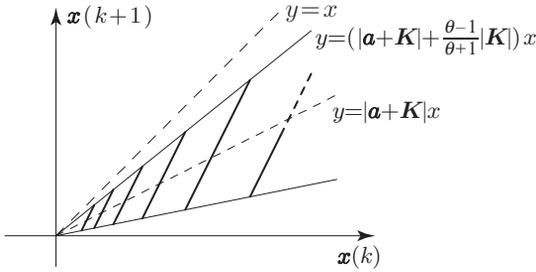


Fig. 6.6 Closed-loop dynamics in Example 6.3 ($a = 2$, $K = -3/2$ and $\theta = 3/2$).

is too large (i.e. if $\gamma_e(\theta)$ is too close to 1), such a set may be empty. Indeed, the minimum of $\gamma_s(K)$ in the set of the gains $K \in \mathbb{R}$ such that $|a + K| < 1$ is attained for $K = -a$, yielding $\gamma_s(-a) = |a|$. Hence, a sufficient condition for the existence of $K \in \mathbb{R}$ ensuring closed-loop Lyapunov stability is $|a|^{\frac{\theta-1}{\theta+1}} < 1$, that is

$$\theta < \frac{|a| + 1}{|a| - 1}.$$

If one has the freedom to choose both the quantized set \mathcal{U} and the control gain K , then, $\forall |a| > 1$, there exists a logarithmically quantized set, constructed using a parameter θ with $\theta \in]1, \frac{|a|+1}{|a|-1}[$, such that the system is stabilizable. Since $\frac{|a|+1}{|a|-1}$ is decreasing with $|a| > 1$, strongly unstable systems require *dense* quantization (corresponding to small feasible values for θ). \square

Nonlinear behaviors in quantized linear systems Because of the presence of discrete variables, the closed-loop dynamics of a quantized linear system is nonlinear and may exhibit such features as multiple isolated equilibria, limit cycles and chaotic behaviors. For instance, referring to Remark 6.3, when K is chosen so that $\gamma_s(K) \cdot \gamma_e(\theta) = 1$, the closed-loop dynamics has multiple isolated equilibria and/or limit cycles depending on the sign of a and $a + K$. Whereas, if $a \in \mathbb{Z}$, the closed-loop behavior within Ω_1 in case 1 of Example 6.2 is the prototype of discrete-time chaotic dynamics.

6.2.3 Control under communication constraints

This section deals with quantized feedback systems where the quantizers are introduced to cope with communication constraints. An important field of application is provided by networked control systems where the components of the control loop are connected by a digital communication network.

Part 1: Joint design of the control law and of the communication protocol One of the most important methodological results of the recent literature states that, for

control problems under communication constraints, the two issues of control synthesis and communication protocol design have to be jointly considered [668]. We show this fact by an example that is illustrative of a methodology applicable to more general cases.

Example 6.4 *Control of moving agents over a wireless connection*

Consider a continuous-time version of system (6.1) and assume that the controller is embedded into the plant but receives its sensor data transmitted through a channel capable of transmitting at rate R (namely, only $2^{R\tau}$ symbols can be exchanged during the time interval τ). As a reference example for this model, consider the feedback control of some agents moving in an environment whose current state is detected by remote sensors like a video camera that can communicate the state of the system to the controller through a finite rate digital communication bus or through a wireless connection.

A stabilizing control law can be constructed as follows. The system is sampled at times $k\tau$ ($k \in \mathbb{N}$, $\tau > 0$ is fixed) at which the controller receives a quantized measurement of the state. Therefore, we are in the quantized state framework ($\phi_y = \phi_x$) but the set \mathcal{U} is not quantized. The control law $u(t)$ and the communication protocol are defined below.

Control law

1. Consider $K \in \mathbb{R}^{m \times n}$ such that $A + BK$ is a Hurwitz matrix.
2. Fix $M \in \mathbb{N}$, $M \geq 2$, such that $N := M^n \leq 2^{R\tau}$ (assume this can be done).
3. Assume that $x_0 \in B_\infty^{(0)}(0, E_0) := \{x \in \mathbb{R}^n \mid \|x\|_\infty \leq E_0\}$, $E_0 > 0$. Divide each edge of this set (box) into M equal segments so that $B_\infty^{(0)}(0, E_0)$ turns out to be divided into N equal square sub-boxes. Let

$$\phi_x(x, 0) := \hat{x}_0,$$

where \hat{x}_0 is the center of the sub-box containing x_0 (Fig. 6.7). It holds that $\|x_0 - \hat{x}_0\|_\infty \leq E_0/M$.

4. For $t \in [0, \tau[$, let $\hat{x}(t) := e^{(A+BK)t} \hat{x}_0$ and

$$u(t) := K\hat{x}(t).$$

5. Let

$$A := \max_{0 \leq t \leq \tau} \|e^{At}\|_\infty$$

and $c(1) := \lim_{t \rightarrow \tau^-} \hat{x}(t)$: it holds that $x(t) - \hat{x}(t) = e^{At}(x_0 - \hat{x}_0)$, thus

$$\forall t \in [0, \tau[, \quad \|x(t) - \hat{x}(t)\|_\infty \leq A\|x_0 - \hat{x}_0\|_\infty \leq \frac{A}{M}E_0 := E(1).$$

In particular, $x(\tau) \in B_\infty^{(1)}(c(1), E(1)) := \{x \in \mathbb{R}^n \mid \|x - c(1)\|_\infty \leq E(1)\}$.

6. Iterate the procedure, that is

(a) Assume that at time $k\tau$ the state of the system belongs to the box

$$B_\infty^{(k)}(c(k), E(k)) := \{x \in \mathbb{R}^n \mid \|x - c(k)\|_\infty \leq E(k)\},$$

for some $c(k) \in \mathbb{R}^n$ and $E(k) > 0$, divide this box into N equal square sub-boxes and let $\phi_x(x, k) := \hat{x}(k\tau)$, where $\hat{x}(k\tau)$ is the center of the sub-box containing $x(k\tau)$. It holds that $\|x(k\tau) - \hat{x}(k\tau)\|_\infty \leq E(k)/M$.

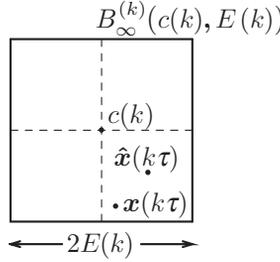


Fig. 6.7 State quantizer ϕ_x with $n = 2$ and $M = 2$ ($N = 4$).

(b) For $t \in [k\tau, (k+1)\tau]$, let $\hat{x}(t) := e^{(A+BK)(t-k\tau)} \hat{x}(k\tau)$ and

$$u(t) := K\hat{x}(t).$$

(c) Let $c(k+1) := \lim_{t \rightarrow (k+1)\tau^-} \hat{x}(t)$: it holds that

$$\forall t \in [k\tau, (k+1)\tau], \quad \|\mathbf{x}(t) - \hat{\mathbf{x}}(t)\|_\infty \leq \frac{\Lambda}{M} E(k) := E(k+1). \quad (6.6)$$

In particular, $\mathbf{x}((k+1)\tau) \in B_\infty^{(k+1)}(c(k+1), E(k+1))$.

Communication protocol If the controller knows E_0 , Λ , and the updating rule of the boxes, then, at each time $k\tau$, the controller knows the N possible values taken by $\phi_x(\mathbf{x}, k)$. Therefore, once a rule for the enumeration of the sub-boxes has been established, to implement the control law the controller only needs to know the index corresponding to the sub-box where the current state is. Namely, only one symbol among $\{1, \dots, N\}$ is to be transmitted to the controller at each sampling time. \square

The correctness of the procedure given in the example is shown in the following:

Proposition 6.1 [401] *In the above setting, if*

$$\frac{\Lambda}{M} < 1, \quad (6.7)$$

then $\mathbf{x} = 0$ is a locally asymptotically stable equilibrium point for the closed-loop system and $\forall \mathbf{x}_0 \in B_\infty^{(0)}(0, E_0)$, $\lim_{t \rightarrow +\infty} \mathbf{x}(t) = 0$.

Proof Let us prove convergence: with the control law defined above, the closed-loop dynamics is $\dot{\mathbf{x}}(t) = (\mathbf{A} + \mathbf{BK})\mathbf{x}(t) + \mathbf{e}(t)$, where $\mathbf{e}(t) = \mathbf{BK}(\hat{\mathbf{x}}(t) - \mathbf{x}(t))$. If inequality (6.7) is satisfied, then $\lim_{k \rightarrow +\infty} E(k) = 0$ and, by inequality (6.6), $\lim_{t \rightarrow +\infty} \mathbf{e}(t) = 0$. Since $\mathbf{A} + \mathbf{BK}$ is a Hurwitz matrix and an asymptotically stable linear system is input-to-state stable, then also $\lim_{t \rightarrow +\infty} \mathbf{x}(t) = 0$.

Lyapunov stability can be proved by considering a quadratic Lyapunov function $V(\mathbf{x}) = \mathbf{x}^\top \mathbf{P}\mathbf{x}$ for the ideal (i.e. nonquantized) system $\dot{\mathbf{x}}(t) = (\mathbf{A} + \mathbf{BK})\mathbf{x}(t)$. \square

Remark 6.1 The result above provides only local stability because it requires that the initial state be contained in the ball $B_\infty^{(0)}(0, E_0)$. We can overcome this limitation and produce a global stability result by extending the stabilizing control law and communication protocol with a “zooming-out” phase: if $\|x_0\|_\infty > E_0$, then transmit the symbol $N + 1$ to the controller and, for $t \in [0, \tau[$, let $u(t) = 0$. Let $E(1) := \alpha E_0$, with fixed $\alpha > \Lambda$. Repeat until k is found such that $\|x(k\tau)\|_\infty \leq \alpha^k E_0$ (such a k exists because $\alpha > \Lambda$, that is, the expansion of the box dominates the divergence rate of the system), then proceed with the control and communication protocol described above.

Remark 6.2 The presence of a communication constraint in the stabilization of a continuous-time system has a profound impact. Indeed, if the constraint $N \leq 2^{R\tau}$ is not imposed, then condition (6.7), and hence stabilization, can be always achieved even if M is fixed: it is sufficient to reduce the sampling period (so that $\Lambda \simeq 1$). In other words, it is possible to approximate any continuous-time signal by switching very fast between discrete values, as it happens with the pulse width modulator control of an on/off DC motor. On the contrary, the presence of a communication constraint induces a lower bound on the sampling period inhibiting this possibility.

The extension of the proposed technique to the discrete-time case is straightforward: it is sufficient to replace Λ with $\|A\|_\infty$. Also the extension to the output quantization case can be derived by the construction of a state observer. Finally, the crucial property in the proof of Proposition 6.1 is the input-to-state stability of the ideal (i.e. nonquantized) underlying dynamics: this fact makes for the generalization of this technique to more general nonlinear systems [402, 515].

Condition (6.7) can be rephrased by saying that closed-loop asymptotic stability is achievable provided that the expansion Λ due to the instability of the open-loop system can be outweighed by a sufficiently fine partition of the state space and this is possible only if the capacity of the channel is sufficient.

The same idea, and suitable adjustments of the technique described above, has been used to determine the necessary and sufficient condition on the rate R of the channel so that closed-loop asymptotic stabilization can be achieved [480, 615]. For discrete-time systems, the condition is

$$R > \sum_{|\lambda_i(\mathbf{A})| > 1} \log_2 |\lambda_i(\mathbf{A})| := R_{\min}. \tag{6.8}$$

In [408], a similar analysis is offered allowing for the possibility of packet losses.

Part 2: Minimum density of quantization to achieve Lyapunov stability Let us go further inside the problem of understanding the minimum information needed to carry out a stabilization task. In this respect, the question we want to answer is the following:

Stabilization problem with minimum feedback information

For a given stabilizable discrete-time linear system $\Sigma(\mathbf{A}, \mathbf{B}, \mathbf{C})$, what is the minimum *density* for quantizers ϕ_u and ϕ_y so that stabilizability is preserved for the resulting quantized control system?

It has been proved in [219] that this goal is achieved by logarithmic quantizers. This result formalizes a natural intuition: if the system state is far away from the equilibrium we can use a coarse control action, whereas if the state is approaching the equilibrium we need “fine-grained” corrections. Let us formally state the problem.

Given a quantized set $\mathcal{U} \subset \mathbb{R}$, for $0 < \epsilon \leq 1$, denote by $\#\mathcal{U}[\epsilon]$ the cardinality of $\mathcal{U} \cap [\epsilon, 1/\epsilon]$. The *density* of \mathcal{U} is defined by

$$\eta(\mathcal{U}) := \limsup_{\epsilon \rightarrow 0^+} \frac{\#\mathcal{U}[\epsilon]}{-\log \epsilon}.$$

Consider the system (6.1) and assume that $\mathbf{y} = \mathbf{x}$, the system is single-input ($u \in \mathbb{R}$) and the pair (\mathbf{A}, \mathbf{B}) is stabilizable. Let $V(\mathbf{x}) = \mathbf{x}^T \mathbf{P} \mathbf{x}$ be a quadratic control Lyapunov function for the system and, hence, $\forall \mathbf{x} \in \mathbb{R}^n \setminus \{0\}, \exists \mathbf{w}(\mathbf{x}) \in \mathbb{R}$ such that $V(\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{w}(\mathbf{x})) - V(\mathbf{x}) < 0$. The problem is to find the least dense quantized set \mathcal{U} such that V remains a quadratic control Lyapunov function also if the control is forced to take values in \mathcal{U} . More precisely, in a first stage a static input quantizer ϕ_u and a state feedback $\mathbf{x} \mapsto \mathbf{w}(\mathbf{x}) \in \mathbb{R}$ are to be designed so that $\forall \mathbf{x} \in \mathbb{R}^n \setminus \{0\}, \Delta(V(\mathbf{x})) := V(\mathbf{A}\mathbf{x} + \mathbf{B}\phi_u(\mathbf{w}(\mathbf{x}))) - V(\mathbf{x}) < 0$ and the density of \mathcal{U} is minimized. In a second stage, the problem consists in minimizing this density over all quadratic control Lyapunov functions, thus providing the coarsest quantization that preserves stabilizability.

The control sets solving these problems are logarithmically quantized. Specifically, the least dense set \mathcal{U} solving the problem for a given $V(\mathbf{x}) = \mathbf{x}^T \mathbf{P} \mathbf{x}$ is logarithmic of parameter θ given by

$$\theta = \frac{\gamma_s(\mathbf{K}) + 1}{\gamma_s(\mathbf{K}) - 1}, \quad (6.9)$$

with $\gamma_s(\mathbf{K}) = 1/\sqrt{\mathbf{K}\mathbf{M}^{-1}\mathbf{K}^T} < 1$, where

$$\mathbf{M} = \frac{\mathbf{A}^T \mathbf{P} \mathbf{B} \mathbf{B}^T \mathbf{P} \mathbf{A} - (\mathbf{B}^T \mathbf{P} \mathbf{B})(\mathbf{A}^T \mathbf{P} \mathbf{A} - \mathbf{P})}{(\mathbf{B}^T \mathbf{P} \mathbf{B})^2} > 0$$

and

$$\mathbf{K} = -\frac{\mathbf{B}^T \mathbf{P} \mathbf{A}}{\mathbf{B}^T \mathbf{P} \mathbf{B}}.$$

Correspondingly, $\eta(\mathcal{U}) = 2/\log \theta$, $\phi_u : \mathbb{R} \rightarrow \mathcal{U}$ is a nearest-neighbor quantizer and a feasible state feedback is $\mathbf{w}(\mathbf{x}) = \mathbf{K}\mathbf{x}$.

The minimum density of \mathcal{U} over all quadratic control Lyapunov functions is attained by a logarithmically quantized set of parameter θ^* given by

$$\theta^* = \frac{\gamma^* + 1}{\gamma^* - 1}, \quad (6.10)$$

where

$$\begin{aligned} \gamma^* &= \prod_{|\lambda_i(\mathbf{A})| > 1} |\lambda_i(\mathbf{A})| \\ &= \inf\{\|\mathbf{G}_{\mathbf{K}}\|_{\infty} \mid \mathbf{K} \text{ such that } \mathbf{A} + \mathbf{BK} \text{ is Schur}\} \end{aligned} \quad (6.11)$$

and $\mathbf{G}_{\mathbf{K}}(z) := \mathbf{K}(z\mathbf{I} - \mathbf{A} - \mathbf{BK})^{-1}\mathbf{B}$. As before, $\eta(\mathcal{U}) = 2/\log \theta^*$, ϕ_u is a nearest-neighbor quantizer and a feasible state feedback is $\mathbf{w}(\mathbf{x}) = \mathbf{K}^*\mathbf{x}$, where \mathbf{K}^* is such that $\|\mathbf{G}_{\mathbf{K}^*}\|_{\infty}$ is close to the infimum in (6.11).

Remark 6.3 To be really precise, the reported results are such that $\Delta(V(\mathbf{x})) \leq 0$ (rather than $\Delta(V(\mathbf{x})) < 0$). In other words, the infimum of the considered minimization problems is not a minimum. To guarantee the strict decrease of V it is sufficient to choose a slightly larger density perturbing θ with $\theta - \epsilon$, where $\epsilon > 0$ is arbitrarily small. The same for θ^* .

Actually, both equations (6.9) and (6.10) can be interpreted as small-gain conditions in H_{∞} , namely, in terms of the ℓ_2 -gain of the closed-loop system and of the quantization error (cf. condition (6.5) in the degenerate case $\gamma_s(\mathbf{K}) \cdot \gamma_e(\theta) = 1$). Taking advantage of robust control techniques, it is then possible to extend the presented results to more general cases [255] such as systems with multiple inputs or with output quantization. Moreover, it is possible to include performance requirements such as guaranteed quadratic cost under quantized outputs or H_{∞} performance under quantized inputs.

Results for continuous-time systems The analysis shown above can be applied also to continuous-time systems as illustrated in [219]. In this case, the problem is to find the pair (τ, θ) , where τ is the sampling period and θ the density of the quantizer that simultaneously minimizes the densities of sampling and quantization. It turns out that the optimal pair (τ^*, θ^*) is such that τ^* is a function of the unstable poles of the open-loop system, while $\theta^* = 1 + \sqrt{2}$ is independent of the system and may be referred to as the universal constant of logarithmic quantization for single-input continuous-time systems.

In [338] some of the basic ideas on logarithmic quantization have been extended to continuous-time models. In [163] the robust control approach is further extended to nonlinear models by resorting to the theory of dissipative systems.

A related issue is that considered in [132], where the following problem is studied: given a continuous-time linear system $\Sigma(\mathbf{A}, \mathbf{B})$, a stabilizing control gain \mathbf{K} and an integer M , find a static state quantizer $\phi_{\mathbf{x}}$ taking M values and such that a so-called “destabilizing measure” is minimized (thus, ensuring good practical stability properties for the closed-loop dynamics $\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{BK}\phi_{\mathbf{x}}(\mathbf{x}(t))$). It is then shown that the quantizer design can be conveniently cast as a *locational optimization* problem and different algorithms to solve it are discussed.

Part 3: Performance/complexity trade-off Let us go back to condition (6.8) on the minimum rate of the channel that preserves stabilizability. When $R \simeq R_{\min}$, it has been shown in [32] that chaotic behaviors arise in the closed-loop dynamics.

The emergence of chaos in linear systems under quantized control (previously studied also in [207]) bears an unsuspected design opportunity for constructing efficient practically stabilizing controllers [225]. The idea is as much brilliant as it is simple: consider two neighborhoods $\Omega_1 \subset \Omega_0 \subset \mathbb{R}^n$ of the equilibrium and two control laws $k_i = \phi_u \circ K_i : \Omega_i \rightarrow \mathcal{U}$ ($i = 0, 1$) for some $K_i : \Omega_i \rightarrow \mathbb{R}^m$ and static input quantizer ϕ_u (while $\mathbf{y} = \mathbf{x}$). Assume that the following properties hold: for $i = 0, 1$, k_i ensures the invariance of Ω_i ; the closed-loop system under k_0 is such that, for any open set $S \subseteq \Omega_0$ and for almost all initial condition \mathbf{x}_0 , the corresponding trajectory eventually enters S . This property is implied by the existence of an *ergodic* measure defined on Ω_0 and absolutely continuous with respect to the Lebesgue measure. It is then clear that the control law defined by

$$k(\mathbf{x}) := \begin{cases} k_0(\mathbf{x}), & \text{if } \mathbf{x} \in \Omega_0 \setminus \Omega_1, \\ k_1(\mathbf{x}), & \text{if } \mathbf{x} \in \Omega_1, \end{cases}$$

has the property that for almost all $\mathbf{x}_0 \in \Omega_0$, $\exists k \in \mathbb{N}$ such that $\mathbf{x}(k) \in \Omega_1$. Hence, the closed-loop system is *almost surely* (Ω_0, Ω_1) -stable. Note that the convergence towards Ω_1 only depends on k_0 , it does not depend on Ω_1 . Thus, Ω_1 can be chosen to be any arbitrarily small neighborhood of the origin.

For scalar systems and for some cases of higher order, conditions on the matrices (\mathbf{A}, \mathbf{B}) have been provided ensuring that a control law k_0 can be constructed minimizing the number of control values and guaranteeing both the invariance of Ω_0 and the ergodicity property. Thus, with the effort needed to guarantee the invariance of Ω_0 , also convergence towards any neighborhood Ω_1 is ensured. As one expects, the smaller Ω_1 is, the larger is the mean time of convergence T taken by the trajectories to enter Ω_1 .

In more general terms, there exists a trade-off between *performance* of the closed-loop system and *complexity* of the controller. Let $\mathbf{u}(\mathbf{x}) = \phi_u(\mathbf{w}(\mathbf{x}))$ be a quantized controller so that the closed-loop dynamics is (Ω_0, Ω_1) -stable. The performance of the closed-loop system may be measured in terms of transient behavior and steady state. A good performance metric for the transient behavior is the mean time of convergence T . On the other hand, for the steady state, we can use as a performance metric the amount of contraction $C := \text{Volume}(\Omega_0)/\text{Volume}(\Omega_1)$. The complexity of the controller may be measured by the number N of control values taken by $\mathbf{u}(\mathbf{x})$. A triple (T, C, N) is feasible if there exists an (Ω_0, Ω_1) -stabilizing controller taking N values and so that the corresponding performance parameters are T and C . It can be shown that a necessary condition for the feasibility of a triple (T, C, N) is

$$\log C \leq T \log N.$$

Such an inequality eloquently points out the trade-off between performance and complexity, since a large contraction C or a low convergence time T entail large values for N . A more detailed analysis, including sufficient conditions of feasibility, can be found in [208, 226].

6.2.4 Control with quantized sensors and actuators

This section considers quantized feedback systems where the quantizers are modeling sensors providing discrete measurement information and actuators having discrete inputs.

Part 1: Practical stability We start by illustrating the analysis of the achievable stability properties as a function of the quantizers *resolution*. Consider system (6.1) and assume that a static state quantizer ϕ_x is assigned, the system is single-input with $\mathcal{U} \subset \mathbb{R}$ being a strictly quantized and assigned set, and the pair (\mathbf{A}, \mathbf{B}) is reachable. There is no further assumption on the structure of the assigned input and state quantization.

A controller yielding practical stability and the analysis of the resulting closed-loop dynamics can be remarkably simplified if we use the state-space model in controllable canonical form. Assume that $z^n - a_n z^{n-1} - \dots - a_2 z - a_1$ is the characteristic polynomial of \mathbf{A} . Let $\alpha := \sum_{i=1}^n |a_i|$ and assume $\alpha > 1$ (for $\alpha \leq 1$ the open-loop system is stable). For $\Delta > 0$, let

$$\mathcal{Q}_n(\Delta) := [-\Delta/2, \Delta/2]^n \subset \mathbb{R}^n$$

be the hypercube of edge length Δ . If $\mathbf{x}(k) \in \mathcal{Q}_n(\Delta)$ and $\mathbf{u} \in \mathcal{U}$, due to the canonical form of (\mathbf{A}, \mathbf{B}) ,

$$\mathbf{x}(k+1) = (x_2(k), \dots, x_n(k), \sum_i a_i x_i(k) + \mathbf{u})^T$$

and

$$\mathbf{x}(k+1) \in \mathcal{Q}_n(\Delta) \Leftrightarrow |\sum_i a_i x_i(k) + \mathbf{u}| \leq \Delta/2.$$

Hence, the invariance analysis for hypercubes is *reduced to a scalar problem* and practical stability is conveniently studied in terms of invariant hypercubes. Let

$$\mathbf{w}(\mathbf{y}) := \mathbf{K}_D \mathbf{y} = - \sum_{i=1}^n a_i y_i$$

be the deadbeat controller, consider the *qdb-controller*

$$\mathbf{u}(\mathbf{x}) := \phi_u(\mathbf{w}(\mathbf{y})),$$

where $\mathbf{y} = \phi_x(\mathbf{x})$ and ϕ_u is a nearest-neighbor quantizer. The closed-loop system is described by

$$\mathbf{x}(k+1) = (\mathbf{A} + \mathbf{B}\mathbf{K}_D)\mathbf{x}(k) + \mathbf{B}[\mathbf{K}_D\phi_{x_e}(\mathbf{x}(k)) + \phi_{u_e}(\mathbf{K}_D\phi_x(\mathbf{x}(k)))],$$

where $\phi_{u_e}(\mathbf{w}) := \phi_u(\mathbf{w}) - \mathbf{w}$ and $\phi_{x_e} := \phi_x(\mathbf{x}) - \mathbf{x}$ are the input and state quantization error, respectively. The practical stability properties of the closed-loop dynamics depend on the properties of the nonlinear functions ϕ_{u_e} and ϕ_{x_e} , which, in turn, are directly related to the geometric structure of the quantized input set \mathcal{U} and of the state-space partition induced by ϕ_x .

The relevant characteristics of the quantization errors can be easily expressed in terms of scalar functions $\rho_u(\Delta)$ and $H_y(\Delta)$ as follows. For a given $\Delta > 0$, the possibility of ensuring the invariance of the hypercube $\mathcal{Q}_n(\Delta)$ only depends on a bounded subset $\mathcal{U}(\Delta)$ of the whole control set \mathcal{U} . The function $\rho_u(\Delta)$ is defined as the maximum gap between consecutive elements of $\mathcal{U}(\Delta)$. Similarly, $H_y(\Delta)$ is related to the size of the elements \mathcal{C}_y 's of the state-space partition induced by ϕ_x intersecting $\mathcal{Q}_n(\Delta)$. Two auxiliary functions $m_u(\Delta) = \min \mathcal{U}(\Delta)$ and $M_u(\Delta) = \max \mathcal{U}(\Delta)$ which take the control authority into account are also considered. The main result is summarized by the following:

Proposition 6.2 [520] *In the above setting, let $\Delta_0 > 0$ be such that the following conditions are all satisfied:*

$$\rho_u(\Delta_0) + H_y(\Delta_0) < \Delta_0, \quad (6.12)$$

$$m_u(\Delta_0) < -\frac{\Delta_0}{2}(\alpha - 1), \quad (6.13)$$

$$M_u(\Delta_0) > \frac{\Delta_0}{2}(\alpha - 1). \quad (6.14)$$

Consider $\Delta_1 := \max\{\Delta < \Delta_0 \mid \rho_u(\Delta) + H_y(\Delta) = \Delta\}$, then the closed-loop dynamics under the qdb-controller is $(\mathcal{Q}_n(\Delta_0), \mathcal{Q}_n(\Delta_1))$ -stable.

This technique can be extended to the case of an assigned output quantization ϕ_y by including a state observer in the controller (cf. [520] for the details). Furthermore, also inequality (6.12) can be interpreted as a small-gain condition. This fact permits the presented approach to be extended to systems with assigned multi-input quantized sets $\mathcal{U} \subset \mathbb{R}^m$, as well as the consideration the quantization of other feedback laws $w(y) = Ky$ different from the deadbeat controller [519].

Other stabilization methods rely on the adjustment to quantized systems of traditional techniques based on Lyapunov theory [207]. However, the corresponding practical stability analysis is often conservative.

Part 2: Optimal control Different approaches are those based on optimal control techniques. In this case, the control law is derived by the solution of an optimization problem. Among these methods, a suitable framework is provided by model-predictive control techniques (MPC), due to their ability to effectively deal with constraints in the control action. A detailed study of this application of MPC techniques to quantized control can be found in [546], where the practical stabilizing control law is provided in closed form and analysis methods are included to determine the invariant sets for the corresponding closed-loop dynamics. The so-called *dual mode* MPC scheme is considered in [521]. In this case, the closed-loop invariance of a-priori determined sets can be enforced and the results on the invariance analysis of hypercubes presented in Part 1 of this section (see p. 000) can be profitably used.

Another method is that presented in [610], where the design of practically stabilizing controllers is converted into a nonlinear programming problem. Such an

approach can be applied to a wide class of hybrid systems including quantized input models as a particular case.

6.2.5 Channel sharing for systems with input constraints

In [522], the two cases of control under communication constraints and control under assigned quantization are blended and also a shared resource allocation problem is considered. A set of N linear systems under assigned input quantization is considered and the N control loops are supposed to be closed over a shared communication channel of total capacity R . The correct assignment of the shared resource allowing for stabilization is part of the problem. In this case, quantization is imposed at the actuators but the designer has the freedom to choose a subset of the assigned control values to the purpose of bounding the data rate requested by each control loop. Therefore, quantization is both a physical constraint *and* a means to enable communication over a finite capacity shared channel.

Formally, a set of N scalar linear systems is considered:

$$\begin{cases} \dot{\mathbf{x}}_j(t) = \mathbf{a}_j \mathbf{x}_j(t) + \mathbf{u}_j(t) + \mathbf{d}_j(t) \\ j = 1, \dots, N, \end{cases}$$

where the control function $\mathbf{u}_j(t)$ can take values in a uniformly quantized set $\mathcal{U}_j \subseteq \epsilon_j \mathbb{Z}$ ($\epsilon_j > 0$ is assigned) while $\mathbf{d}_j(t)$ represents an exogenous noise term ranging in the bounded set $\mathcal{I}(d_j) := [-d_j/2, d_j/2]$. The different systems are periodically sampled with period τ_j , which is to be properly chosen.

In this scenario, one has to face three different but related problems:

1. How can the performance of the control systems be quantified?
2. How is it related to the resource assignment and to the design parameters?
3. How is it possible to identify the best trade-off in the assignment of the shared communication capacity between the different loops?

Concerning the first problem, we know from the discussion in (Section 6.2.2) that the presence of an assigned strict quantization allows for practical stabilization only. Here we focus on the invariance of a target set $\mathcal{I}(\Delta_j) = [-\Delta_j/2, \Delta_j/2]$. Therefore, the size Δ_j of the sets that can be made invariant is a coherent notion of performance.

As far as the second problem is concerned, the main design variable is the fraction \mathcal{R}_j of the shared resource capacity that can be assigned to the different loops. This parameter is influenced by the sampling period and by the cardinality of the control set \mathcal{U}_j by the obvious relation: $\#\mathcal{U}_j \leq 2^{\lceil \mathcal{R}_j \tau_j \rceil}$. Therefore, the combined choice of the sampling period τ_j and of the control set \mathcal{U}_j leads to an allocated rate \mathcal{R}_j and to a subsequent performance Δ_j . More formally, a triple $(\mathcal{R}_j, \tau_j, \Delta_j)$ is feasible if there exists a $\mathcal{U}_j \subset \epsilon_j \mathbb{Z}$ so that the interval $\mathcal{I}(\Delta_j)$ can be made invariant.

Given this background, it is possible to formalize the design problem as an optimization in which decision variables are the vectors $\vec{\Delta} = [\Delta_1, \dots, \Delta_N]$, $\vec{\tau} = [\tau_1, \dots, \tau_N]$, $\vec{\mathcal{R}} = [\mathcal{R}_1, \dots, \mathcal{R}_N]$:

$$\begin{cases} \min_{(\vec{\mathcal{R}}, \vec{\tau}, \vec{\Delta})} f(\vec{\Delta}) \\ \text{subject to: } \sum_{j=1}^N \mathcal{R}_j \leq R, \quad (\vec{\mathcal{R}}, \vec{\tau}, \vec{\Delta}) \text{ feasible.} \end{cases} \quad (6.15)$$

One possibility for the choice of the cost function f is to relate it to the attainment of a specified vector $\vec{\Delta}^{(0)}$, e.g. $f(\vec{\Delta}) := \|\vec{\Delta}^{(0)} - \vec{\Delta}\|/\|\vec{\Delta}^{(0)}\|$.

An effective strategy for solving this problem is based on the explicit computation of the minimal cardinality $l_j(\Delta_j, \tau_j)$ of a control set achieving a given performance Δ_j with sampling period τ_j , for each loop. Based on this computation, it is possible to identify a function $R_j^{(\min)}(\Delta_j)$, which provides a lower bound for the rates \mathcal{R}_j for which there exists a sampling period τ_j such that the triple $(\mathcal{R}_j, \Delta_j, \tau_j)$ is feasible:

$$R_j^{(\min)}(\Delta_j) := \mathbf{a}_j / \log \left(1 + \frac{\mathbf{a}_j \Delta_j}{2\epsilon_j \left\lceil \frac{\mathbf{a}_j \Delta_j + d_j}{2\epsilon_j} \right\rceil + d_j} \right).$$

This expression is related to condition (6.8) and provides insight to the third question above. These computations make for a radical simplification of problem (6.15), which can be solved in two steps. First, a solution $\vec{\Delta}^*$ is found to the simplified problem:

$$\begin{cases} \min_{\vec{\Delta}} f(\vec{\Delta}) \\ \text{subject to: } \sum_{j=1}^N R_j^{(\min)}(\Delta_j) \leq R, \end{cases}$$

by using a standard branch and bound scheme (observe that $R_j^{(\min)}(\Delta)$ is only piecewise continuous). In the second step, using the expression for $l_j(\Delta_j, \tau_j)$, one can find the sampling periods $\vec{\tau}$ and the number of levels attaining the solution $\vec{\Delta}^*$.

6.3 Discrete-event modeling and diagnosis

6.3.1 Diagnostic problem

This section presents a method for fault diagnosis of systems whose input \mathbf{u} and output \mathbf{y} can only be accessed through quantizers (Fig. 6.8). The quantization is denoted by $[\cdot]$. The quantizers generate sequences

$$[\mathbf{U}(0\dots k_h)] = ([\mathbf{u}(0)], [\mathbf{u}(1)], \dots, [\mathbf{u}(k_h)]), \quad (6.16)$$

$$[\mathbf{Y}(0\dots k_h)] = ([\mathbf{y}(0)], [\mathbf{y}(1)], \dots, [\mathbf{y}(k_h)]), \quad (6.17)$$

of quantized input and output values. This section shows how quantized systems can be represented by stochastic automata and how the diagnostic problem can be solved by means of this model. For the diagnostic purposes, the model $\mathcal{M}(f)$ has to describe the system subject to the faults $f \in \mathcal{F}$, where \mathcal{F} denotes the set of possible faults.

Problem 6.1 (Diagnostic problem for quantized systems)

Given: Sequences (Eq. (6.16)), (Eq. (6.17)) of quantized input and output values.

Model $\mathcal{M}(f)$ of the quantized system subject to faults $f \in \mathcal{F}$.

Find: Current fault \bar{f} .

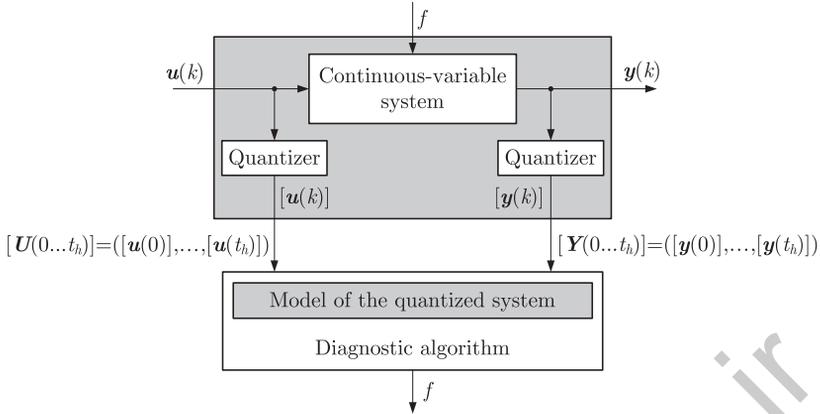


Fig. 6.8 Diagnosis of quantized systems.

As in quantized feedback systems, quantizers represent the effect of measurement uncertainties or are deliberately introduced to reduce the information to be processed by the diagnostic algorithm. Its use follows the guideline to solve fault diagnostic tasks with reasonable effort by ignoring as much information as possible.

6.3.2 Hybrid model of quantized systems

This subsection introduces the hybrid model consisting of a continuous-variable state-space model and quantizers.

The core of a quantized system is a nonlinear continuous-variable continuous-time system

$$\dot{\mathbf{x}}(t) = \mathbf{g}_c(\mathbf{x}(t), \mathbf{u}(t)), \quad \mathbf{x}(0) = \mathbf{x}_0, \tag{6.18}$$

$$\mathbf{y}(t) = \mathbf{h}_c(\mathbf{x}(t), \mathbf{u}(t)), \tag{6.19}$$

with input vector $\mathbf{u} \in \mathcal{R}^m$, output vector $\mathbf{y} \in \mathcal{R}^r$ and the vector of the internal state $\mathbf{x} \in \mathcal{R}^n$. The *quantizers* introduce partitions of the signal spaces \mathcal{R}^m , \mathcal{R}^n and \mathcal{R}^r into a finite number of disjoint sets $\mathcal{Q}_u(v)$ ($v \in \mathcal{V} = \{0, 1, \dots, M\}$) $\mathcal{Q}_x(z)$ ($z \in \mathcal{Z} = \{0, 1, \dots, N\}$) and $\mathcal{Q}_y(w)$ ($w \in \mathcal{W} = \{0, 1, \dots, R\}$). $\mathcal{Q}_u(v)$, denotes the set of input values \mathbf{u} , with the same quantized values v . The mapping invoked by the quantizers is symbolized by brackets $[\cdot]$:

$$[\mathbf{u}] = v \iff \mathbf{u} \in \mathcal{Q}_u(v), \tag{6.20}$$

$$[\mathbf{x}] = z \iff \mathbf{x} \in \mathcal{Q}_x(z), \tag{6.21}$$

$$[\mathbf{y}] = w \iff \mathbf{y} \in \mathcal{Q}_y(w). \tag{6.22}$$

The numbers v , z , or w are called the quantized input, quantized state, and quantized output, respectively.

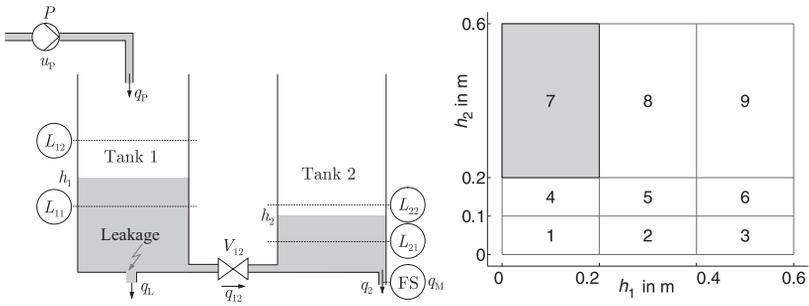


Fig. 6.9 Quantized tank system.

Example 6.5 *Quantized tank system*

The right part of Fig. 6.9 shows the partition of the two-dimensional state space of the tank system depicted in the left part. The grey region includes all states $\mathbf{x} = (h_1, h_2)'$ with the same quantized value $[x] = 7$. The part of the two-dimensional state space outside the nine partitions constitutes the unbounded partition $Q_x(0)$. As the tank system has discrete level sensors $LS1...LS4$, only the quantized level values are accessible for measurement. \square

Discrete-event quantized systems vs. discrete-time quantized systems Two alternative view points can be adopted when considering quantized systems. The first one considers equidistant sampling of the continuous system (6.18), (6.19) at time points $t_k = kT$, where T denotes the sampling time. The system is called a discrete-time quantized system, whose continuous subsystem can be described by the sampled version of the model (6.18), (6.19):

$$\mathbf{x}(k+1) = \mathbf{g}(\mathbf{x}(k), \mathbf{u}(k)), \quad \mathbf{x}(0) = \mathbf{x}_0, \quad (6.23)$$

$$\mathbf{y}(k) = \mathbf{h}(\mathbf{x}(k), \mathbf{u}(k)). \quad (6.24)$$

The second view point highlights the fact that as long as a signal does not leave a partition of the signal space, its quantized value remains the same and a new value is obtained at time points t_k at which the signal crosses a partition border. If the sequences (Eq. (6.16)), (Eq. (6.17)) concern these time points t_k , the system is called a discrete-event quantized system. Note that the time points t_k are not determined by a clock but by the quantized system itself.

The following investigations concern discrete-time quantized systems but may be transformed to discrete-event quantized systems as mentioned in the bibliographical remarks.

6.3.3 Properties of quantized systems

An important issue is the fact that it is impossible to predict the quantized state and output sequences of a quantized system unambiguously for a given quantized initial

state and a quantized input sequence. The reason for this is given by the fact that the system (6.23), (6.24) may start from any initial state $\mathbf{x}(0)$ with the given quantized value $z(0) = [\mathbf{x}(0)]$ and may obtain any input sequence \mathbf{U} which belongs to the given quantized sequence $[\mathbf{U}]$. For these sets of initial states and input sequences, a set of output sequences \mathbf{Y} result, which has, in general, more than one element. This phenomenon is referred to as the *nondeterminism of the quantized behavior*.

This existence of this nondeterminism is explained now for the autonomous system

$$\mathbf{x}(k+1) = \mathbf{g}(\mathbf{x}(k)), \quad \mathbf{x}(0) = \mathbf{x}_0 \quad (6.25)$$

with the quantized state space shown in Fig. 6.10. As only the quantized state information $[\mathbf{x}(0)] = 2$ is given, the bundle of all state trajectories that start in the set $\mathcal{X}(0) = \mathcal{Q}_x(2)$ have to be considered. The set

$$\mathcal{X}(1) = \{\mathbf{x}' = \mathbf{g}(\mathbf{x}) \mid \mathbf{x} \in \mathcal{X}(0)\}$$

of successor states overlaps with three state partitions $\mathcal{Q}_x(z)$, ($z = 1, 2, 5$). Hence, the quantized value of the state $\mathbf{x}(1)$ cannot be unambiguously predicted but is only known to have one of these three values:

$$[\mathbf{x}(1)] \in \{1, 2, 5\}.$$

This nondeterminism occurs because the initial state of the system is not precisely known. For systems with inputs, the nondeterminism is further introduced by the fact that only the quantized value of the input is given.

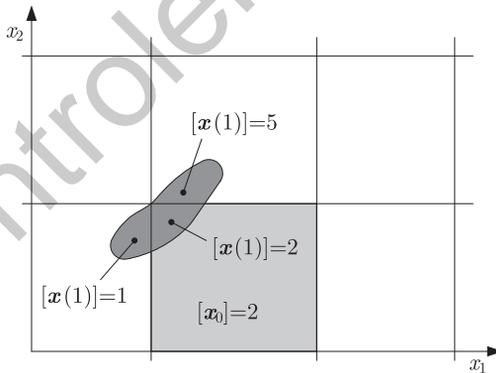


Fig. 6.10 Sets of states reached by the system for quantized initial state $\mathbf{x}(0) \in \mathcal{Q}_x(2)$.

The following theorem shows that only for a very restricted class of systems this nondeterminism does not occur.

Theorem 6.1 [415] *Linear autonomous systems*

$$x(k + 1) = Ax(k), \tag{6.26}$$

whose state space \mathcal{R}^n is uniformly quantized with resolution q_{x_i} in the direction of x_i , have a deterministic discrete-event behavior if and only if the matrix A can be represented in the form

$$A = \text{diag}q_{x_i} \text{diag}(2z_i + 1)^{-1} P \text{diag}q_{x_i}^{-1}, \tag{6.27}$$

where P is a permutation matrix and z_i ($i = 1, \dots, n$) are integer numbers.

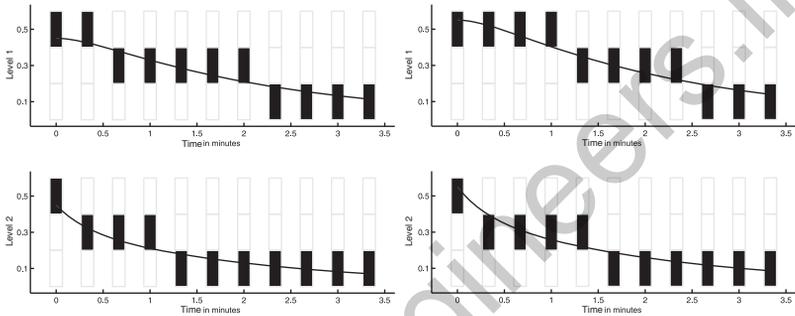


Fig. 6.11 Two quantized trajectories of the tank system starting in the same quantized initial state $[x_0]$.

Figure 6.11 shows how this nondeterminism becomes obvious in experiments with the tank system considered in Example 6.5. The black rectangles show the intervals $Q_{x_i}(z(k))$ that correspond to the quantized values $z(k)$ of the two tank levels. The two parts of the figure concern two experiments, in which the initial liquid levels have the same quantized values, but differ quantitatively. The thin lines show the quantitative tank behavior. Obviously, the resulting quantized trajectories are different because the initial tank levels are different, albeit their quantized values are the same in both experiments.

This fact has an important consequence for the modeling task considered subsequently:

|| As the quantized behavior is nondeterministic, the discrete-event model of quantized systems has to be nondeterministic.

Consequently, in the next subsection, stochastic automata are used to represent the behavior of quantized systems.

Markov property An important property of nondeterministic dynamical systems is the Markov property, which for autonomous quantized systems (6.25) claims that the relation

$$\begin{aligned} \text{Prob}([\mathbf{x}(k+1)] = z' | [\mathbf{x}(k)] = z) &= \\ \text{Prob}([\mathbf{x}(k+1)] = z' | [\mathbf{x}(k)] = z, [\mathbf{x}(k-1)] = z_{k-1}, \dots, [\mathbf{x}(0)] = z_0) &= \end{aligned} \quad (6.28)$$

holds for all z_0, z_1, \dots, z, z' . Accordingly, the transition probability from the current quantized state z towards the next quantized state z' must not depend on the earlier quantized states z_{k-1}, \dots, z_0 . If this relation is valid, the probability distribution $\text{Prob}([\mathbf{x}(k+1)] = z')$ with $z' \in \mathcal{Z}$ can be determined in terms of the probability distribution $\text{Prob}([\mathbf{x}(k)] = z)$ of the predecessor state.

Quantized systems, in general, do not possess the Markov property. For linear systems (6.26) it has been proved that (6.28) holds if and only if the matrix A has the property (6.27).

As the quantized system does not possess the Markov property with respect to the quantized input, state and output signals, every discrete-event model that possesses the Markov property, can only be an approximate representation of the quantized system.

Consequently, the modeling problem solved in the next subsection cannot claim to get a precise discrete-event representation.

6.3.4 Discrete-event modeling of quantized systems

The behavior of the quantized system is the set of all pairs $([U], [Y])$ of any length k_h that are consistent with the model (6.18), (6.19) and the quantisers (6.20)–(6.22). For diagnostic purposes the model has to be *complete* in the following sense:

Definition 6.3 (Completeness)

A model \mathcal{M} with behavior \mathcal{B}_M is called *complete* if it satisfies the relation

$$\mathcal{B}_M \supseteq \mathcal{B}_{\text{qual}}. \quad (6.29)$$

From (6.29) it follows that there may exist pairs

$$\begin{aligned} (V(0 \dots k_h), W(0 \dots k_h)) &\in \mathcal{B}_M, \\ (V(0 \dots k_h), W(0 \dots k_h)) &\notin \mathcal{B}_{\text{qual}}, \end{aligned}$$

which are consistent with the model but not with the quantized system. These pairs are called *spurious*. Their existence is a typical phenomenon encountered in abstraction-based modeling. Spurious solutions occur because the quantized model has the Markov property whereas the quantized system does not possess this property.

The importance of the completeness of the model used for fault diagnosis is given by the following corollary, which will become obvious in the next subsection:

A model is suitable for solving the diagnostic problem if and only if it is complete.

The following part shows how a model can be obtained that satisfies the completeness requirement (6.29). A stochastic automaton

$$\mathcal{S} = (\mathcal{Z}, \mathcal{V}, \mathcal{W}, L, \text{Prob}(z(0)))$$

is used whose state, input, and output sets \mathcal{Z} , \mathcal{V} , and \mathcal{W} are identical to the sets of quantized signal values. The behavioral relation L and the initial state probability $\text{Prob}(z(0))$ have to be chosen as follows:

Theorem 6.2 (Complete model of the quantized system)

A stochastic automaton is a complete model of the quantized system if and only if the following conditions are satisfied:

$$L(z', w | z, v) > 0$$

$$\Leftrightarrow \text{Prob}([\mathbf{x}(1)] = z', [\mathbf{y}(0)] = w | [\mathbf{x}(0)] = z, [\mathbf{u}(0)] = v) > 0 \quad (6.30)$$

$$\text{Prob}(z(0) = z) > 0 \text{ for all } z = [\mathbf{x}_0] \quad \mathbf{x}_0 \in \mathcal{X}_0. \quad (6.31)$$

L is best chosen according to the equation

$$L(z', w | z, v) = \text{Prob}([\mathbf{x}(1)] = z', [\mathbf{y}(0)] = w | [\mathbf{x}(0)] = z, [\mathbf{u}(0)] = v). \quad (6.32)$$

Note that the right-hand side of (6.30) can be determined by means of the state-space model (6.23), (6.24) and the definitions of the quantisers. The initial state $\mathbf{x}(0)$ and the input $\mathbf{u}(0)$ are distributed over the sets $\mathcal{Q}_x(z)$ or $\mathcal{Q}_u(v)$ and the task is to determine the probability that $[\mathbf{x}(1)] = z'$ and $[\mathbf{y}(0)] = w$ holds for given z, v, z' and w . An approximation of L can be obtained by mapping a grid of initial states for selected input values and “counting” those points $\mathbf{x}(1)$ and $\mathbf{y}(0)$ that fall into the sets $\mathcal{Q}_x(z')$ or $\mathcal{Q}_y(w)$, respectively.

To solve the diagnostic problem given in Subsection 6.3.1, the discrete-event model has to be built for all faults $f \in \mathcal{F}$. The result is a behavioral relation $L(z', w, | z, v, f)$ that depends on the fault f . For the simplicity of presentation it is assumed that the fault f does not change during the application of the diagnostic method.

6.3.5 Diagnosis of quantized systems

The principle of consistency-based diagnosis, which is applied here to the discrete-event model of the quantized system, is to test for which fault f the measured sequences (Eq. (6.16)), (Eq. (6.17)) are consistent with the behavioral relation L for $v(k) = [\mathbf{u}(k)]$ and $w(k) = [\mathbf{y}(k)]$. That is, it has to be tested whether a state sequence

$$Z(0 \cdots k_h + 1) = (z(0), \dots, z(k_h + 1))$$

exists such that

$$L(z(k+1), [\mathbf{y}(k)] | z(k), [\mathbf{u}(k)], f) > 0 \quad \text{for } k = 0, 1, \dots, k_h$$

holds. If this requirement is satisfied, the fault probability

$$p(f | k_h) := \text{Prob}(F = f | [\mathbf{U}(0 \cdots k_h)], [\mathbf{Y}(0 \cdots k_h)]),$$

which describes the probability of the presence of the fault f under the condition that the measured input and output sequences have occurred, can be determined as shown in the following.

The first step of the diagnostic algorithm is to determine the joint probability distribution

$$\text{Prob}(F = f | Z(k) = z | [\mathbf{U}(0 \cdots k)], [\mathbf{Y}(0 \cdots k)]) =: p(f, z | k)$$

of the current state $Z(k)$ and the fault F for all time k , which is abbreviated in the following formulas by $p(f, z | k)$:

$$p(f, z' | k-1) = \frac{\sum_z L(z', [\mathbf{y}(k)] | z, [\mathbf{u}(k)], f) \cdot p(f, z | k-2)}{\sum_{z', z, f} L(z', [\mathbf{y}(k)] | z, [\mathbf{u}(k)], f) \cdot p(f, z | k-2)} \quad (k = 1, 2, \dots, k_h), \quad (6.33)$$

$$p(f, z' | -1) = \text{Prob}(F = f) \cdot \text{Prob}([\mathbf{x}(0)] = z). \quad (6.34)$$

$\text{Prob}(F = f)$ is the a priori fault probability and $\text{Prob}([\mathbf{x}(0)] = z)$ the probability distribution of the initial state. Then the fault probability $p(f | k_h)$ can be determined by

$$p(f | k) = \frac{\sum_{z', z} L(z', [\mathbf{y}(k)] | z, [\mathbf{u}(k)], f) \cdot p(f, z | k-1)}{\sum_{z', z, f} L(z', [\mathbf{y}(k)] | z, [\mathbf{u}(k)], f) \cdot p(f, z | k-1)} \quad (k = 0, \dots, k_h) \quad (6.35)$$

Theorem 6.3 (Diagnosis of quantized systems)

If the stochastic automaton \mathcal{S} is a complete model of the quantized system, the fault that occurs in the quantized systems belongs to the set

$$\mathcal{F}(k_h) = \{f : p(f | k_h) > 0\}, \quad (6.36)$$

where $p(f | k_h)$ is described by (6.34) and (6.35).

The interpretation of the formulas is to determine the probability with which the system has made the state transitions that are necessary to produce the currently measured output for the measured input.

Corollary 6.1 (Diagnostic results for the quantized system)

- *The quantized system is known to be subject to some fault $f \in \mathcal{F}(k_h)$.*
- *Fault detection: If $f_0 \notin \mathcal{F}(k_h)$ holds, the quantized system is known to be faulty (where f_0 denotes the faultless system).*
- *Fault identification: If $\mathcal{F}(k_h) = \{f_i\}$ is a singleton, the system is known to be subjected to fault f_i provided that the occurring fault belongs to the set \mathcal{F} .*

The diagnostic method is summarized in Algorithm 6.1.

Algorithm 6.1 Diagnosis of quantized systems

Given:

Complete model \mathcal{S} of the quantized system.

Fault model \mathcal{S}_f .

Initial state probability distribution $\text{Prob}([\mathbf{x}(0)] = z)$ for all $z \in \mathcal{Z}$.

Initial fault probability distribution $\text{Prob}(F = f)$ for all $f \in \mathcal{F}$.

Init:

$$p(f, z | -1) = \text{Prob}(F = f) \cdot \text{Prob}([\mathbf{x}(0)] = z) \text{ for } f \in \mathcal{F}, z \in \mathcal{Z} \\ k = 0.$$

1: Measure $[\mathbf{u}(k)]$ and $[\mathbf{y}(k)]$.

2: For all $f \in \mathcal{F}$ and $z \in \mathcal{Z}$ determine $p(f, z' | k)$ by (6.33)

3: For all $f \in \mathcal{F}$ and $z \in \mathcal{Z}$ determine $p(f | k)$ by (6.35)

4: Determine $\mathcal{F}(k_h)$ according to (6.36).

5: $k := k + 1$

Continue with Step 1.

Result: $p(f | k_h)$ and $\mathcal{F}(k)$ for increasing time horizon k .

6.3.6 Example: Diagnosis of the air path of a diesel engine

The air path of a diesel engine with a single turbo charger is shown in Fig. 6.12. The hot-wire air flow meter (HFM) measures the incoming air flow, which is compressed and cooled before it arrives in the container. After the combustion of the fuel, the exhaust gas may be recirculated through the exhaust gas recirculation (EGR) valve or drives turbine with a variable geometry (VTG).

The air path is subject to the input signals $n_E(t)$ (engine speed), $m_F(t)$ (fuel flow per stroke), $A_{\text{EGR}}(t)$ (effective area of the EGR valve) and $A_{\text{VTG}}(t)$ (effective area

of the turbine). The incoming air flow $q_1(t)$, the container pressure $p_2(t)$ and the temperature in the container $T_2(t)$ can be measured. The continuous-variable state-space model (6.23), (6.24) has seven state variables, and four input and three output signals.

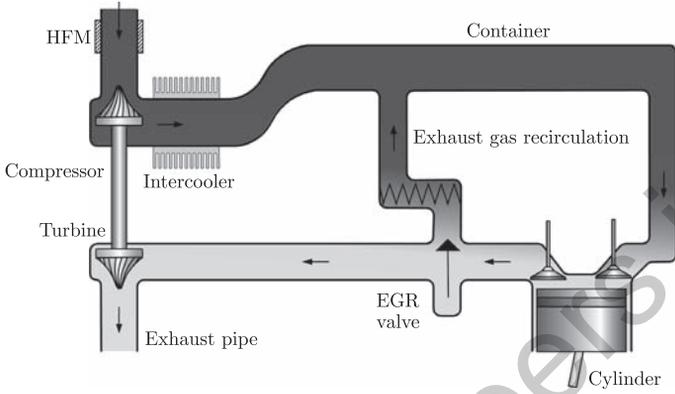


Fig. 6.12 Air path of a diesel engine.

Faults that occur in the components of this system influence the emission, which is subject to legal restrictions. Therefore, the sensor and actuator faults like a blockage of the EGR valve or offsets of the HFM or T_2 sensor have to be found on-board the car.

The airpath is considered as a quantized system. For all measured signals, a partition of the signal space is introduced. For example, Fig. 6.13 shows the discretization of the air flow q_1 into six regions starting from “very low” up to “maximum.” The rectangles indicate in which interval the signal q_1 lies at the respective time instant. As can be seen in the figure, the trajectories of the signal $q_1(t)$ for the two fault cases f_0 and f_3 can be distinguished easily in spite of the rough discretization.

The quantized model has been determined by (6.32).

Results For the evaluation of the diagnostic method, the input signals from test series with an experimental car have been used. The diagnostic result describes the probability for the occurrence of each fault, which is marked by the intensity of the bars in the following diagram.

If the air path is affected by the fault f_1 (HFM offset + 0.02 kg/s), the faults f_0 (faultless case), f_1 , and f_3 (the HFM drift) are stated possible when the diagnosis starts (Fig. 6.14). During the diagnosis the probability of f_3 decreases continuously and the probability of f_1 increases accordingly. At 1.9 seconds a measurement has occurred that is impossible in the faultless case. Therefore, the faultless case is excluded and the diagnostic system returns the correct result.

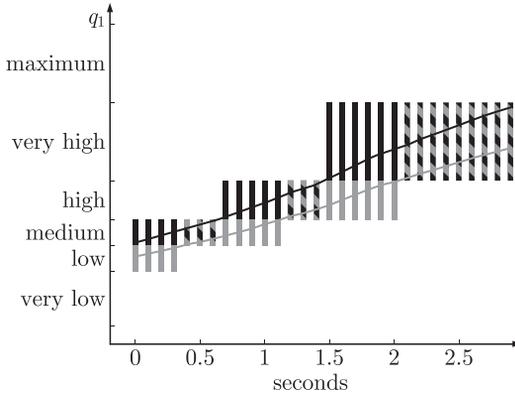


Fig. 6.13 Air flow q_1 during faultless operation f_0 (black) and during fault f_3 (grey).

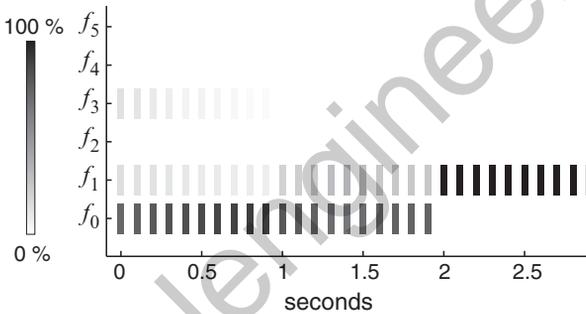


Fig. 6.14 Identification of the fault f_1 .

6.4 Abstraction-based supervisory control design

6.4.1 Motivation of abstraction-based design

The motivation for the topic of this section is given by the complexity of the control design task for hybrid systems. As hybrid systems consist of interacting continuous and discrete-event components, a straightforward application of standard control synthesis methods from either the continuous or the discrete world are impossible. The situation changes, however, if continuous components are approximated and “replaced” by discrete ones. This effectively transforms the given hybrid control problem into a purely discrete one, which can subsequently be addressed using established methods from the area of discrete-event systems (DES).

Of course, this is only an acceptable strategy if it can be proved that the resulting discrete control scheme will enforce the desired specifications not only for

the discrete approximation of the underlying hybrid problem, but also for the hybrid problem itself. This philosophy is usually called “abstraction-based control design,” and it is understood that an abstraction is an approximation with a particular set of properties that facilitates the required proof.

Abstraction-based control of hybrid systems has been a popular topic over the last ten years, and various approaches have been reported in the literature, e.g. [190, 363, 423, 606]. In this section, we will concentrate on a particular approach developed in [464, 466, 550], which exhibits a number of convenient properties:

- It employs a special set of abstractions, called l -complete approximations, which are exclusively defined in terms of (measurable) outputs and inputs of the continuous system to be approximated.
- The suggested concept allows for systematic approximation refinement—this is extremely important in practice, as it may well turn out that the currently employed approximation is too coarse for *any* controller to achieve the given specifications and, therefore, needs to be “improved,” or refined.
- We choose an exposition which is set within J. C. Willems, behavioral framework, e.g. [661, 662]. Within this framework, the presentation of conceptual and computational details can be separated to a large extent. It is, therefore, possible to describe the underlying ideas without “bothering too much” about cumbersome computations.

This section is organized as follows: we first describe the supervisory control problem treated in this section. We then recall some definitions and concepts from behavioral systems theory (Sect. 6.4.3). In (Section 6.4.4), we discuss supervisory control in a behavioral setting. In (Section 6.4.5), we explain if and why abstraction-based control will also work for the underlying hybrid control problem. Finally, in (Section 6.4.6), we briefly mention how complexity problems may be alleviated within an abstraction-based control approach.

6.4.2 Control problem

The simplest scenario where our approach applies is shown in Fig. 6.15. The continuous communicates with its environment exclusively via discrete events. Input events from the finite set \mathcal{U} may switch the continuous dynamics, and output events from a finite set \mathcal{Y} are typically generated by some sort of quantization mechanism. We assume the existence of solutions in the following sense: if, in an arbitrary initial plant state $\mathbf{x} \in \mathbb{R}^n$, a control event occurs, the plant state will evolve over time and will eventually generate an output event. A typical example for this scenario is when the plant state \mathbf{x} or a continuous output signal ξ produces an output event by “hitting” the boundary of a set associated with the last input symbol (“invariant”). This is usually referred to as event-driven sampling.

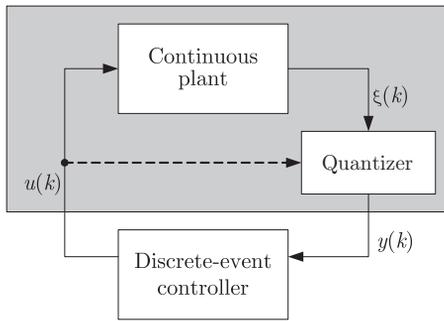


Fig. 6.15 Continuous plant under discrete control.

Alternatively, sampling could be clock-driven. In this case $y(k)$ would be a quantized version of the plant state (or a continuous plant output signal) at fixed sampling times $k\Delta$.

Note that we do *not* need to specify at this point whether sampling is event or clock-driven. In both cases, the input and the output signal are sequences of discrete events/symbols, denoted by u and y , respectively. Upon generation of a measurement event by the plant, a new input event or set of input events will be provided instantaneously by the controller.

The stated assumptions imply that the plant can be realized as a so-called I/S -machine. Formally this is a state machine with state set \mathcal{X} , input and output event sets \mathcal{U} and \mathcal{Y} , an initial state set \mathcal{X}_0 and a transition relation δ with the additional property that for any reachable state $x \in \mathcal{X}$ and every $\mu \in \mathcal{U}$, there exist $\nu \in \mathcal{Y}$ and $x' \in \mathcal{X}$ such that $(x, (\mu, \nu), x') \in \delta$.

Following the supervisory control philosophy proposed in [551, 552], the control task is to guarantee that certain events or sequences of events which are deemed to be dangerous or forbidden will never occur. This is to be achieved by a supervisor which “sees” the sequence y of events generated by the plant and, in response to each measurement event $y(k)$, may disable certain input events from the set \mathcal{U} .

6.4.3 Behaviors

In the terminology of behavioral system theory, the (external) behavior of a dynamical system is the set of external signals that the system can evolve on. Hence, with $w := (u, y)$ and $\mathcal{W} := \mathcal{U} \times \mathcal{Y}$, the external plant behavior \mathfrak{B}_p is a set of maps $w: \mathbb{N}_0 \rightarrow \mathcal{W}$; i.e. $\mathfrak{B}_p \subseteq \mathcal{W}^{\mathbb{N}_0}$, where \mathbb{N}_0 is the set of nonnegative integers and $\mathcal{W}^{\mathbb{N}_0} := \{w: \mathbb{N}_0 \rightarrow \mathcal{W}\}$ represents the set of all sequences in \mathcal{W} . From an external point of view, a dynamical system is completely defined by specifying the notion of time and the signal space for the external signals, and the (external) behavior. As in the following discussion time (\mathbb{N}_0) and external signal space ($\mathcal{U} \times \mathcal{Y}$) are fixed, we can use the terms “system” and “behavior” synonymously.

In the following, we will also restrict attention to *time-invariant* systems. Following [661] a system with behavior \mathfrak{B} is called time-invariant, if $\sigma\mathfrak{B} \subseteq \mathfrak{B}$, where σ represents the backward shift operator, i.e. $(\sigma w)(k) = w(k+1)$ for all $k \in \mathbb{N}_0$.

To keep notation as simple as possible, we introduce the restriction operators $(\cdot)|_{[k_1, k_2]}$ and $(\cdot)|_{[k_1, k_2]}$ that map sequences $w \in \mathcal{W}^{\mathbb{N}_0}$ to finite strings:

$$\begin{aligned}
 w|_{[k_1, k_2]} &:= w(k_1)w(k_1+1) \cdots w(k_2-1) \in \mathcal{W}^{k_2-k_1}, \\
 w|_{[k_1, k_2]} &:= w(k_1)w(k_1+1) \cdots w(k_2) \in \mathcal{W}^{k_2-k_1+1},
 \end{aligned}$$

where $\mathcal{W}^0 := \{\epsilon\}$ and ϵ denotes the empty string. For $\mathcal{W} = \mathcal{U} \times \mathcal{Y}$, the symbols \mathcal{P}_U and \mathcal{P}_Y denote the *natural projection* operators to the respective component, i.e. $\mathcal{P}_U w = u$ and $\mathcal{P}_Y w = y$ for $w = (u, y)$, $u \in \mathcal{U}^{\mathbb{N}_0}$, $y \in \mathcal{Y}^{\mathbb{N}_0}$. Finally, we use $\tilde{w}|_{[0, k]} \approx_y w|_{[0, k]}$ as an abbreviation for the two strings w and \tilde{w} to be identical up to the last output event, i.e. $\mathcal{P}_U \tilde{w}|_{[0, k]} = \mathcal{P}_U w|_{[0, k]}$ and $\mathcal{P}_Y \tilde{w}|_{[0, k]} = \mathcal{P}_Y w|_{[0, k]}$.

To clarify the input/output structure, we use a slightly weakened version of Willems' I/O-behaviors:

Definition 6.4 (I/-behavior) A behavior $\mathfrak{B} \subseteq \mathcal{W}^{\mathbb{N}_0}$ is said to be a (strict) I/-behavior with respect to $(\mathcal{U}, \mathcal{Y})$, if

- (i) the input is free, i.e. $\mathcal{P}_U \mathfrak{B} = \mathcal{U}^{\mathbb{N}_0}$; and
- (ii) the output does (strictly) not anticipate the input, i.e.

$$\begin{aligned}
 \mathcal{P}_U \tilde{w}|_{[0, k]} &= \mathcal{P}_U \hat{w}|_{[0, k]} \\
 &\Rightarrow (\exists w \in \mathfrak{B}) \mathcal{P}_Y w|_{[0, k]} = \mathcal{P}_Y \tilde{w}|_{[0, k]} \text{ and } \mathcal{P}_U w = \mathcal{P}_U \hat{w}
 \end{aligned}$$

for all $k \in \mathbb{N}_0$, $\tilde{w}, \hat{w} \in \mathfrak{B}$; for the strict case the premise on the l.h.s. is weakened to $\mathcal{P}_U \tilde{w}|_{[0, k]} = \mathcal{P}_U \hat{w}|_{[0, k]}$.

Loosely speaking, item (ii) in this definition says that we can change the future (and, in the strict case, the present) of the input without affecting present and past of the output.

In the following, we will also need the notion of complete systems:

Definition 6.5 (Complete behavior) [661] A behavior $\mathfrak{B} \subseteq \mathcal{W}^{\mathbb{N}_0}$ is complete if

$$w \in \mathfrak{B} \Leftrightarrow \forall k \in \mathbb{N}_0 : w|_{[0, k]} \in \mathfrak{B}|_{[0, k]}.$$

Hence, to decide whether a signal w belongs to a complete behavior \mathfrak{B} , it is sufficient to look at “finite length portions” of w .

It is easily verified that the external behavior induced by an I/S/-machine is an I/-behavior. It should be noted however that, in general, it is not complete.

6.4.4 Supervisory control

We now focus on the role of a controller, or supervisor, evolving on the same signal space as the plant model. Adopting the concepts of supervisory control theory

[551, 552] to the behavioral framework, the task of a supervisor $\mathfrak{B}_{\text{sup}} \subseteq \mathcal{W}^{\mathbb{N}_0}$ is to restrict the plant behavior $\mathfrak{B}_p \subseteq \mathcal{W}^{\mathbb{N}_0}$ such that the closed-loop behavior contains only acceptable signals. The closed-loop behavior is given by

$$\mathfrak{B}_{\text{cl}} = \mathfrak{B}_p \cap \mathfrak{B}_{\text{sup}},$$

because a signal $w \in \mathfrak{B}_p$ “survives” closing the loop if and only if it is also in $\mathfrak{B}_{\text{sup}}$. We collect all acceptable signals in the specification behavior $\mathfrak{B}_{\text{spec}}$ and say that the supervisor $\mathfrak{B}_{\text{sup}}$ *enforces the specification* if $\mathfrak{B}_{\text{cl}} \subseteq \mathfrak{B}_{\text{spec}}$.

It is immediately clear that any supervisor must exhibit two additional properties:

- it must respect the I/O structure of the plant, i.e. it may restrict the plant input but then has to accept whatever output event the plant generates;
- it must ensure that, at any instant of time, there is a possible future evolution for the closed loop.

This is formalized by the following definition:

Definition 6.6 (Admissible supervisor) A supervisor $\mathfrak{B}_{\text{sup}} \subseteq \mathcal{W}^{\mathbb{N}_0}$ is admissible to the plant $\mathfrak{B}_p \subseteq \mathcal{W}^{\mathbb{N}_0}$ if:

- $\mathfrak{B}_{\text{sup}}$ is generically implementable, i.e. $k \in \mathbb{N}_0$, $w|_{[0,k]} \in \mathfrak{B}_{\text{sup}}|_{[0,k]}$, $\tilde{w}|_{[0,k]} \in \mathcal{W}^{k+1}$, $\tilde{w}|_{[0,k]} \approx_y w|_{[0,k]}$ implies $\tilde{w}|_{[0,k]} \in \mathfrak{B}_{\text{sup}}|_{[0,k]}$; and
- \mathfrak{B}_p and $\mathfrak{B}_{\text{sup}}$ are nonconflicting, i.e. $\mathfrak{B}_p|_{[0,k]} \cap \mathfrak{B}_{\text{sup}}|_{[0,k]} = (\mathfrak{B}_p \cap \mathfrak{B}_{\text{sup}})|_{[0,k]}$ for all $k \in \mathbb{N}_0$.

This leads to the following formulation of supervisory control problems:

Definition 6.7 (Supervisory control problem) Given a plant $\mathfrak{B}_p \subseteq \mathcal{W}^{\mathbb{N}_0}$, $\mathcal{W} = \mathcal{U} \times \mathcal{Y}$, and a specification $\mathfrak{B}_{\text{spec}} \subseteq \mathcal{W}^{\mathbb{N}_0}$, the pair $(\mathfrak{B}_p, \mathfrak{B}_{\text{spec}})_{\text{cp}}$ is a supervisory control problem. A supervisor $\mathfrak{B}_{\text{sup}} \subseteq \mathcal{W}^{\mathbb{N}_0}$ that is admissible to \mathfrak{B}_p and that enforces $\mathfrak{B}_{\text{spec}}$ is said to be a solution of $(\mathfrak{B}_p, \mathfrak{B}_{\text{spec}})_{\text{cp}}$.

In [463, 464], we have adapted the set-theoretic argument of [552] to show the unique existence of the *least restrictive solution* for our class of supervisory control problems. Formally, the set of all solutions of $(\mathfrak{B}_p, \mathfrak{B}_{\text{spec}})_{\text{cp}}$ is a complete upper semi-lattice with join operator “ \cup ” and partial order “ \subseteq ”, and the supremal element of the lattice is referred to as the least restrictive supervisor. Clearly, the action of this supervisor will be least restrictive in the sense of disabling the least number of input events from the set \mathcal{U} at any sampling instant k .

Note that the least restrictive solution may turn out to be trivial, i.e. $\mathfrak{B}_{\text{sup}} = \emptyset$, as this is admissible to the plant and, because of $\mathfrak{B}_p \cap \emptyset \subseteq \mathfrak{B}_{\text{spec}}$, enforces the specifications. Obviously, only nontrivial solutions are of interest. Therefore, if \emptyset turns out to be the least restrictive solution of $(\mathfrak{B}_p, \mathfrak{B}_{\text{spec}})_{\text{cp}}$, one would conclude that the specifications are “too strict” for the given plant model.

6.4.5 Abstraction-based supervisory control

If both \mathfrak{B}_p and $\mathfrak{B}_{\text{spec}}$ could be realized by finite automata, we could easily compute (a realization of) the least restrictive solution of $(\mathfrak{B}_p, \mathfrak{B}_{\text{spec}})_{\text{CP}}$ by appropriately modifying standard DES tools (e.g. [146, 552]). The required modifications would be straightforward and take into account that in the framework suggested here an I/O-structure has been employed for plant and supervisor instead of the standard partition of external events into controllable and uncontrollable events. Details can be found in [466]. While a finite automaton realization of $\mathfrak{B}_{\text{spec}} \subseteq \mathcal{W}^{\mathbb{N}_0}$ is quite common for finite \mathcal{W} , the hybrid plant is, of course, in general *not* realizable on a finite state space. In [464, 550], we suggest to approach this problem by replacing \mathfrak{B}_p with an appropriate *abstraction* or, if needed, a sequence of increasingly accurate abstractions that can be realized by finite automata. This procedure is based on the notion of l -completeness:

Definition 6.8 (l -complete behavior) [662] A behavior $\mathfrak{B} \subseteq \mathcal{W}^{\mathbb{N}_0}$ is l -complete if

$$w \in \mathfrak{B} \quad \Leftrightarrow \quad \forall k \in \mathbb{N}_0 : \sigma^k w|_{[0,l]} \in \mathfrak{B}|_{[0,l]}.$$

Hence, whether a signal $w \in \mathcal{W}^{\mathbb{N}_0}$ is contained in a given l -complete behavior can be decided on the basis of strings of length $l + 1$. Note that the external signal space \mathcal{W} has been assumed finite. Therefore, $\mathfrak{B}|_{[0,l]}$ is also a finite set for any $l \in \mathbb{N}_0$. It is then a straightforward exercise to realize a given l -complete behavior by a finite-state machine, where the state memorizes the last l values of the external signal [466].

It is now natural to introduce the notion of a *strongest l -complete approximation* of a given system with behavior $\mathfrak{B} \in \mathcal{W}^{\mathbb{N}_0}$. Roughly speaking, this is another system evolving on the same time axis \mathbb{N}_0 and within the same signal space as the original system, and with the smallest l -complete behavior that covers \mathfrak{B} . Formally this is captured in the following definition [464, 466]:

Definition 6.9 (l -complete approximation) [464] Consider two dynamical systems with behaviors \mathfrak{B} and \mathfrak{B}_l , both evolving on time \mathbb{N}_0 and within the external signal space \mathcal{W} . Then \mathfrak{B}_l is said to be a *strongest l -complete approximation* for \mathfrak{B} if:

- (i) $\mathfrak{B}_l \supseteq \mathfrak{B}$, \mathfrak{B}_l is l -complete;
- (ii) $\mathfrak{B}'_l \supseteq \mathfrak{B}$, \mathfrak{B}'_l is l -complete $\implies \mathfrak{B}'_l \supseteq \mathfrak{B}_l$.

If only item (i) holds, \mathfrak{B}_l is referred to as an *l -complete approximation*.

The motivation for Definition 6.9 is the following: we want to synthesize supervisory control for a system with behavior \mathfrak{B} on the basis of the approximation \mathfrak{B}_l . Clearly, we need condition (i) to hold; otherwise, \mathfrak{B} could contain unacceptable trajectories which could not be predicted by the approximation \mathfrak{B}_l and hence not be suppressed by a control strategy based on the approximate model. Note that it has become accepted terminology to call an approximation that covers the behavior of the system to be approximated as an *abstraction*. Therefore, by definition, each

l -complete approximation is an abstraction. It is also obvious that we want condition (ii) to hold: the smaller \mathfrak{B}_l , the more “accurate” the approximation, and the better the chances for a suitable supervisor to exist.

Existence and uniqueness of strongest l -complete approximations are easily established within the behavioral framework (for a formal proof cf. [466]). The approximation behavior is given by

$$\mathfrak{B}_l = \{w \mid w \in \mathcal{W}^{\mathbb{N}_0}, \sigma^k w|_{[0,l]} \in \mathfrak{B}|_{[0,l]} \forall k \in \mathbb{N}_0 \}, \quad (6.37)$$

i.e. it consists of all sequences w which, on intervals of length $l + 1$, coincide with the underlying system behavior. It is an immediate consequence of (6.37) that for all $l \in \mathbb{N}$

$$\mathfrak{B}_l \supseteq \mathfrak{B}_{l+1} \quad (6.38)$$

i.e. by increasing l we are guaranteed to obtain an approximation which is at least as accurate as the previous one. This is also referred to as approximation or abstraction *refinement*.

Definition 6.9 and property (6.38) suggest the following iterative procedure for abstraction-based supervisory control:

Algorithm 6.2 Abstraction-based supervisory control

Init: Set $l = 1$.

- 1: Compute a (strongest) l -complete approximation \mathfrak{B}_l for the plant model \mathfrak{B}_p and realize it as a finite-state machine.
- 2: Compute the least restrictive solution $\mathfrak{B}_{\text{sup}}$ for the supervisory control problem $(\mathfrak{B}_l, \mathfrak{B}_{\text{spec}})_{\text{cp}}$. As finite-state realizations for both \mathfrak{B}_l and $\mathfrak{B}_{\text{spec}}$ are known, this can be done using slight modifications of standard methods from DES theory (e.g. [146, 552]). Note that the resulting supervisor $\mathfrak{B}_{\text{sup}}$ is realized by a finite-state machine and is, therefore, complete.
- 3: If $\mathfrak{B}_{\text{sup}} \neq \emptyset$ (i.e. a nontrivial solution has been found), terminate. Otherwise increase l by 1 and go to Step 1; the latter corresponds to the fact that on the basis of the abstraction \mathfrak{B}_l a nontrivial solution cannot be found, and, therefore, the abstraction solution will be refined.

Result: $\mathfrak{B}_{\text{sup}}$ (or exhaustion of computational resources).

Two main questions remain open at this point:

- How do we compute \mathfrak{B}_l ?
- If a nontrivial solution $\mathfrak{B}_{\text{sup}}$ for $(\mathfrak{B}_l, \mathfrak{B}_{\text{spec}})_{\text{cp}}$ exists, can we guarantee that it is also a solution for the underlying problem $(\mathfrak{B}_p, \mathfrak{B}_{\text{spec}})_{\text{cp}}$?

We will answer the second question first, which is indeed the central one in the context of abstraction-based control design.

Suppose $\mathfrak{B}_{\text{sup}}$ is a nontrivial solution for the control problem $(\mathfrak{B}_l, \mathfrak{B}_{\text{spec}})_{\text{cp}}$. This implies that $\mathfrak{B}_l \cap \mathfrak{B}_{\text{sup}} \subseteq \mathfrak{B}_{\text{spec}}$. Clearly, because of $\mathfrak{B}_p \subseteq \mathfrak{B}_l$ the supervisor also enforces the specifications for the underlying plant model \mathfrak{B}_p , i.e.

$\mathfrak{B}_p \cap \mathfrak{B}_{\text{sup}} \subseteq \mathfrak{B}_{\text{spec}}$. To make $\mathfrak{B}_{\text{sup}}$ a solution of $(\mathfrak{B}_p, \mathfrak{B}_{\text{spec}})_{\text{cp}}$, it is also required to be admissible to \mathfrak{B}_p (Definition 6.7). This, in turn, requires generic implementability and nonconflictingness of $\mathfrak{B}_{\text{sup}}$ and \mathfrak{B}_p (Definition 6.6). Generic implementability is guaranteed as $\mathfrak{B}_{\text{sup}}$ is a solution of the approximated control problem $(\mathfrak{B}_l, \mathfrak{B}_{\text{spec}})_{\text{cp}}$. To establish that $\mathfrak{B}_{\text{sup}}$ and \mathfrak{B}_p are nonconflicting, we employ the following Lemma from [465]:

Lemma 6.1 *Let \mathfrak{B}_p be an I-behavior and additionally assume that it is either complete or can be realized by an I/S/-machine. If $\mathfrak{B}_{\text{sup}}$ is complete and generically implementable, then \mathfrak{B}_p and $\mathfrak{B}_{\text{sup}}$ are nonconflicting.*

The central result in our abstraction-based supervisory control synthesis approach then follows immediately:

Theorem 6.4 *Let $\mathfrak{B}_l \subseteq \mathcal{W}^{\mathbb{N}_0}$, $\mathcal{W} = \mathcal{U} \times \mathcal{Y}$, be an l -complete abstraction of an I-behavior $\mathfrak{B}_p \subseteq \mathcal{W}^{\mathbb{N}_0}$. Assume furthermore that \mathfrak{B}_p can be realized by an I/S/-machine. Let $\mathfrak{B}_{\text{sup}} \subseteq \mathcal{W}^{\mathbb{N}_0}$ be a complete nontrivial solution of the supervisory control problem $(\mathfrak{B}_l, \mathfrak{B}_{\text{spec}})_{\text{cp}}$. Then $\mathfrak{B}_{\text{sup}}$ is a nontrivial solution of $(\mathfrak{B}_p, \mathfrak{B}_{\text{spec}})_{\text{cp}}$.*

It remains to discuss how \mathfrak{B}_l is computed. Computing \mathfrak{B}_l amounts to determining whether strings $(\mu_0, \nu_0), \dots, (\mu_l, \nu_l)$, with $\mu_i \in \mathcal{U}$, $\nu_i \in \mathcal{Y}$, $i = 0, \dots, l$, are contained in the restricted plant behavior $\mathfrak{B}_p|_{[0,l]}$ (compare (6.37)). As the plant behavior \mathfrak{B}_p cannot be expected to be finite, this problem needs to be settled on the basis of a realization. Both for the event-driven and the time-driven sampling case, one then asks whether there is an initial state $\mathbf{x} \in \mathbb{R}^n$ such that applying the string μ_0, \dots, μ_l of input events will make the plant respond with the string ν_0, \dots, ν_l of output events.

Clearly, in both cases, this is a reachability problem. Reachability issues for nonlinear continuous systems are in general highly nontrivial, and one may have to settle for approximate solutions. This is not a fundamental problem in general, if we can guarantee that all strings contained in $\mathfrak{B}_p|_{[0,l]}$ are accepted. (Falsely) accepting additional strings will enlarge the approximation behavior—we still get an l -complete approximation, but not the strongest one. On the basis of this approximation we can carry out our abstraction-based synthesis as before.

6.4.6 Extensions

The basic setup shown in Fig. 6.15 includes a plant with continuous dynamics and discrete inputs and outputs. Often, the plant to be controlled will additionally contain discrete dynamics. A popular example are plants that are modelled as hybrid automata with state (\mathbf{x}, q) , where $\mathbf{x}(t) \in \mathbb{R}^n$, $q(t) \in \mathcal{Q}$ and \mathcal{Q} is a finite set. Our approach can easily be extended to cover this case. Using the procedure outlined above, we “replace” the continuous part of the plant dynamics by a discrete abstraction. In

combination with the discrete part of the plant dynamics this provides a discrete abstraction of the overall hybrid plant, and we can proceed to synthesize a discrete supervisory controller for this abstraction.

The described approach is conceptually elegant but, as with other approaches that involve the use of discrete abstractions, it suffers from the problem of computational complexity. Roughly speaking, computational effort grows exponentially with $|\mathcal{U}|$ (the number of control events), $|\mathcal{Y}|$ (the number of measurement events), and l (the parameter determining approximation accuracy). Therefore, it may well turn out that in the iterative procedure outlined in (Section 6.4.5) computational resources are exhausted before a nontrivial solution to a supervisory control problem $(\mathfrak{B}_l, \mathfrak{B}_{\text{spec}})_{\text{CP}}$ (and hence to the underlying problem $(\mathfrak{B}_p, \mathfrak{B}_{\text{spec}})_{\text{CP}}$) is found. Several approaches to alleviate this problem have been suggested:

- [468] describes how uniform abstraction refinement (by increasing the parameter l) can be replaced by a more subtle, specification dependent, refinement procedure. Roughly speaking, the suggested algorithm analyzes why supervisory control synthesis for a specific abstraction and the required specifications fails. It then refines (only) the aspects of the abstraction that are deemed “too coarse” for the specification and hence provides an abstraction that is tailored to the supervisory control problem at hand.
- [463] investigates the use of modular and decentralized control architectures within the abstraction-based supervisory control design framework described above.
- [467] discusses how the described abstraction-based approach can be extended to cover hierarchical control architectures. Within this framework, several control layers, working on different time axes and with different levels of information aggregation cooperate to enforce an overall specification for the given plant. Proper use of hierarchical decomposition allows the successful treatment of problems that are far beyond the reach of the unstructured basic approach treated within this section. This hierarchical extension is summarized in (Section 14.3.1) of this handbook and applied to the control of a multiproduct batch plant.

6.5 Control design by means of embedded maps

6.5.1 Stabilization problem for discretely controlled continuous systems

This section outlines two complementary approaches for stabilizing the periodic operation of discretely controlled continuous systems (DCCS– (Section 1.1.2)). Both procedures concentrate on the event generator, which is designed on the basis of an abstract model, which only captures the system behavior at the switching instants.

As depicted in Fig. 6.16, DCCS represent an important class of hybrid systems, where the control output of a continuous plant is regulated by the control actions of a purely discrete-event controller. The plant provides a finite number of *operation*

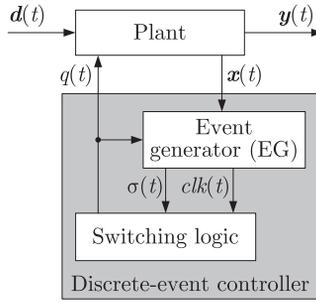


Fig. 6.16 Structure of a discretely controlled continuous system.

modes $q \in \mathcal{Q}$, with each mode being associated to different dynamical properties of the plant. While state jumps do not occur, a mode transition typically induces parameter and structural changes in the plant dynamics, which can be exploited to shape the output trajectory $y(t)$ as desired.

The discrete controller consists of an event generator, which translates continuous-time signals into discrete events $\sigma \in \Sigma$, and a discrete switching logic that defines all admissible mode transitions.

In contrast to switched systems, the plant does not exhibit an equilibrium, which is common to all operation modes. This demands for novel controller design concepts, which are particularly not based on the concept of equilibrium stability.

The primary control objective of DCCS is to achieve region stability 3.4.2, i.e. to drive the continuous state $x(t) \in \mathbb{R}^n$ or the output $y(t) \in \mathbb{R}^r$ from an initial region \mathcal{X}_I into a terminal set \mathcal{X}_T of the state space and to maintain it there at stationary operation (Fig. 6.17) despite the influence of strong disturbances. To successfully execute this transition, the discrete controller must generate a suitable mode sequence $q(t)$ with the modes being arranged in the correct order and the transitions occurring at the right time instants. Since the terminal set \mathcal{X}_T usually does not contain any equilibrium state of the plant, the controller must persistently switch the plant among its modes forever. Consequently, even simple DCCS exhibit a variety of complex behaviors [79, 167, 444].

Design challenges of DCCS Discretely controlled continuous system appear throughout many application domains, such as power electronics, process engineering, robotics, mechatronics, production management, traffic management or even biology [186, 196, 275, 444]. There, switching between different operation modes does not only serve the purpose of improving the loop's performance, but is indeed fundamental to generate otherwise impossible forms of behavior.

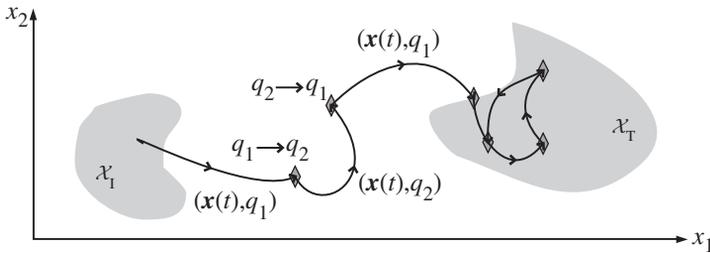


Fig. 6.17 Control task of DCCS.

A major challenge in the controller design of a DCCS is to systematically derive an event generator, i.e. a switching law, that translates the evolution of the continuous and the discrete state into a suitable event sequence $\sigma(t)$, such that the system specifications are met.

Example 6.6 DC-DC converter

The working principle of switching power converters relies on switching the circuit topology perpetually. The simple boost converter described in (Section 1.3), for instance, only accomplishes the transformation a low DC input voltage into a high DC output voltage, because the inductor L can be charged over the source in less time, than it takes for the capacitor C to discharge over the load R . Hence, only by alternating between the circuit topologies at very high frequencies, the energy stored in the capacitor accumulates until it reaches the required level.

A boost converter operating in continuous conduction mode certainly is a very simple bimodal hybrid systems with alternating modes $\mathcal{Q} = \{\text{on}, \text{off}\}$. Here, the only control influence, by which the average output voltage of the plant can be controlled as desired, is given by the switching times $t(k)$. \square

Surprisingly, the latter situation generalizes to many other real-world applications, even to those, which possess a large number of operation modes. Typically, the determination of a proper cyclic mode sequence is an essential part of the initial plant design. If the mode sequence is not fixed a priori, at least all executions, which evolve in the vicinity of a stationary trajectory, generate the same mode sequence.

Since even at stationary operation, switching the mode is continued infinitely long, the state $x(t)$ persistently oscillates around an *average* value. It *never* converges towards an equilibrium point and its motion may be chaotic. For many applications, however, maintaining $x(t)$ inside \mathcal{X}_T is not enough. Additionally, those applications require the plant to operate periodically in order to comply with switching frequency restrictions and other essential loop specifications.

6.5.2 Modeling the behavior of discretely controlled continuous systems

Hybrid automaton model of a DCCS As illustrated in Fig. 6.16, a DCCS can be separated into three distinct components:

1. a continuous subsystem, which constitutes the plant;
2. an event generator, which acts as an interface between the continuous-valued and the discrete-valued signal space (Sect. 1.1.3); and
3. a switching logic, which determines the successor mode to be activated at the subsequent switching.

The closed-loop behavior can be represented by a hybrid automaton $H = (\mathcal{Q}, \mathcal{X}, \mathbf{f}, \text{Init}, \text{Inv}, \mathcal{E}, \mathcal{G}, \mathcal{R})$ defined in (Section 3.1). \mathcal{Q} and $\mathcal{X} \subseteq \mathbb{R}^n$ denote the sets of discrete and continuous state variables, $q(t)$ and $\mathbf{x}(t)$ represent the corresponding trajectories.

The continuous subsystem Σ_c is described by a piecewise-Lipschitz nonlinear vector field

$$\frac{d\mathbf{x}}{dt}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{d}(t), q(t)), \quad \mathbf{x}(t_0) = \mathbf{x}_0, \quad (6.39)$$

with inputs $q(t) \in \mathcal{Q}$ and $\mathbf{d}(t) \in \mathbb{R}^d$ being the mode and the disturbance signals. The set of initial states $\text{Init} = \{(x_0, q_0)\}$ is usually single valued and the domain $\text{Inv}(q)$ of each mode q is identical to the \mathbb{R}^n .

The discrete subsystem Σ_d is represented by a deterministic transition function

$$\bar{q}(k+1) = \Theta(\bar{q}(k), \sigma(k+1)), \quad \bar{q}(0) = q_0, \quad (6.40)$$

which maps the currently activated mode $\bar{q}(k)$ and the potentially generated event symbol $\bar{\sigma}(k+1)$ onto the successor mode $\bar{q}(k+1)$. In the sequel, all signals sampled at switching instants $\bar{t}(k)$ are indicated by a bar and enumerated by a counter k .

Each edge $e \in \mathcal{E}$ of the graph is labeled by a unique tuple (q, σ) and the related guard set \mathcal{G} of the extended state space $\mathbb{R}^n \times \mathbb{R}$ is implicitly defined as

$$\mathcal{G}(q, \sigma) = \{(\mathbf{x}, t) : \Phi(\mathbf{x}(t), t, q, \sigma) = 0\}. \quad (6.41)$$

Definition Eq. (6.41) is based upon the event function

$$\Phi(\mathbf{x}(t), t, q(t), \sigma(t)) = \begin{cases} \Phi_{\sigma_1}(\mathbf{x}(t), t, q(t)) & \text{if } \sigma(t) = \sigma_1, \\ \vdots & \\ \Phi_{\sigma_V}(\mathbf{x}(t), t, q(t)) & \text{if } \sigma(t) = \sigma_V, \end{cases} \quad (6.42)$$

which is composed of switching conditions that evaluate to zero, i.e. $\Phi(\bullet, \bullet, q, \sigma) = 0$, whenever an event σ occurs in mode q . At each time instant the event signal

$$\sigma(t) = \arg \min_{\sigma_i} |\Phi(\mathbf{x}(t), t, q(t), \sigma_i)| \quad (6.43)$$

obtains the value σ_i , for which $|\Phi|$ is minimal. The time elapsed in between two consecutive mode transitions $\bar{\tau}(k) = \bar{t}(k+1) - \bar{t}(k)$ is referred to as a mode's activation

duration or dwell time. Furthermore, the execution of a DCCS starting in (x_0, q_0) is denoted by the collection $\chi(x_0, q_0, t_0) = (t, \mathbf{x}(t), q(t))$.

Classification of switching laws Concerning the structure of the event function (6.42), every switching law belongs to one of the three following classes:

Definition 6.10 (Event-driven, clock-driven and mixed switching)

A switching law is called:

- *event-driven, if all switching conditions are independent of time, i.e. $\Phi(\mathbf{x}(t), t, q(t), \sigma(t)) = \Phi(\mathbf{x}(t), q(t), \sigma(t))$;*
- *clock-driven, if all switching conditions are independent of the continuous state $\mathbf{x}(t)$, i.e. $\Phi(\mathbf{x}(t), t, q(t), \sigma(t)) = \Phi(t, q(t), \sigma(t))$;*
- *a mixed law, if it is neither purely clock-driven nor purely event-driven.*

Depending on the type of switching law, it may or may not be possible to explicitly solve (6.42) for the switching times $\bar{t}(k)$ or the activation durations $\bar{\tau}(k)$. This aspect essentially determines, whether or not a closed form expression of an execution χ can be obtained.

Besides the type of switching law, an event function (6.42) may as well be classified according to the geometry of the guard sets, which it defines. The case of a piecewise affine function (6.42) in \mathbf{x} and t is particularly relevant for many applications.

Definition 6.11 (Switching plane) The guard sets $\mathcal{G}(q, \sigma)$ represent switching planes $\mathcal{S}(q, \sigma)$, iff the event function

$$\Phi(\mathbf{x}(t), t, q(t), \sigma(t)) = \mathbf{n}_x^T(q, \sigma) \mathbf{x}(t) + n_T(q, \sigma) \tau - r(q, \sigma) \quad (6.44)$$

is piecewise affine, where the vectors $(\mathbf{n}_x^T(q, \sigma) \ n_T(q, \sigma))$ coincide with the planes' normal directions in the extended state space $\mathbb{R}^n \times \mathbb{R}$ and $r(q, \sigma)$ define the planes' distances to the origin.

Sampling signals at switching instants The hybrid automaton representation of a DCCS may be well suited for accurately simulating the system's behavior. Due to its complexity and generality, however, this model is only of little value for solving the design task. Instead, this task requires a model, which uniquely relates the event function parameters $\mathbf{n}_x^T(q, \sigma)$ and $r(q, \sigma)$ to the system's stationary behavior. As control actions only affects the system at the switching times $\bar{t}(k)$, it is convenient to sample an execution exclusively at these instants and obtain the sequence

$$\bar{\chi}(x_0, q_0, t_0) = ((\bar{\mathbf{x}}(0) \ \bar{q}(0) \ \bar{t}(0))^T, (\bar{\mathbf{x}}(1) \ \bar{q}(1) \ \bar{t}(1))^T, \dots) \quad (6.45)$$

of hybrid switch points $(\bar{\mathbf{x}}(k) \ \bar{q}(k) \ \bar{t}(k))^T$. The model, from which this sequence can be computed, is called the embedded map of a DCCS.

Definition 6.12 (Embedded map) *The mapping*

$$(\bar{\mathbf{x}}(k+1) \quad \bar{q}(k+1) \quad \bar{t}(k+1))^T = \mathbf{H}(\bar{\mathbf{x}}(k), \bar{q}(k), \bar{t}(k)), \quad (6.46)$$

which relates the current hybrid switch points $(\bar{\mathbf{x}}(k) \quad \bar{q}(k) \quad \bar{t}(k))^T$ to the next one, is called the embedded map of a DCCS.

The mapping (6.46) can be decomposed into three parts $\mathbf{H} = (\mathbf{H}_x \quad \mathbf{H}_q \quad \mathbf{H}_t)^T$, one for each element of the hybrid switch point vector.

Existence of analytic expressions for the embedded map To obtain a closed form expression of the embedded map (6.46) requires to solve the switching conditions (6.41) for the switching times $\bar{t}(k)$ symbolically. As a necessary condition for this, an analytic solution of the continuous vector field (6.39) must be found, which only exists for a very restricted class of differential equations. Assuming the event function to be piecewise affine (6.44), the embedded map can be expressed analytically for the following types of DCCS:

1. DCCS with n -dimensional piecewise integrator dynamics $d\mathbf{x}/dt = \mathbf{B}(q)$.
2. DCCS with one-dimensional piecewise affine continuous dynamics $(dx/dt) = -ax(t) + b$ and a purely event- or clock-driven switch law.
3. DCCS with n -dimensional piecewise affine continuous dynamics $d\mathbf{x}/dt = \mathbf{A}(q)\mathbf{x}(t) + \mathbf{B}(q)$ and a clock-driven switching law.

For all other DCCS it highly depends upon the structure and the parameters of the continuous subsystem Σ_c as well as the event function Φ , whether or not it is possible to obtain an explicit expression of \mathbf{H} .

For arbitrary switching laws and arbitrary state space dimensions $n > 1$, only the first class of DCCS permits a closed-form representation of the embedded map. Such systems are referred to as cyclic linear automata (CDLA) [444], if the successor mode $\bar{q}(k+1) = \phi(\bar{q}(k))$ uniquely follows from the current mode $\bar{q}(k)$.

Theorem 6.5 [444] *The embedded map of a cyclic linear automaton (CDLA) is given by*

$$\mathbf{H}(\bar{\mathbf{x}}(k), \bar{q}(k), \bar{t}(k)) = \begin{pmatrix} \left(\mathbf{I} - \frac{\mathbf{B}(\bar{q}(k))\mathbf{n}_x^T(\bar{q}(k))}{\mathbf{n}_x^T(\bar{q}(k))\mathbf{B}(\bar{q}(k)) + n_T(\bar{q}(k))} \right) \bar{\mathbf{x}}(k) + \frac{r(\bar{q}(k))\mathbf{B}(\bar{q}(k))}{\mathbf{n}_x^T(\bar{q}(k))\mathbf{B}(\bar{q}(k)) + n_T(\bar{q}(k))} \\ \Theta(\bar{q}(k)) \\ \bar{t}(k) + \frac{r(\bar{q}(k)) - \mathbf{n}_x^T(\bar{q}(k))\bar{\mathbf{x}}(k)}{\mathbf{n}_x^T(\bar{q}(k))\mathbf{B}(\bar{q}(k)) + n_T(\bar{q}(k))} \end{pmatrix}. \quad (6.47)$$

As can be seen from (6.47), the continuous component \mathbf{H}_x is an affine expression of $\bar{\mathbf{x}}(k)$, but it already depends nonlinearly on the design parameters $\mathbf{n}_x^T(\bar{q}(k))$ and $n_T(\bar{q}(k))$. Although the second class of systems is still very simple, an explicit expression of the embedded map can yet only be found for purely clock- or event-driven switching laws.

Theorem 6.6 *The embedded map of 1-D piecewise linear DCCS with an event function according to (6.44) and $n_T = 0$ is equal to*

$$\mathbf{H}(\bar{x}(k), \bar{q}(k), \bar{t}(k)) = \begin{pmatrix} r(\bar{q}(k))/n_x(\bar{q}(k)) \\ \Theta(\bar{q}(k)) \\ \bar{t}(k) - \ln\left(\frac{r(\bar{q}(k))a-b}{n_x(\bar{q}(k))\bar{x}(k)a+n_x(\bar{q}(k))b}\right) \frac{1}{a} \end{pmatrix}. \quad (6.48)$$

Because of the simple dynamics, the first element $\bar{x}(k+1)$ of (6.48) is trivial. However, the third entry of (6.48) clearly highlights the complex relation between the design parameters and the switching times, which usually prevents a closed form representation of \mathbf{H} for event-driven switching laws and $n > 1$.

Linear approximation of the embedded map If the continuous dynamics are not composed of decoupled integrators, the embedded map can only be evaluated numerically by simulation. Given two nominal switch points $\mathbf{x}^*(k)$ and $\mathbf{x}^*(k+1)$, which belong to an admissible solution $\mathbf{x}^*(t)$ of (6.39), a locally valid first-order approximation of the embedded map [172]

$$\delta\bar{\mathbf{x}}(k+1) = \left(\mathbf{I} - \frac{\mathbf{f}(\bar{\mathbf{x}}^*(k+1), \bar{q}(k)) \frac{\partial\Phi}{\partial\mathbf{x}}}{\frac{\partial\Phi}{\partial\mathbf{x}} \mathbf{f}(\bar{\mathbf{x}}^*(k+1), \bar{q}(k)) + \frac{\partial\Phi}{\partial t}} \right) \frac{\partial\bar{\mathbf{x}}^*(k+1)}{\partial\bar{\mathbf{x}}^*(k)} \delta\bar{\mathbf{x}}(k), \quad (6.49)$$

$$\frac{\partial\Phi}{\partial\bullet} = \frac{\partial\Phi}{\partial\bullet}(\bar{\mathbf{x}}^*(k+1), \bar{q}(k), \bar{\sigma}(k+1)) \quad (6.50)$$

can be computed. As illustrated in Fig. 6.18, it maps the current local deviation $\delta\mathbf{x}(k) = \bar{\mathbf{x}}(k) - \bar{\mathbf{x}}^*(k)$ from the nominal trajectory $\mathbf{x}^*(t)$ onto the deviation at the next switching.

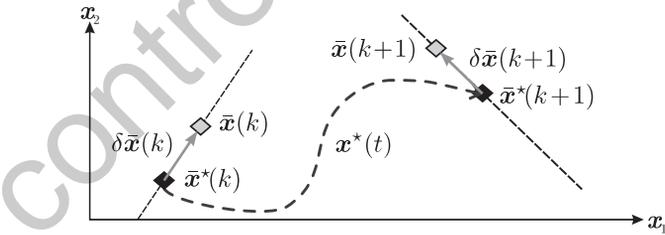


Fig. 6.18 Continuous part \mathbf{H}_x of the embedded map.

The event $\bar{\sigma}(k+1)$ is issued at $\mathbf{x}^*(k+1)$ and $\partial\bar{\mathbf{x}}^*(k+1)/\partial\bar{\mathbf{x}}^*(k)$ denotes the state transition matrix $\mathbf{Z}(t)$, which is obtained as a solution to the sensitivity equation

$$\frac{d\mathbf{Z}}{dt}(t) = \frac{\partial\mathbf{f}}{\partial\mathbf{x}}(\mathbf{x}(t), t, q(t)) \mathbf{Z}(t), \quad \mathbf{Z}(0) = \mathbf{I}$$

at $t = \bar{\tau}(k)$. If the event function (6.42) is piecewise affine, (6.49) becomes

$$\delta \bar{\mathbf{x}}(k+1) = \left(\mathbf{I} - \frac{\mathbf{f}(\bar{\mathbf{x}}^*(k+1), \bar{q}(k)) \mathbf{n}_x^T(\bar{q}(k), \bar{\sigma}(k+1))}{\mathbf{n}_x^T(\bar{q}(k), \bar{\sigma}(k+1)) \mathbf{f}(\bar{\mathbf{x}}^*(k+1), \bar{q}(k)) + n_t(\bar{q}(k), \bar{\sigma}(k+1))} \right) \frac{\partial \bar{\mathbf{x}}^*(k+1)}{\partial \bar{\mathbf{x}}^*(k)} \delta \bar{\mathbf{x}}(k), \quad (6.51)$$

which closely resembles the first element of (6.47).

6.5.3 Stabilization of periodic stationary solutions

Periodic stationary executions An important control objective of DCCS is to stabilize a periodic stationary execution, which is defined as follows:

Definition 6.13 (p -periodic execution) [444] *An execution*

$$\chi^*(\mathbf{x}_0, q_0, t_0) = (t, \mathbf{x}^*(t), q^*(t))$$

of the model (6.39)–(6.42) is called p -periodic, if there exists a time $T = \sum_{k=0}^{p-1} \bar{\tau}(k)$, such that

$$q^*(t+T) = q^*(t), \quad \mathbf{x}^*(t+T) = \mathbf{x}^*(t)$$

holds for all $t \geq 0$ and T is minimal. A p -periodic solution is called elementary, if

$$\bar{q}^*(k+l) \neq \bar{q}^*(k+m), \quad \forall l, m < p$$

is satisfied.

Definition 6.14 (Isolated closed orbit) [248] *A closed orbit Γ traced out by the continuous component $\mathbf{x}^*(t)$ of the periodic solution $\chi^*(\mathbf{x}_0, q_0, t_0)$ is called a limit cycle, if it is isolated, meaning there does not exist any other periodic solution in its neighborhood.*

Without loss of generality, any limit cycle is assumed to be elementary from here on. Eventually, this requires a re-definition of the plant's operation modes. To simplify notation, the switch points $\bar{\mathbf{x}}^*(k+pl)$, $l \in \mathbb{N}$ associated to the activation of mode \bar{q}_k^* will be referred to as $\bar{\mathbf{x}}^*(\bar{q}_k^*)$. Likewise, the activation duration of mode \bar{q}_k^* is referred to as $\bar{\tau}^*(\bar{q}_k^*)$.

Proposition 6.3 *Any sampled periodic execution $\bar{\chi}^*$, which generates an elementary limit cycle Γ , is uniquely represented by the finite sequences $Q_\Gamma = (\bar{q}_0^*, \dots, \bar{q}_{(p-1)}^*)$, $X_\Gamma = (\bar{\mathbf{x}}^*(\bar{q}_0^*), \dots, \bar{\mathbf{x}}^*(\bar{q}_{(p-1)}^*))$ and $T_\Gamma = (\bar{\tau}^*(\bar{q}_0^*), \dots, \bar{\tau}^*(\bar{q}_{(p-1)}^*))$.*

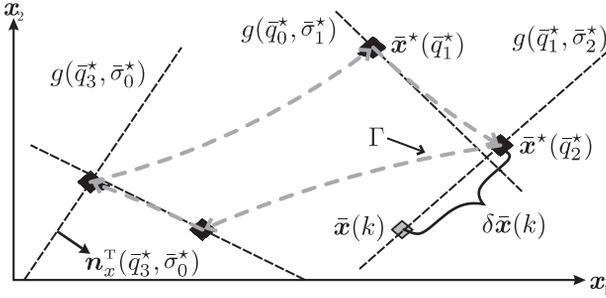


Fig. 6.19 Sample limit cycle of a two-dimensional DCCS of order $p = 4$.

Proposition 6.3 is intuitively clear, as the evolution of $\mathbf{x}(t)$ in between two switching instants is by definition unique and autonomous. A sample fourth-order limit cycle of a two-dimensional DCCS is depicted in Fig. 6.19. The closed dashed grey curve corresponds to $\mathbf{x}^*(t)$ and the black diamonds indicate the four points $\bar{\mathbf{x}}^*(\bar{q}_k^*)$, at which a mode switch occurs when the trajectory intersects with the switching planes $g(q, \sigma)$.

Stability of a periodic executions To analyze the *orbital* stability of a periodic execution χ^* , the stability of the associated limit cycle Γ must be determined.

Definition 6.15 (Asymptotical orbital stability) [248] A periodic execution χ^* is called asymptotically orbitally stable, if for any $\epsilon > 0$ there exists a $\delta > 0$, such that any execution χ , whose continuous trajectory $\mathbf{x}(t)$ starts at $\mathbf{x}(0)$ with $\|\mathbf{x}(0) - \mathbf{x}^*(0)\| < \delta$, converges towards χ^* . The latter implies that $\text{dist}(\mathbf{x}(t), \Gamma) < \epsilon, \forall t > 0$ and $\lim_{t \rightarrow \infty} \text{dist}(\mathbf{x}(t), \Gamma) = 0$.

In this context, the return map, which is obtained by concatenation of the embedded map over a complete cycle Q_Γ , provides a powerful analysis tool.

Definition 6.16 (Return map) The mapping

$$\begin{aligned} & (\bar{\mathbf{x}}((k+1)p+l), \bar{q}((k+1)p+l), \bar{t}((k+1)p+l))^T \\ &= \mathbf{P}(\bar{\mathbf{x}}(kp+l), \bar{q}(kp+l), \bar{t}(kp+l)) \\ &= \mathbf{H} \circ \dots \circ \mathbf{H}(\bar{\mathbf{x}}(kp+l), \bar{q}(kp+l), \bar{t}(kp+l)), l \in [0, p-1] \end{aligned} \quad (6.52)$$

which relates the hybrid switch point $(\bar{\mathbf{x}}(kp+l), \bar{q}(kp+l), \bar{t}(kp+l))^T$ to the hybrid switch point, at which mode $\bar{q}(kp+l) = \bar{q}((k+1)p+l)$ is first reactivated, is called the first return map of a DCCS.

Like the embedded map, the return map consists of three components

$$\mathbf{P} = (\mathbf{P}_x \ \mathbf{P}_q \ \mathbf{P}_t)^T.$$

The continuous component \mathbf{P}_x describes how the switch points evolve inside the switching surface of a particular mode.

Proposition 6.4 All hybrid switch points $(\bar{x}^*(\bar{q}_k^*) \ \bar{q}_k^* \ \bar{\tau}^*(\bar{q}_k^*))^\top$ of a sampled periodic execution $\bar{\chi}^*$ define fixpoints

$$(\bar{x}^*(\bar{q}_k^*) \ \bar{q}_k^* \ \bar{\tau}^*(\bar{q}_k^*))^\top = P(\bar{x}^*(\bar{q}_k^*), \bar{q}_k^*, \bar{\tau}^*(\bar{q}_k^*))$$

of the first return map P .

Since this proposition especially holds for the continuous component P_x , asymptotic orbital stability of a limit cycle Γ implies asymptotic stability of the corresponding fixpoint of P_x and vice versa. As long as this fixpoint $\bar{x}^*(\bar{q}_k^*)$ is not located in the intersection of two switching planes, any trajectory starting in this point's neighborhood returns to the same switching plane after a complete switching cycle Q_Γ . Then, the return map is a local diffeomorphism [172], which is continuously differentiable in a vicinity around the orbit Γ . This property allows conclusions to be made about local stability from the return map's Jacobian

$$\frac{dP_x}{dt}(\bar{x}^*(\bar{q}_k^*), \bar{q}_k^*, \bar{\tau}^*(\bar{q}_k^*)) = \prod_{i=k}^{k+p-1} \frac{dH_x}{dt}(\bar{x}^*(\bar{q}_i^*), \bar{q}_i^*, \bar{\tau}^*(\bar{q}_i^*)), \quad (6.53)$$

which is computed by taking the product of all approximations (6.49) over a cycle Q_Γ .

Theorem 6.7 [248] A periodic trajectory $x^*(t)$ is locally orbitally asymptotically stable, iff the eigenvalues

$$m_i = \lambda_i \left(\prod_{k=0}^{p-1} \frac{dH_x}{dt}(\bar{x}(k), \bar{q}(k), \bar{\tau}(k)) \right), \quad (6.54)$$

of the return map's Jacobian evaluated at any hybrid switch point

$$(\bar{x}^*(\bar{q}_k^*) \ \bar{q}_k^* \ \bar{\tau}^*(\bar{q}_k^*))^\top$$

are stable, i.e. $|m_i| < 1$.

Given a completely defined DCCS, all fixpoints of the return map (6.52) can be retrieved numerically, for instance by application of shooting methods [215]. The eigenvalues m_i of (6.53) are called *characteristic multipliers* of Γ and are identical for all hybrid switch points $(\bar{x}^*(\bar{q}_k^*) \ \bar{q}_k^* \ \bar{\tau}^*(\bar{q}_k^*))^\top$.

Checking for stability of periodic executions χ^* is loosely related to checking stability of common equilibria via Lyapunov's indirect method (Section 3.4.5) and gain automata, as introduced in (Section 3.4.1).

Design of stabilizing switching planes Since the return map explicitly relates the design parameters $n_x(q, \sigma)$, $n_d(q, \sigma)$ to the stability of the limit cycle, it can be used to solve the inverse problem, namely to determine an event function, which guarantees orbital stability.

Design problem. Given a plant with continuous dynamics (6.39) and an admissible stationary limit cycle Γ represented by the finite sequences Q_Γ , X_Γ , and T_Γ , find an event function (6.42), such that Γ is orbitally stable.

Except for cyclic linear automata, for which global orbital stability can be achieved, the linearized return map (6.53) only allows the derivation of locally stabilizing switching planes. To compute such planes, two complementary approaches are admissible:

1. Dynamically adjust nominal switching planes at run-time to reduce a deviation from the limit cycle and thereby achieve orbital stability.
2. Determine static switching plane orientations that guarantee at least local orbital stability.

In both approaches, the discrete controller responds to a deviation from the nominal trajectory by exclusively adjusting all subsequent switching times, whereas the order of the modes in the periodic sequence Q_Γ is preserved.

Dynamic adjustment of switching planes by means of a switching surface controller

The first concept is centered around classical feedback control theory. To begin with, the nonlinear dynamics, represented by the embedded map (6.46) are linearized around the desired stationary orbit (all $\bar{x}^*(\bar{q}_k^*) \in X_\Gamma$) and the nominal control inputs $\bar{\tau}^*(\bar{q}_k^*) \in T_\Gamma$. Instead of directly affecting the activation duration, $\bar{\tau}^*(\bar{q}_k^*)$ is corrected implicitly by adjusting the corresponding switching plane's orientation and/or its distance to the origin (Fig. 6.20). Thereby, the event-driven nature of the switching law is preserved and with it the good disturbance attenuation properties of the control loop. As a result of the linearization, a local approximation

$$\begin{aligned}
 \delta \bar{x}(k+2) &= \frac{d\mathbf{H}_x^{(2)}}{d\mathbf{x}}(\bar{x}^*(\bar{q}_k^*), \bar{x}^*(\bar{q}_k^*), \bar{\tau}^*(\bar{q}_k^*), \mathbf{n}_0(\bar{q}_k^*), r_0(\bar{q}_k^*))\delta \bar{x}(k) \\
 &+ \frac{d\mathbf{H}_x^{(2)}}{d\mathbf{n}}(\bar{x}^*(\bar{q}_k^*), \bar{x}^*(\bar{q}_k^*), \bar{\tau}^*(\bar{q}_k^*), \mathbf{n}_0(\bar{q}_k^*), r_0(\bar{q}_k^*))\delta \bar{\mathbf{n}}(k) \\
 &+ \frac{d\mathbf{H}_x^{(2)}}{d\mathbf{r}}(\bar{x}^*(\bar{q}_k^*), \bar{x}^*(\bar{q}_k^*), \bar{\tau}^*(\bar{q}_k^*), \mathbf{n}_0(\bar{q}_k^*), r_0(\bar{q}_k^*))\delta \bar{\mathbf{r}}(k) \quad (6.55)
 \end{aligned}$$

of the two-step *controlled embedded map* is obtained [574], where $\delta \bar{\mathbf{n}}(k)$, $\delta \bar{\mathbf{r}}(k)$ denote the perturbations of the switching plane parameters from their nominal values $\mathbf{n}_0(\bar{q}_k^*)$, $r_0(\bar{q}_k^*)$. From the concatenation of (6.55) over a complete cycle Q_Γ , the linearized *controlled return map* is obtained, which provides the model for determining the feedback law $(\delta \bar{\mathbf{n}}(k), \delta \bar{\mathbf{r}}(k))^T = -\mathbf{K}^T(k)\delta \mathbf{x}(k)$ by application of results from periodic control systems [23, 602, 643]. The overall procedure of the first control approach is summarized in Algorithm 6.3.

The nominal switching planes to be determined in the initialization phase of the algorithm are expected to render the target set \mathcal{X}_T control invariant, but the continuous trajectory may be either chaotic or periodic. Hence, the limit cycle Γ , which is subsequently computed in the second step, may or may not be stable. If it is already orbitally stable, then usually the convergence rate and other important loop properties

Algorithm 6.3 Dynamic adjust of switching planes

Given:

Vector field $f(x, q)$ of plant and its set \mathcal{Q} of operation modes.

Init:

- Determine nominal switching planes $\mathcal{G}_0(q, \sigma)$, which account for the system specs.
- Use $\mathcal{G}_0(q, \sigma)$ to determine an admissible limit cycle Γ contained in \mathcal{X}_T and obtain its sampled representation Q_Γ, X_Γ , and T_Γ .
- Parametrize the switching surfaces with respect to δn , δr and compute the controlled embedded map (6.55).
- Compute the gain $K^T(k)$ of the switching surface controller by means of the controlled embedded map and methods from periodic control systems.

- 1: Sample the current deviation $\delta \bar{x}(k) = \bar{x}(k) - \bar{x}^*(\bar{q}(k))$ between the execution χ and the desired limit cycle Γ at the switching instants.
- 2: Apply a switching surface controller (SSC) to pre- or postpone the next switching instant $\bar{i}(k + 1)$, i.e. adjust the nominal activation duration $\bar{\tau}^*(\bar{q}_k^*)$ of the currently activated mode $\bar{q}(k) = \bar{q}_k^*$, by perturbing the orientation and/or the location of the switching plane $g_0(\bar{q}_k^*, \bar{\sigma}_{k+1}^*)$ to be subsequently intersected (Fig. 6.21).
- 3: Wait for the next two mode transitions and then repeat procedure at step 1

Result: Event generator that locally stabilizes the limit cycle Γ .

can still be improved by a dynamical adaptation of the switching planes. If it is unstable, then its stabilization inevitably requires to adjust the nominal planes. The steps of the dynamical adjustment and their influence on the run of $x(t)$ are illustrated in Fig. 6.20. In Fig. 6.20(b), the subsequently intersected plane $g_0(\bar{q}_k^*, \bar{\sigma}^*(k + 1))$ is rotated around the point $x_{RP}(\bar{q}_k^*)$, such that the current deviation $\delta \bar{x}(k)$ is compensated within the next two mode transition. Clearly, the same result can be achieved by a mere relocation of the plane, while maintaining its nominal orientation.

Computation of static switching planes The stabilization approach explained in the previous paragraph exhibits two major drawbacks, namely:

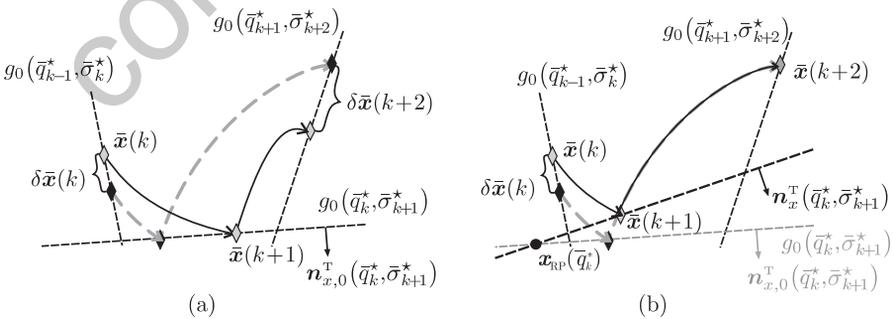


Fig. 6.20 Stabilization of a limit cycle by means of dynamical switching plane adjustments: (a) trajectory evolution without adjustment; (b) rotation of $g_0(\bar{q}_k^*, \bar{\sigma}_{k+1}^*)$ around $x_{RP}(\bar{q}_k^*)$, such that $\delta \bar{x}(k+2)$ vanishes.

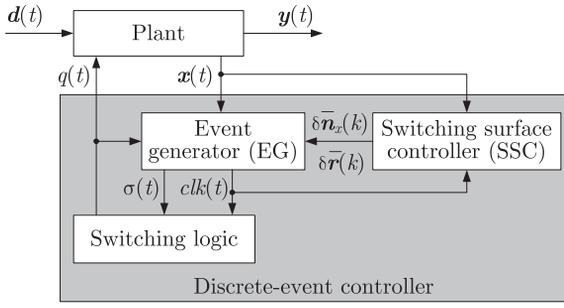


Fig. 6.21 DCCS augmented by a switching surface controller, which acts on the parameters of the event function.

1. any control action affects the continuous flow $x(t)$ not earlier than the switching instant one step ahead, thereby causing a potentially large reaction delay;
2. the planes' initial nominal orientations n_x strongly influence the amplitude of the control actions, which in turn determines the size of the region in which the controller performs acceptably.

A remedy to both issues is to compute nominal switching planes, which are in a sense oriented optimally with respect to the linearized dynamics and a given cost criterion. It shows that like in the previous approach, the original control problem can be reformulated as a constrained periodic control problem. Indeed, the switching plane orientations n_x can be interpreted as feedback gains, which properly update the activation durations $\bar{\tau}(k)$. The following result can be retrieved by enforcement of the constraints

$$n_x^T(\bar{q}_k^*, \bar{\sigma}_{k+1}^*) f(\bar{x}(\bar{q}_{k+1}^*), \bar{q}_k^*) = 1 \tag{6.56}$$

and application of a periodic state transformation to (6.51) [575].

Theorem 6.8 The p -periodic linear system $\Sigma = \{A_d(k), b_d(k)\}$

$$\zeta(k+1) = A_d(k)\zeta(k) + b_d(k)u(k), \text{ with} \tag{6.57}$$

$$A_d(k+p) = A_d(k) = \partial \bar{x}(\bar{q}_{k+2}^*) / \partial \bar{x}(\bar{q}_{k+1}^*), \tag{6.58}$$

$$b_d(k+p) = b_d(k) = A_d(k) f(\bar{x}(\bar{q}_{k+1}^*), \bar{q}_k^*), \tag{6.59}$$

under a p -periodic state feedback

$$u(k) = -k^T(k) \zeta(k) \text{ with } k^T(k+p) = k^T(k), \tag{6.60}$$

is equivalent to the system (6.53), iff for all k the following is true:

$$1. |\alpha(k)| = |n_x^T(\bar{q}_k^*, \bar{\sigma}_{k+1}^*) f(\bar{x}(\bar{q}_{k+1}^*), \bar{q}_k^*)| > 0, \tag{6.61}$$

$$2. k^T(k) = n_x^T(\bar{q}_k^*, \bar{\sigma}_{k+1}^*) / \alpha(k), \tag{6.62}$$

$$3. \det(A_d(k) - b_d(k)k^T(k)) = 0. \tag{6.63}$$

Hence, admissible normal directions n_x are once more obtained by solving a well-known periodic control problem, either by application of explicit pole-placement algorithms [602, 643] or by application of periodic H_2 and H_∞ design methods [23]. According to (6.62) the gain $k^T(k)$ and the associated normal $n_x(\bar{q}_k^*)$ are colinear vectors, scaled by a factor $\alpha(k)$. Such a scaling does not influence the multipliers m_i .

Theorem 6.9 *The piecewise-affine event function*

$$\Phi(x(t), t, \bar{q}_k^*, \bar{\sigma}_{k+1}^*) = k^T(k) / \|k^T(k)\| (x(t) - \bar{x}(\bar{q}_{k+1}^*)), \quad (6.64)$$

with $k^T(k)$ being a stabilizing periodic state feedback of the equivalent periodic system Σ , guarantees local orbital stability of the limit cycle Γ .

The steps of the design procedure are summarized in Algorithm 6.4. In contrast to the dynamic switching plane adjustment, the implementation of such static switching planes requires less computational power at run-time. Furthermore, the resulting discrete controller adjusts the activation duration instantaneously, such that the loop behavior is free of any delay.

Algorithm 6.4 Determination of stabilizing switching surfaces

Given: Vector field $f(x, q)$ of plant and its set \mathcal{Q} of operation modes.

- 1: Compute the periodic matrices $A_d(k)$, $b_d(k)$ according to (6.58), (6.59).
- 2: Verify stabilizability of the equivalent periodic system Σ .
- 3: Solve the constrained periodic state-feedback problem to obtain gains $k^T(k)$.
- 4: Set the event function parameters to $n^T(\bar{q}_k^*, \bar{\sigma}_{k+1}^*) = k^T(k) / \|k^T(k)\|$ and compute $r(\bar{q}_k^*) = n^T(\bar{q}_k^*, \bar{\sigma}_{k+1}^*) \bar{x}(\bar{q}_{k+1}^*)$.

Result: Event generator that locally stabilizes the limit cycle Γ .

6.5.4 Application: Event generator design of a boost converter

To illustrate the performance of both design approaches outlined in the last subsection, each procedure is used to determine an event-driven switching strategy, which stabilizes a periodic stationary operation of the boost converter described in (Section 1.3.3). This second order system clearly exemplifies the benefits gained from a model-based design. In contrast to (Section 13), the emphasis here is put on the microscopic rather than the macroscopic converter behavior.

The converter is designed to operate in continuous conduction mode meaning that at stationary operation the modes alternate between 1 and 2. Using the parameters from Table 1.6, the converter may execute a limit cycle Γ with an elementary mode sequence $Q_\Gamma = (1, 2)$ and switch points $X_\Gamma = ((3.446 \text{ A } 30.21 \text{ V})^T, (4 \text{ A } 22.82 \text{ V})^T)$.

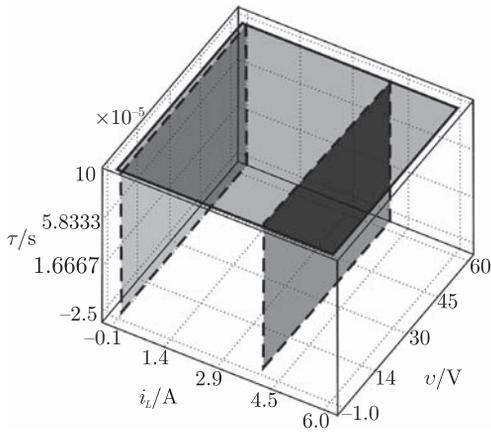


Fig. 6.22 Paraxial switching planes resulting from current-mode control.

Following traditional current-mode control a mixed switching law is obtained, where the switch S is opened whenever the inductor current i_L exceeds an upper threshold, in this case $i_{\max} = 4\text{A}$, and is closed again every T seconds. The corresponding switching planes are depicted in Fig. 6.22. This heuristic switching law already stabilizes the orbit Γ , however, by incorporating more information, i.e. the values of all state variables in the decision for or against a mode transition, the transient behavior as well as the robustness of the loop is improved considerably.

Design of static switching planes Static switching planes are computed by application of Algorithm 6.4. The design is executed for all edges in between mode one and two, while the switching conditions of all other edges remain unchanged. After computing the equivalent periodic system Σ , the stabilizability of Γ is verified numerically [603]. The periodic gains $k^T(k)$ are chosen to minimize the H_2 -norm of Σ [23].

Figure 6.23 shows the resulting switching plane configuration. The hatched planes are obtained by the model-based design procedure and replace the old paraxial ones. The sample trajectories of Fig. 6.24, which start in

$$\mathbf{x}_0 = (0\text{ A } 1\text{ A } 32\text{ V})^T, q_0 = 1,$$

indicate the fast convergence towards Γ . In this figure, the grey dotted line represents the trajectory obtained under conventional current mode control. As seen, with the model-based switching planes the continuous trajectory is forced onto the limit cycle Γ within six mode transitions or approximately $600\ \mu\text{s}$, while for current mode control, it takes more than 50 mode transitions to achieve something similar.

Dynamic adjustment of paraxial switching planes Using the paraxial switching planes of Fig. 6.22 as nominal ones, a switching surface controller was designed

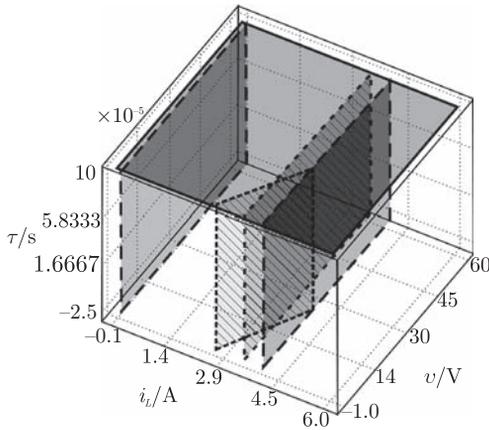


Fig. 6.23 Stabilizing switching planes for the boost converter.

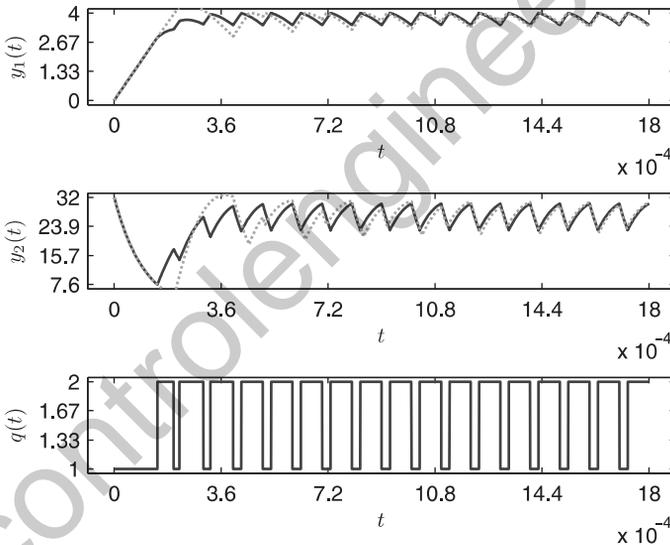


Fig. 6.24 Signals of the boost converter illustrating the fast convergence towards Γ .

according to Algorithm 6.3 and afterwards integrated into the control loop. Compared to the static plane design, a slightly worse local loop behavior is obtained (Fig. 6.25). Comparing the transient behavior once more to the one achieved under heuristic switching (grey dotted curves), a significantly improved convergence rate towards Γ is observed, despite the fact that all corrections applied by the SSC are small.

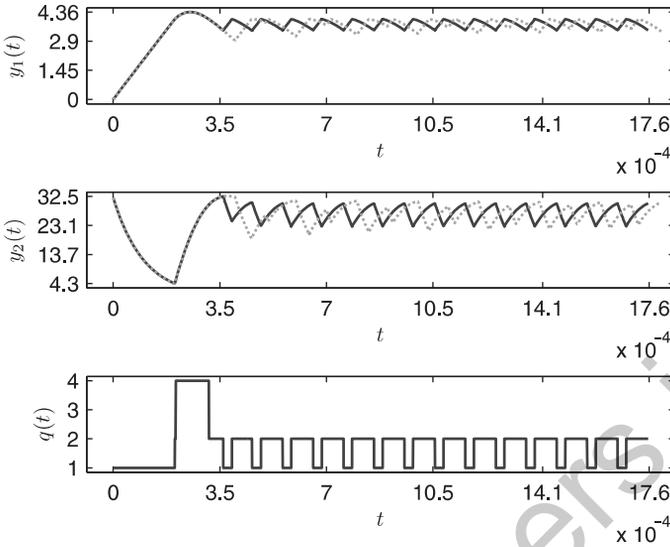


Fig. 6.25 Signals the boost converter obtained by application of a switching surface controller that adjusts the nominal switching planes at run-time.

Bibliographical notes

Diagnosis of quantized systems Surveys of analysis methods for quantized systems can be found in [576] and [84]. In [415, 416, 424] it has been shown that quantized systems have, in general, a nondeterministic quantized behavior and do not possess the Markov property.

Methods for abstracting discrete-event representations of discrete-time or discrete-event quantized systems have been proposed in [364, 366, 415, 416, 424, 525, 548, 609] all of which aim at finding complete models in the sense defined in (Section 6.3.4). [548] showed that, by using different definitions of the model state, a hierarchy of discrete abstractions can be obtained, which generate different numbers of spurious solutions.

There are only preliminary results concerning the question how to partition the signal spaces in order to obtain abstractions with a small number of spurious solutions [362, 414, 424]. The problem how to partition the state space in order to avoid the nondeterminism is still open [414]. A connection of the stochastic automaton as discrete-event description of quantized systems and the Frobenius–Perron operator is given in [421].

The complete solution to the diagnostic problem for stochastic automata given in [420] is the basis for the results reviewed in (Section 6.3.5). First results for quantized systems have been described in [422]. In [243] it has been shown that discrete-event representations of quantized systems can be used for diagnosis if and only if they are complete. This reference also gives an example to demonstrate that different models can be used for the same quantized system, all of which are complete but differ concerning the number of spurious solutions and, hence, yield diagnostic results of different precision.

The diagnosis of discrete-event quantized systems has been developed in [243, 419] and [418], where the latter reference presents a method that takes the temporal event distances into account. The corresponding abstraction methods are developed in [245] and [417].

The subject of this chapter has interesting connections to the literature on set-oriented control methods that have been developed as computational methods to evaluate the

properties of nonlinear dynamical systems. The basic idea of cell-to-cell mapping is described in the monograph [327] and the application of this methods for controller design in [286, 592]. Contrary to the state space partitions induced by quantizers described in this chapter, cell-to-cell mapping is based on the use of “sufficiently small” cells, such that the behavior for the whole cell is represented by the single trajectory starting in the center point of the cell. Hence, it is sufficient to investigate this single trajectory rather than the trajectory bundle that originates in the cell.

Application examples of fault diagnosis of quantized systems are presented in [422] (neutralization process), [406] (H_2 compressor), [242, 244] (diesel injection system), and [246, 485] (air path of a diesel motor), which is summarized here in (Section 6.3.6).

Discretely controlled continuous systems Discretely controlled continuous systems have been extensively investigated over the past decades by various research communities. Triggered by practically observed problems in power electronic devices, bifurcation analysis of nonsmooth systems has been addresses in many publications dating back as far as the late 70’s. A very exhaustive overview of nonsmooth bifurcation phenomena is presented in [79] and references therein. Approaches for locating limit cycles and analyzing their local or global stability properties are discussed in [215, 240, 269, 274, 321, 444, 560]. The monograph [444] provides an excellent and exhaustive treatment of the analysis of a simple class of DCCS.

Control approaches for DCCS can generally be divided into two groups: one group of methods focussed on the start-up and the execution of large transitions in the continuous state space and the other group of methods focussed on the stabilization of stationary solutions. Belonging to the first group, off-line optimal control of DCCS was developed in [29, 96, 218]. There, the switching times or switching surfaces are optimized off-line for a given initial state x_0 and a desired terminal state x_T with respect to a smooth cost functional. Extensions have been elaborated that allow for a dynamical update of the switching times or surfaces in case of disturbances or uncertainties in x_0 .

Recent approaches, which deal with the global stabilization of switched (average) equilibria, are published in [100, 186, 675]. They all rely on the existence of a quadratic Lyapunov-like energy function and achieve to drive any trajectory into a neighborhood of the average equilibrium. Inside this neighborhood, however, the execution may evolve arbitrarily.

Control of chaotic systems is an ongoing field of research in nonlinear dynamical systems. The problem of stabilizing unstable periodic orbits embedded in chaotic attractors has been investigated for many decades, resulting in many different control strategies that require different amounts of the knowledge about the limit set to be stabilized.

Stochastic hybrid systems

J. Lygeros and M. Prandini

Stochastic hybrid systems involve the interaction of continuous discrete and probabilistic dynamics, and thus pose considerable conceptual, theoretical, and practical challenges. In this chapter an overview of the modeling issues that arise in the study of stochastic hybrid systems is presented. Based on this discussion, a study of the problem of reachability analysis for stochastic hybrid systems is presented.

Chapter contents

7.1	Nondeterminism in hybrid systems	page 250
7.1.1	Deterministic and nondeterministic models	250
7.1.2	Hybrid automata	251
7.1.3	Executions	252
7.2	Continuous-time stochastic hybrid systems	255
7.2.1	Stochastic hybrid automata	255
7.2.2	Special cases and systems with inputs	259
7.3	Discrete-time stochastic hybrid systems	261
7.4	Reachability analysis	264
7.4.1	Reachability problem	264
7.4.2	Probabilistic safety analysis of stochastic finite automata	265
7.4.3	Probabilistic safety analysis of stochastic hybrid systems	268
7.4.4	Extensions	274

7.1 Nondeterminism in hybrid systems

7.1.1 Deterministic and nondeterministic models

Much of the work on hybrid systems has focussed on *deterministic* models that completely characterize the future of the system without allowing any uncertainty. In practice, it is often desirable to introduce uncertainty in the models, to allow, for example, under-modeling of certain parts of the system, external unmodeled disturbances, etc. To address this need, researchers in discrete-event and hybrid systems have introduced what are known as *nondeterministic* models. Here the evolution is defined in a *declarative* way (the system specifies what solutions are allowed) as opposed to the *imperative* way more common in continuous dynamical/control systems (the system specifies what the solution must be).

Nondeterministic hybrid systems allow uncertainty to enter in a number of places: choice of continuous evolution (modeled, for example, by a differential inclusion), choice of discrete transition destination, or choice between continuous evolution and a discrete transition. “Choice” in this setting may reflect disturbances that add uncertainty about the system evolution, but also control inputs that can be used to steer the system evolution.

Deterministic and nondeterministic hybrid systems are very versatile, can capture a wide range of behaviors encountered in practice and have proved invaluable in a number of applications. They do, however, have their limitations. In particular, nondeterministic systems provide no way of distinguishing between their many possible solutions. This implies that only worst-case analysis is possible with nondeterministic hybrid systems, as one can only pose *qualitative*, yes–no type questions. For example, a safety question for nondeterministic hybrid systems admits only one of two answers: “The system is safe” (if none of the solutions of the system ever reaches an unsafe state) or “the system is not safe” (if some solution reaches some unsafe state). In some applications this type of analysis may be too coarse, however, as the following example indicates.

Example 7.1 Air traffic management

In air traffic management (ATM) the question “Is it possible for an accident to happen in the ATM system?” is not particularly meaningful. Surely the answer must be “yes,” since several accidents happen every year. The intuitive notions of safety in air traffic are better captured by questions like “What is the probability that a fatal accident happens in the ATM system?” and “How can the probability of a fatal accident be reduced?” This provides the motivation for developing model classes that can provide *quantitative*, hence more useful, answers to questions such as these through stochastic models of hybrid systems. □

In this chapter we survey the main issues and alternatives that arise in the area of stochastic hybrid systems. Because the coverage is rather wide, the presentation in this chapter will be by necessity terse and will overlook several important technical issues. We also provide a more thorough view of a specific problem in the area of stochastic hybrid control, the so-called stochastic reachability problem: computing

(and in the presence of control inputs maximizing or minimizing) the probability that the system state will remain in a given subset of the state space.

The chapter is arranged four sections. The rest of the present section presents a brief overview of continuous-time nondeterministic hybrid models. The aim is to outline ways in which uncertainty can enter the dynamics of the system and hence relate the subsequent discussion of stochastic hybrid system to other chapters in the volume. In Section 7.2 we discuss how these sources of uncertainty can be replaced by probabilities and outline the different classes of stochastic hybrid models that arise in the process. In Section 7.3 we establish a connection to discrete-time stochastic hybrid models, which still provide a powerful modeling framework without some of the technical complications associated with their continuous-time counterparts. Finally, the concluding section presents in detail the problem of reachability for stochastic hybrid systems.

7.1.2 Hybrid automata

We start by giving a high level overview of a general class of nondeterministic hybrid systems. The main aim of the discussion is to outline the different types of uncertainty that can arise in this class of systems, so that the differences between nondeterministic and stochastic hybrid systems can be highlighted in the subsequent discussion. Therefore, we will not enter into questions like how general this class of systems is and under what conditions are models in this class well-posed. We will also restrict our attention to nondeterministic hybrid systems that evolve in continuous-time. We first recall the definition of the hybrid automata introduced in Chapter 3:

Definition 7.1 (Hybrid automaton) A hybrid automaton is a collection $H = (\mathcal{Q}, \mathcal{X}, \mathcal{F}, \text{Init}, \text{Dom}, \mathcal{E}, \mathcal{G}, \mathcal{R})$, where:

- \mathcal{Q} is a finite set of discrete states;
- $\mathcal{X} = \mathbb{R}^n$ is a set of continuous states;
- $\mathcal{F}(\cdot, \cdot) : \mathcal{Q} \times \mathcal{X} \rightarrow 2^{\mathcal{X}}$ is a differential inclusion;
- $\text{Init} \subseteq \mathcal{Q} \times \mathcal{X}$ is a set of initial states;
- $\text{Dom}(\cdot) : \mathcal{Q} \rightarrow 2^{\mathcal{X}}$ is a domain;
- $\mathcal{E} \subseteq \mathcal{Q} \times \mathcal{Q}$ is a set of edges;
- $\mathcal{G} : \mathcal{E} \rightarrow 2^{\mathcal{X}}$ is a guard condition;
- $\mathcal{R} : \mathcal{E} \times \mathcal{X} \rightarrow 2^{\mathcal{X}}$ is a reset map.

Recall that $2^{\mathcal{X}}$ denotes the power set (set of all subsets) of \mathcal{X} . The notation of Definition 7.1 suggests, for example, that the function Dom assigns a set of continuous states $\text{Dom}(q) \subseteq \mathcal{X}$ to to each discrete state $q \in \mathcal{Q}$. We refer to $(q, x) \in \mathcal{Q} \times \mathcal{X}$ as the state of H . The discrete state $q \in \mathcal{Q}$ is referred to as the mode of H .

Hybrid automata define possible evolutions for their state. Informally, starting from an initial value $(q_0, x_0) \in \text{Init}$, we select a solution of the differential inclusion $\dot{x} \in \mathcal{F}(q_0, x)$ and follow it (continuous evolution). While doing this the discrete state q remains constant $q(t) = q_0$. Continuous evolution can go on as long as x remains in $\text{Dom}(q_0)$. If at some point the continuous state x reaches the guard $\mathcal{G}(q_0, q_1) \subseteq \mathcal{X}$

of some edge $(q_0, q_1) \in E$, the discrete state may change value to q_1 . At the same time the continuous state gets reset to some value in $\mathcal{R}(q_0, q_1, \mathbf{x}) \subseteq \mathcal{X}$. After this discrete transition, continuous evolution resumes and the whole process is repeated.

Hybrid automata can be visualized as directed graphs $(\mathcal{Q}, \mathcal{E})$ with vertices \mathcal{Q} and edges \mathcal{E} . With each vertex $q \in \mathcal{Q}$, we associate a set of initial states $\{\mathbf{x} \in \mathcal{X} \mid (q, \mathbf{x}) \in \text{Init}\}$, a differential inclusion $\mathcal{F}(q, \cdot): \mathcal{X} \rightarrow 2^{\mathcal{X}}$ and a domain $\text{Dom}(q) \subseteq \mathcal{X}$. An edge $(q, q') \in \mathcal{E}$ starts at $q \in \mathcal{Q}$ and ends at $q' \in \mathcal{Q}$. With each edge $(q, q') \in E$, we associate a guard $\mathcal{G}(q, q') \subseteq \mathcal{X}$ and a reset function $\mathcal{R}(q, q', \cdot): \mathcal{X} \rightarrow 2^{\mathcal{X}}$. The directed graphs contain exactly the same information as Definition 7.1 and can therefore be treated as informal definitions of hybrid automata.

7.1.3 Executions

To formally define the solutions of hybrid automata we recall the definition of a *hybrid time set* [27, 427]; effectively equivalent notions of “supper-dense” hybrid time can be found in [209, 271, 430].

Definition 7.2 (Hybrid time set) *A hybrid time set is a sequence of intervals $\tau = \{I_0, I_1, \dots, I_N\} = \{I_i\}_{i=0}^N$, finite or infinite (i.e. $N = \infty$ is allowed) such that:*

- $I_i = [\tau_i, \tau'_i]$ for all $i < N$;
- if $N < \infty$ then either $I_N = [\tau_N, \tau'_N]$ or $I_N = [\tau_N, \tau'_N)$; and
- $\tau_i \leq \tau'_i = \tau_{i+1}$ for all i .

Notice that the right endpoint, τ'_i , of the interval I_i coincides with the left endpoint, τ_{i+1} of the interval I_{i+1} . The interpretation is that these are the times at which discrete transitions of the hybrid system take place. τ'_i corresponds to the time instant just before a discrete transition, whereas τ_{i+1} corresponds to the time instant just after the discrete transition.

Discrete transitions are assumed to be instantaneous, therefore $\tau'_i = \tau_{i+1}$. The advantage of this convention is that it allows one to model situations where multiple discrete transitions take place one after the other at the same time instant, in which case $\tau'_{i-1} = \tau_i = \tau'_i = \tau_{i+1}$. Since all the primitives in Definition 7.1 do not explicitly depend on time, we can without loss of generality take $\tau_0 = 0$.

Definition 7.3 (Hybrid automaton execution) *An execution, (τ, q, x) , of a hybrid automaton, H , comprises a hybrid time set $\tau = \{I_i\}_{i=0}^N$ and two sequences of functions, $q = \{q_i(\cdot)\}_{i=0}^N$ and $x = \{x_i(\cdot)\}_{i=0}^N$ with $q_i(\cdot) : I_i \rightarrow \mathcal{Q}$ and $x_i(\cdot) : I_i \rightarrow \mathcal{X}$ such that:*

- initial condition: $(q_0(\tau_0), x_0(\tau_0)) \in \text{Init}$;
- discrete evolution: for all i ,

$$(q_i(\tau'_i), q_{i+1}(\tau_{i+1})) \in \mathcal{E}, x_i(\tau'_i) \in \mathcal{G}(q_i(\tau'_i), q_{i+1}(\tau_{i+1})),$$

and $x_{i+1}(\tau_{i+1}) \in \mathcal{R}(q_i(\tau'_i), q_{i+1}(\tau_{i+1}), x_i(\tau'_i))$;

- continuous evolution: for all i ,
 1. $q_i(t) = q_i(\tau_i)$ for all $t \in I_i$;

2. $x_i(\cdot) : I_i \rightarrow \mathcal{X}$ is a solution to the differential inclusion

$$\frac{dx_i}{dt}(t) \in \mathcal{F}(q_i(t), x_i(t))$$

over I_i starting at $x_i(\tau_i)$; and

3. for all $t \in [\tau_i, \tau'_i]$, $x_i(t) \in \text{Dom}(q_i(t))$.

As expected several conditions need to be imposed on the elements of Definition 7.1 to ensure that executions in the sense of Definition 7.3 exist (see, for example, [27, 271, 427]). Definition 7.3 imposes a number of restrictions on the behaviors that the hybrid automaton finds “acceptable.” The first restriction dictates that the executions should start at an acceptable initial state in *Init*.

The second restriction determines when discrete transitions can take place and what the state after discrete transitions can be. The requirements relate the state before the discrete transition $(q_i(\tau'_i), x_i(\tau'_i))$ to the state after the discrete transition $(q_{i+1}(\tau_{i+1}), x_{i+1}(\tau_{i+1}))$: they should be such that $(q_i(\tau'_i), q_{i+1}(\tau_{i+1}))$ is an edge, $x_i(\tau'_i)$ belongs to the guard of this edge, and $x_{i+1}(\tau_{i+1})$ belongs to the reset map of this edge. In this context, it is convenient to think of the guard $\mathcal{G}(e)$ as *enabling* a discrete transition $e \in \mathcal{E}$: the execution *may* take a discrete transition $e \in \mathcal{E}$ from a state x as long as $x \in \mathcal{G}(e)$.

The third restriction determines what happens along continuous evolution, and when continuous evolution must give way to a discrete transition. The first part dictates that along continuous evolution the discrete state remains constant. The second part requires that along continuous evolution the continuous state flows according to the differential inclusion $\dot{x} \in \mathcal{F}(q, x)$. The third part requires that along continuous evolution the state must remain in the domain, $\text{Dom}(q)$, of the discrete state. In this context, it is convenient to think of $\text{Dom}(q)$ as *forcing* discrete transitions: the execution *must* take a transition if the state is about to leave the domain.

Freedom in the execution of hybrid automata Considerable freedom is allowed when defining the solution in this “declarative” way:

- There is a choice for the initial condition; any state in the set *Init* will work.
- The direction of the continuous motion at any point in time is in general not unique; any direction in $\mathcal{F}(q, x)$ will work.
- The mode after a discrete transition may not be uniquely defined. If, for example, $x \in \mathcal{G}(q, \hat{q}) \cap \mathcal{G}(q, \hat{q}')$ then from state (q, x) it is possible to transition either to mode \hat{q} , or to mode \hat{q}' .
- The continuous state after a discrete transition may not be uniquely defined. If from state (q, x) a discrete transition to mode \hat{q} takes place the continuous set can at the same time change to any value in the set $\mathcal{R}(q, \hat{q}, x)$.

- There may be a choice between continuous evolution and a discrete transition. For example, if we have $x \in \text{Dom}(q) \cap \mathcal{G}(q, q')$, then from state (q, x) it may be possible either to evolve continuously in mode q , or to take a discrete transition to mode q' .

Example 7.2 Thermostat

As an example, consider a room being heated by a radiator controlled by a thermostat. Assume that when the radiator is off the temperature, $x \in \mathbb{R}$, of the room decreases exponentially toward 0 degrees with an uncertain rate of decay

$$\dot{x} \in [ax, bx], \quad (7.1)$$

for some $a < b < 0$. When the thermostat turns the radiator on the temperature increases exponentially towards 30 degrees with an uncertain rate

$$\dot{x} = [a(x - 30), b(x - 30)]. \quad (7.2)$$

Assume that the thermostat is trying to keep the temperature at around 20 degrees. To avoid “chattering” (i.e. switching the radiator on and off all the time) the thermostat does not attempt to turn the heater on until the temperature falls below 19 degrees. Due to some uncertainty in the radiator dynamics, the temperature may fall further, to 18 degrees, before the room starts getting heated. Likewise, the thermostat does not attempt to turn the heater off until the temperature rises above 21 degrees. Due to some uncertainty in the radiator dynamics the temperature may rise further, to 22 degrees, before the room starts to cool down. Finally, assume that initially the room temperature is somewhere in the range $x \in [19, 21]$.

It is easy to write a formal definition of this system in the notation of Definition 7.1:

- $\mathcal{Q} = \{\text{ON}, \text{OFF}\}$;
- $\mathcal{X} = \mathbb{R}$;
- $F(\text{ON}, x) = [a(x - 30), b(x - 30)]$ and $F(\text{OFF}, x) = [ax, bx]$;
- $\text{Init} = \{\text{ON}, \text{OFF}\} \times [19, 21]$;
- $\text{Dom}(\text{ON}) = (-\infty, 22]$ and $\text{Dom}(\text{OFF}) = [18, \infty)$;
- $\mathcal{E} = \{(\text{ON}, \text{OFF}), (\text{OFF}, \text{ON})\}$;
- $\mathcal{G}(\text{ON}, \text{OFF}) = [21, \infty)$ and $\mathcal{G}(\text{OFF}, \text{ON}) = (-\infty, 19]$;
- $\mathcal{R}(\text{ON}, \text{OFF}, x) = \mathcal{R}(\text{OFF}, \text{ON}, x) = \{x\}$.

It is even more convenient to compactly describe the system using the directed graph notation as shown in Fig. 7.1.

The system exhibits several types of nondeterministic uncertainty: in its initial condition, continuous evolution, and choice between continuous and discrete evolution. One can pose several yes/no type questions for this system. For example, it is easy to show that the set $\mathcal{Q} \times [18, 22]$ is an invariant set for the system: all executions of the system always stay in this set [27]. On the other hand, the set $\mathcal{Q} \times [19, 21]$ is not an invariant set: all executions start in it, but, given any arbitrarily small time horizon, there exist executions that leave the set during that horizon. One can see, however, that the set $\mathcal{Q} \times [19, 21]$ is viable: from all initial conditions, there exist executions that stay in the set for ever [27]. Finally, sets of the form $\mathcal{Q} \times [19 + \epsilon, 21 - \epsilon]$ are neither viable nor invariant for any small $\epsilon > 0$.

Notice that the above statements are “all-or-nothing” in spirit. In the nondeterministic framework it is impossible to argue about how likely it is that the state will remain in a set; either it does, or it does not. In the following section we introduce a class of hybrid systems that aims to alleviate this drawback. \square

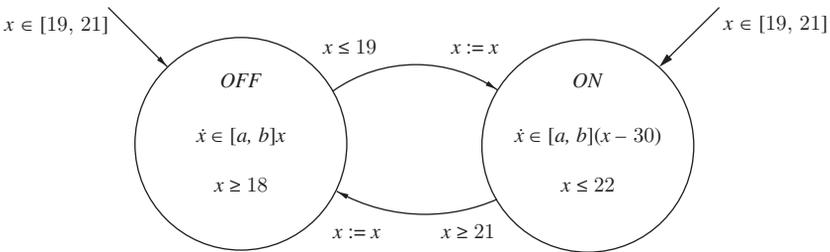


Fig. 7.1 Directed graph notation for the thermostat system.

7.2 Continuous-time stochastic hybrid systems

What the nondeterministic framework outlined in the previous section cannot accommodate is randomness. All the choices listed above are possible under the nondeterministic framework, but there is nothing to distinguish between them. Selecting among these choices leads to different system executions, but there is no implication that some of these executions are more likely than others. As a consequence, all analysis and control problems formulated for nondeterministic systems are of the “yes-or-no,” “worst case” type, which, as we have seen, may be too coarse for some applications.

To show how this difficulty can be alleviated we introduce in this section a class of *stochastic* hybrid systems. By necessity this introduction will be terse and overlook most of the important technical challenges that stochastic hybrid systems pose. For a more thorough treatment, as well as an overview of several application areas of stochastic hybrid methods the interested reader is referred to [86, 145]. We again start with continuous-time stochastic hybrid systems; a brief introduction to discrete-time stochastic hybrid systems is given in Section 7.3.

7.2.1 Stochastic hybrid automata

Given a finite set \mathcal{Q} , $\mathcal{X} = \mathbb{R}^n$ and a set valued map $Dom(\cdot) : \mathcal{Q} \rightarrow 2^{\mathcal{X}}$ assigning to each $q \in \mathcal{Q}$ an open subset of \mathbb{R}^n we define the state space of our stochastic process by

$$S = \bigcup_{q \in \mathcal{Q}} \{q\} \times Dom(q) \subseteq \mathcal{Q} \times \mathcal{X}.$$

Assuming that \mathcal{Q} is endowed with the discrete topology (all subsets are open) one can see that the set S is an open subset of the topological space $\mathcal{Q} \times \mathcal{X}$. Indeed, it is even possible to define a metric (equivalent to the Euclidean metric on $Dom(q)$ for each q) such that S becomes a metric space [203]. We use $\mathcal{B}(S)$ to denote the Borel σ -algebra of S , i.e. the σ -algebra generated by all open subsets of S in this metric topology. Let also \bar{S} denote the closure of the set S in $\mathcal{Q} \times \mathcal{X}$.

Definition 7.4 (Stochastic hybrid automaton) A stochastic hybrid automaton is a collection $H = (\mathcal{Q}, \mathcal{X}, Dom, f, g, Init, \lambda, R)$ where:

- \mathcal{Q} is a finite set of discrete states;
- $\mathcal{X} = \mathbb{R}^n$ is a set of continuous states;
- $Dom(\cdot) : \mathcal{Q} \rightarrow 2^{\mathcal{X}}$ is a set valued map assigning to each $q \in \mathcal{Q}$ an open subset of \mathbb{R}^n . We define $S = \bigcup_{q \in \mathcal{Q}} \{q\} \times Dom(q)$ and endow S with its Borel σ -algebra as above;
- $f : S \rightarrow \mathbb{R}^n$ is a vector field;
- $g : S \rightarrow \mathbb{R}^{n \times m}$ is a diffusion coefficient;
- $Init : \mathcal{B}(S) \rightarrow [0, 1]$ is an initial probability measure on $(S, \mathcal{B}(S))$;
- $\lambda : S \rightarrow \mathbb{R}_+$ is a transition rate function;
- $R : \overline{S} \times \mathcal{B}(S) \rightarrow [0, 1]$ is a transition measure.

Informally, the idea is similar to nondeterministic hybrid systems, with all choices being replaced by probabilistic events. Assume without loss of generality that the system evolution starts at time $t = 0$. We start by extracting an initial state $(q(0), x(0)) = (q_0, x_0)$ according to the probability measure $Init$, i.e.

$$P\{(q_0, x_0) \in A\} = Init(A), \text{ for all } A \in \mathcal{B}(S).$$

We then let the continuous state evolve along a continuous solution of the stochastic differential equation

$$dx(t) = f(q_0, x(t))dt + g(q_0, x(t))dw(t), \tag{7.3}$$

where $w(t)$ denotes the standard m -dimensional Wiener process. While this is going on, the discrete state is assumed to remain constant:

$$q(t) = q_0.$$

The discrete state can only change value when a discrete transition takes place. Discrete transitions can come about in one of two distinct ways:

1. When the continuous state $x(t)$ attempts to leave the set $Dom(q_0)$. Since $x(t)$ is assumed to be continuous as a function of time and the set $Dom(q_0)$ is assumed to be open, such a transition will take place at the time instant

$$T_f = \inf\{t \geq 0 \mid (q_0, x(t)) \notin S\}.$$

This is a well-defined stopping time for the diffusion process (7.3). Discrete transitions of this type are known as *forced transitions*.

2. Spontaneously, in the interior of the set S . The easiest way to think of such transitions is as those of a Poisson arrival process, with a time varying rate $\lambda(q_0, x(t))$ which depends on the value of the continuous state $x(t)$. Discrete transitions of this type are known as *spontaneous transitions*.

In either case, once the discrete transition takes place the state (both continuous and discrete) is reset according to the transition kernel R . In other words, given that a transition (either forced or spontaneous) takes place at time T_1 ,

$$P\{(q(T_1), x(T_1)) \in A \mid q(T_1^-), x(T_1^-)\} = \mathcal{R}(q(T_1^-), x(T_1^-), A), \text{ for all } A \in \mathcal{B}(S),$$

where, as usual,

$$(q(T_1^-), x(T_1^-)) = \lim_{t \nearrow T_1} (q(t), x(t)).$$

Once the state $(q(T_1), x(T_1))$ after the first discrete transition has been selected, the process is repeated to the second transition time T_2 . Notice that the resulting stochastic process is defined over the usual time axis $t \in \mathfrak{R}_+$ and not over the “superdense” time sets of Definition 7.2.

Clearly several assumptions need to be introduced for the above intuitive discussion to make mathematical sense: Lipschitz continuity of the vector fields, measurability of the transition rates and transition kernels, etc. For details, the interested reader is referred to [129, 368].

There are several ways in which the above informal definition of the stochastic process can be formalized. Recall that (under some mild assumptions) forced transitions can be characterized in terms of a well defined stopping time for the diffusion process defined by

$$dx(t) = f(q(t), x(t))dt + g(q(t), x(t))dw(t),$$

namely the exit time from the set $Dom(q(t))$. A trick one can use to account for spontaneous transitions in a similar fashion is to introduce an auxiliary continuous state variable $x_{n+1} \in \mathfrak{R}$ and turn them into exit times as well. To see how this can be done, consider again the first discrete transition of the stochastic process. Consider a random variable z uniformly distributed in $[0, 1]$, set $x_{n+1}(0) = \ln(z)$, and define

$$\frac{dx_{n+1}}{dt}(t) = \lambda(q_0, x(t)).$$

Notice that $x_{n+1}(0) < 0$ and $x_{n+1}(t)$ is nondecreasing as a function of time (since by assumption $\lambda(q, x) \geq 0$ for all $(q, x) \in S$). Let

$$T_s = \inf\{t \geq 0 \mid x_{n+1}(t) \geq 0\}.$$

Under mild assumptions, this is also a well-defined stopping time. Moreover, for each solution $x(t)$ of the stochastic differential equation

$$P\{T_s > t\} = e^{-\int_0^t \lambda(q_0, x(s))ds}$$

as required. The next transition time can then be defined as

$$T_1 = \min\{T_f, T_s\},$$

which is again a well-defined stopping time for the extended system.

More compactly, consider the extended state system with discrete state $q \in \mathcal{Q}$ and continuous state $(x, x_{n+1}) \in \mathbb{R}^{n+1}$. For each $q \in \mathcal{Q}$ define $\hat{Dom}(q) = Dom(q) \times (-\infty, 0)$ (notice that this is an open subset of \mathbb{R}^{n+1} as required); then

$$\hat{S} = S \times (-\infty, 0) \subseteq \mathcal{Q} \times \mathbb{R}^{n+1}.$$

Initialize $x(0)$ according to the measure $Init$ and $x_{n+1}(0) = \ln(z)$ for z uniform in $[0, 1]$. On the extended state space, consider the continuous solution of the stochastic differential equation

$$\begin{pmatrix} dx(t) \\ dx_{n+1}(t) \end{pmatrix} = \begin{pmatrix} f(q(t), x(t)) \\ \lambda(q(t), x(t)) \end{pmatrix} dt + \begin{pmatrix} g(q(t), x(t)) \\ 0 \end{pmatrix} dw(t).$$

When the exit time

$$T_1 = \inf\{t \geq 0 \mid (q(t), x(t), x_{n+1}(t)) \notin \hat{S}\} \quad (7.4)$$

is reached, extract $(q(T_1), x(T_1))$ according to the transition kernel R , re-extract z uniform in $[0, 1]$ independently, set $x_{n+1}(T_1) = \ln(z)$ and repeat the process.

Definition 7.5 (Stochastic hybrid automaton execution) A stochastic process $(q(t), x(t))$ is called an execution of the stochastic hybrid automaton $H = (\mathcal{Q}, \mathcal{X}, Dom, f, g, Init, \lambda, R)$ if there exists a sequence of stopping times $T_0 = 0 < T_1 < T_2 \leq \dots$ and an auxiliary stochastic process $x_{n+1}(t)$ such that:

- (q_0, x_0) is an S -valued random variable extracted according to the probability measure $Init$;
- and, for all $i = 0, 1, \dots$,
- For all $t \in [T_i, T_{i+1})$, $q(t) = q(T_i)$, $x(t)$ is a continuous solution of the stochastic differential equation

$$dx(t) = b(q(T_i), x(t))dt + g(q(T_i), x(t))dw(t) \quad (7.5)$$

starting at $x(T_i)$, and

$$x_{n+1}(t) = x_{n+1}(T_i) + \int_{T_i}^t \lambda(q(T_i), x(s))ds;$$

- $T_{i+1} = \inf\{t \geq T_i \mid (q(t), x(t), x_{n+1}(t)) \notin \hat{S}\}$;
- $x(T_{i+1})$ is distributed according to $\mathcal{R}(q(T_i), x(T_{i+1}^-), \cdot)$ and $e^{x_{n+1}(T_i)}$ is uniform in $[0, 1]$.

All random extractions in the above definition are assumed to be mutually independent. Under relatively mild assumptions, it can be shown that this defines a strong Markov process on the hybrid state space S , whose sample paths are continuous from the right with left limits [129]. Moreover, using methods from stochastic

analysis, existence and uniqueness conditions for this Markov process can be established [368].

7.2.2 Special cases and systems with inputs

The class of stochastic hybrid systems defined above is very general and encompasses a very wide range of stochastic phenomena. Naturally at this level of generality detailed analysis of the properties of a model is difficult. Over the years researchers have been able, however, to establish subclasses of stochastic hybrid systems whose special structure allows the development of sophisticated analysis and control methods. Many of these classes differ in the way that stochasticity enters the process. For example:

- If the stochastic differential equation is replaced by an ordinary differential equation, the resulting stochastic process exhibits deterministic continuous evolution, but has a rich random structure in its discrete transitions. Both spontaneous and forced transitions are possible, as well as a random destination for the discrete and continuous states after the discrete transition. This class of stochastic processes are known as piecewise deterministic Markov processes [203].
- If $Dom(q) = \mathcal{X}$ for all $q \in \mathcal{Q}$ and $\sum_{q' \in \mathcal{Q}} \mathcal{R}(q, x, \{q', x\}) = 1$ for all $(q, x) \in \bar{\mathcal{S}}$, the resulting stochastic process exhibits no forced transitions and is such that the continuous state does not change when discrete transitions take place. This class of processes are known as switching diffusion processes [266] or stochastic differential equations with Markovian switching [438] (often under the further assumption that $\lambda(q, x)$ is constant as a function of x for each $q \in \mathcal{Q}$).

The situation gets even more complicated if one considers inputs in addition to the stochastic terms. Input variables are necessary in many applications to allow for the introduction of control, or nondeterministic (as opposed to stochastic) disturbances, such as an adversary in a stochastic game. Input variables essentially introduce an extra element of choice into the system and require a modeling formalism that can accommodate both stochastic and nondeterministic features. Similar to the stochastic terms, input variables can enter the continuous motion and the timing and destination of discrete transitions. For an indication of how input variables can be introduced see the discrete-time discussion in Section 7.3.

Table 7.1 attempts to summarize the modeling choices made in some of the key references in the literature on continuous-time stochastic hybrid systems. The table does not, of course, tell the whole story. The fact that one modeling framework subsumes another in terms of the phenomena it can capture does not make it more useful in practice, since more powerful modeling formalisms generally lead to models that are more difficult to analyze. For example, even though optimal control methodologies have been developed for the models of [203, 266] and stability results for the models of [438] very little is known about these issues when it comes to the models of [129, 368].

Table 7.1 Overview of stochastic hybrid models. A [203], B [266], C [76, 449], D [438], E [316], F [329], G [129, 368].

Characteristics	A	B	C	D	E	F	G
Stochastic differential equation		√	√	√		√	√
Probabilistic resets	√		√		√	√	√
Spontaneous transitions	√	√		√	√		√
Forced transitions	√		√			√	√
Continuous control	√	√	√				
Transition rate control	√	√					
Forcing transition control	√		√				
Continuous reset control	√		√				

Example 7.3 Thermostat

To illustrate the various elements that enter the definition of stochastic hybrid automata we consider a stochastic variant of the thermostat system defined in Example 7.2. Let again $\mathcal{Q} = \{\text{ON}, \text{OFF}\}$ and $\mathcal{X} = \mathbb{R}$ and as before define $\text{Dom}(\text{ON}) = (-\infty, 22)$ and $\text{Dom}(\text{OFF}) = (18, \infty)$. The state space of the stochastic hybrid automaton then becomes

$$S = \{\text{ON} \times (-\infty, 22)\} \cup \{\text{OFF} \times (18, \infty)\}.$$

Notice that the system will exhibit forced transitions whenever the heater is on and the temperature rises above 22 degrees, or whenever the heater is off and the temperature falls below 18 degrees.

For the continuous evolution consider instead of the differential inclusion, the stochastic differential equations

$$dx(t) = \begin{cases} \frac{a+b}{2}x(t)dt + dw(t) & \text{if } q(t) = \text{OFF}, \\ \frac{a+b}{2}(x(t) - 30)dt + dw(t) & \text{if } q(t) = \text{ON}, \end{cases}$$

where $w(t)$ denotes the standard one-dimensional Wiener process; in other words for $x \in \mathcal{X}$ set $f(\text{ON}, x) = \frac{a+b}{2}x$, $f(\text{OFF}, x) = \frac{a+b}{2}(x - 30)$, and $g(\text{ON}, x) = g(\text{OFF}, x) = 1$.

For the initial condition set $q(0) = \text{ON}$ with probability 0.5 (else set $q(0) = \text{OFF}$) and extract $x(0)$ uniformly on $[19, 21]$; in other words let Init be the uniform measure on $\{\text{ON}, \text{OFF}\} \times [19, 21] \subseteq S$.

For the spontaneous transitions, let

$$\lambda(q, x) = \begin{cases} 0 & \text{if } (q, x) \in \{\text{OFF} \times [19, \infty)\} \cup \{\text{ON} \times (-\infty, 21]\}, \\ 1 & \text{otherwise.} \end{cases}$$

In this way (and keeping in mind the definition of S given above), with probability 1 no transitions will take place when the heater is off and the temperature is above 19 degrees or when the heater is on and the temperature is below 21 degrees. At all other times spontaneous transitions will take place at a fixed rate of 1.

Finally, whenever a discrete transition takes place the continuous state is reset deterministically: the heater switches its discrete state from on to off or vice versa and the continuous state remains the same. In other words, the transition measure $R(\text{ON}, x, \cdot)$ is the Dirac measure concentrated on $(\text{OFF}, x) \in S$ defined as

$$R(\text{ON}, x, A) = \begin{cases} 1 & \text{if } (\text{OFF}, x) \in A \\ 0 & \text{otherwise} \end{cases} \quad \text{for all } A \in \mathcal{B}(S), x \in \mathcal{X}.$$

Likewise

$$R(\text{OFF}, x, A) = \begin{cases} 1 & \text{if } (\text{ON}, x) \in A \\ 0 & \text{otherwise} \end{cases} \quad \text{for all } A \in \mathcal{B}(S), x \in \mathcal{X}.$$

Even though it is easy to infer certain properties of this system by intuition, the formal analysis of the resulting stochastic process is far from trivial. For example, one can see that the set $\mathcal{Q} \times [18, 22]$ is no longer an invariant set for the system. Even over finite horizons there is a nonzero probability that the system execution will exit this set. This is because, due to the uncertainty introduced by the stochastic differential equation, the temperature in the room may fall below 18 degrees even though the heater is on, or rise above 22 degrees even though the heater is off. Indeed one can argue that over an infinite horizon the probability that the state remains in the set $\mathcal{Q} \times [18, 22]$ for ever is zero!

Formalizing such statements is not a trivial matter, however. In general it is not even clear whether questions like “what is the probability that $(q(t), x(t)) \in \mathcal{Q} \times [18, 22]$ for all $t \in [0, T]$?” are mathematically well-posed. To ask this question one needs to ensure that the set of executions that have this property form an event (measurable set) in the underlying probability space. This is by no means obvious, since the definition of this set of executions involves taking an intersection over the uncountable set $[0, T]$. This turns out to be the case for several important classes of stochastic hybrid automata [131], although technical conditions need to be imposed. Moreover, even assuming that the questions are well-posed mathematically, finding the answers is generally very difficult. The reader is referred to the last section of this chapter for a treatment of methods that can be used to address this problem. \square

7.3 Discrete-time stochastic hybrid systems

We conclude this chapter with a brief treatment of stochastic hybrid systems in discrete-time. The main aim of the section is to show that in discrete-time stochastic hybrid systems can be thought of as a special case of discrete-time, Markov chains on general state spaces. Hence the powerful machinery that has been developed for discrete-time Markov chains in terms of tools for stability analysis, optimal control, etc., can be deployed. In the process we will also give an indication of how control variables can enter stochastic hybrid processes. Again this is only one of many possible ways one can think of discrete-time stochastic hybrid systems; comparable modeling frameworks can be found for example in [1, 59].

We define a discrete-time stochastic hybrid automaton (DTSHA for short) as the discrete-time counterpart of the continuous-time stochastic hybrid automaton model described in Section 7.2. The state of a DTSHA again comprises a discrete component taking values in a finite set of modes \mathcal{Q} and a continuous component taking values in $\mathcal{X} = \mathbb{R}^n$. The hybrid state space is thus

$$S = \bigcup_{q \in \mathcal{Q}} \{q\} \times \mathcal{X}.$$

As before, S can be turned into a metric space; let again $\mathcal{B}(S)$ denote the Borel σ -algebra generated by the corresponding open sets. Let also $\mathcal{B}(\mathcal{X})$ denote the Borel σ -algebra of \mathcal{X} with the standard Euclidean topology.

Definition 7.6 (Discrete-time stochastic hybrid automaton (DTSHA))

A discrete-time stochastic hybrid automaton is a collection

$$H = (\mathcal{Q}, \mathcal{X}, U, \Sigma, \text{Init}, T_x, T_q, R),$$

where:

- \mathcal{Q} is a finite set of discrete states;
- $\mathcal{X} = \mathbb{R}^n$ is a set of continuous states. Define $S = \cup_{q \in \mathcal{Q}} \{q\} \times \mathcal{X}$;
- U is a compact Borel space representing the transition control space;
- Σ is a compact Borel space representing the reset control space;
- $\text{Init} : \mathcal{B}(S) \rightarrow [0, 1]$ is an initial probability measure on $(S, \mathcal{B}(S))$;
- $T_x : \mathcal{B}(\mathcal{X}) \times S \times U \rightarrow [0, 1]$ is a Borel measurable stochastic kernel on \mathcal{X} given $S \times U$, which assigns to each $(q, x) \in S$ and $u \in U$ a probability measure $T_x(\cdot | (q, x), u)$ on the Borel space $(\mathcal{X}, \mathcal{B}(\mathcal{X}))$;
- $T_q : \mathcal{Q} \times S \times U \rightarrow [0, 1]$ is a discrete stochastic kernel on \mathcal{Q} given $S \times U$, which assigns to each $(q, x) \in S$ and $u \in U$, a discrete probability distribution $T_q(\cdot | (q, x), u)$ over \mathcal{Q} ;
- $R : \mathcal{B}(\mathcal{X}) \times S \times \Sigma \times \mathcal{Q} \rightarrow [0, 1]$ is a Borel measurable stochastic kernel on \mathcal{X} given $S \times \Sigma \times \mathcal{Q}$, that assigns to each $(q, x) \in S$, $\sigma \in \Sigma$, and $q' \in \mathcal{Q}$, a probability measure $R(\cdot | (q, x), \sigma, q')$ on the Borel space $(\mathcal{X}, \mathcal{B}(\mathcal{X}))$.

As for continuous-time stochastic hybrid automata, the execution semantics of a DTSHA can be algorithmically defined.

Definition 7.7 (Discrete-time stochastic hybrid automaton execution)

A process $\{q(k), x(k)\}_{k=0}^N$ is an execution of a stochastic automaton

$$H = (\mathcal{Q}, \mathcal{X}, U, \Sigma, \text{Init}, T_x, T_q, R)$$

over the horizon $\{0, 1, \dots, N\}$ associated with an input sequence $\{u(k), \sigma(k)\}_{k=0}^{N-1}$ if it takes values in $S = \cup_{q \in \mathcal{Q}} \{q\} \times \mathcal{X}$ and its sample paths are obtained according to Algorithm 7.1.

It is easy to see that by appropriately defining the discrete transition kernel T_q , it is possible to model *spontaneous transitions* that may occur during the continuous state evolution, as well as *forced transitions* that must occur when the continuous state exits some prescribed domain. For the spontaneous transitions, if a discrete transition from q to $q' \neq q$ is enabled at $(q, x) \in S$ by the control input $u \in U$,

Algorithm 7.1 DTSHA execution

Given: Automaton $H = (\mathcal{Q}, \mathcal{X}, U, \Sigma, Init, T_x, T_q, R)$,

- 1: Extract $(q(0), x(0)) \in S$ according to $Init$, and set $k = 0$;
- 2: **while** $k < N$ **do**
- 3: extract $q(k+1) \in \mathcal{Q}$ according to $T_q(\cdot | (q(k), x(k)), u(k))$;
- 4: **if** $q(k+1) = q(k)$ **then**
- 5: extract $x(k+1) \in \mathcal{X}$ according to $T_x(\cdot | (q(k), x(k)), u(k))$;
- 6: **else**
- 7: extract $x(k+1) \in \mathcal{X}$ according to $R(\cdot | (q(k), x(k)), \sigma(k), q(k+1))$;
- 8: **end if**
- 9: **end while**
- 10: increment k .

Result: Automaton execution.

then this can be encoded by the condition $T_q(q' | (q, x), u) > 0$ (cf. $\lambda(q, x) > 0$ for continuous-time stochastic hybrid automata). For the forced transitions, the domain set $Dom(q)$ associated with mode $q \in \mathcal{Q}$ can be expressed in terms of T_q by forcing $T_q(q | (q, x), u)$ to be equal to zero for all $x \notin Dom(q)$, irrespective of the value of the control input $u \in U$. Thus, while the system evolves in mode q , a jump from q to some $q' \neq q$ is forced as soon as $x \notin Dom(q)$.

The above discussion suggests that the definition of DTSHA allows one to capture in discrete-time the same qualitative features as continuous-time stochastic hybrid automata. We now show how DTSHA can be written more compactly as controlled Markov chains. Introduce a stochastic kernel $T : \mathcal{B}(\mathcal{X}) \times S \times U \times \Sigma \times \mathcal{Q} \rightarrow [0, 1]$ on $(\mathcal{X}, \mathcal{B}(\mathcal{X}))$ given $S \times U \times \Sigma \times \mathcal{Q}$ defined by

$$T(\cdot | (q, x), u, \sigma, q') = \begin{cases} T_x(\cdot | (q, x), u), & \text{if } q' = q, \\ R(\cdot | (q, x), \sigma, q'), & \text{if } q' \neq q. \end{cases}$$

T can be used in the DTSHA algorithm to randomly select a value for the continuous state at time $k+1$, given the values taken by the hybrid state and the control input at time k , and that of the discrete state at time $k+1$. Based on T we can introduce the Borel measurable stochastic kernel $P : \mathcal{B}(S) \times S \times U \times \Sigma \rightarrow [0, 1]$ on $(S, \mathcal{B}(S))$ given $S \times U \times \Sigma$ as

$$P((dx', q') | (q, x), (u, \sigma)) = T(dx' | (q, x), u, \sigma, q') T_q(q' | (q, x), u), \quad (7.6)$$

Then, the DTSHA algorithm in Definition 7.7 can be rewritten in a more compact form as Algorithm 7.2.

We have thus shown that a DTSHA can be described as a controlled Markov process with state space $S = \cup_{q \in \mathcal{Q}} \{q\} \times \mathcal{X}$, control space $A = U \times \Sigma$, and controlled transition probability function $P : \mathcal{B}(S) \times S \times A \rightarrow [0, 1]$ defined in (7.6). Hence the extensive literature on discrete-time (controlled) Markov processes that has developed over the years can be utilized to address questions such as stability (see, for example, [450]), optimal control (see, for example, [81]), or reachability [1].

Algorithm 7.2 Simplified DTSHA execution

Given: Automaton $H = (\mathcal{Q}, \mathcal{X}, U, \Sigma, Init, T_x, T_g, R)$.

- 1: Extract $s(0) \in S$ according to $Init$, and set $k = 0$;
- 2: **while** $k < N$ **do**
- 3: extract $s(k+1) \in S$ according to $P(\cdot | s(k), u(k), \sigma(k))$;
- 4: **end while**
- 5: increment k .

Result: Automaton execution.

7.4 Reachability analysis

7.4.1 Reachability problem

In general, reachability analysis deals with the problem of evaluating whether a given system will reach a certain region of the state space during a possibly infinite time horizon, starting from some set of initial conditions. By encoding the correct/safe behavior for a system as a reachability specification, system verification can be reduced to reachability analysis; if the outcome of the analysis is unsatisfactory, some action will be taken to modify the system.

Recall that while reachability for deterministic systems is a problem with a qualitative, yes/no-type answer, for stochastic systems a continuous spectrum of “soft” reachability problems with quantitative answers, such as the hitting probability, the expected hitting time, and the hitting distribution, can be formulated. Posing a reachability problems in quantitative terms makes sense in the verification of those systems for which a misbehavior is quite unlikely to occur, but cannot be ruled out.

Example 7.4 Reachability analysis of stochastic automata

Consider the stochastic finite automaton with state space $\mathcal{V} = \{1, 2, 3\}$ represented by the graph in Fig. 7.2(a). The nodes in the graph are the different states of the automaton. The labels on the oriented arc are the values taken by the transition probability function $p(\cdot | \cdot) : \mathcal{V} \times \mathcal{V} \rightarrow [0, 1]$, which assigns to each $v \in \mathcal{V}$ a probability distribution over \mathcal{V} : $p(\cdot | v)$ with $\sum_{v' \in \mathcal{V}} p(v' | v) = 1$.

The execution of the stochastic finite automaton associated with the initial condition $v_0 \in \mathcal{V}$ is a Markov chain stochastic process $\{v_k, k \geq 0\}$, satisfying $\text{Prob}(v_0 = v_0) = 1$ and $\text{Prob}(v(k+1) = v' | v_k = v) = p(v' | v)$, $v, v' \in \mathcal{V}$, $k \geq 0$. The trajectories of the automaton starting from v_0 are the realizations of this Markov chain.

Now, suppose that $\mathcal{D} = \{3\}$ represents an unsafe set for the system and that the system is initialized at $v_0 = 2$. There are two possible kinds of realizations starting from $v_0 = 2$: (i) 2 followed by all 1's, and (ii) 2 followed by a number of 3's and then all 1's. Only the latter kind leads to the unsafe set \mathcal{D} and the event where this occurs is easily seen to have probability 10^{-3} . Thus, the answer to the qualitative, worst-case, question “Is the system always operating safely?” is “No...,” but the probability that the system eventually enters the unsafe set is actually small. \square

Most of the effort and progress in the study of reachability of hybrid systems concerns deterministic hybrid systems. Reachability analysis of stochastic hybrid

systems is generally much more challenging: analytical solutions are difficult if not impossible to obtain, and few effective general algorithms exist for the numerical simulation of stochastic hybrid systems, especially compared with the many software packages for deterministic hybrid systems.

In this section we discuss reachability analysis for stochastic hybrid systems with reference to the *probabilistic safety analysis* problem of estimating the probability that the state of a given system will enter some unsafe region during a certain time horizon, starting from an arbitrary safe state. This allows to identify the set of initial conditions that guarantee a certain safety level. As a preliminary step, we start by describing an iterative reachability computation algorithm for probabilistic safety analysis of stochastic finite automata. This algorithm is indeed a key ingredient for the numerical solution to the probabilistic safety analysis problem of stochastic hybrid systems.

7.4.2 Probabilistic safety analysis of stochastic finite automata

Consider a stochastic automaton with finite state space \mathcal{V} and transition probability function $p(\cdot|\cdot) : \mathcal{V} \times \mathcal{V} \rightarrow [0, 1]$, and let $\mathcal{D} \subset \mathcal{V}$ be an unsafe set for the system. Our objective is determining the probability that the Markov chain generated as execution of the automaton for the initial condition v_0 will enter \mathcal{D} within the time horizon $[0, k_f]$, for any $v_0 \in \mathcal{V} \setminus \mathcal{D}$.

To this purpose, we modify the automaton by making all the unsafe states in \mathcal{D} absorbing, so that, once the system enters the unsafe set, it stays there forever. The probability of interest can then be written as follows:

$$\mathcal{P}_{v_0} := \text{Prob}(\tilde{v}_k \in \mathcal{D} \text{ for some } 0 \leq k \leq k_f) = \text{Prob}(\tilde{v}_{k_f} \in \mathcal{D}), \quad (7.7)$$

where $\{\tilde{v}_k, k \geq 0\}$ is the Markov chain generated by the modified automaton with state space \mathcal{V} and transition probability function $\tilde{p}(\cdot|\cdot) : \mathcal{V} \times \mathcal{V} \rightarrow [0, 1]$

$$\tilde{p}(v'|v) = \begin{cases} p(v'|v), & v \in \mathcal{V} \setminus \mathcal{D}, \\ 1, & v \in \mathcal{D}, v' = v, \\ 0, & v \in \mathcal{D}, v' \neq v, \end{cases} \quad (7.8)$$

$v' \in \mathcal{V}$, initialized with $v_0 \in \mathcal{V} \setminus \mathcal{D}$.

Example 7.5 Modified automaton for probabilistic safety analysis

Consider the stochastic automaton in Fig. 7.2(a) and the unsafe set $\mathcal{D} = \{3\}$. The modified automaton for probabilistic safety analysis is represented in Fig. 7.2(b). \square

We now describe an algorithm to compute the probability \mathcal{P}_{v_0} in (7.7).

Let $P^{(k)} : \mathcal{V} \rightarrow [0, 1]$ with

$$P^{(k)}(v) := \text{Prob}(\tilde{v}_{k_f} \in \mathcal{D} \mid \tilde{v}_{k_f-k} = v) \quad (7.9)$$

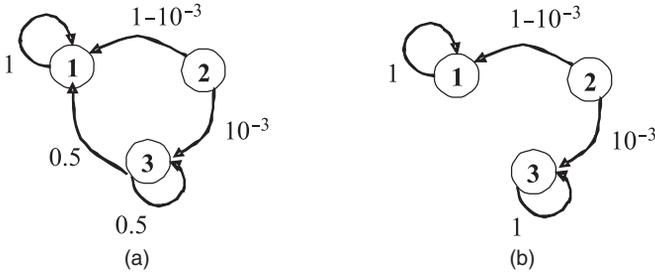


Fig. 7.2 Graphs representing a stochastic finite automaton (a) and the modified stochastic automaton when $\mathcal{D} = \{3\}$ (b).

be a set of probability maps defined on \mathcal{V} , indexed by $k = 0, 1, \dots, k_f$. Note that $\mathbf{P}^{(k)}(v)$ represents the conditional probability that a Markov chain execution of the automaton that takes the value v at time $k_f - k$ will enter \mathcal{D} during the residual time horizon $[k_f - k, k_f]$. Clearly, $\mathbf{P}^{(0)}(v) = 1_{\mathcal{D}}(v)$, $v \in \mathcal{V}$, where $1_{\mathcal{D}}(\cdot)$ is the indicator function of set \mathcal{D} . Moreover, the quantity of interest \mathcal{P}_{v_0} can be expressed as $\mathcal{P}_{v_0} = \mathbf{P}^{(k_f)}(v_0)$.

Fix a time k such that $0 \leq k < k_f$. It is easily seen that the map $\mathbf{P}^{(k)} : \mathcal{V} \rightarrow [0, 1]$ satisfies the following recursive equation

$$\begin{aligned} \mathbf{P}^{(k+1)}(v) &= \text{Prob}(\tilde{\mathbf{v}}_{k_f} \in \mathcal{D} \mid \tilde{\mathbf{v}}_{k_f-(k+1)} = v) \\ &= \sum_{v' \in \mathcal{V}} \text{Prob}(\tilde{\mathbf{v}}_{k_f-k} = v' \mid \tilde{\mathbf{v}}_{k_f-(k+1)} = v) \text{Prob}(\tilde{\mathbf{v}}_{k_f} \in \mathcal{D} \mid \tilde{\mathbf{v}}_{k_f-k} = v') \\ &= \sum_{v' \in \mathcal{V}} \tilde{p}(v'|v) \mathbf{P}^{(k)}(v'), \quad v \in \mathcal{V}. \end{aligned}$$

Recalling the definition of $\tilde{p}(\cdot|\cdot) : \mathcal{V} \times \mathcal{V} \rightarrow [0, 1]$ in (7.8), the fact that any $v \in \mathcal{D}$ is an absorbing state and that, for each $k \in [0, k_f]$, $\mathbf{P}^{(k)}(v) = 1$ if $v \in \mathcal{D}$, we get

$$\mathbf{P}^{(k+1)}(v) = \begin{cases} \sum_{v' \in \mathcal{V}} p(v'|v) \mathbf{P}^{(k)}(v'), & v \in \mathcal{V} \setminus \mathcal{D}, \\ 1, & v \in \mathcal{D}. \end{cases} \quad (7.10)$$

In the finite horizon case ($k_f < \infty$), the probability map $\mathbf{P}^{(k_f)}$ can then be computed by iterating (7.10) k_f times starting from $k = 0$ with $\mathbf{P}^{(0)}(v) = 1_{\mathcal{D}}(v)$, $v \in \mathcal{V}$. The probabilities of reaching the unsafe set \mathcal{D} are propagated backwards in time through the transition probability function starting from the indicator function of \mathcal{D} .

As for the infinite horizon case, by arranging the sequence $\{\mathbf{P}^{(k)}(v), v \in \mathcal{V} \setminus \mathcal{D}\}$ into a column vector $\mathbf{P}^{(k)}$ according to some fixed ordering of the points in $\mathcal{V} \setminus \mathcal{D}$, the iterative equation (7.10) relating $\mathbf{P}^{(k+1)}$ to $\mathbf{P}^{(k)}$ can be rewritten in a more compact matrix form as

$$\mathbf{P}^{(k+1)} = \mathbf{A}\mathbf{P}^{(k)} + \mathbf{b}, \quad (7.11)$$

Algorithm 7.3 Probabilistic safety analysis

Given: the state space \mathcal{V} and the transition probability function $p(\cdot|\cdot) : \mathcal{V} \times \mathcal{V} \rightarrow [0, 1]$; the unsafe set $\mathcal{D} \subset \mathcal{V}$; the time horizon length k_f .

Init: $P^{(0)}(\cdot) \Leftarrow 1_{\mathcal{D}}(\cdot)$; $k \Leftarrow 0$

- 1: **while** $k < k_f$ **do**
- 2: **if** $v \in \mathcal{D}$ **then**
- 3: $P^{(k+1)}(v) \Leftarrow 1$;
- 4: **else**
- 5: $P^{(k+1)}(v) \Leftarrow \sum_{v' \in \mathcal{V}} p(v'|v)P^{(k)}(v')$;
- 6: **end if**
- 7: $k \Leftarrow k + 1$;
- 8: **end while**

Result: $P^{(k_f)}(\cdot)$

where the square matrix A and column vector b are chosen appropriately. Matrix A is a sparse positive matrix with the property that the sum of its elements on each row is smaller than or equal to 1, where equality holds if and only if that row corresponds to a state $v \in \mathcal{V} \setminus \mathcal{D}$ not communicating with \mathcal{D} , which is such that $p(v'|v) = 0$, for all $v' \in \mathcal{D}$. As for b , it is a positive vector with nonzero elements on exactly those rows corresponding to states $v \in \mathcal{V} \setminus \mathcal{D}$ communicating with \mathcal{D} .

Equation (7.11) can be interpreted as the dynamical equation of a discrete-time system with state $P^{(k)}$, dynamical matrix A , and constant input equal to b , evolving over the time horizon $[0, k_f]$ starting from $k = 0$. As a consequence of the asymptotic stability of this system, the solution to (7.11) converges to some (unique) P value, irrespectively of the initialization. The probability map $P^{(k_f)}$ of interest in the infinite horizon case is exactly that P , and can then be determined by solving the fixpoint equation associated with (7.11).

Proposition 7.1 Consider equation

$$P^{(k+1)} = AP^{(k)} + b. \quad (7.12)$$

- (i) There is a unique P satisfying $P = AP + b$.
- (ii) Starting from any initial value $P^{(0)}$ at $k = 0$, the solution $P^{(k)}$ to (7.12) converges to the fixpoint P as $k \rightarrow \infty$.

Example 7.6 Probabilistic safety analysis

We now compute the probability \mathcal{P}_{v_0} that the execution of the stochastic finite automaton in Fig. 7.2(b) associated with $v_0 = 1$ and $v_0 = 2$ eventually ends up in the unsafe set $\mathcal{D} = \{3\}$. To this purpose we introduce the bidimensional column vector $P^{(k)} = [P^{(k)}(1) P^{(k)}(2)]'$, and iterate equation (7.11) starting from $P^{(k)} = [0 \ 0]'$, with

$$A = \begin{pmatrix} p(1|1) & p(2|1) \\ p(1|2) & p(2|2) \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0.999 & 0 \end{pmatrix}, \quad b = \begin{pmatrix} p(3|1) \\ p(3|2) \end{pmatrix} = \begin{pmatrix} 0 \\ 0.001 \end{pmatrix}.$$

We then get: $P^{(1)} = AP^{(0)} + b = b$, $P^{(2)} = AP^{(1)} + b = b = P^{(1)}$, and, hence, $P^{(k+1)} = P^{(k)} = b, \forall k > 0$.

The fixpoint of (7.11) is given by $P = b$ and, in this case, convergence to the fixpoint is obtained in a finite number of iterations (one time step). As is actually easy to see directly from the graph in Fig. 7.2(b), the probability \mathcal{P}_{v_0} of reaching \mathcal{D} starting from $v_0 \in \mathcal{V}$ is independent of the finite horizon length, and equal to $\mathcal{P}_{v_0} = 0$, if $v_0 = 1$, and $\mathcal{P}_{v_0} = 0.001$, if $v_0 = 2$. \square

As a final remark, note that, in both the finite and infinite horizon cases, one can directly refer to the transition probability function of the original stochastic automaton to compute the probability of interest \mathcal{P}_{v_0} . The introduction of the modified automaton is just instrumental to the derivation of the reachability computation procedure.

7.4.3 Probabilistic safety analysis of stochastic hybrid systems

Consider a switching diffusion system whose state $s = (x, q)$ takes values in the hybrid state space $\mathcal{H} = \mathcal{X} \times \mathcal{Q}$, $\mathcal{Q} = \{1, 2, \dots, M\}$, and is governed by the stochastic differential equations

$$\begin{aligned}
 dx(t) &= a(x(t), q(t))dt + b(x(t), q(t)) \Sigma dw(t), \\
 dq(t) &= \int_{\Gamma} r_q(x(t^-), q(t^-), \gamma) \mathbf{p}(dt, d\gamma),
 \end{aligned}
 \tag{7.13}$$

where $w(\cdot)$ is a standard n -dimensional Brownian motion with variance modulated by the diagonal positive definite matrix $\Sigma \in \mathbb{R}^{n \times n}$, and $\mathbf{p}(\cdot, \cdot)$ is a Poisson random measure of intensity $h(dt, d\gamma) = \lambda dt \times \Pi(d\gamma)$, where λ is a positive constant and Π is the uniform distribution over the set $\Gamma = [0, \lambda)$. The Brownian motion and the Poisson random measure are independent.

The Poisson random measure generates two sequences of random variables: the sequence $\{t_i, i \geq 1\}$ of jump times of a standard Poisson process with intensity λ , and the sequence $\{\gamma_i, i \geq 1\}$ of independent random variables distributed according to Π and independent of $\{t_i, i \geq 1\}$. $\mathbf{p}(d\tau, d\gamma)$ assigns unit mass to (τ, γ) if there exists $i \geq 1$ such that $t_i = \tau$ and $\gamma_i = \gamma$. As a result, q solving (7.13) is a pure jump process characterized by piecewise constant and right continuous realizations, given by

$$q(t) = q_0 + \sum_{i \geq 1: t_i \leq t} r_q(x(t_i^-), q(t_i^-), \gamma_i).$$

The function $r_q : \mathcal{X} \times \mathcal{Q} \times \Gamma \rightarrow \{0, \pm 1, \pm 2, \dots, \pm M - 1\}$

$$r_q(x, q, \gamma) = \begin{cases} q' - q, & \text{if } \gamma \in \Gamma_{qq'}(x), \\ 0, & \text{otherwise,} \end{cases} \tag{7.14}$$

determines the entity of the jump in q starting from $(x, q) \in \mathcal{X} \times \mathcal{Q}$, for $\gamma \in \Gamma$. $\Gamma_{qq'}(x) \subseteq \Gamma$ appearing in (7.14) is an interval of length $0 \leq \lambda_{qq'}(x) \leq \lambda$.

For each $x \in \mathcal{X}$, $\{\Gamma_{qq'}(x) : q', q \in \mathcal{Q}, q' \neq q\}$ is a partition of $\Gamma(x) := [0, \sum_{q', q \in \mathcal{Q}, q' \neq q} \lambda_{qq'}(x)] \subseteq \Gamma$.

Since a jump of zero entity occurs at time t_i when $r_q(x(t_i^-), q(t_i^-), \gamma_i) = 0$, then, the actual jump rate is different from the intensity λ of the Poisson process generating the jump times, and depends on the value taken by $(q(t_i^-), x(t_i^-))$. More specifically, the jump rate from $s = (x, q) \in \mathcal{H}$ is given by the jump intensity function $\Lambda : \mathcal{H} \rightarrow [0, +\infty)$

$$\Lambda(s) = \lambda \int_{\{\gamma \in \Gamma : r_q(x, q, \gamma) \neq 0\}} \Pi(d\gamma) = \sum_{q' \in \mathcal{Q}, q' \neq q} \lambda_{qq'}(x) \leq \lambda. \quad (7.15)$$

Also, when a jump occurs from $s = (x, q) \in \mathcal{H}$ such that $\Lambda(s) \neq 0$, the probability distribution of the discrete state q over $\mathcal{Q} \setminus \{q\}$ is given by the reset function $\mathcal{R} : \mathcal{H} \times \mathcal{Q} \rightarrow [0, 1]$:

$$R((x, q), q') = \frac{\int_{\{\gamma \in \Gamma : r_q(x, q, \gamma) = q' - q\}} \Pi(d\gamma)}{\int_{\{\gamma \in \Gamma : r_q(x, q, \gamma) \neq 0\}} \Pi(d\gamma)} = \frac{\lambda_{qq'}(x)}{\Lambda(s)}, \quad q' \neq q. \quad (7.16)$$

During each time interval $[t_i, t_{i+1})$ when the discrete state component q is constant, the continuous state component x evolves according to the first equation in (7.13), initialized with $x(t_i^-) = \lim_{h \rightarrow 0^+} x(t_i - h)$ at time t_i , with local properties determined by the drift and diffusion terms $a(\cdot, q(t_i)) : \mathcal{X} \rightarrow \mathcal{X}$ and $b(\cdot, q(t_i)) : \mathcal{X} \rightarrow \mathbb{R}^{n \times n}$.

Under the assumption that $a(\cdot, q)$, $b(\cdot, q)$, and $\lambda_{qq'}(\cdot)$, $q', q \in \mathcal{Q}$, $q' \neq q$, are bounded and Lipschitz continuous, then the switching diffusion system initialized with $s_0 = (x_0, q_0) \in \mathcal{H}$ admits a unique strong solution $s(t) = (x(t), q(t))$, $t \geq 0$, which satisfies the Markov property and has continuous trajectories of the x component. The stochastic process $s(t)$, $t \geq 0$, is the execution of the switching diffusion system associated with the initial condition s_0 .

Probabilistic safety analysis problem

Given an open safe set $\mathcal{A} \subset \mathcal{H}$, the problem is to estimate the probability that the execution of the switching diffusion system associated with $s_0 \in \mathcal{A}$ will enter the unsafe set $\mathcal{D} := \mathcal{H} \setminus \mathcal{A}$ during the time interval $[0, t_f]$:

$$\mathcal{P}_{s_0} := \text{Prob}(s(t) \in \mathcal{D} \text{ for some } t \in [0, t_f]). \quad (7.17)$$

Based on \mathcal{P}_{s_0} , one can compute the set $S(\varepsilon)$ of initial conditions s_0 for the system that guarantee a safety level $1 - \varepsilon$, $\varepsilon \in (0, 1)$:

$$S(\varepsilon) := \{s_0 \in \mathcal{V} : \mathcal{P}_{s_0} \leq \varepsilon\}.$$

Here, we suppose that the probability (7.17) is well-defined, without discussing the conditions for the event “ $s(t) \in \mathcal{D}$ for some $t \in [0, t_f]$ ” to be measurable.

For ease of explanation, we consider the case when $\mathcal{A} = A \times \mathcal{Q}$. The probability (7.17) can then be rewritten as

$$\mathcal{P}_{s_0} = \text{Prob}(x(t) \in D, \text{ for some } t \in [0, t_f]), \quad (7.18)$$

with $D := \mathcal{X} \setminus A$, and the evolution of x in (7.13) initialized with $s_0 \in \mathcal{A}$ can be confined to the open domain A with stopping on the boundary ∂A of A . We assume that $A \subset \mathcal{X}$ is bounded. If this were not the case, the system evolution would be anyway confined to some bounded region for numerical computation purposes.

The probability (7.18) can be estimated by a numerical approximation scheme that is based on the introduction of a stochastic finite automaton which approximates the switching diffusion system: the reachability probability for the Markov chain execution $\{v_k, k \geq 0\}$ of the automaton converges to the corresponding probability for the execution $\{s(t), t \geq 0\}$ of the switching diffusion system as the gridding parameter δ determining the quality of the approximation tends to zero. We distinguish two components of the Markov chain execution $v_k = (z_k, m_k), k \geq 0$, corresponding to the two components of the switching diffusion execution $s(t) = (x(t), q(t)), t \geq 0$. We now describe how to build the approximating automaton.

Fix a value $\delta > 0$ for the space gridding parameter.

The state space of the approximating stochastic automaton is given by $\mathcal{V}_\delta := \mathcal{Z}_\delta \times \mathcal{Q}$, where \mathcal{Z}_δ is a finite set obtained by gridding A . More precisely, $\mathcal{Z}_\delta = A \cap \mathbb{Z}_\delta^n$, where \mathbb{Z}_δ^n denotes the integer grids of \mathcal{X} scaled according to δ and the positive diagonal entries of matrix $\Sigma = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_n)$ as follows $\mathbb{Z}_\delta^n = \{(m_1\eta_1\delta, m_2\eta_2\delta, \dots, m_n\eta_n\delta) \mid m_i \in \mathbb{Z}, i = 1, \dots, n\}$, with $\eta_i = \sigma_i / \max_k \sigma_k, i = 1, \dots, n$. The interpolation time interval Δ_δ is a positive function of δ which tends to zero faster than δ : $\Delta_\delta = o(\delta)$.

The transition probability function $p_\delta(\cdot|\cdot) : \mathcal{V}_\delta \times \mathcal{V}_\delta \rightarrow [0, 1]$ of the approximating automaton is defined so as to mimic the characteristics of the switching diffusion system over A . To this purpose, observe that the instantaneous behavior of a switching diffusion execution depends on the fact whether a jump in the discrete component q occurs or not. In the former case, the continuous component x remains constant and, if the jump is of nonzero entity, q changes value according to the reset function (7.16). In the latter case, q remains constant and x evolves according to (7.13) with absorption on the boundary of A . To account for this, the transition probability function p_δ is chosen as the weighted sum of the probabilities $p_{z,\delta}(q'|q)$ and $p_{q,\delta}(z'|z)$ respectively governing the transitions of the Markov chain execution $v_k = (z_k, m_k), k \geq 0$, when a jump (possibly of zero entity) occurs in the state component m_k , and when no jump occurs in m_k , from $v = (z, q)$:

$$p_\delta((z', q')|(z, q)) = \begin{cases} (1 - e^{-\lambda\Delta_\delta})p_{z,\delta}(q'|q) + e^{-\lambda\Delta_\delta}p_{q,\delta}(z'|z), & z' = z, q' = q, \\ (1 - e^{-\lambda\Delta_\delta})p_{z,\delta}(q'|q), & z' = z, q' \neq q, \\ e^{-\lambda\Delta_\delta}p_{q,\delta}(z'|z), & z' \neq z, q' = q, \\ 0, & z' \neq z, q' \neq q, \end{cases}$$

for all $(z, q), (z', q') \in \mathcal{Z}_\delta \times \mathcal{Q}$.

The weight $1 - e^{-\lambda\Delta_\delta} = \lambda\Delta_\delta + o(\Delta_\delta)$ represents the probability of a jump (possibly of zero entity) for the m_k component of the approximating automaton execution, which tends as $\delta \rightarrow 0$ to the probability of a single jump (possibly of

zero entity) within a time interval of length Δ_δ for the q component of the original switching diffusion execution.

Fix $v = (z, q) \in \mathcal{V}$. We set

$$p_{z,\delta}(q'|q) = \begin{cases} \frac{\lambda_q q'(z)}{\lambda}, & q' \neq q, \\ 1 - \frac{1}{\lambda} \sum_{q^* \in \mathcal{Q}, q^* \neq q} \lambda_q q^*(z), & q' = q, \end{cases}$$

so that the probability distribution governing jumps of nonzero entity in \mathbf{m}_k from v is equal to the reset function $\mathcal{R}(\cdot, \cdot)$ in (7.16), while, at the same time, the probability of a jump of nonzero entity ($q' \neq q$) from v is given by $\Lambda(v)\Delta_\delta + o(\Delta_\delta)$ where $\Lambda(\cdot)$ is the jump intensity function in (7.15).

To define $p_{q,\delta}(z'|z)$, we introduce the immediate neighbors set $\mathcal{N}_{q,\delta}(z)$ of $z \in \mathbb{Z}_\delta^n$ as an appropriate subset of the grid points in \mathbb{Z}_δ^n whose distance from z along the coordinate axis x_i is at most $\eta_i \delta$, $i = 1, \dots, n$, i.e. $\mathcal{N}_{q,\delta}(z) = \{z + (i_1 \eta_1 \delta, \dots, i_n \eta_n \delta) \in \mathbb{Z}_\delta^n \mid (i_1, \dots, i_n) \in \mathcal{I}\}$, where $\mathcal{I} \subseteq \{0, 1, -1\}^n \setminus \{(0, 0, \dots, 0)\}$. The interior $\mathcal{Z}_{q,\delta}^\circ$ of \mathcal{Z}_δ for the given q consists of all those points in \mathcal{Z}_δ which have all their neighbors in \mathcal{Z}_δ . The boundary $\partial \mathcal{Z}_{q,\delta} = \mathcal{Z}_\delta \setminus \mathcal{Z}_{q,\delta}^\circ$ of \mathcal{Z}_δ is the set of points with at least one neighbor outside A . The probability $p_{q,\delta}(z'|z)$ is such that:

- each state z in $\partial \mathcal{Z}_{q,\delta}$ is an absorbing state;
- only transitions to $\mathcal{N}_{q,\delta}(z)$ are allowed from any state z in $\mathcal{Z}_{q,\delta}^\circ$, according to probabilities determined by its current location:

$$p_{q,\delta}(z'|z) = \begin{cases} \pi_{q,\delta}(z'|z), & z' \in \mathcal{N}_{q,\delta}(z) \cup \{z\} \\ 0, & \text{otherwise} \end{cases} \quad z \in \mathcal{Z}_{q,\delta}^\circ. \quad (7.19)$$

An example is shown in Fig. 7.3 for $n = 2$. This figure refers to the case when the safe set A is ellipsoidal, $\eta_1 = \eta_2 = \eta$, and the immediate neighbors set $\mathcal{N}_{q,\delta}(z)$ is confined to the set of points along the x_1 and x_2 directions whose distance from q is $\eta\delta$. Each grid point has four immediate neighbors, say z_{1-} , z_{1+} , z_{2-} , z_{2+} : two (z_{1-} and z_{1+}) at a distance $\eta\delta$ along direction x_1 , and two (z_{2-} and z_{2+}) at a distance $\eta\delta$ along direction x_2 .

For each $v = (z, q) \in \mathcal{Z}_{q,\delta}^\circ \times \{q\}$, the immediate neighbors set $\mathcal{N}_{q,\delta}(z)$ and the probability distribution $\{\pi_{q,\delta}(z'|z), z' \in \mathcal{N}_{q,\delta}(z) \cup \{z\}\}$ are chosen based on the drift and diffusion terms $a(\cdot, q)$ and $b(\cdot, q)$ in (7.13), so that the evolution of the \mathbf{z}_k component of the Markov chain when no jump occurs in the \mathbf{m}_k component resembles the evolution of the x component of the switching diffusion execution when no jump occurs in q . More specifically, for each $q \in \mathcal{Q}$, the following local consistency property should hold

$$\frac{1}{\Delta_\delta} m_{q,\delta}(z) \rightarrow a(x, q), \quad \frac{1}{\Delta_\delta} V_{q,\delta}(z) \rightarrow b(x, q) \Sigma^2 b(x, q)^\top, \quad \text{as } \delta \rightarrow 0,$$

for all $x \in A$, where, for any δ , z is a point in $\mathcal{Z}_{q,\delta}^\circ$ closest to x , and

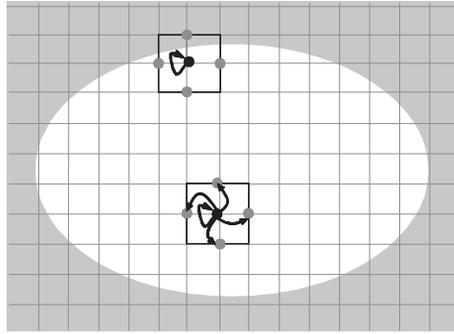


Fig. 7.3 Approximating stochastic automaton: grid points belonging to $\partial Z_{q,\delta}$ and $Z_{q,\delta}^\circ$, with the immediate neighbors set $\mathcal{N}_{q,\delta}(z)$, for an ellipsoidal safe region A .

$$m_{q,\delta}(z) = \sum_{z' \in \mathcal{N}_{q,\delta}(z)} (z' - z) \pi_{q,\delta}(z'|z), \quad V_{q,\delta}(z) = \sum_{z' \in \mathcal{N}_{q,\delta}(z)} (z' - z)(z' - z)^T \pi_{q,\delta}(z'|z)$$

represent the conditional one-step mean and variance of the z_k component of the Markov chain execution when no jump occurs in the m_k component from $v = (z, q)$. Different choices are possible that satisfy the local consistency property.

Example 7.7 A possible choice for local consistency

Suppose that the diffusion matrix $b : \mathcal{X} \times \mathcal{Q} \rightarrow \mathbb{R}^{n \times n}$ in (7.13) has the following form: $b(x, q) = \beta(x, q)I$, where $\beta : \mathcal{X} \times \mathcal{Q} \rightarrow \mathbb{R}$ and I is the identity matrix of size n . The first equation in (7.13) then takes the form: $dx(t) = a(x(t), q(t))dt + \beta(x(t), q(t))\Sigma dw(t)$. Since each component of the n -dimensional Brownian motion $w(\cdot)$ directly affects a single component of $x(\cdot)$, the immediate neighbors set $\mathcal{N}_{q,\delta}(z)$ can be taken as the set of points along each one of the x_i , $i = 1, \dots, n$, directions whose distance from z is $\eta_i \delta$, $i = 1, \dots, n$, respectively. For each z , $\mathcal{N}_{q,\delta}(z)$ is then composed of the following $2n$ elements:

$$\begin{aligned} z_{1+} &= z + (+\eta_1 \delta, 0, \dots, 0), & z_{1-} &= z + (-\eta_1 \delta, 0, \dots, 0), \\ z_{2+} &= z + (0, +\eta_2 \delta, \dots, 0), & z_{2-} &= z + (0, -\eta_2 \delta, \dots, 0), \\ &\vdots & &\vdots \\ z_{n+} &= z + (0, 0, \dots, +\eta_n \delta), & z_{n-} &= z + (0, 0, \dots, -\eta_n \delta). \end{aligned}$$

Figure 7.3 refers to the case when $n = 2$.

As for the probability distribution $\{\pi_{q,\delta}(z'|z), z' \in \mathcal{N}_{q,\delta}(z) \cup \{z\}\}$:

$$\pi_{q,\delta}(z'|z) = \begin{cases} c_q(z) \xi_q^0(z), & z' = z, \\ c_q(z) e^{+\delta \xi_q^i(z)}, & z' = z_{i+}, i = 1, \dots, n, \\ c_q(z) e^{-\delta \xi_q^i(z)}, & z' = z_{i-}, i = 1, \dots, n, \end{cases} \quad (7.20)$$

where

$$\begin{aligned}\xi_q^0(z) &= \frac{2}{\rho\sigma^2\beta(z,q)^2} - 2n, \\ \xi_q^i(z) &= \frac{[a(z,q)]_i}{\eta_i\sigma^2\beta(z,q)^2}, \quad i = 1, \dots, n, \\ c_q(z) &= \left(2 \sum_{i=1}^n \cosh(\delta\xi_q^i(z)) + \xi_q^0(z)\right)^{-1}\end{aligned}$$

with $\sigma := \max_k \sigma_k$. ρ is a positive constant that has to be chosen small enough such that $\xi_q^0(z)$ defined above is positive for all (z, q) . In particular, this is guaranteed if $0 < \rho \leq (n\sigma^2 \max_{s \in \mathcal{A}} \beta(s)^2)^{-1}$. As for Δ_δ , $\Delta_\delta = \rho\delta^2$. \square

Weak approximation of the switching diffusion execution by the (interpolated) Markov chain execution of the automaton can be proved as $\delta \rightarrow 0$. In turn, weak convergence implies convergence with probability one to the probability of interest \mathcal{P}_{s_0} of the corresponding quantity for the approximating Markov chain. Technically speaking, this result requires some regularity conditions to hold in order to avoid situations where the trajectory of the process x touches the absorbing boundary ∂A_q without leaving A_q , for some $q \in \mathcal{Q}$. Such pathological situations are known to be critical also for the discrete-time approximation schemes used in the simulation of stochastic hybrid systems, and are studied in the related literature.

Proposition 7.2 *Consider the stochastic finite automaton approximation of the switching diffusion system corresponding to a certain gridding scale parameter δ . Let $\{\mathbf{v}_k, k \geq 0\}$ be the Markov chain execution of the automaton initialized at a point $v_0 \in \mathcal{V}_\delta^\circ := \cup_{q \in \mathcal{Q}} \mathcal{Z}_{q,\delta}^\circ \times \{q\}$ closest to $s_0 \in \mathcal{A}$, and $k_f := \lfloor t_f/\Delta_\delta \rfloor$ be the largest integer not exceeding t_f/Δ_δ ($k_f = \infty$ if $t_f = \infty$). Then, the probability*

$$\hat{\mathcal{P}}_{s_0} := \text{Prob}(\mathbf{v}_k \in \partial\mathcal{V}_\delta \text{ for some } 0 \leq k \leq k_f), \quad (7.21)$$

where $\partial\mathcal{V}_\delta = \mathcal{V}_\delta \setminus \mathcal{V}_\delta^\circ$, converges to the probability of interest \mathcal{P}_{s_0} in (7.17) with probability one as $\delta \rightarrow 0$. \square

By Proposition 7.2, probabilistic safety analysis of switching diffusion systems is reduced to probabilistic safety analysis of stochastic automata.

The numerical approximation scheme for probabilistic safety analysis over the finite time horizon $[0, t_f]$ of a switching diffusion system consists of the following steps:

1. Fix the gridding scale parameter $\delta > 0$ and the interpolation time interval Δ_δ , and determine the state space \mathcal{V}_δ and the transition probability function $p_\delta(\cdot|\cdot) : \mathcal{V}_\delta \times \mathcal{V}_\delta \rightarrow [0, 1]$ of the approximating stochastic automaton corresponding to δ .
2. Determine the probability map $\mathbf{P}^{(k_f)} : \mathcal{V}_\delta \rightarrow [0, 1]$ by executing Algorithm 7.3 with $\mathcal{V} = \mathcal{V}_\delta$, $p(\cdot|\cdot) = p_\delta(\cdot|\cdot)$, $\mathcal{D} = \partial\mathcal{V}_\delta$, $k_f = \lfloor t_f/\Delta_\delta \rfloor$.
3. Compute an estimate of the probability of interest \mathcal{P}_{s_0} as $\hat{\mathcal{P}}_{s_0} = \mathbf{P}^{(k_f)}(v_0)$, where $v_0 \in \mathcal{V}_\delta^\circ$ is a grid point closest to s_0 , and an estimate of the set of initial conditions $\mathcal{S}(\varepsilon)$ with safety level $1 - \varepsilon$, $\varepsilon \in (0, 1)$, as $\hat{\mathcal{S}}(\varepsilon) = \{s_0 \in \mathcal{A} : \hat{\mathcal{P}}_{s_0} \leq \varepsilon\}$.

In the model checking approach to safety analysis of deterministic hybrid systems, safety is verified by constructing backward reachable sets starting from the unsafe set. Here, probabilities rather than sets are propagated backwards through the system dynamics. For each safe state $s_0 \in \mathcal{A}$ an estimate of the probability \mathcal{P}_{s_0} that the switching diffusion system will enter the unsafe set $\mathcal{H} \setminus \mathcal{A}$ starting from s_0 is computed by appropriately propagating the transition probabilities of the approximating automaton backwards in time starting from the unsafe set during the time horizon of interest, according to Algorithm 7.3 in Section 7.4.2. The issue of representation and propagation of probabilities is solved by approximating the stochastic hybrid system through a stochastic automaton. Similarly, abstraction to deterministic automata is adopted for reachability computations of deterministic hybrid systems.

Barrier certificate method An interesting alternative approach to address probabilistic safety analysis of switching diffusion systems consists in computing an upper bound on the probability \mathcal{P}_{s_0} by means of the introduction of an appropriate function called *barrier certificate*. A set of initial conditions has to be specified, and the certified upper bound on \mathcal{P}_{s_0} is valid for s_0 belonging to that initial set.

A main advantage of this approach is that one does not have to compute and propagate probabilities through the system dynamics, which eliminates the need to approximate the system by a stochastic automaton and, consequently, the scalability issue caused by gridding. However, a computational procedure to construct a barrier certificate is currently available only for the case when the switching diffusion system is polynomial and all the sets involved are described by polynomial inequalities. As the complexity of the (polynomial) barrier certificate is increased, the upper bound gets tighter.

7.4.4 Extensions

Uncertain initial state and time-varying safe set If the initial state $s(0)$ is uncertain and independent of the other random quantities involved in the system evolution, the probability of entering the unsafe region can be computed by integrating numerically the estimate $\hat{\mathcal{P}}_{s_0}$ with respect to the initial state probability distribution.

If the safe set is time-varying during the finite time horizon of interest, the system dynamics should be confined to the union of the sets obtained as time varies along the time horizon, and Algorithm 7.3 should be simply adapted by considering at each iteration the corresponding set and initializing the recursion with the appropriate indicator function.

Control problems Besides safety analysis, the described approach is actually well-suited for assessing the efficacy of a control system whose objective is steering the state of the system close to some operating condition. In this case, the “safe” set should be constructed as a time-varying region shrinking to a small neighborhood around the desired operating condition and accounting for the admissible deviations from that condition in the transient.

Further classes of stochastic hybrid systems The described approach to reachability analysis can be extended to the class of switching diffusion systems with hybrid jumps. The modeling power of switching diffusions systems, however, is limited by the fact that only spontaneous transitions (transitions that occur according to a probabilistic rate) are allowed. In a quite recent study, general stochastic hybrid systems (GSHS) models allowing also forced transitions (transitions that are driven by a boundary hitting times) are approximated by switching diffusion systems, replacing forced transitions with spontaneous transitions characterized by properly defined state-dependent transition intensities. As a result, the described approach to reachability computations for switching diffusions appears to be useful for more general classes of continuous-time stochastic hybrid systems. Interestingly, the iterative procedure for probabilistic safety analysis in Algorithm 7.3 remains valid also for discrete-time stochastic hybrid systems, and for this class of systems has been extended to address the case when a control input is available and can be chosen so as to maximize the safety level. This is obtained by applying dynamic programming to an optimal stochastic control reformulation of the problem.

Bibliographical notes

The described scheme for reachability analysis was first introduced in [328], and further developed in [537], for systems described by stochastic differential equations with coefficients changing value at a-priori known time instants. The methodology was extended in [538] to the stochastic hybrid setting. The theoretical issues regarding the measurability of the reachability event are addressed in [131]. The conditions for weak convergence of the Markov chain approximation to imply convergence with probability one of the probability of entering the unsafe set are studied in, e.g., [367] and [372]. Approximation of general stochastic hybrid systems by switching diffusion systems is discussed in [2]. Model checkers for verifying probabilistic reachability specifications of stochastic automata are described in [354].

In [130], an upper bound on the probability of reachability events is derived theoretically based on the theory of Dirichlet forms associated with a right-Markov process. In [536], barrier certificates are used to derive an upper bound. Differently from [130], a computational technique is provided, though only for a specific class of stochastic hybrid systems.



A “mean-square” notion of reachability is introduced in [213]. Motivated by air traffic control applications, in [425, 539] randomized algorithms are used for estimating the probability of entering some unsafe set starting from some given initial condition. In [1] reachability is studied for a quite general class of discrete-time stochastic hybrid systems whose dynamics can be influenced by a control input. A methodology to compute and minimize the probability that the system enters an unsafe region is developed.

controlengineers.ir

Part II

controlengineers.ir Tools

controlengineers.ir

Overview of tools development and open problems

D. A. van Beek and S. Engell

For specific classes of hybrid formalisms, powerful algorithms and tools for analysis and synthesis have been developed in the recent years. This chapter gives a brief overview of the functionality of tools for control design, verification, simulation, optimization, and model transformation. The tools are presented in more detail in the next chapters

Chapter contents

8.1	Control of switched linear systems	page 280
8.2	Verification of linear hybrid automata	281
8.3	Modeling, simulation, and optimization of general hybrid systems	281
8.4	Model transformation	282
8.5	Concluding remarks and open issues	283

In the last decade, many tools for the design and control of hybrid systems have been developed. This chapter gives an overview of some important classes of such tools. The classes are ordered according to the expressivity of the underlying mathematical model. In the end, the interconnection of tools by means of a compositional interchange format is sketched and open issues are discussed. The tools are discussed in more detail in [Chapter 9–12](#).

8.1 Control of switched linear systems

The *MATLAB/Simulink*-based tools discussed in [Chapter 10](#) are based on (deterministic) discrete-time piecewise affine (PWA) mathematical models of hybrid systems. These tools offer the following support for the design and analysis of control systems:

- *Modeling*: Two modeling languages with associated tools are presented: *HYSDEL* and *MLD*. *HYSDEL* models are based on interconnections of linear dynamical systems that are specified by means of finite-state discrete-time automata, *IF-THEN-ELSE* rules, and propositional logic statements. *HYSDEL* models can be transformed into *MLD* (mixed logical dynamical) models, which in turn can be transformed into discrete-time PWA models.
- *Identification*: For inferring PWA models based on data, the hybrid identification toolbox (*HIT*) and the piecewise affine system identification toolbox (*PWAID*) are available.
- *Control design*: For control design, two toolboxes are available: the *multi-parametric toolbox* and the *hybrid toolbox*. The *multi-parametric toolbox* can be viewed as a unifying repository of hybrid systems design tools utilizing optimization packages. It allows users to design optimization-based controllers. The *hybrid toolbox* supports the design of model-predictive controllers (MPC) based on on-line mixed-integer linear or quadratic programming (MILP/MIQP) for hybrid systems in *MLD* form.
- *Simulation and verification*: Both toolboxes support simulation, verification, and code generation for the developed control systems. Verification in this context is different from the verification of linear hybrid automata as discussed in [Section 8.2](#). Since PWA models are deterministic, reachability properties of a model with given initial values, input functions, and horizon length can be verified by executing one simulation run. Verification in this context means to assess the validity of a property for a given (finite) set of initial conditions and/or input functions.

For control system design based on continuous-time PWAs, see the tool *ConPAHS* [[184](#)].

8.2 Verification of linear hybrid automata

The tools *PHAVer* and *HyTech* that are discussed in Chapter 9 are tools based on (nondeterministic) linear hybrid automata (LHA) models. They support modeling and verification of hybrid systems:

- Modeling*: The tools support modeling of networks of linear hybrid automata. Networks are defined as a parallel composition of automata, based on the synchronous execution of shared action labels and the interleaving execution of non-shared action labels. In this context, linear refers to the specification of the dynamics by bounds on linear combinations of derivatives, e.g. $\dot{x} + 2\dot{y} \leq 1$.

Hybrid automata with non-LHA dynamics can be over-approximated arbitrarily close by LHA using phase-portrait approximation. This, however, may lead to a large number of discrete states.
- Verification*: Linear hybrid automata are in general nondeterministic. The specification by linear constraints over derivatives in principle admits infinitely many solutions for the continuous dynamics (time-dependent behavior). The linear constraints over the values of the variables before and after a jump may admit infinitely many solutions for the discrete dynamics (action-dependent behavior). The verification of the properties of LHA is based on reachability analysis for a given set of initial states.

8.3 Modeling, simulation, and optimization of general hybrid systems

Traditionally, the dominant industrial tool for computer-based system analysis and design has been simulation. It still is the dominant tool of choice since complex and large hybrid systems with possibly nonlinear continuous dynamics often cannot be analyzed in a rigorous fashion. In comparison to the tool support for hybrid optimal control or for the verification of hybrid systems, the tool base for simulation is rather large and mature, mostly because hybrid simulation is comparatively straightforward and can directly be applied to many systems of practically relevant size. Requirements that are posed on hybrid simulation tools range from the support of hierarchical and reusable models to intuitive user interfaces and facilities for tool integration.

Dynamic optimization tools are often integrated into simulation tools. There are two reasons for this. First, simulation forms the basis of many dynamic optimization techniques. Second, the most time-consuming task in the formulation of a dynamical optimization problem is the definition of the dynamic simulation model.

Since dynamical optimization of general hybrid systems is considerably more demanding than simulation from both the theoretical and the computational point of view, the tool support for dynamical optimization is considerably smaller and less mature. However, due to significant improvements of optimization techniques,

several tools are now available that are capable of optimizing large and complex systems.

Chapter 11 gives a comprehensive overview of the challenges and concepts of hybrid systems simulation and optimization and describes a large variety of tools that have been developed for this purpose. After the introduction of the major modeling approaches for hybrid simulation and optimization, many of the often intricate issues that arise in the joint execution of the possibly very complex continuous dynamics and the discrete dynamics are discussed in detail.

Over twenty different hybrid simulation tools and environments are discussed and compared. The presented tools cover the complete spectrum of hybrid systems simulation and range from rather prototypical academic tools that mostly serve as test beds for hybrid systems research, to integrated modeling and simulation environments that are capable of real-time simulation of highly complex models with hundreds of thousands of equations on modern computing hardware. The major modeling concepts are illustrated by two commercial tools. The de-facto industrial standard tool set *MATLAB/Simulink* with the toolboxes *Stateflow* and *SimEvents* is used to clarify the block-diagram-based modeling approach, which is a form of causal modeling, and the modeling and simulation environment *gPROMS* is used to illustrate equation based (acausal) modeling.

8.4 Model transformation

The current state of technology with respect to tool support for the integrated design of complex controlled systems is characterized by *islands of support*. This means that the tools mentioned above can be used to analyze properties of parts of the overall system on some level of detail (and accuracy) of the models, but abstraction and refinement of models, as well as the interconnection of heterogeneous models, are not supported and require manual transformations. One step towards tool-based support of the design of larger and more complex systems is the precise definition and implementation of interchange formats, and in particular the compositional interchange format for hybrid systems (*CIF*), as described in Chapter 12. The purpose of an interchange format is to establish the interoperability of a wide range of tools by means of model transformations. In this way, the implementation of many bilateral translators between specific formalisms can be avoided. The compositional interchange format has the following characteristics:

- *Modeling*: The basic element of the *CIF* is an interchange automaton with an operator for parallel composition. Parallel composition provides synchronization by means of shared action labels, and communication by means of shared variables and CSP channels. Different forms of urgency are available, so that that execution of actions can have priority over passage of time. Furthermore, the concepts of inputs and outputs are defined, and scoping of variables, channels, and actions is available to allow modular and hierarchical specifications.

- *Semantics*: The semantics of the *CIF* is compositional, allowing property preserving model transformations. Parts of a model can be replaced by equivalent parts without changing the meaning of the model.
- *Transformations*: Transformations between the *CIF* and several other languages (*UPPAAL*, *PHAVer*, *Chi*, *MATLAB/Simulink*, *gPROMS*, *Modelica*, and *Sequential Function Charts*) will be defined in the European Multiform project (www.ict-multiform.eu).

8.5 Concluding remarks and open issues

Tool support for the analysis and synthesis of hybrid systems has improved considerably in the last decade. Systems of moderate complexity in terms of the number of continuous state variables can be rigorously analyzed and large, complex hybrid systems can be simulated faithfully in reasonable computing times. Dynamic optimization is feasible even for nonlinear DAE systems as long as the number of discrete variables is moderate and no autonomous switches occur.

More development is required, inter alia, in the following areas:

- Compositional analysis of systems that consist of several subsystems for which properties can be proven rigorously.
- Over-approximation of fine-grained models by more abstract ones, e.g. by timed automata.
- Rigorous, global analysis of systems with nonlinear dynamics and of systems described by differential-algebraic equations.
- Optimization of systems with discrete inputs and autonomous switches over long horizons.
- Robust analysis and optimization, taking into account the fact that models are only approximations of the behaviors of real systems and that safety and optimality are demanded for the real system, not for the model.

controlengineers.ir

Verification tools for linear hybrid automata

G. Frehse

Linear hybrid automata are of interest in verification because sets of successor states can be computed exactly and efficiently using integer arithmetic. This fact is exploited in a number of formal verification algorithms that upon termination give a mathematically sound answer on whether a property holds or not. This chapter gives an overview of verification algorithms for safety properties of linear hybrid automata, and some of the over-approximation techniques that help to make the tools applicable in practice.

Chapter contents

9.1	Formal verification and linear hybrid automata	page 286
9.2	Modeling systems using linear hybrid automata	287
9.3	Run semantics and path constraints	288
9.4	Reachability analysis	289
9.4.1	Polyhedral computations	290
9.4.2	Over-approximating polyhedra	291
9.5	Over-approximation of affine dynamics	294

9.1 Formal verification and linear hybrid automata

Formal verification tries to establish with absolute certainty, in the sense of being mathematically provable, whether or not a given property is satisfied or violated by a system. In this chapter we consider algorithmic approaches to verification of safety properties such as whether a set of forbidden states is reachable. If a verification algorithm is *exact* (no over-approximations, no floating point computations, no rounding of any kind, etc.) and terminates, it gives the provably correct answer: safe or unsafe. Exact verification algorithms are of limited use for hybrid systems in general, since the solution of a differential equation can only be computed up to some accuracy. One may have to resort to an *over-approximative* algorithm, i.e. use over-approximations in the sense that when the algorithm terminates with result *safe*, the system really is safe. If the algorithm finds a violation, one might not know whether the violation is real or an artifact of over-approximations. Nondeterministic models and set-based computations can be used when inputs, initial conditions or model parameters are only known up to a certain range.

Linear hybrid automata (LHA) [309] are characterized by piecewise constant bounds on the derivatives. They are of interest in formal verification because their dynamics are so simple that basic operators such as discrete and continuous successor states can be computed with exact integer arithmetic, and relatively efficiently over an infinite time horizon. This has led to tools such as *HyTech* [310] and *PHAVer* [250], which use exact polyhedral computations to obtain the set of reachable states, synthesize controllers [669], and verify equivalence and abstraction between automata using assume/guarantee reasoning [252]. Other verification approaches for LHA try to avoid polyhedral computations altogether. They exploit the fact that for LHA reachability along a given path can be formulated as satisfiability of a conjunction of linear constraints, and can therefore be computed very efficiently using linear programming techniques. This approach has been taken in counter example guided abstraction refinement [369] and to synthesise parameters [253]. With a simple model transformation (introducing a variable for the time that elapses in each location), a LHA can be modeled and verified by any model checker able to handle linear constraints over the rationals. This approach has been used to verify LHA using abstraction refinement with the tool ARMC [528].

From a theoretical point of view, exact algorithms for LHA suffer from the fact that even basic properties such as reachability are in general undecidable, so termination can not be guaranteed. In practice, slow convergence and computation costs that increase explosively with the size of the system are limiting exact approaches to relatively small problems. Over-approximation techniques such as those used in *PHAVer* have pushed the envelope, but it remains a challenge to find over-approximations for which reachability algorithms terminate in acceptable time and are yet accurate enough to show the safety of the system.

9.2 Modeling systems using linear hybrid automata

Hybrid automata are modeling formalism in which the state consists of a discrete component, called *location*, and a set of continuous variables, whose evolution with time is governed by differential equations (or inclusions) associated with each location. The system can jump instantaneously from one location to another according to a set of transitions, which can also modify the continuous variables (for details, see Section 3.1).

The class of *Linear hybrid automata* (LHA) is syntactically defined: the continuous dynamics are given by *flow constraints*, which are conjunctions of (strict or nonstrict) linear constraints over the derivative \dot{x} ,

$$\mathbf{a}^T \dot{x} \leq b, \quad \mathbf{a} \in \mathbb{Z}^n, b \in \mathbb{Z}, \quad (9.1)$$

(n being the number of continuous variables of the automaton); the discrete dynamics (jumps) are given by conjunctions of linear (strict or nonstrict) constraints over the values x before the jump, and the values x' after,

$$\mathbf{a}^T x + \mathbf{a}'^T x' \leq b, \quad \mathbf{a}, \mathbf{a}' \in \mathbb{Z}^n, b \in \mathbb{Z}. \quad (9.2)$$

The set of pairs (x, x') that satisfy these constraints is called the *jump relation* μ . Invariants and initial states are given for each location by conjunctions of linear constraints over the variables.

Various nondeterministic types of dynamics can be expressed in the above form, e.g. differential inclusions such as $\dot{x} \in [\text{const}_1, \text{const}_2]$, and conservation laws such as $\dot{x}_1 + \dot{x}_2 = 0$. If a transition is defined in terms of a guard condition \mathcal{G} and a reset map \mathcal{R} , its jump relation is

$$\mu = \{(x, x') \mid x \in \mathcal{G} \wedge x' = \mathcal{R}(x)\}. \quad (9.3)$$

The discrete states of a hybrid automaton can be used to model abrupt changes such as a collision, but they are also useful in building abstractions. In the following example, we use discrete states to abstract some complex dynamics, which we know little about, with very simple LHA dynamics. By using differential inclusions with conservative bounds (similar to interval analysis), we can assure that the actual, complex behavior is contained in the range of behaviors of our nondeterministic model.

Example 9.1 Model of Lake Mead

The water level of the Lake Mead/Lake Powell system is decreasing every year, and it could completely dry up with the next two decades [42]. We take a (grossly simplified) look at the problem and compare a worst-case estimate based on constant rates with an estimate based on a hybrid model with switching rates. Lake Mead is fed by the Colorado River at an average rate of $r_{\text{in}} = 15$ million acre feet per year (maf/yr), a figure which is expected to decrease by up to 30% over a timespan of $t_{\text{dec}} = 50$ years. Water is taken out of the lake at a rate of 14 maf/yr, plus 1.7 maf/yr due to evaporation and infiltration, resulting in a constant

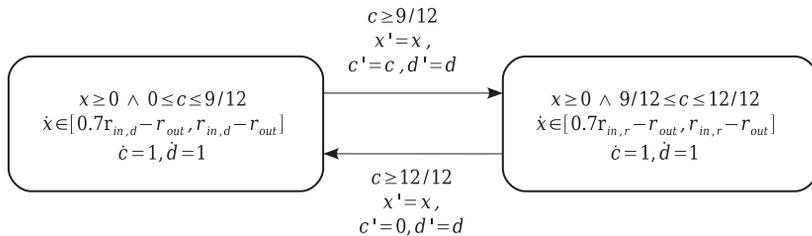


Fig. 9.1 Linear hybrid automata model of the lake level.

loss of $r_{out} = 15.7 \text{ maf/yr}$. Let x be the water level. As of June 2007, the reservoir contains $x_{ini} = 25.7 \text{ maf}$. For an average-rate estimate we assume a fill rate reduced to 70% and obtain a net rate of $\dot{x} = 0.7r_{in} - r_{out} = -5.2 \text{ maf/yr}$, which means that Lake Mead could be dry in $(0 - x_{ini})/\dot{x} = 4.94$ years. However, the feed from the Colorado River is varying strongly with the seasons. In the rainy season, April to June, $r_{in,r} = 30 \text{ maf/yr}$, while in the rest of the year, dry season, $r_{in,d} = 10 \text{ maf/yr}$. This variation in feed causes a corresponding seasonal variation in Lake Mead’s water level, so Lake Mead might dry up sooner than the average-rate estimate suggests. Assuming as worst case the constant rate of the dry season, i.e. $\dot{x} = 0.7r_{in,d} - r_{out}$, yields a very conservative estimate of 2.95 years.

To obtain a refined estimate, we use the hybrid automaton model in Fig. 9.1, which includes the seasonal switch in dynamics. The dry and rainy season are each represented by a location. In each location the dynamics are given by a differential inclusion, which models the net rate to be in the interval between today’s rate and the rate with inflow reduced to 70%.

Since the change of season depends on the time of year, we introduce a *clock variable* c , i.e. a variable with $\dot{c} = 1/\text{yr}$. Another clock d is used to measure the time that elapses until x reaches zero. The change from dry to rainy season is modeled by a transition whose guard $c \geq 9/12$ means that the change is *possible* after 9 months. The system can also remain in location *dry* as long as its invariant is satisfied. The invariant being $c \leq 9/12$, guard and invariant overlap only on $c = 9/12$. Since $\dot{c} \neq 0$, c takes this value only at a single instant in time, and the system transitions immediately from *dry* to *rainy* when the clock reaches this boundary. The values of c and x do not change during the transition, so its reset map is simply the identity map. The change from rainy back to dry season is modeled with a similar transition, except that it resets the clock c to zero in order to restart the cycle when a full year has elapsed. □

9.3 Run semantics and path constraints

We formally define the behavior of a LHA in terms of *runs*, and show how the existence of a run can be reduced to satisfiability of a conjunction of linear constraints. This satisfiability can be checked by linear programming, for which very powerful standard methods and tools are available.

A *run* is an alternating series of continuous trajectories and jumps of the automaton, i.e. a (finite or infinite) sequence

$$(q_0, \mathbf{x}_0) \xrightarrow{\delta_0, \xi_0(t)} (q_0, \mathbf{x}'_0) \xrightarrow{\alpha_0} (q_1, \mathbf{x}_1) \xrightarrow{\delta_1, \xi_1(t)} (q_1, \mathbf{x}'_1) \cdots,$$

where first state (q_0, \mathbf{x}_0) is an initial state of the automaton, δ_i is a real value specifying the time spent in location q_i , $\xi_i(t)$ is the trajectory of x for $t \in [0, \delta_i]$, and α_i is the label of the transition taken. During the timespan $[0, \delta_i]$, the trajectory $\xi_i(t)$ with $\xi_i(0) = \mathbf{x}_i$, $\xi_i(\delta_i) = \mathbf{x}'_i$ must be in the invariant of q_i and $\frac{d}{dt}\xi_i(t)$ must satisfy the flow constraints of q_i . For each jump, $(\mathbf{x}'_i, \mathbf{x}_{i+1})$ must be in the jump relation μ of a corresponding transition, and \mathbf{x}_{i+1} must be in the invariant of q_{i+1} . A state (q, \mathbf{x}) is *reachable* if there exists a run with $(q_i, \mathbf{x}_i) = (q, \mathbf{x})$ for some i .

As we will see in the following, these conditions can be simplified to a set of linear constraints [399]. While the definition of a run admits trajectories that can be arbitrarily curved within the bounds imposed by the flow constraints and the invariant, the straight line from \mathbf{x}_i to \mathbf{x}'_i also satisfies these constraints [15]. As a consequence, the question whether

$$(q_0, \mathbf{x}_0) \xrightarrow{\delta_0, \xi_0} (q_0, \mathbf{x}'_0) \xrightarrow{\alpha_0} (q_1, \mathbf{x}_1)$$

is a run reduces to finding an \mathbf{x}'_0 such that $(\mathbf{x}'_0 - \mathbf{x}_0)/\delta_0 \in \text{Flow}$, $\mathbf{x}'_0 \in \text{Inv}(q_0)$, $(\mathbf{x}'_0, \mathbf{x}_1) \in \mu_0$, and $\mathbf{x}_1 \in \text{Inv}(q_1)$. Since $\text{Inv}(q_0)$, $\text{Inv}(q_1)$, and μ_0 are given by linear constraints, the latter three expressions correspond to checking satisfiability of a linear program. To achieve the same with the first expression, it is reformulated as follows.

Recall that $\text{Flow}(q)$ is given by a conjunction of linear constraints, which we may write in matrix form as

$$\mathbf{A}_q^{\text{flow}} \dot{\mathbf{x}} \leq \mathbf{b}_q^{\text{flow}}.$$

Because we need only consider straight lines as trajectories, we may substitute $\dot{\mathbf{x}} = (\mathbf{x}'_0 - \mathbf{x}_0)/\delta_0$. Multiplying the result with δ_0 we get

$$\mathbf{A}_q^{\text{flow}} \mathbf{x}'_0 - \mathbf{A}_q^{\text{flow}} \mathbf{x}_0 - \mathbf{b}_q^{\text{flow}} \leq \mathbf{0}, \quad (9.4)$$

which is a linear constraint over the variables \mathbf{x}_0 , \mathbf{x}'_0 , and δ_0 . The argument can be extended to runs of arbitrary length, resulting in a conjunction of linear constraints over the variables \mathbf{x}_i , \mathbf{x}'_i , and δ_i , with i ranging over the length of the run. These constraints are called *path constraints*. For a given sequence of locations and transitions, deciding whether there exists a corresponding run from an initial to a final set of states can therefore be reduced to checking satisfiability of the corresponding path constraints.

9.4 Reachability analysis

We describe the basic operators for computing the set of reachable states using polyhedra, and present over-approximation techniques that ensure termination and help reduce the computational complexity.

9.4.1 Polyhedral computations

A polyhedron \mathcal{P} can be described in matrix form as

$$\mathcal{P} = \{x \mid Ax \leq b\},$$

which is referred to as the *constraint representation* of the polyhedron. An alternative form is the *generator representation*, which describes the polyhedron as the convex hull of a finite set $\mathcal{V} \subseteq \mathbb{R}^n$ of *vertices* and a finite set $\mathcal{S} \subseteq \mathbb{R}^n$ of *rays*, i.e.

$$\mathcal{P} = \left\{ \sum_{\mathbf{v}_i \in \mathcal{V}} \lambda_i \mathbf{v}_i + \sum_{\mathbf{r}_i \in \mathcal{R}} \mu_i \mathbf{r}_i \mid \lambda_i, \mu_i \geq 0, \sum_i \lambda_i = 1 \right\}. \quad (9.5)$$

A matrix representation is advantageous for operations such as intersection, while the generator representation is more suitable for checking emptiness, convex hull, and time elapse. The conversion from one representation to the other has a complexity exponential in n .

Recall from Section 3.3 that the set of reachable states $\text{Reach}(\text{Init})$ can be computed by recursively adding the successor states of time elapse, Succ_C , and of discrete transitions, Succ_D . Formally, $\text{Reach}(\text{Init})$ is the smallest fixed-point of the equation

$$\mathcal{R} = \mathcal{R} \cup \text{Succ}_C(\mathcal{R}) \cup \text{Succ}_D(\mathcal{R}), \quad \text{Init} \subseteq \mathcal{R}.$$

In the following we write the invariant of a location q as $A_q^{\text{Inv}} x \leq b_q^{\text{Inv}}$, and the jump relation for the transition (q, q') in matrix form as $A_{q,q'} x + A'_{q,q'} x' \leq b_{q,q'}$. Using the results of Section 9.3, the successor operators are

$$\begin{aligned} \text{Succ}_C(\mathcal{R}) = \{ & (q, x') \mid \exists (q, x) \in \mathcal{R}, \delta \in \mathbb{R}^{\geq 0} : \\ & A_q^{\text{flow}} x' - A_q^{\text{flow}} x - b_q^{\text{flow}} \delta \leq \mathbf{0} \wedge A_q^{\text{Inv}} x' \leq b_q^{\text{Inv}} \}, \end{aligned} \quad (9.6)$$

$$\begin{aligned} \text{Succ}_D(\mathcal{R}) = \{ & (q', x') \mid \exists (q, x) \in \mathcal{R} : \\ & (q, q') \in \mathcal{E}, A_{q,q'} x + A'_{q,q'} x' \leq b_{q,q'} \wedge A_q^{\text{Inv}} x' \leq b_q^{\text{Inv}} \}. \end{aligned} \quad (9.7)$$

Polyhedra are closed with respect to conjunction and existential quantification, so if \mathcal{R} is a set of polyhedra, then so is are its timed and discrete successors. The corresponding geometric operations, intersection and projection, are of exponential complexity with respect to the number of variables in the constraints ($2n + 1$ for Succ_D).

The timed successor operator can be implemented more efficiently by using the generator representation of polyhedra [294]. Let $(\mathcal{V}, \mathcal{S})$ be the generator representation of \mathcal{R} , and $(\mathcal{V}_q^{\text{flow}}, \mathcal{S}_q^{\text{flow}})$ be the generator representation of $\text{Flow}(q)$, then its timed successors in generator representation are

$$\text{Succ}_C(q \times (\mathcal{V}, \mathcal{S})) = q \times (\mathcal{V}, \mathcal{S} \cup (\mathcal{V}_q^{\text{flow}}, \mathcal{S}_q^{\text{flow}})) \cap \text{Inv}. \quad (9.8)$$

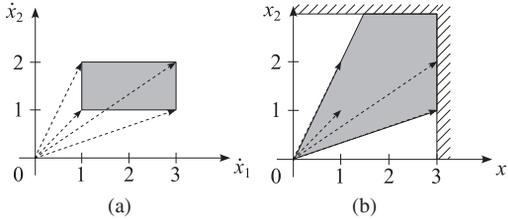


Fig. 9.2 Time elapse using generator representations: (a) derivatives, (b) states.

Example 9.2 Time elapse using generator representation

We compute the time elapse for the states $\mathcal{R} = q \times \{x_1 = x_2 = 0\}$, with $Flow(q) = \{1 \leq \dot{x}_1 \leq 3 \wedge 1 \leq \dot{x}_2 \leq 2\}$ and $Inv(q) = \{x_1 \leq 3, x_2 \leq 3\}$. The generator representation of the flow is

$$Flow(q) = (\{(1, 1), (3, 1), (3, 2), (1, 2)\}, \{\}),$$

as shown in Fig. 9.2(a). To apply time elapse, the vertices of F in the derivative space are reinterpreted as rays in the state space, shown as dashed arrows in Fig. 9.2(b). These rays are added to the generator representation of $\mathcal{R} = q \times (\{(0, 0)\}, \{\})$ and we obtain

$$Succ_C(\mathcal{R}) = q \times (\{(0, 0), \{(1, 1), (3, 1), (3, 2), (1, 2)\}\} \cap \{x_1 \leq 3 \wedge x_2 \leq 3\}),$$

which is the shaded region in Fig. 9.2(b). \square

Example 9.3 Reachable states of Lake Mead

The above successor operators are implemented in the tools *HyTech* and *PHAVer*, as well as an enhanced version of the reachability algorithm 3.1. Figure 9.3 shows the reachable states projected onto the variables x and d . To find the earliest time at which the lake is dry, we intersect the reachable states with the states $\{\text{dry, rainy}\} \times \{x = 0\}$, and use linear programming to find the smallest value of d in the set. According to the result, Lake Mead could be dry in as little as 4.6 years, which is 4 months earlier than the estimate based on the average feed and 20 months later than the worst-case estimate based on the dry season feed. \square

9.4.2 Over-approximating polyhedra

The successor operators from the previous section can be computed exactly for polyhedra with rational or integer coefficients. In principle, this suffices to run a fixed-point computation that returns the exact set of reachable states if it terminates. In practice, this computation is often fatally slow. As it turns out, the polyhedra produced by the successor operators increase in complexity with each iteration. For each polyhedron, this manifests itself in two ways: (i) the number of constraints keeps growing, and (ii) the coefficients of each of the constraints get bigger and bigger, often at an exponential rate. The latter problem would not occur if standard floating

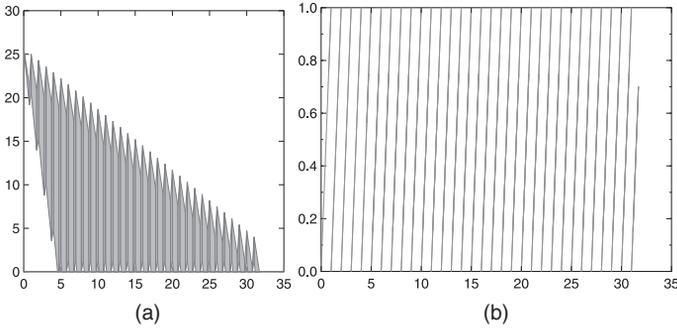


Fig. 9.3 Reachable states of the Lake Mead model with constant bounds: (a) water level x over clock d , (b) clock c over clock d .

point computations were used, but then it is very difficult to guarantee that the result is an over-approximation, and checking emptiness or containment are very sensitive to computation errors.

In the following we present two heuristics, one for each of the above problems. Both take as input a complex polyhedron, and return a simplified polyhedron that is a guaranteed over-approximation of the input. In experiments, they have been used to solve problems that are intractable for exact reachability algorithms. Their downside is that the over-approximation error can be unbounded, and may slow down convergence of the fixpoint computation. It remains a challenge to find the right parameters for both methods such that the fixpoint computation terminates fast while being accurate enough so that the properties of interest can still be shown.

Limiting the number of constraints The method presented in this section assumes that redundant constraints have already been eliminated. The basic idea is to reduce the number of constraints of a polyhedron to some number z by classifying the constraints according to an estimate of their importance, and keeping only the z most important ones. Exact measures of the importance of a constraint, such as comparing the volume of the polyhedron with and without the constraint, are rather expensive. A very fast approximative measure is to compare the angle between the constraints. Consider two constraints, $\mathbf{a}_1^T \dot{\mathbf{x}} \leq b_1$ and $\mathbf{a}_2^T \dot{\mathbf{x}} \leq b_2$. Intuitively speaking, the angle is a measure of importance because if the two constraints point into nearly the same direction, removing one of them will not change the volume of the polyhedron much, depending on their position and the other constraints in the polyhedron. Under this measure, a constraint is important if it is most opposite, or has the largest angle to, the other constraints. Recalling that $\mathbf{a}_1^T \mathbf{a}_2 = \cos \phi$, where ϕ is the angle between the constraints, it suffices to rank the constraints by their respective scalar products instead of computing the angle itself.

Using this measure of importance, the polyhedron is reconstructed from scratch as follows. We start with some initial constraint, e.g. the one with the smallest

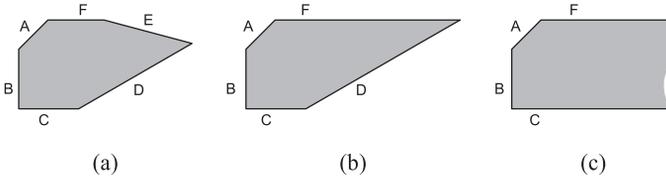


Fig. 9.4 Limiting the number of constraints: (a) original, (b) reduced to 5, (c) reduced to 4.

coefficients. Then we iteratively add the constraint that is most opposite to the ones already selected. We stop when z constraints have been added.

Example 9.4 Reduction of the constraint set

Consider the polyhedron shown in Fig. 9.4(a). It has six constraints A–F. The angle-based reconstruction with C as the initial choice adds constraints in the order F, B, A, D, E. Stopping after five constraints yields the polyhedron shown in Fig. 9.4(b). Note that in terms of volume, a better choice of five constraints would have been C, F, B, D, E. Stopping after five constraints yields Fig. 9.4(c). The polyhedron is unbounded, which illustrates the fact that the approximation error of this method is unbounded, too. \square

Limiting the size of coefficients The size of integer coefficients is relevant because the reachability algorithm can only iterate indefinitely without terminating if the coefficients grow unbounded. The proof is simple: if the coefficients are bounded in size, the number of possible combinations of coefficients, and therefore the number of possible polyhedra, is finite, and will eventually be enumerated by the algorithm in finite time.

Bounding the coefficients limits the possible directions of the constraints to a relatively small number, and thus possibly accelerates the computation. Of course, if a polyhedron has large coefficients, the only valid over-approximation with coefficients smaller than a given limit might be the entire space \mathbb{R}^n . The challenge is to find a limit on the size that is small enough for fast calculations, but large enough so that the accuracy is sufficient. We illustrate the method intuitively with the following example.

Example 9.5 Limiting the size of coefficients

Consider the triangular polyhedron shown in Fig. 9.5(a), which has a constraint with 7-bit coefficients. Rounding the coefficients to 3 bits gives the slightly tilted constraint shown in Fig. 9.5(b). Linear programming is used to push the constraint towards the outside of the polyhedron, see Fig. 9.5(c), thus making the approximation conservative. Further rounding yields the final constraint, and the resulting polyhedron is outlined in Fig. 9.5(d). \square

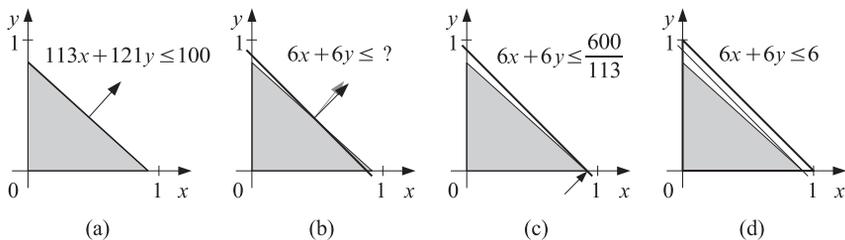


Fig. 9.5 Limiting the size of coefficients in a constraint.

9.5 Over-approximation of affine dynamics

Hybrid automata with non-LHA dynamics, invariants and initial states can be over-approximated with a LHA up to arbitrary accuracy by constructing a LHA that is a *phase-portrait approximation* [314]. This method exploits the fact that modifying a hybrid automaton by over-approximating its invariants, jump relations or *Flow* only adds more behavior, and so the modified automaton is a conservative over-approximation of the original. To increase the accuracy, one can partition the state space (representing each partition by a separate location) and apply the over-approximation separately in each of the partitions. The smaller the partitions, the more accurately the modified automaton reflects the behavior of the original.

If the dynamics are of the form $\dot{x} = Ax + b$ or $M\dot{x} = Ax + b$, a generalized form of *affine dynamics*, a tight LHA-over-approximation can be obtained using quantifier elimination. More generally, we consider *Flow* to be given as a conjunction of constraints of the form

$$a^T \dot{x} + \hat{a}^T x \leq b, \quad a, \hat{a} \in \mathbb{Z}^n, b \in \mathbb{Z}. \quad (9.9)$$

Applying quantifier elimination over x to a flow of the form (9.9) yields LHA-style flow constraints. To obtain a more accurate result, one may include a-priori knowledge about the possible values of x by constraining x to some bounded set \mathcal{S} before the elimination. By definition, the variables must always satisfy the invariant, so one may choose $\mathcal{S} = \text{Inv}$. Similarly, \mathcal{S} can be the corresponding partition created by the phase-portrait approximation, and further refinement of \mathcal{S} is possible [250]. The over-approximated flow is

$$\text{Flow}_{pr}(\mathcal{S}) = \{ \dot{x} \mid \exists x : \dot{x} \in \text{Flow}(x) \wedge x \in \mathcal{S} \}. \quad (9.10)$$

When *Flow* is given as a conjunction of constraints of the form (9.9) and \mathcal{S} is given by linear constraints, (9.10) involves only standard operations on polyhedra in the space of x and \dot{x} , and is, therefore, straightforward to implement. When $\text{Flow}_{pr}(\mathcal{S})$ is too complex to be useful, a construction based on finding bounds for each constraint individually may help [250].

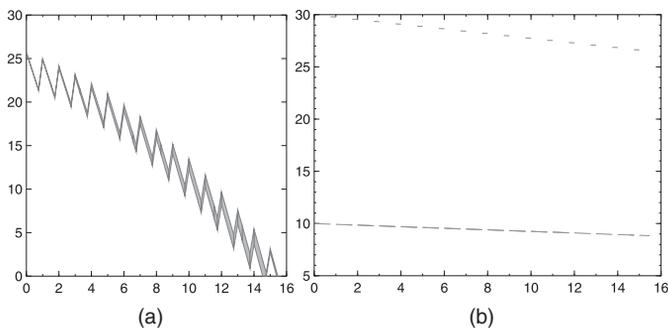


Fig. 9.6 Reachable states of a LHA-over-approximation of a Lake Mead model with affine dynamics: (a) water level x over clock d , (b) feed v over clock d .

Example 9.6 Behaviour of Lake Mead

In the LHA model of Lake Mead, we assumed that the feed of the Colorado river has a constant lower bound corresponding to a feed rate r_{in} reduced by 30%. The reduction actually takes place gradually over $t_{\text{dec}} = 50$ years. We refine our estimate by replacing the constant rate r_{in} with a new state variable v , and letting v decrease linearly over time:

$$\dot{x} = v - r_{\text{out}}, \quad (9.11)$$

$$\dot{v} = -0.3r_{\text{in}}/t_{\text{dec}}, \quad (9.12)$$

with initial values $x_0 = 25.7$ and $v_0 = 2/3 \cdot 15$ since the system is initially in dry season. A change to rainy season can be modeled by tripling v in a discrete transition, and a change back to dry season by dividing it by 3, preserving the original ration between wet and dry season feeds. The resulting hybrid automaton model thus needs only one location. However, (9.11) is not a LHA-style flow constraint, but affine. It contains both a variable and a derivative of a state variable, while LHA-style flow constraints can only be over the derivatives alone.

The affine dynamics of (9.11) can be over-approximated with LHA dynamics by existential quantification over the state variables. The derivative of the feed, \dot{v} , already has a constant bound in (9.12). To obtain constant bounds on the derivative of x we partition the state space along the v -axis into K segments in some interval, say $[0, 32]$. In the i -th segment, v is restricted by the constraints $32i/K \leq v \leq 32(i+1)K$, which substituted in (9.11) yields the bounds

$$32i/K - r_{\text{out}} \leq \dot{x} \leq 32(i+1)K - r_{\text{out}},$$

Figure 9.6 shows the reachable states for $K = 256$. Increasing K improves the precision to an arbitrary degree. \square

Bibliographical notes

The fundamentals of linear hybrid automata are presented in [309]. Their algorithmic analysis is discussed in [15], and its application to nonlinear hybrid systems in [314].

controlengineers.ir

Tools for modeling, simulation, control, and verification of piecewise affine systems

A. Bemporad, S. Di Cairano, G. Ferrari-Trecate, M. Kvasnica,
M. Morari, and S. Paoletti

Tools for mixed logical dynamical and piecewise affine systems based on the model description language HYSDEL are described. They concern various modeling, identification, analysis, control design, and verification tasks. The data exchange format explained in this chapter facilitates the combination of these tools.

Chapter contents

10.1	<i>HYSDEL</i>	page 299
10.1.1	Modeling aim	299
10.1.2	The <i>HYSDEL</i> language	300
10.1.3	<i>HYSDEL</i> 3.0 enhancements	301
10.2	Multi-Parametric Toolbox	303
10.2.1	Functionality of the toolbox	303
10.2.2	Modeling of linear and hybrid systems	305
10.2.3	Control of constrained systems	305
10.2.4	Analysis of hybrid systems	306
10.2.5	Simulations and code generation	307
10.2.6	Summary	307
10.3	<i>Hybrid Toolbox</i>	308
10.3.1	Toolbox features	308
10.3.2	Verification of safety properties	311
10.3.3	Hybrid model-predictive control design	313
10.3.4	Explicit hybrid model-predictive control	313
10.3.5	Applications	314
10.4	Tools for identification	316
10.4.1	<i>Hybrid Identification Toolbox</i>	316
10.4.2	<i>Piecewise Affine System Identification Toolbox</i>	317



10.5	Data interchange format	320
10.5.1	Interchange architecture	320
10.5.2	Platform-independent format for dt-PWA systems	322
10.5.3	XML scheme of the platform-independent format for PWA systems	323
10.5.4	Implementation of the architecture-specific format in <i>MATLAB</i>	324

controlengineers.ir

This chapter describes *MATLAB/Simulink* tools for modeling, identifying, simulating, analyzing, and controlling discrete-time hybrid systems with piecewise affine dynamics, described in mixed logical dynamical (MLD) or piecewise affine (PWA) form. Such tools include:

- *modeling and identification* tools for describing the hybrid model in a high-level user-friendly way, or for identifying hybrid models from data;
- *simulation* tools for open-loop simulation and validation of hybrid models;
- *analysis* tools for characterization of stability and reachability properties of hybrid models;
- *control design* tools for designing hybrid model-predictive controllers, simulating closed-loop performances, and generating real-time control code.

(Section 10.1) describes the modeling language *HYSDEL*, which ease the process of formulating a hybrid model. Sections 10.2 and 10.3 describe two toolboxes for analysis, simulation, and control of hybrid systems, the *Multi-Parametric Toolbox* and the *Hybrid Toolbox*. (Section 10.4) describes two tools for identification of hybrid piecewise-affine models from data, the *Hybrid Identification Toolbox* and the *PieceWise Affine Identification Toolbox*. Finally, (Section 10.5) describes an interchange format for transferring models among all the aforementioned tools.

10.1 HYSDEL

10.1.1 Modeling aim

HYSDEL (HYbrid System DEscription Language) [632] allows modeling a class of hybrid systems described by interconnections of linear dynamical systems, finite-state automata, IF-THEN-ELSE rules, and propositional logic statements. It allows hybrid models to be formulated in a manner appealing to the application engineer. *HYSDEL* uses simple, natural language statements to model complex relations between continuous and discrete elements of the model. Based on the textual description provided by the user, a compiler associated with *HYSDEL* creates an appropriate mathematical representation of the plant using linear relations involving both continuous and discrete decision variables. The generated mathematical model can either take the form of mixed logical dynamical (MLD) models, which can be subsequently converted into an equivalent piecewise affine (PWA) representation [56, 265]. Both these models can be readily used for plant optimization or verification. *HYSDEL* also generates a *MATLAB* simulator of the described hybrid dynamics.

Unlike general-purpose optimization modeling languages (such as *AMPL*), *HYSDEL* is specifically tailored for modeling dynamical systems. Therefore, the *HYSDEL* compiler can exploit the specific problem structure to derive higher quality models, compared to high-level tools. As a consequence, the optimization problems

formulated using *HYSDEL* models can be solved more efficiently compared to other approaches.

HYSDEL is currently available in version 2.0.5. Version *HYSDEL* 3.0 is under development as a joint project of the Automatic Control Laboratory at ETH Zürich, Switzerland, and ABB, Switzerland.

10.1.2 The *HYSDEL* language

The *HYSDEL* language allows to model hybrid systems using difference equations, on/off switches, IF-THEN-ELSE rules, and finite-state automata. The modeling principle is based on so-called discrete hybrid automata (DHA) [632], which are the interconnection of a finite-state machine and a switched linear dynamical system through a mode selector and an event generator.

The textual representation of the *HYSDEL* model consists of an INTERFACE part, where state, input, and output variables of the model are declared, and an IMPLEMENTATION part, which describes relations between the declared variables. The syntax of the model is, therefore, as follows:

```

SYSTEM name {
  /* example of {\sl HYSDEL} file structure */
  INTERFACE {
    /* declaration of variables */
  }
  IMPLEMENTATION {
    /* relations between declared variables */
  }
}

```

Declared variables are distinguished by their type (state, input, output) and by their kind (real or binary). In addition, real-valued or symbolic parameters can be also declared in the INTERFACE part.

The IMPLEMENTATION part is composed of specialized sections describing the relations among the declared variables. The IMPLEMENTATION part starts with an optional AUX section which contains the declarations of the internal signals of the hybrid system, called also auxiliary variables. The declaration follows the general syntax of the variable declaration. Linear relations between real-valued variables can be captured in the LINEAR section, while the CONTINUOUS section serves to describe linear state-update equations (continuous dynamics). Similarly, logic statements involving purely Boolean variables can be captured in the LOGIC section. State update equations involving Boolean state variables can be similarly described in the AUTOMATA section. Moreover, IF-THEN-ELSE rules can be defined in a DA (discrete-to-analog) section, while the AD (analog-to-discrete) section serves to define the switching conditions as linear constraints. The OUTPUT section allows one to specify static linear and logic relations for the output vector of the hybrid model. Finally, *HYSDEL* allows one more section: the MUST section. This section specifies

arbitrary linear and logic constraints on continuous and Boolean states, inputs, and outputs.

A complete description of the syntax of *HYSDEL* is available in the manual accompanying the compiler [633].

The hybrid models automatically generated by *HYSDEL* can be used for solving several analysis (stability, observability, safety/reachability) and design (control, state estimation) tasks. An example of a hybrid system modeled in *HYSDEL* is provided in (Section 10.3).

10.1.3 *HYSDEL* 3.0 enhancements

HYSDEL has been used in hundreds of engineering applications and is a very stable and reliable tool. However, some improvements on *HYSDEL* can be done. For instance, one of the main limitations of the *HYSDEL* 2.0.5 language is its scalar orientation, which means that all state, input, and output variables of the model have to be declared and used separately. This renders the modeling of systems with several variables time consuming and error prone. Another limitation of the *HYSDEL* 2.0.5 language, and one closely related to its scalar orientation, is the lack of support for FOR-loops. Finally, *HYSDEL* 2.0.5 does not allow multiple independent models to be merged together in order to utilize existing models as building blocks when creating more complex models.

To address the main limitations of *HYSDEL* 2.0.5 as described in (Section 10.1.2), *HYSDEL* 3.0 extends the prior version by the following new features:

1. Variables of the model can be declared as vectors or matrices.
2. Iterations through vectorized signals can be achieved by means of nested FOR-loops.
3. Models of complex systems can be obtained by defining interconnections between subsystems.
4. The compiler uses optimization packages to improve quality of the generated models.

As mentioned earlier, *HYSDEL* 3.0 also allows the use of (nested) FOR-loops to iterate through certain elements of a vector variable. To illustrate this feature, we again consider a state vector with three elements, where the task is to model the following relation: $x(k + 1) = Ax(k)$ where the index k takes integer values from the interval $1, \dots, N$. In *HYSDEL* 3.0 this can be easily achieved as follows:

```

SYSTEM model {
  INTERFACE {
    STATE { REAL x(3, N); }
    PARAMETER { REAL A, N; }
  }
  IMPLEMENTATION {
    AUX { INDEX k; }
    LINEAR {

```

```

        FOR (k = 1:N-1) {
            x(k+1) = A*x(k);
        }
    }
}
}

```

Important to notice is that even the dimensions of the declared variables can be defined by means of symbolic parameters, the value of which is only known at the time the model is compiled into a corresponding mathematical representation.

HYSDEL 3.0 also allows one to split the textual description of the plant model into several independent files, which can be subsequently interconnected on a higher level.

Example 10.1 *HYSDEL model of a tank system*

The task is to model the dynamical behavior of a system which consists of two liquid tanks, where the liquid flowing out of tank no. 1 enters directly into tank no. 2. This task can be easily solved if the behavior of a single tank is captured by a standalone *HYSDEL* file, say `single_tank.hys`. The user then creates a so-called “master” file, where she first creates two instances of the `single_tank` object (T1 and T2), and then, in the second step, connects respective input and output variables according to the configuration of the tanks:

```

SYSTEM two_tanks {
    INTERFACE {
        MODULE { single_tank T1, T2; }
        INPUT { REAL u; }
        OUTPUT { REAL y; }
    }
    IMPLEMENTATION {
        LINEAR {
            T1.inflow = u;
            T2.inflow = T1.outflow;
        }
        OUTPUT {
            y = T2.outflow;
        }
    }
}

```

□

In *HYSDEL* 2.0.5 the compiler, which parses the input source file to generate the MLD mathematical model, is written in C++. The *HYSDEL* 3.0 compiler has been rewritten in *MATLAB*, which ease maintenance and extensions, and the access to external optimization packages needed to improve the quality of the generated mathematical models. In particular, the *HYSDEL* 3.0 compiler is based on the *YALMIP* [412] optimization toolbox, which takes care of the generation of the MLD model.

During the MLD generation procedure, YALMIP automatically optimizes parameters of the mathematical equivalents of individual *HYSDEL* language statements.

Example 10.2 Representation of logic statements by inequalities

To illustrate the idea, consider a propositional logic statement of the following form:

$$[f(x) \leq 0] \leftrightarrow [\delta = 1]. \tag{10.1}$$

When $f : \mathcal{R}^n \mapsto \mathcal{R}$ is a linear function in the real variable x , it is well known [62] that the statement (10.1) can be captured by the following two linear inequalities in the continuous variable x and in the binary variable δ :

$$f(x) \leq M(1 - \delta), \tag{10.2}$$

$$f(x) \geq \epsilon + (m - \epsilon)\delta. \tag{10.3}$$

where $\epsilon \geq 0$ is an arbitrarily small number (e.g. the machine precision), and M and m denote, respectively, the maximal and minimal values the function $f(x)$ can take on a certain domain $x \in \mathcal{X}$. □

In *HYSDEL* 2.0.5, the M and m bounds are calculated by assuming known min/max boundaries of the variable x . However, if the domain \mathcal{X} which limits the admissible values of x is a generic polyhedron, tighter M and m can be calculated through linear programming, which improves the quality of the mixed-integer programming models generated for control and verification purposes [373].

10.2 Multi-Parametric Toolbox

10.2.1 Functionality of the toolbox

Optimal control of constrained linear and hybrid systems has garnered great interest in the research community due to the ease with which complex problems can be stated and solved. The aim of the *Multi-Parametric Toolbox* (MPT) is to provide efficient computational means to obtain feedback controllers for these types of constrained optimal control problems in a *MATLAB* programming environment. By multi-parametric programming a linear or quadratic optimization problem is solved off-line. The associated solution takes the form of a piecewise affine state feedback law. In particular, the state space is partitioned into polyhedral sets and for each of those sets the optimal control law is given as an affine function of the state. In the on-line implementation of such controllers, computation of the controller action reduces to a simple set-membership test, which is one of the reasons why this method has attracted so much interest in the research community.

As shown in [67] for quadratic objectives, a feedback controller may be obtained for constrained linear systems by applying multi-parametric programming techniques. The linear objective was tackled in [64] by the same means. The multi-parametric algorithms for constrained finite time-optimal control (CFTOC) of linear systems contained in the MPT are based on [39] and are similar to [631].

It is current practice to approximate the constrained infinite time optimal control (CITOC) by receding-horizon control (RHC) – a strategy where the CFTOC problem is solved at each time step, and then only the initial value of the optimal input sequence is applied to the plant. The main problem of RHC is that it does not, in general, guarantee stability or constraint satisfaction. In order to obtain these properties, certain conditions have to be added to the original problem [448]. The extensions to guarantee these properties are a part of the MPT. It is furthermore possible to impose a minimax optimization objective which allows for the computation of robust controllers for linear systems subject to polytopic and additive uncertainties [68, 356]. As an alternative to computing suboptimal stabilizing controllers, the procedures to compute the infinite time optimal solution for constrained linear systems [277] are also provided.

Even though the multi-parametric approaches rely on off-line computation of a feedback law, the computation can quickly become prohibitive for larger problems. This is not only due to the high complexity of the multi-parametric programs involved, but mainly because of the exponential number of transitions between regions which can occur when a controller is computed in a dynamic programming fashion [104, 357]. The MPT, therefore, also includes schemes to obtain sub-optimal controllers of low complexity for linear and PWA systems as presented in [276, 278, 279, 281].

In addition to control tools, the toolbox provides extensive functionality for polytope manipulation: convex hulls, convex unions and envelopes, Minkowski sums, and Pontryagin differences, as well as many other operations, can be performed efficiently by MPT. Most of the functionality supports both single polytopes as well as non-convex unions thereof. Other commercial software, such as the Geometric Bounding Toolbox (GBT) [647], provides similar functionality on the level of polytope manipulation. MPT, however, explicitly takes advantage of object-oriented programming to provide a transparent and easy to use interface.

MPT can be viewed as a unifying repository of hybrid systems design tools from international experts utilizing state-of-the-art optimization packages. The included software packages concern linear programming (*CDD*, *GLPK*), quadratic programming (*CLP*), mixed-integer linear programming (*GLPK*), and semi-definite programming (*SeDuMi*). In addition, MPT ships with a dedicated solver for computing projections of convex polytopes, a Boolean optimization package *ESPRESSO*, as well as with the *HYSDEL* modeling language [632].

The main factor which distinguishes this toolbox from other alternatives is the emphasis on efficient formulation of the problems which are being solved. This means that the toolbox provides implementation of novel control design and analysis algorithms, but also offers the user an easy way to use them without the need to be an expert in the respective fields. MPT aims at providing tools which can be used in the whole chain of the process of successful control design. It allows users not only to design optimization-based controllers, but also to formally verify that they behave as desired, investigate the behavior of the closed-loop system, and to post-process the resulting feedback laws in order to simplify them without losing prescribed design properties.

In the following sections we describe the main building blocks which MPT offers for the modeling, control, analysis, simulations, and deployment of optimization-based control strategies.

10.2.2 Modeling of linear and hybrid systems

MPT can design control laws for discrete-time constrained linear, switched linear and hybrid systems. Hybrid systems can be described in piecewise-affine or mixed logical dynamical representations and an efficient algorithm is provided to switch from one representation to the other form and vice-versa. To increase user's comfort, models of dynamical systems can be imported from various sources:

- models of hybrid systems generated by the *HYSDEL* [632] and language;
- state-space and transfer function objects of the Control System Toolbox;
- system Identification Toolbox objects;
- MPC toolbox objects.

Models of dynamical systems can be imported into MPT using the following function call:

```
model=mpt_sys(object, Ts)
```

where `object` can be either a string (in which case the model is imported from a corresponding *HYSDEL* source file), or it can be a variable of one of the above mentioned object types. The second input parameter `Ts` denotes sampling time and can be omitted, in which case $T_s = 1$ is assumed.

In addition, MPT allows to define different types of constraints which the user wishes to impose on certain model variables. These include, but are not limited to

- Min/Max or polytopic constraints on system outputs;
- Min/Max or polytopic constraints on system states;
- Min/Max or polytopic constraints on manipulated variables;
- Min/Max or polytopic constraints on slew rate of manipulated variables.

The obtained models can then be used either for control design or, alternatively, for analysis and simulation.

10.2.3 Control of constrained systems

Once the model of the controlled system is available, MPT can be used to synthesize optimal and sub-optimal control laws either in implicit form, where an optimization problem of finite size is solved on-line at every time step and is used in a receding horizon control (RHC) manner or, alternatively, solve the optimal control problem in a multi-parametric fashion. If the latter approach is used, an explicit representation of the control law is obtained.

Solutions to the following problems can be obtained depending on the properties of the system model and the optimization problem:

- constrained finite time-optimal control (CFTOC) problem;
- constrained infinite time-optimal control problem (CITOC);
- constrained minimum time-optimal control (CMTOC) problem;
- low complexity setup.

The problem that will be solved depends on the parameters of the system model and the problem setup. Specifically, different algorithms are used for different types of models (linear or hybrid), prediction horizon (finite or infinite), and the level of optimality which is desired (optimal solution or suboptimal solutions with reduced complexity).

In addition, users can freely modify the underlying optimization problem. This can be done either by adding custom constraints (such as constraints involving logic decisions, norms, move blocking, collision avoidance constraints, etc.), or by modifying the objective function.

Once a particular form of the optimization problem to solve has been chosen by the user, the optimal control problem can be solved by calling a single function `mpt_control`. If the problem should be solved in a parametric fashion, the user calls the function as follows:

```
controller = mpt_control(model, problem)
```

On-line MPC controllers can be generated by calling

```
controller = mpt_control(model, problem, 'online')
```

Here, the variable `model` describes the model of the controlled system and the variable `problem` represents the parameters of the optimization problem. Once the problem is solved, the resulting solution is returned in the variable `controller`. The returned control law can be subsequently used either for analysis or for simulation purposes.

10.2.4 Analysis of hybrid systems

The MPT toolbox offers broad functionality for analysis of hybrid systems and verification of safety and liveness properties of explicit control laws. In addition, stability of closed-loop systems can be verified using different types of Lyapunov functions.

In particular, MPT can compute forward N -steps reachable sets for linear and hybrid systems assuming the system input either belongs to some bounded set of inputs, or when the input is driven by some given explicit control law. These sets are calculated using exact polytopic operations.

Reachability computation can be directly extended to answer the following question:

Do the states of a dynamical system (whose inputs either belong to some set of admissible inputs, or whose inputs are driven by an explicit control law) enter some set of “unsafe” states in a given number of steps?

MPT gives an answer to this question by either calculating a subset of the admissible state space, which enter the set of unsafe states in, at most, N steps, or provides a certificate that no such states exist.

In terms of stability analysis, MPT offers functions which aim at identifying quadratic, sum-of-squares, piecewise quadratic, piecewise affine or piecewise polynomial Lyapunov functions. If such a function is found, it can be used to show stability of the closed-loop systems even in cases where no such guarantee can be given a-priori based on the design procedure.

MPT also addresses the issue of complexity reduction of the resulting explicit control laws. Although the effort of evaluating look-up table styled solution on-line is usually small, it can become prohibitive for complex controllers with several thousands or even more regions. Therefore, MPT allows to reduce this complexity by simplifying the controller partitions over which the control law is defined. This simplification is performed by merging regions which contain the same expression of the control law. By doing so, the number of regions may be greatly reduced, while maintaining the same performance as the original controller [263].

10.2.5 Simulations and code generation

MPT provides simple, yet powerful, capabilities to simulate optimization-based controllers, regardless whether in implicit or in an explicit form. The simulation can either be carried out directly in *MATLAB*, or the *Simulink* modeling tool can be used in connection with Multi-Parametric Toolbox. If the latter approach is used, the MPT *Simulink* library provides several building blocks, which allow control laws and models of dynamical systems to be used in *Simulink*.

In particular, the *MPT Controller* block supplies the control action as a function of the measured state. If the controller is an explicit one, it is possible to directly compile a *Simulink* model which includes one or more of the *MPT Controller* blocks using the Real Time Workshop.

The *Dynamical System* block serves for simulations of constrained linear and hybrid systems described by the system model. The user must specify initial values of the state vector in a dialog box.

Finally, the *In polytope* block returns *true* if a input point lies inside of a given polytope, *false* otherwise. If the polytope variable denotes a polytope array, the output of this block will be the index of a region which contains a given point. If no such region exists, 0 (zero) will be returned.

10.2.6 Summary

Over the past 3 years, MPT has attracted the attention of important industrial and government entities, such as ABB, DaimlerChrysler, Ford, Honeywell, and NASA. Successful applications include control of DC-DC converters, driver assistance systems, control of electronic throttles, or identification of wing movements in experimental aircrafts. More than 20 universities world-wide use our toolbox either in their research, or directly utilize MPT's building blocks to create their own

tools. The popularity of the Multi-Parametric Toolbox is demonstrated by more than 12,000 downloads over the last 3.5 years. The toolbox can be obtained from <http://control.ee.ethz.ch/~mpt/>.

10.3 Hybrid Toolbox

The *Hybrid Toolbox* [57] is a *MATLAB/Simulink* toolbox for modeling, simulating, and verifying hybrid dynamical systems, for designing and simulating model-predictive controllers for hybrid systems subject to constraints, and for generating linear and hybrid MPC control laws in piecewise affine form that can be directly embedded as C-code in real-time applications.

The toolbox was developed at the Department of Information Engineering of the University of Siena and can be freely downloaded for non-commercial use from <http://www.dii.unisi.it/hybrid/toolbox>.

10.3.1 Toolbox features

The toolbox offers the following features:

Hybrid model design, simulation, and reachability analysis Hybrid models can be conveniently described in *HYSDEL*. *HYSDEL* models are converted to MLD models that are handled as *MATLAB* objects. MLD objects can be automatically converted into PWA objects, using an implementation of the conversion algorithm described in [56]. MLD and PWA objects can be simulated in open loop for validation purposes through command-line functions or *MATLAB* scripts, or in *Simulink*. Safety properties can be verified through reachability analysis based on mixed-integer programming.

Control design Model-predictive controllers (MPC) based on on-line mixed-integer linear or quadratic programming (MILP/MIQP) can be designed for hybrid systems converted to MLD form. MPC performance functions based on both quadratic and infinity norms are supported.

Explicit control design The toolbox provides a multi-parametric quadratic programming (mpQP) solver based on the algorithm described in [631], a multi-parametric linear programming (mpLP) solver based on a similar implementation, and a multi-parametric hybrid optimal control solver based on the method described in [10]. Several functions for manipulation and visualization of polyhedral objects and polyhedral partitions are provided by the toolbox.

MPC controllers designed for hybrid systems and for constrained linear systems can be converted to their equivalent explicit piecewise affine form via off-line optimization (multiparametric programming). Linear MPC controllers designed with the Model-Predictive Control Toolbox for *MATLAB* [69] can be also converted into piecewise affine form via multiparametric quadratic programming.

C-code generation Explicit controllers can be easily exported as C-code for direct implementation and rapid prototyping, by either simply running Real Time Workshop or by embedding the generated C code in the application.

Simulink library The toolbox provides *Simulink* blocks for controllers based on on-line optimization (MILP/MIQP or QP), explicit piecewise linear controllers (for hybrid or linear systems), and for simulation and validation of MLD and PWA models.

Solver support The *Hybrid Toolbox* supports several solvers for linear, quadratic, and mixed-integer optimization. The list of supported solvers includes the GNU Linear Programming Kit for LP/MILP [434] through the GLPKMEX *MATLAB* interface, the QP solver of the MPC Toolbox, the LP/QP/MILP/MIQP solvers of Cplex [334] through the CPLEXMEX interface, the LP/QP/MILP/MIQP solvers of Xpress-MP [198] through the MEXPRESS interface, the LP/QP solvers of the NAG Foundation Toolbox, the LP/QP solvers of the Optimization Toolbox, and the free MIQP solver MIQP.M [61]. The source code of the MEX interfaces can be downloaded from <http://www.dii.unisi.it/hybrid/tools>.

Demos Several demos are provided to highlight different functionalities of the toolbox. In the following section we briefly describe the features of the toolbox through simple illustrating examples.

Example 10.3 *Temperature control*

Consider the problem of regulating the temperatures T_1 and T_2 of two bodies in a room (see demo `heatcool.m` in the `demos/hybrid/` directory of the *Hybrid Toolbox*). The room is equipped with a heater and an air conditioning system, which are automatically activated according to the following rules:

- body #1 turns the heater on whenever its temperature T_1 is below a certain threshold $T_{cold,1}$, and similarly turns the air conditioning system on when T_1 is above a certain threshold $T_{hot,1}$;
- body #2 turns the heater on whenever its temperature T_2 is below a certain threshold $T_{cold,2}$, unless body #1 is hot ($T_1 \geq T_{hot,1}$); similarly, body #2 turns the air conditioning system on whenever T_2 is above a certain threshold $T_{hot,2}$, unless body #1 is cold ($T_1 \leq T_{cold,1}$);
- otherwise, both the heater and the air conditioning system are off.

The dynamics can be interpreted as a double thermostat, with thermostat #2 having lower priority than thermostat #1.

By letting u_{hot} be the amount of heat produced by the heater, and u_{cold} the amount of heat subtracted by the air conditioner, the dynamics of T_1, T_2 is

$$\dot{T}_1 = -\alpha_1(T_1 - T_{amb}) + k_1(u_{hot} - u_{cold}), \tag{10.4}$$

$$\dot{T}_2 = -\alpha_2(T_2 - T_{amb}) + k_2(u_{hot} - u_{cold}), \tag{10.5}$$

where the ambient temperature T_{amb} is treated as a continuous input to the system (assume for instance that the ambient temperature of the room can be changed by opening a window), and $k_1 = 0.8, k_2 = 0.4, \alpha_1 = 1, \alpha_2 = 0.5, U_c = U_h = 2, T_{cold,1} = 15, T_{hot,1} = 30,$

```

/* Heat and cool example - (C) 2003 by A. Bemporad */
SYSTEM heatcool {
INTERFACE {
    STATE { REAL T1 [-10,50];
            REAL T2 [-10,50];
        }
    INPUT { REAL Tamb [-50,50];
        }
    OUTPUT {REAL y1;
            REAL y2;};
    PARAMETER {
        REAL Ts, alpha1, alpha2, k1, k2;
        REAL Thot1, Tcold1, Thot2, Tcold2, Uc, Uh;
    }
}
IMPLEMENTATION {
    AUX { REAL uhot, ucold;
        BOOL hot1, hot2, cold1, cold2;
    }
    AD { hot1 = T1>=Thot1;
        hot2 = T2>=Thot2;
        cold1 = T1<=Tcold1;
        cold2 = T2<=Tcold2;
    }
    DA { uhot = {IF cold1 | (cold2 & ~hot1) ...
        THEN Uh ELSE 0};
        ucold = {IF hot1 | (hot2 & ~cold1) ...
        THEN Uc ELSE 0};
    }
    CONTINUOUS { T1 = T1+Ts*(-alpha1*(T1-Tamb)+...
        k1*(uhot-ucold));
        T2 = T2+Ts*(-alpha2*(T2-Tamb)+...
        k2*(uhot-ucold));
    }
    OUTPUT {y1=T1;
        y2=T2;
    }
}
}

```

Fig. 10.1 *HYSDEL* 2.0.5 model for the hybrid thermal system (heatcoolmodel.hys).

$T_{\text{cold},2} = 10$, $T_{\text{hot},2} = 35$ (no physical units are provided here as the model is a toy example with little physical significance).

To obtain a *HYSDEL* model of the system, we sample the continuous dynamics with sampling time $T_s = 0.5$, by replacing \dot{T} with $\frac{T(k+1)-T(k)}{T_s}$ in (10.5) (Euler approximation). The overall hybrid dynamics is described by the *HYSDEL* model `heatcoolmodel.hys` reported in Figure 10.1.

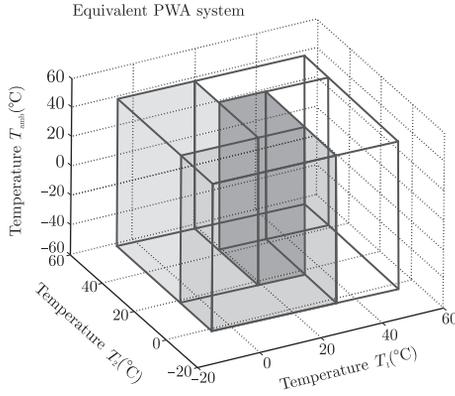


Fig. 10.2 PWA equivalent of thermal system. Note that the partition does not depend on the input variable T_{amb} .

The corresponding MLD model is simply obtained through the MATLAB command

```
>> S=mlc('heatcoolmodel', Ts);
```

which returns the MLD object S containing all MLD matrices, dimensions of system variables, etc. In this case the system has 2 continuous states, 1 continuous input, 2 continuous auxiliary variables, 6 binary auxiliary variables, and consists of 20 mixed-integer linear inequalities.

The PWA equivalent P of S is obtained as follows:

```
>> P=pwa(S);
```

and consists of five regions in the three-dimensional space (T_1, T_2, T_{amb}) , as depicted in Fig. 10.2

The MLD system S or its equivalent PWA system P can be now simulated in open loop for inspecting the dynamical behavior of the hybrid system and for validation purposes. For example, given the initial condition $T_1(0) = 30, T_2(0) = 20$ and input signal $T_{amb}(t) = 20 + 10 \cos(t/10), t \in \mathbb{R}$, the command

```
>> [X, T, D, Z, Y]=sim(S, [30;20], Tamb);
```

generates the trajectories of state, auxiliary, and output vectors of the MLD system, and similarly

```
>> [X, T, I, Y]=sim(P, [30;20], Tamb);
```

generates state, mode, and output trajectories of the PWA system. Clearly, the state and output trajectories generated by the MLD and PWA system coincide. \square

10.3.2 Verification of safety properties

Simulation tools allow probing a model for a certain initial condition and input excitation. Reachability analysis aims at detecting whether a hybrid model will eventually reach unsafe state configurations for *all* possible initial conditions and input excitations within a prescribed set.

For MLD systems the problem is to test whether a certain terminal unsafe set $X_f = \{x : S_x x \leq T_x\}$ can be reached after exactly N steps, starting from a set of initial states $X_0 = \{x : S_0 x \leq T_0\}$ for some input sequence $u(0), \dots, u(N-1)$ with $u_{\min} \leq u(k) \leq u_{\max}, \forall k = 0, \dots, N-1$. Such a sequence exists if the following set of mixed-integer linear inequalities is feasible:

$$\begin{cases} S_0 x(0) \leq T_0, \\ \mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{B}_1 \mathbf{u}(k) + \mathbf{B}_2 \delta(k) + \mathbf{B}_3 \mathbf{z}(k), k = 0, 1, \dots, N-1, \\ \mathbf{E}_2 \delta(k) + \mathbf{E}_3 \mathbf{z}(k) \leq \mathbf{E}_1 \mathbf{u}(k) + \mathbf{E}_4 \mathbf{x}(k) + \mathbf{E}_5, k = 0, 1, \dots, N-1, \\ \mathbf{u}_{\min} \leq \mathbf{u}(k) \leq \mathbf{u}_{\max}, k = 0, 1, \dots, N-1, \\ S_x \mathbf{x}(N) \leq T_x, \end{cases} \quad (10.6)$$

with respect to the unknowns $\mathbf{u}(0), \delta(0), \mathbf{z}(0), \dots, \mathbf{u}(N-1), \delta(N-1), \mathbf{z}(N-1), \mathbf{x}(0)$. Complex conditions can be posed in the reachability analysis by adding constraints (such as linear constraints, logical constraints, and combinations of both) in the MUST section of the *HYSDEL* file that defines the MLD model.

Example 10.4 Temperature control problem

Consider again the thermal example. We want to test if the set of states $\mathcal{X}_f = \{(T_1, T_2) : 10 \leq T_1, T_2 \leq 15\}$ can be reached after $N = 10$ steps from an initial state $\mathbf{x}(0)$ in the set $\mathcal{X}_0 = \{(T_1, T_2) : 35 \leq T_1, T_2 \leq 40\}$ with a bounded input $10 \leq T_{\text{amb}} \leq 30$. The reachability analysis question is answered through the following code:

```

>> Xf.A=[eye(2);-eye(2)];
>> Xf.b=[15;15;-10;-10];
>> X0.A=[eye(S.nx);-eye(S.nx)];
>> X0.b=[40;40;-35;-35];
>> umin=10;
>> umax=30;
>> [flag,x0,U,xf,X,T]=reach(S,N,Xf,X0,umin,umax);
    
```

□

The answer is `flag=1`, which means that the set of states \mathcal{X}_f is reachable from \mathcal{X}_0 under the above dynamics and input bounds. The code also returns an example of initial state $\mathbf{x}(0) = \mathbf{x}_0$, sequence of inputs $[\mathbf{u}(0) \mathbf{u}(1) \dots \mathbf{u}(N-1)] = \mathbf{U}$, and the corresponding final state $\mathbf{x}(N) = \mathbf{x}_f$ and state sequence $[\mathbf{x}(0) \mathbf{x}(1) \dots \mathbf{x}(N-1)] = \mathbf{X}$, satisfying the reachability analysis problem. With the lower bound $T_{\text{amb}} \geq 20$, the MILP problem (10.6) is infeasible, meaning that no input exists driving \mathbf{x}_0 to \mathcal{X}_f .

A more extended “safety” question is whether the state can never reach \mathcal{X}_f at any time $k, 1 \leq k \leq N$, can be also tested. Instead of solving this extended problem by solving the MILP (10.6) for all horizons between 1 and N , the *Hybrid Toolbox* provides the answer with one MILP, by introducing an auxiliary binary variable as described in [57]. The corresponding command is

```

>> [flag,x0,U]=reach(S,[1 N],Xf,X0,umin,umax);
    
```

10.3.3 Hybrid model-predictive control design

The design of a model-predictive controller for a hybrid system is demonstrated by the following example.

Example 10.5 MPC design

Consider the problem to command T_{amb} so that T_2 tracks a given reference trajectory $r(t) = 15 \sin(\frac{2}{5}t)$ under the constraint $T_1(t) \geq 25, \forall t \geq 0$. To this end we design the hybrid MPC controller based on the optimization problem

$$\min_{\{u, \delta, z\}_0^{N-1}} \sum_{k=1}^{N-1} |Q_x(T_2(t+k|t) - r(t))|, \quad (10.7)$$

$$\text{s. t. } x(t+k|t) \geq 25, \quad k = 1, \dots, N. \quad (10.8)$$

Using the *Hybrid Toolbox* the problem is formulated as

```

>> refs.x=2;      % only state x(2) is weighted
>> Q.x=1;        % weight on state x(2)
>> Q.rho=Inf;    % hard constraints
>> Q.norm=Inf;   % infinity norm
>> N=2;         % prediction horizon
>> limits.xmin=[25;-Inf];
>> C=hybcon(S,Q,N,limits,refs);
    
```

where C is now an MPC controller object of type `@hybcon`. A closed-loop simulation over a time interval of $T_{stop} = 100$ time units from the initial condition $T_1(0) = 30, T_2(0) = 30$ is obtained through the commands

```

>> r.x=30+15*sin(2/5*(0:Ts:Tstop-Ts)'/5);
>> [X,U,D,Z,T,Y]=sim(C,S,r,[30;20],Tstop);
    
```

producing the closed-loop trajectories reported in Fig. 10.3 The closed-loop control system can also be simulated in *Simulink* using the “Hybrid Controller” block reported.

In case we want to use the quadratic penalty $(T_2(t+k|t) - r(t))^2$ and the MIQP solver of CPLEX, it is enough to set

```

>> Q.norm=2; % use quadratic costs
>> C=hybcon(S,Q,N,limits,refs);
>> C.mipsolver='cplex';
    
```

□

10.3.4 Explicit hybrid model-predictive control

The hybrid MPC controller C can be converted to its equivalent explicit PWA form through the following command:

```

>> E=expcon(C,range,options);
    
```

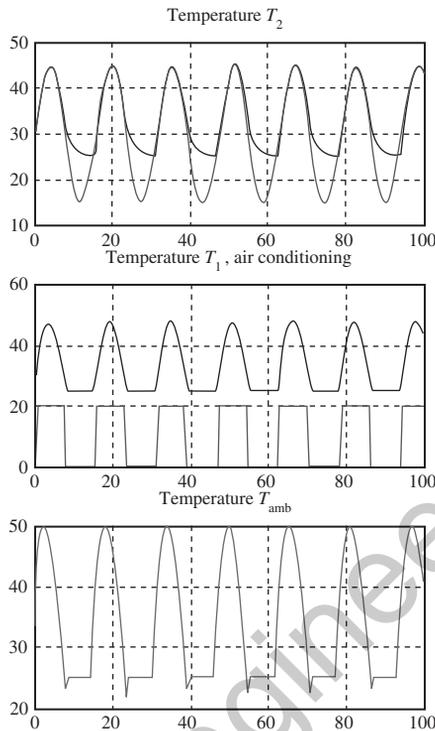


Fig. 10.3 Closed-loop hybrid MPC simulation.

where `range` is a structure defining the bounds of interest for measured states and references, and `options` is a structure defining parameters for the multiparametric solvers. The output is an explicit `@excon` controller object consisting of 12 affine gains, defined over the polyhedral partition depicted in Fig. 10.4. The closed-loop system containing the explicit controller can be either simulated in *MATLAB* through command `sim` or in *Simulink* using the “Explicit Hybrid Controller” block. In the latter case Real-Time Workshop can be employed to rapidly prototype the controller.

The *Hybrid Toolbox* handles constrained *linear* MPC designs `@lincon` and their explicit counterparts `@excon` in a similar fashion.

10.3.5 Applications

Since its first release at the end of 2004 the toolbox has been requested from the author by approximately 2000 users to date, and employed in several industrial

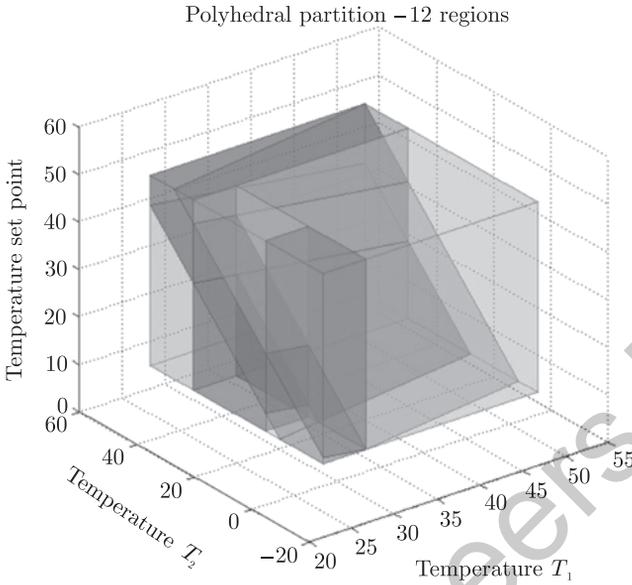


Fig. 10.4 Equivalent explicit MPC controller.

applications in numerous different domains. The modeling and control tools provided by the *Hybrid Toolbox* are mainly useful for the following areas [58]:¹

- *MPC of complex dynamical plants that cannot be handled by linear models due to logical states, switching dynamics, nonlinearities, and to the enforcement of logical rules and other constraints.* A hybrid MPC controller can be designed through the toolbox and embedded in existing *Simulink* diagrams for realistic simulation based on mixed-integer programming solutions. Real-time implementation was successfully carried out through both the xPC-Target and OPC Toolbox for *MATLAB* (sampling time in the second to hour range). Typical applications include on-line rescheduling of production plants and supervisory control of industrial processes.
- *MPC of small-size fast dynamical processes without on-line optimization.* Thanks to the capabilities of the toolbox to convert linear or hybrid MPC laws into a lookup-table of linear gains and automatically generate the corresponding C code, MPC can be smoothly embedded in any real-time application (sampling frequency up to 100 kHz, depending on the control platform). Typical appli-

¹ The Ford Motor Company is acknowledged for continuous support for the development of the *Hybrid Toolbox*, and in particular D. H. Hrovat, I. V. Kolmanovsky, and H. E. Tseng.

cations are in the automotive industry, as testified by numerous successful test cases, as described in [Chapter 15](#).

10.4 Tools for identification

10.4.1 Hybrid Identification Toolbox

The use of real data for inferring models and parameters characterizing hybrid dynamics calls for software for hybrid identification. This was the main motivation underlying the development of HIT [231], a public available, GNU-licensed *MATLAB* toolbox for hybrid identification. The choice of developing the Hybrid Identification Toolbox (HIT) within the *MATLAB* environment has been dictated by two factors. The first one is that *MATLAB* is a powerful language for scientific computing and provides many built-in functions for matrix manipulation, statistical computations, and plotting. The second one is the availability of the MPT toolbox [375], a free *MATLAB* toolbox that provides routines for polyhedral computations and easy-to-use interfaces for accessing various Linear Programming (LP) and quadratic programming solvers. All these tools play a prominent role in hybrid identification software and hence HIT has been linked with some of the functionalities of MPT. Currently, HIT is shipped as part of the most recent versions of MPT.

HIT has been designed for the reconstruction of PWA maps and PWARX models that, as described in ([Section 4.2](#)), provide input/output descriptions of PWA systems. Moreover, HIT implements the clustering-based procedures described in the papers [232, 233, 236]. At a more conceptual level, HIT performs the following basic tasks underlying the reconstruction of PWA maps composed by s affine submodels:

- the classification of the data points that is the attribution of each data point to a submodel;
- the reconstruction of the polytopic regions on which submodels are defined;
- the estimation of the parameter vectors for all submodels;
- an optional refinement of the results.

In addition, HIT provides facilities for plotting and validating the identified model.

In clustering-based procedures, classification is performed through a clustering algorithm. To this purpose, HIT provides two methods: one based on K-means [236] and one based on single-linkage clustering [232]. K-means is the most powerful method but requires the user to specify the number s of submodels composing the system. When s is not known, HIT offers the possibility to identify a number of models for different values of s and select the best one using cross-validation. Single-linkage clustering estimates automatically the number of modes on the basis of a parameter related to the minimal distance of parameter vectors characterizing different submodels. However, when this piece of information is not available a priori this method can not be applied.

Estimating the polyhedral regions amounts to find region boundaries that are given by hyperplanes. As discussed in (Section 4.2) this can be done using linear or piecewise linear classifiers. Methods based on linear classifiers are computationally less demanding, but also less accurate, than methods based on piecewise linear classifiers. In particular, linear classifiers produce regions that are not guaranteed to cover the whole regression set and HIT provides a routine for checking whether this happens or not. More in details, HIT offers the following three methods for region estimation, each one achieving a different balance between accuracy and computational complexity:

- *Multicategory Robust Linear Programming (MRLP)* [75] MRLP is based on piecewise linear classifiers and is the most precise algorithm. However, MRLP requires to solve a single linear program with a number of variables that scales linearly with the number of data points. Hence, for large data sets the procedure becomes computationally demanding.
- *Support Vector Machines (SVM)* [642] SVMs are linear classifiers that estimate a single boundary between two regions at each time. In HIT, each SVM problem requires the solution to an LP with a number of unknown that scales linearly with the number of data attributed to the two considered submodels. Hence, compared with MRLP, multiple LPs of smaller size are solved in order to estimate the regions. This makes possible to estimate regions even when the total number of data is too large for using MRLP.
- *Proximal SVM (PSVM)* [258] Differently from SVM, PSVMs are linear classifiers that do not require to solve an optimization problem. On the one hand, the computational burden of PSVM is much less than the one of SVM and hence PSVMs can be run even on very large data sets. On the other hand, one must bear in mind that the results of PSVM might be less accurate than those of SVM.

As explained in [233], once the identified model is available, it is possible to conduct an a posteriori analysis of the classified data points in order to improve the identification results. HIT offers the possibility of performing this step automatically, at the price of running the whole identification procedure one more time.

Finally, when the underlying model is nonlinear and smooth one can use HIT for getting a data-based piecewise linear approximation of the system. In this case, it might be desirable to obtain a continuous PWA map. To this aim, HIT offers the option of enforcing continuity constraints among submodels on the region boundaries during the estimation of parameter vectors.

10.4.2 Piecewise Affine System Identification Toolbox

The *Piecewise Affine System Identification Toolbox (PWAID)* is a *MATLAB* toolbox for identification of piecewise affine models that has been developed at the University of Linköping, Sweden and the University of Siena, Italy in the framework of the

Network of Excellence HYCON. The toolbox is free software, and is distributed at <http://www.dii.unisi.it/~paoletti/PWAID> under the terms of the GNU General Public License (version 2 or later) as published by the Free Software Foundation.

PWARX model structure PWAID provides functions for fitting to input/output data a piecewise affine ARX model (PWARX) of the following form:

$$y(t) = f(\phi(t)) + e(t), \quad (10.9)$$

$$f(\phi(t)) = \theta_i^T \begin{bmatrix} \phi_1(t) \\ \vdots \\ \phi_{n_s}(t) \end{bmatrix} \quad \text{if } \mathbf{H}_i \begin{bmatrix} \phi_1(t) \\ \vdots \\ \phi_{n_s}(t) \end{bmatrix} \preceq_{[i]} 0, \quad i = 1, \dots, n_s,$$

where $\mathbf{y} = (y_1, \dots, y_{n_y}) \in \mathbb{R}^{n_y}$ is the system output, $\mathbf{u} = (u_1, \dots, u_{n_u}) \in \mathbb{R}^{n_u}$ is the system input, $\mathbf{e} \in \mathbb{R}^{n_y}$ is the error term, and $\phi \in \mathbb{R}^{n_r}$ is the regression vector composed by past outputs and inputs as follows:

$$\begin{aligned} \phi(t) = & \begin{bmatrix} y_1(t-1) \dots y_1(t-n_a(1)) \\ y_2(t-1) \dots y_2(t-n_a(2)) \\ \vdots \\ y_{n_y}(t-1) \dots y_{n_y}(t-n_a(n_y)) \\ u_1(t-n_k(1)) \dots u_1(t-n_k(1)-n_b(1)+1) \\ u_2(t-n_k(2)) \dots u_2(t-n_k(2)-n_b(2)+1) \\ \vdots \\ u_{n_u}(t-n_k(n_u)) \dots u_{n_u}(t-n_k(n_u)-n_b(n_u)+1) \end{bmatrix}^T. \end{aligned} \quad (10.10)$$

In (10.10), $n_a(j)$ is the number of y_j lags in $\phi(t)$, $n_b(j)$ is the number of u_j lags in $\phi(t)$, and $n_k(j)$ is the delay for the first u_j term in $\phi(t)$. It turns out that $n_r = \sum_{j=1}^{n_y} n_a(j) + \sum_{j=1}^{n_u} n_b(j)$. Moreover, in (10.9), n_s is the number of submodels (or modes), $\theta_i \in \mathbb{R}^{(n_r+1) \times n_y}$ is the matrix of parameters of the i -th submodel, and $\mathbf{H}_i \in \mathbb{R}^{\mu_i \times (n_r+1)}$ is the matrix of coefficients of the μ_i linear inequalities defining the i -th polyhedral region \mathcal{X}_i . Note that $\preceq_{[i]}$ denotes a μ_i -dimensional vector whose elements can be the symbols \leq and $<$ in order to avoid that the regions \mathcal{X}_i overlap over common boundaries.

Hinging-hyperplane ARX models (HHARX) form a subclass of (10.9) for which the piecewise affine map f is continuous. Considering the j -th output, we have:

$$y_j(t) = f_j(\phi(t)) + e_j(t)$$

$$f_j(\phi(t)) = \vartheta_{j,0}^T \begin{bmatrix} \phi_1(t) \\ \vdots \\ \phi_{n_r}(t) \end{bmatrix} + \sum_{i=1}^M \sigma_{j,i} \max\{\vartheta_{j,i}^T \begin{bmatrix} \phi_1(t) \\ \vdots \\ \phi_{n_r}(t) \end{bmatrix}, 0\}, \quad (10.11)$$

where $\vartheta_{j,0}, \dots, \vartheta_{j,M} \in \mathbb{R}^{n_r+1}$ are parameter vectors, and $\sigma_{j,i} \in \{-1, 1\}$ are fixed a priori to allow for both convex and nonconvex functions. The number of submodels n_s of (10.11) is bounded by the quantity $\sum_{j=0}^{n_r} \binom{M}{j}$, which only depends on the length n_r of the regression vector, and the number M of hinge functions.

Identification algorithms implemented in PWAID PWAID implements two identification algorithms. The mixed-integer programming algorithm, presented in [559], is an algorithm for optimal identification of the HHARX models (10.11). Given the pairs $(\mathbf{y}(t), \phi(t))$, $t = 1, \dots, N$, the optimal model for the j -th output is selected by solving:

$$(\vartheta_{j,0}, \dots, \vartheta_{j,M}) = \arg \min_{\vartheta_0, \dots, \vartheta_M} \sum_{t=1}^N |y_j(t) - f_j(\phi(t))|^p, \quad (10.12)$$

where $p = 1$ or 2 . Assuming a priori known bounds on $\vartheta_0, \dots, \vartheta_M$ (which can be taken arbitrarily large), (10.12) can be reformulated as a mixed-integer linear or quadratic program that can then be solved for the global optimum. The optimality of the described approach comes at the cost of a theoretically very high worst-case computational complexity. For this reason, the mixed-integer programming algorithm is suitable in situations when it is of importance to get a model which is as good as possible and relatively few data are available (e.g. when it is very costly to obtain data).

The bounded-error algorithm, presented in [70], allows the user to impose a bound $\delta > 0$ on the norm of the identification error $e(t)$ in (10.9), and then searches for a PWARX model with minimum number of modes such that the following bounded-error condition is satisfied:

$$\|\mathbf{y}(t) - f(\phi(t))\|_\infty \leq \delta, \quad \forall t = 1, \dots, N, \quad (10.13)$$

where $\|\cdot\|_\infty$ is the infinity norm of a vector. The bounded-error algorithm can only guarantee suboptimal solutions, but can cope with large data sets. It is well suited when no prior knowledge on the physical system is available, and one needs to identify a model with a prescribed accuracy.

Description of the toolbox PWAID has been designed as a stand-alone toolbox. A minimum requirement is the availability of at least one solver for (mixed-integer) linear and quadratic programs. Given the input/output data, the user is only asked to select the model structure, choose the identification algorithm, and set the parameters of the selected algorithm, if any. The identified model is then automatically returned.

The main function of the toolbox is the following:

```
[model, status, details] = pwaid(y, u, modelstruct, algopts)
```

where \mathbf{y} is a matrix containing the output data, \mathbf{u} is a matrix containing the input data, `modelstruct` defines the model structure through the following fields (see also Table 10.1):

- `na`: number of \mathbf{y} lags in (10.10);
- `nb`: number of \mathbf{u} lags in (10.10);
- `nk`: delays for the first lag of each \mathbf{u} element in (10.10),

and `algopts` is a structure containing the identification algorithm chosen and the corresponding options. For instance, `algopts.name = 'hhoft'` selects

Table 10.1 Fields defining the PWARX model structure in PWAID.

<code>ns</code> (1×1)	number of submodels
<code>ny</code> (1×1)	number of outputs
<code>nu</code> (1×1)	number of inputs
<code>na</code> ($1 \times ny$)	vector of y lags
<code>nb</code> ($1 \times nu$)	vector of u lags
<code>nk</code> ($1 \times nu$)	vector of delays for the first u terms
<code>nr</code> (1×1)	dimension of the regression vector: $nr = \text{sum}(na) + \text{sum}(nb)$
<code>theta</code> ($1 \times ns$)	cell array of parameter matrices
<code>H</code> ($1 \times ns$)	cell array of region coefficients
<code>indSTR</code> ($1 \times ns$)	cell array of indexes of strict inequalities

the mixed-integer programming algorithm, and additional fields `algopts.Mp` and `algopts.Mm` fix the number of positive ($\sigma_{j,i} = 1$) and negative ($\sigma_{j,i} = -1$) hinges, respectively. On the other hand, `algopts.name = 'pwaid'` selects the bounded-error algorithm, and the additional field `algopts.delta` specifies the bound δ on the identification error. If no error occurs during the computation (i.e. `status = 1`), the output argument `model` contains the identified PWARX model. By default, the model is reported as a structure with the fields listed in Table 10.1. Note that

- `theta{i}` is the $(n_r + 1) \times n_y$ matrix of parameters of the i -th submodel;
- `H{i}` is the matrix of coefficients of the linear inequalities defining the i -th polyhedral region (`H{i}` has $n_r + 1$ columns);
- `indSTR{i}` contains the indexes of strict inequalities in the definition of the i -th polyhedral region.

However, the data interchange format developed within HYCON (cf. (Section 10.5)) is also supported to facilitate the data exchange with other tools (e.g. tools for analysis, verification and control of hybrid systems). If an error occurs during the computation, it is captured by the flag `status`, while the structure `details` provides additional information on the identification result.

PWAID provides also interfaces to several LP/MILP/MIQP solvers, and functions for model conversion. The software is compatible for joint use with the *Multi-Parametric Toolbox* ((Section 10.2)), the *Hybrid Toolbox* ((Section 10.3)), and the *Hybrid Identification Toolbox* ((Section 10.4.1)). Some functionalities of the mentioned toolboxes can be accessed directly from PWAID. More details can be found in the toolbox manual available with the software.

10.5 Data interchange format

10.5.1 Interchange architecture

In order to allow to exchange of models of hybrid systems among various tools and environments, we have implemented a three-layer model interchange architecture

consisting of a global readable format, a platform-specific format and a tool-specific format. The interchange architecture uses as the base model PWA systems because as discussed in [136, 305] several hybrid models can be automatically converted to PWA systems (cf. Chapter 4).

The interchange architecture based on the PWA model was chosen in order to allow the developer of each tool to obtain full integration by providing only two conversion functions. The developer only needs to implement a `fromPWA` function that converts from the common PWA model to its internal tool representation, and a `toPWA` function that converts from the internal tool representation to the common PWA model. The layered architecture allows the developer to write the conversion function in the platform-dependent language that is the language the developer is more familiar with. The conversion from the platform format to the global format and from the global format to the platform format is expected to be implemented by an expert in the platform language. Thus, if M platforms are given, each with n_i tools, $i = 1, \dots, M$, the proposed architecture requires in total $2M + 2 \sum_{i=1}^M n_i$ conversion functions, instead of $\left(\sum_{i=1}^M n_i\right)^2$.

The purpose of the global readable format is to pass model descriptions among different environments in a format which is easy to understand and to recognize. Therefore, we aim for the XML (eXtensible Markup Language) format, which allows data to be stored in a self-describing way. The XML format internally stores all data in a text file and adds markup elements which describe the type and logical interconnections of stored data.

The platform-specific format describes a given object in a form which is directly understandable by a given platform. We consider as platforms those environments that allow developer to build their own toolboxes, such as *MATLAB* and *Dymola*. In the case of *MATLAB*, the architecture-specific layer is represented by a *MATLAB* object, which can be viewed as a structure with various fields. The advantage of the architecture-specific layer is that it allows one to access various elements of the objects in a way which is convenient for a given platform. To convert objects from an architecture-specific format to the global readable format, the XML Toolbox for *MATLAB* can be used [239].

The tool-specific layer consists of an interface from individual tools to the architecture-specific format and vice-versa. The implementation of this layer is left to individual tools. Therefore its description is not included in this report. A typical work flow based on this three-layer structure as shown in Fig. 10.5 is as follows:

- the *origin* tool on platform P1 converts the tool-specific format into the platform-specific format of P1;
- a unified set of tools converts the platform-specific format of P1 into the global format;
- the global format is converted to the platform-specific format of P2;

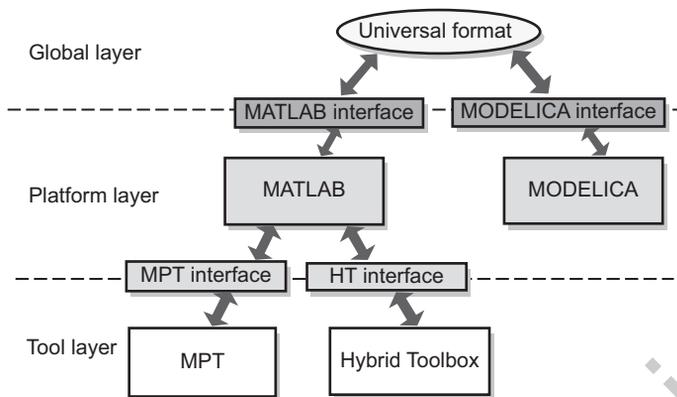


Fig. 10.5 Schematic of the interchange architecture.

- the *destination* tool on platform P2 converts from platform-specific format to its tool-specific format.

Obviously, only two steps are needed if the origin and destination tools are on the same platform, as for the conversion from the *Hybrid Toolbox* [57] to the Multi-Parametric Toolbox [375] schematized in Fig. 10.5.

10.5.2 Platform-independent format for dt-PWA systems

This section defines the information which a platform-independent format needs to store in order to fully describe piecewise affine systems in discrete-time (dtPWA). The description of dt-PWA systems consists of following blocks of information: System dynamics and guards, system constraints (for control design and optimization), information about nature of system inputs and system states, possible additional information.

We consider an extended model of a PWA system

$$x(k+1) = A_{i(k)}x(t) + B_{i(k)}u(t) + P_{i(k)}w(t) + f_{i(k)}, \quad (10.14a)$$

$$y(k) = C_{i(k)}x(t) + D_{i(k)}u(t) + Q_{i(k)}w(t) + g_{i(k)}, \quad (10.14b)$$

$$i(k) : H_{i(k)}x(t) + J_{i(k)}u(t) + M_{i(k)}t + R_{i(k)}w(t) \leq K_{i(k)}, \quad (10.14c)$$

where the terms $P_{i(k)}w(k)$ and $Q_{i(k)}w(k)$ represent the uncertainties in the state and output update equations, while the term $R_{i(k)}w(k)$ represents the uncertainty in the partitions, i.e., the possibility that the operation mode is not a deterministic function of the state and input vectors, as required for modeling linear hybrid automata [136].

A dt-PWA system as given by (10.14d) is defined by the following elements which will be stored in the interchange format: $A, B, C, D, f, g, H, J, M$, and K . Since some of the guardlines (10.14c) may be defined by strict inequalities, a vector t is introduced, whose elements describe the type of respective constraints. Matrices Q, P , and R modeling the uncertainties are stored as well. In addition, as an option, various constraints can be defined for PWA systems for control design, constraints on states, inputs, and outputs in the form $\alpha_{\min} \leq \alpha(k) \leq \alpha_{\max}$; slew-rate constraints on states, inputs, and outputs in the form $\Delta\alpha_{\min} \leq \alpha(k) - \alpha(k-1) \leq \Delta\alpha_{\max}$. To fully describe the behavior of PWA systems, information about the nature (i.e. continuous/discrete) of system inputs, states and outputs has to be kept as well. This are described in the additional fields `inputtype`, `statetype`, or `outputtype`, respectively.

10.5.3 XML scheme of the platform-independent format for PWA systems

XML documents are capable of encapsulating arbitrary data in a self-describing way. The description is composed of so-called tags, which are special sequences of characters which denote the type of the data described. The data itself is then provided in a textual representation which is inserted into the tags. The XML format used to describe PWA models has the following form:

```

<pwa>
  <ident>$string$</ident>
  <field>
    <ident>$string$</ident>
    <matrix>
      <row>
        <column>$float$</column>
      </row>
    </matrix>
  </field>
  <dimensions>
    <state>$integer$</state>
    <input>$integer$</input>
    <output>$integer$</output>
  </dimensions>
  <inputtype>$matrix$</inputtype>
  <statetype>$matrix$</statetype>
  <outputtype>$matrix$</outputtype>
</pwa>
    
```

Here, `pwa` denotes the root element of the XML file. The `ident` tag describes the name of the system, provided as a string argument (type `$string$`). The PWA system is composed of several field, where each `field` connects the corresponding

variable name (e.g. A_i) to its numerical value. The value is provided as a matrix, which is further composed of several rows and columns. The values in each row-column pair are given as floating point numbers (type `$float$`). In addition, the `dimensions` tag encapsulates the dimensions of the state, input, and output vectors, respectively. The value of each such parameter is provided as an integer.

Finally, the `inputtype`, `statetype`, and `outputtype` tags define, respectively, the hybrid nature of system inputs, states, and outputs. Such information is stored by means of the `$matrix$` datatype, which is a shortcut to the `<matrix><row><column></column></row></matrix>` syntax. We allow three possible types of variables:

1. If variable i is Boolean, then `variabletype{i} = [0, 1]`.
2. If variable i is from a finite alphabet, then `variabletype{i} = [a1, a2, ..., am]` enumerates all possible values which the given signal can take.
3. If variable i is continuous, then `variabletype{i} = [-Inf, Inf]`.

The right-hand-side values are further encoded as matrices using the `matrix`, `row`, and `column` tags as described above.

10.5.4 Implementation of the architecture-specific format in MATLAB

In the *MATLAB* environment the architecture-specific format is implemented as an object which can be viewed as an abstraction of structures with various fields. Each element of the platform-independent format described earlier is stored in one field of the object. Because of this, it is easy for individual tools to read data from, or write data to the object directly, without the need of implementing a conversion from/to the platform-independent format in every single tool.

The object-oriented implementation of the interchange format in *MATLAB* allows one to easily import, validate, and export PWA models into the XML representation. To import a PWA model from a given XML file, the user can call the constructor function as follows

```
>> obj = hycpwa('filename.xml');
```

where `filename.xml` is the name of the XML file. The `hycpwa` constructor will automatically parse the XML file and create an internal representation of the PWA model. The user can now change the model, e.g. by altering the numerical data of certain fields. The following command will set the type of the 1st control input to binary:

```
>> obj.inputtype{1} = [0, 1]
```

After each user-invoked modification, the object automatically validates all modifications and prompts the user if he/she entered invalid data. Once the object has been modified according to the user's needs, he can then export the modified PWA model back to an XML representation. This is achieved by calling the `xml()` method as follows:

```
>> xml(obj, 'newfilename.xml')
```

11

Modeling, simulation, and optimization environments

C. Sonntag

This chapter gives an overview of tools and environments for the modeling, simulation, and optimization of hybrid systems. These tasks are based on different modeling formalisms some of which have already found their way to applications.

Chapter contents

11.1	Introduction and overview	page 326
11.2	Modeling and simulation tools and environments	329
11.2.1	Overview	329
11.2.2	Numerical simulation of hybrid systems	329
11.2.3	<i>MATLAB, Simulink, Stateflow, and SimEvents</i>	332
11.2.4	<i>gPROMS</i>	338
11.2.5	Other tools and environments	346
11.3	Dynamic optimization	350
11.3.1	Optimization problem	350
11.3.2	Numerical optimization of nonlinear hybrid systems	351
11.3.3	Tools for the optimization of nonlinear hybrid systems	353

11.1 Introduction and overview

Although the techniques and tools for the algorithmic analysis and design of hybrid systems that have been presented in the previous chapters have already been applied successfully to a variety of industrial case studies, to this day the dominant industrial tool for computer-based system analysis and design is simulation. The main reason for this lies in the large complexity of many sophisticated technological systems such as cars or chemical plants – an accurate model of a large chemical plant often consists of tens of thousands of nonlinear equations. In addition, such systems may contain hundreds of low-level continuous and logic-based controllers that ensure efficient operation or system safety, and that implement sequential procedures such as production recipes, start-up, or shut-down. For such large-scale hybrid systems, sufficiently accurate piecewise linear or affine abstractions or approximations can often not be determined, or the resulting models are too complex for the application of the techniques described above.

In the last decades, a large number of modeling and simulation tools and environments for hybrid systems have been developed, ranging from rather prototypical academic tools that mostly serve as test beds for hybrid systems research to integrated modeling and simulation environments that are capable of the real-time simulation of highly complex models with tens or even hundreds of thousands of equations using modern computing hardware. However, advanced simulation capabilities are only one requirement that an industrially successful modeling and simulation environment must fulfill. Additional requirements are:

- *Support of hierarchy, modularity, and re-usability:* Most complex technological systems can be decomposed into smaller and simpler sub-components which, in turn, may again be decomposable into smaller components. The most comfortable (and the least error-prone) way to model such systems is *compositional* or *object-oriented modeling*: first, the modeler designs (relatively simple) models of the basic sub-components of the system. The dynamic equations of the sub-components are encapsulated within these model components, and the interaction with the environment and other model components occurs only via well-defined interfaces. Subsequently, the modeler interconnects these basic model components into larger sub-components of the overall system which, in turn, can be interconnected with other components until the overall system model is obtained. This approach has several advantages:
 - the correctness of the basic components can be verified independently;
 - the model structure can be chosen to reflect the physical structure of the system which simplifies the modeling process considerably;
 - the model components can be assembled in model libraries so that they can be re-used in different models and settings. Many of the modeling environments presented in this chapter even provide such libraries that contain large varieties of pre-defined model components for different application domains.

- Usability:** A modeling environment should offer an intuitive user interface that supports the user in the modeling process. This might include an intuitive but powerful modeling language as well as graphical editors for model composition.
- Expressiveness:** Although ordinary differential equations (ODEs) can be used to model the continuous dynamics of most technological systems accurately, they often do not allow for a comfortable and “natural” description of the underlying physical properties. To simplify the modeling process for the user, an environment should support the expression of model behaviors with algebraic constraints, e.g. to comfortably describe the interconnection of different model components or the thermodynamic properties of a chemical compound (see, e.g., Section 14.2). This approach results in continuous dynamics that are described by systems of differential-algebraic equations (DAEs). In particular in the process industries, it is advantageous to support even more complex model structures, such as integral equations or partial differential equations which leads to systems of integral-partial-differential-algebraic equations (IPDAEs) that allow to define continuous dynamics that vary not only with time, but also in non-temporal domains (e.g. along the length of a reactor).
- Model integration:** While some modeling environments have been designed for multi-domain modeling, i.e. the creation of heterogeneous models from physical components of different domains (e.g. mechanics, electronics, biology, thermodynamics, or control), many environments were developed for a specific domain and possess specific strengths in this domain. To create models of complex heterogeneous systems, it is thus often necessary to allow for co-simulation of models (i.e. the synchronized simulation of distinct model components that have been created in different modeling environments). In addition, co-simulation capabilities are necessary to enable hardware-in-the-loop simulation that is often employed in industry. In this setting, a part of the system (e.g. an embedded controller) that is available as a hardware prototype is tested in real time with a simulation model of the remainder of the system.

Several of the modeling environments presented in this chapter provide facilities to integrate and export models from/to other modeling environments, mostly from/to *MATLAB/Simulink* which is currently the de-facto industrial standard environment. In addition, several initiatives exist that aim at the standardization of modeling formats for co-simulation, such as the CAPE-OPEN initiative [180]. An alternative approach to achieve model interoperability is to implement translations between different modeling formalisms. This has been an integral part of the research within HYCON (see, e.g., Chapters 10 and 12) and is an essential part of the emerging field of *computer-automated multi-paradigm modeling* [478].

Although the dominant industrial tool is simulation, technologies for the dynamic optimization of nonlinear continuous and hybrid systems have made an increasing impact in industrial engineering in recent years. The industrial interest stems

mainly from the potential for the maximization of the economic benefit, from the significant improvements of the optimization approaches and algorithms in the last years, and from the increase of the available computing power that today enables the optimization of large and complex systems. Dynamic optimization is employed to solve a variety of tasks, such as optimal process design [241], open-loop design of transition procedures such as start-up or shut-down (Section 14.4), safety analysis (Section 14.2), optimal closed-loop control (Section 14.4), or state and parameter estimation of dynamic models from real-world data.

In principle, the requirements on dynamic optimization tools are similar to those for modeling and simulation environments, such as the support of systems with rich nonlinear continuous dynamics and the provision of intuitive user interfaces. In comparison to the large number of tools and environments that is available for the modeling and the simulation of hybrid systems, the tool base for dynamic optimization is still rather small. The main reason for this is that many of the optimization approaches that are suitable for large-scale optimization have only been developed very recently – the ongoing research effort on dynamic optimization techniques will most likely lead to further significant advances in the coming years. Since in general, the most time-consuming task in the formulation of a dynamic optimization problem is the generation of the dynamic (simulation) model, most of the optimization tools that are presented here employ the powerful modeling facilities that are offered by the available modeling and simulation environments.

The remainder of this chapter is organized as follows: Section 11.2 provides an overview of the capabilities of more than 20 different environments and tools for hybrid systems modeling and simulation. Although the selection of tools is by no means exhaustive which is not possible within the scope of this chapter due to the sheer number of different modeling approaches and tools, the overview covers many different modeling formalisms and areas of hybrid systems simulation. After a general classification of the considered tools and environments in Section 11.2.1, Section 11.2.2 introduces the concepts and challenges of hybrid systems simulation to enable the reader to better judge and compare the capabilities of the presented tools and environments. The overview focuses on the illustration of the main concepts employed by modern advanced modeling and simulation environments that fulfill all or most of the requirements described above. The concepts of the two currently dominating modeling approaches, the causal block-diagram-based approach and the noncausal equation-oriented approach, are illustrated with the implementation of the two-tank example (Section 1.3.1) in the tool sets *MATLAB/Simulink/Stateflow* (Section 11.2.3) and *gPROMS* (Section 11.2.4). In Section 11.2.5, the capabilities and concepts of a large variety of other tools and environments are described.

Subsequently, a detailed overview of some of the available techniques and tools for the optimization of hybrid systems is given in Section 11.3. The chapter concludes with a comprehensive list of references and online resources that provides pointers to further reading material on the concepts of hybrid systems modeling and simulation as well as to several extensive and detailed surveys on tools for hybrid systems. Finally, a list of online resources and texts is given that provides more detailed information and modeling examples for the tools presented in this chapter.

11.2 Modeling and simulation tools and environments

11.2.1 Overview

Table 11.1 gives an overview of the environments and tools that are presented in this section. Many of the environments are based on the object-oriented modeling paradigm introduced in [220], which advocates the creation of structured and reusable model components with concepts such as encapsulation, modularity, hierarchy, and instantiation of model templates or classes.

The tools can be classified into three groups: tools that are based on the extension of finite-state formalisms with continuous dynamics (see Chapter 2), such as *UPPAAL*, χ (Chi), or *SHIFT*, tools that are inherently continuous and have been extended with finite-state concepts, such as *MATLAB/Simulink/Stateflow*, *20-sim*, or *Scilab/Scicos*, and tools and languages that were designed specifically for hybrid systems, such as *Modelica*, *gPROMS*, *ABACUSS II*, *Omola/Omsim*, or *BaSiP*.

While some of the tools have clear and formally defined execution semantics, the semantics of other tools (such as *Simulink* and *Stateflow*) are not clearly defined, are hidden within the simulation engine, and may vary with configuration parameters of the engine. A dangerous error source is the execution priority in block-diagram-based simulation models. It is easy to construct block diagrams in which the result value depends on the order of execution of the blocks. Thus, if the execution priority is not clearly defined or, in the worst case, depends on the position of the blocks within the window of the graphical editor, a model may compute unexpected results. Such errors are difficult to track, particularly if the semantical behavior of the simulation engine is not known in detail.

11.2.2 Numerical simulation of hybrid systems

The numerical simulation of hybrid systems poses two major challenges: the numerical integration of possibly very complex systems of DAEs, and the synchronized execution of the continuous and discrete model dynamics. This section presents some of the requirements that hybrid simulators must fulfill and some of the problems that are introduced by the hybrid dynamics.

Most numerical simulation tools for hybrid systems employ one of the following two modeling approaches:

1. *Noncausal modeling*: In the noncausal (or equation-oriented, implicit, declarative) modeling approach, the causality (i.e. the input/output relation) of the model components is not defined a priori. This approach is advantageous if a physical system is modeled because most physical laws are inherently implicit. For example, a resistor can be modeled with Ohm's law as $u = R \cdot i$, where u , R , and i are the voltage, the resistance, and the electrical current. This equation is declarative because it only defines a relation between u , R , and i , but it does not specify which variable causes a change of the other variables. In general, implicit systems of DAEs can be described as $0 = f(\mathbf{x}(t), \dot{\mathbf{x}}(t), \mathbf{y}(t), \mathbf{u}(t), \mathbf{v}(t), \mathbf{p})$.

Table 11.1 Overview of the tools and environments described in this section.

Name	Brief description
<i>20-sim</i>	Bond-graph-based graphical modeling and simulation environment.
<i>ABACUSS II</i>	Object-oriented noncausal modeling and simulation environment for hybrid systems with continuous IPDAE dynamics.
<i>BaSiP</i>	Modeling and simulation environment for recipe-driven chemical batch processes.
<i>Charon</i>	Graphical modeling, simulation, and verification tool for the modular specification of interacting hybrid systems.
<i>CheckMate</i>	<i>MATLAB/Simulink</i> -based toolbox for the modeling, simulation, and verification of nonlinear hybrid systems.
χ (Chi)	Modular modeling formalism and tool set for general hybrid systems based on process algebra.
<i>EcosimPro</i>	Object-oriented noncausal modeling and simulation environment for hybrid systems with DAE dynamics.
<i>gPROMS</i>	Object-oriented noncausal modeling and simulation environment for hybrid systems with continuous IPDAE dynamics.
<i>HyBrSim</i>	Modeling and simulation tool for hybrid bond graphs.
<i>Jacobian</i>	Commercial offspring of <i>ABACUSS II</i> .
<i>Modelica</i>	Object-oriented multi-domain noncausal modeling language for hybrid systems with DAE dynamics. The tools <i>Dymola</i> , <i>OpenModelica</i> , and <i>MathModelica</i> are based on the <i>Modelica</i> language.
<i>Omola/Omsim</i>	Object-oriented noncausal modeling and simulation environment for hybrid systems with continuous semi-explicit DAE dynamics.
<i>Ptolemy II/HyVisual</i>	Graphical modeling and simulation environment for hybrid and embedded systems.
<i>Scilab/Scicos</i>	Open-source alternative of the <i>MATLAB/Simulink</i> tool set.
<i>SHIFT</i>	Modeling and simulation environment for dynamically changing networks of hybrid automata.
<i>Siconos</i>	Tool for the modeling, simulation, analysis, and controller synthesis of nonsmooth dynamic systems (NSDS).
<i>SimEvents</i>	<i>MATLAB/Simulink</i> -based tool for discrete-event systems.
<i>Simulink</i>	<i>MATLAB</i> -based graphical multi-domain modeling environment for nonlinear dynamic systems.
<i>Stateflow</i>	<i>Simulink</i> -based modeling environment for discrete-event reactive systems that implements an extended version of the Statecharts formalism.
<i>UPPAAL</i>	Integrated graphical environment for modeling, simulation, verification, and design of real-time systems.

Here, $\mathbf{x}(t)$ are the state variables and $\dot{\mathbf{x}}(t)$ are the corresponding time derivatives, $\mathbf{y}(t)$ are the algebraic variables, $\mathbf{u}(t)$ and $\mathbf{v}(t)$ are the continuous and the discrete inputs, and \mathbf{p} are time-invariant parameters. An important subclass of implicit DAEs are semi-explicit DAEs in which the equations are solved explicitly for the time derivatives of the state variables. Semi-explicit DAEs are of the form $\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{y}(t), \mathbf{u}(t), \mathbf{v}(t), \mathbf{p})$ with the algebraic constraints $0 = g(\mathbf{x}(t), \mathbf{y}(t), \mathbf{u}(t), \mathbf{v}(t), \mathbf{p})$. Note that for simulation purposes, partial or integral equations must be discretized (either automatically or manually) along the non-temporal domains. The resulting equations are included into the overall DAE system. Most noncausal modeling environments employ a textual modeling language in which the systems of DAEs can be implemented directly.

2. *Causal modeling*: In the causal (or explicit, imperative) modeling approach, the causality, i.e. the input/output relation, of the model components is clearly defined. Modeling environments that implement this approach usually employ a graphical modeling language in which the model components are represented by blocks that are interconnected by signal paths. Causal models are advantageous if the modeled behavior is an explicit objective of the model design, e.g. if the model represents a control system.

In a noncausal model, all of the implicit continuous model equations must be fulfilled at all times. The numerical simulation of the resulting (often very large) DAE system may require one or several symbolic pre-processing steps:

- The integration of DAE systems involves the simultaneous solution of large systems of nonlinear equations which is very time-consuming. The computation time can be reduced drastically by a symbolic reformulation of the DAE system such that all or most of the model variables can be computed explicitly. This reformulation is called *BLT partitioning*. The corresponding algorithms attempt to reformulate equations and to reorganize variables such that the incidence matrix (i.e. the matrix that indicates which variables appear in which equations) of the system equations is transformed into a block-lower-triangular (BLT) form. Often, this reformulation yields that subsets of the equations cannot be solved explicitly for the variables. These subsets are called *algebraic loops* and must be solved simultaneously. To reduce the size of these equation systems and thus speed up the solution, a variable-substitution technique called *tearing* can be applied.
- Standard numerical DAE solvers, such as, e.g., *DASSL* [518], can only solve DAE systems that have a *differential index* of 1 or less. The differential index indicates how often the algebraic equations (or parts thereof) have to be differentiated with respect to time until an explicit ODE system can be obtained by algebraic reformulation (note that several other definitions of the index of a DAE exist, e.g. see [141]). General *high-index* DAEs, i.e. DAEs with an index that is larger than 1, contain hidden constraints that restrict the initial conditions of the DAE, and such DAEs cannot be solved directly by the existing numerical DAE solvers. To simulate high-index DAEs, symbolic index reduction

techniques based on, e.g., the *Pantelides algorithm* [501] and the *dummy derivative method* [443] must be applied to derive a DAE of index 1.

If hybrid elements are added to the simulation model, such as conditional equations, discontinuous resets of the state variable values, or even full-blown discrete-event systems that run concurrently with the continuous model, the situation becomes even more involved. In hybrid systems, the continuous-time dynamics usually interact with the discrete components via events that are often triggered if some continuous quantity crosses a threshold value (e.g. if a bouncing ball hits the ground, see Chapter 2).

A hybrid simulation engine must be able not only to detect, but also to exactly localize the time of occurrence of such events, e.g. by backtracking the continuous trajectory if the threshold was crossed. While a discontinuous reset of a state variable in an explicit ODE system is not problematic since the new value does not affect the values of the remaining state variables directly, the situation is much more difficult if a state or an algebraic variable is reset in a DAE system. In DAEs, a reset of a variable usually leads to values that do not fulfill the algebraic constraints. Thus, the hybrid simulator must support the automatic computation of consistent initial variable values after a discontinuity which is a non-trivial task. In addition, a simulator should be able to perform event iteration, i.e. to handle several events that occur at the same point in time. This phenomenon may for example occur if the action that is associated with the first event immediately triggers a second event.

Zeno behavior (see Chapter 2) can lead to adverse effects during numerical simulation. For clarification, consider again the bouncing ball example. If the value with which the vertical acceleration is reset becomes smaller than the numerical tolerance of the simulator, the reset may be missed and, in consequence, the ball falls below the floor (see [143] for examples of this phenomenon). However, systems that exhibit Zeno behavior can often be reformulated. For example, if the bouncing ball model is extended with a second discrete location that represents the case $x_1 = 0$, $\dot{x}_1 = 0$, and the transition conditions are changed accordingly, the Zeno behavior is removed. While *chattering* does not lead to such drastic effects, it severely reduces the simulation performance. However, since numerical simulation engines operate with nonzero time steps, the simulation often actually progresses in time.

While all of the tools presented in this chapter support the basic concepts of hybrid systems simulation such as state event detection, tools that offer facilities to deal with chattering or Zeno behavior are extremely rare, particularly since these phenomena are still subject to research.

11.2.3 MATLAB, Simulink, Stateflow, and SimEvents

The interactive tools *Simulink* and *Stateflow* are part of the very popular technical computing environment *MATLAB* (*Matrix Laboratory*) that provides powerful and easy-to-use facilities for technical programming, computation, and visualization. The *MATLAB* tool set is currently the academic and industrial standard for the model-based design and analysis of technical systems in many engineering domains. This

dominance is also reflected in this book (many of the presented tools and techniques have been implemented in *MATLAB* or *Simulink*, e.g. see [Chapter 10](#) and [14–17](#)) and is responsible for the large number of tools that provide interfaces to *MATLAB/Simulink*. Another major strength of the *MATLAB/Simulink* tool set is the availability of a large number of commercial and free tool boxes and model libraries that extend the basic tool set with specialized functionalities for e.g. optimization, distributed computing, symbolic computation, mechanics, control, signal and image processing, biology, financial mathematics, statistics, or real-time code generation.

Simulink is a tool for the modeling and simulation of nonlinear dynamic systems from different domains. It employs a graphical, block-diagram-based modeling language. An outstanding *Simulink* feature is the possibility to combine continuous-time and discrete-time components (even with different sampling rates) within a single model. With *Stateflow*, complex hierarchical state machines and flow diagrams can be developed in a graphical environment. The modeling formalism of *Stateflow* is an extended variant of *Statecharts* [295]. *Stateflow* blocks can be integrated into *Simulink* models as subsystems. Thus, hybrid systems can be modeled by combining *Simulink* and *Stateflow* blocks. It is also possible to model hybrid systems with very simple continuous dynamics directly in *Stateflow*.

A *Simulink* model is assembled by the interconnection of blocks from the (extendible) *Simulink* model library. The basic *Simulink* blocks are classified in different categories, such as signal sources, sinks, and routing blocks, basic continuous-time and discrete-time models (e.g. transfer functions and integrators), blocks that provide logic and mathematical operations and look-up tables, and blocks that implement more advanced functionalities such as linearization. Since *Simulink* employs a causal modeling approach, it can only simulate models in which the computational causality is clearly defined, i.e. which do not contain algebraic loops. However, *Simulink* provides an *Algebraic Constraint* block that can be used to resolve algebraic loops.

One of the reasons for the large popularity of *Simulink* is its *s-function* interface. This interface defines which functionality must be implemented by external software components so that they can be integrated into *Simulink* models. The *s-function* interface enables the integration of a large variety of third-party components, e.g. models that were created in other environments such as *gPROMS* or the *Modelica*-based tool *Dymola* (see below). In addition, arbitrary *MATLAB* functions can be invoked in *Simulink* models so that the complete *MATLAB* functionality is available.

Example 11.1 Two-tank system

In the following, the basic concepts of hybrid systems modeling with *Simulink* and *Stateflow* are illustrated using the two-tank example that is described in [Section 1.3.1](#). Like many other object-oriented modeling environments, *Simulink* offers the possibility to define basic model components that can be interconnected into larger components. These components can be assembled in user-defined model libraries so that they can be re-used. The first modeling step for the two-tank system could be to create a dynamic model of a single tank. Such a model is shown in [Fig. 11.1](#). This model implements the following differential equation:

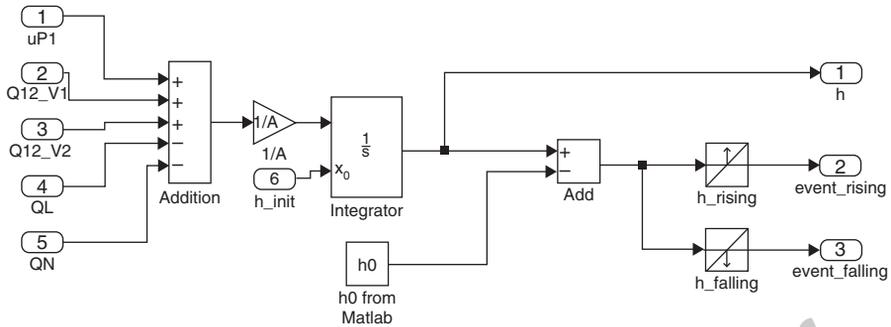


Fig. 11.1 Generic *Simulink* model of a tank of the two-tank system.

$$\dot{h}(t) = \frac{u_{P1}(t) + Q_{12}^{V1}(t) + Q_{12}^{V2}(t) - Q_L(t) - Q_N(t)}{A}. \quad (11.1)$$

Note that this equation is a generalized version of the differential equations given in Section 1.3.1 and that it can represent the volume flow equations of both tanks. During simulation, the model is evaluated from left to right. The five blocks on the leftmost side of Fig. 11.1 are the *input ports* of the model, and the three blocks on the rightmost side are the *output ports*. The central part of the model is the *integrator* block that integrates the signal that enters at the upper left port (\dot{h}). The right-hand side of (11.1) is implemented by the addition block and the triangular multiplier block, and the integrator is initialized at the start of the simulation using a sixth input port (h_{init}).

An essential component for the modeling of hybrid systems in *Simulink* is the *hit crossing* block, which detects and localizes a zero crossing of its input signal. This block indicates a zero crossing by the change of its output value from 0 to 1. The block can be configured to detect zero crossings from below, from above, or in both directions. The configuration is indicated by the arrows in the blocks in Fig. 11.1. Before state event detection, the constant height h_0 of the valve V_1 at which the continuous dynamics of the system changes must be subtracted from h . This value is obtained directly from a variable h_0 in the *MATLAB* workspace. This approach is very convenient since all constant parameters of a *Simulink* model can be defined by simply running a *MATLAB* script before the simulation.

The overall model of the two-tank system can now be constructed hierarchically by including two instances of the tank model as submodels, i.e. as single blocks, as is shown in Fig. 11.2. The overall model contains two additional subsystems, a *Stateflow* chart, which models the discrete dynamics, and another (algebraic) submodel, which computes the flow rates in the system based on the current levels in the tanks, the values of the discrete inputs u_1, u_2, u_3, d_1 , and d_2 , which are obtained from the *MATLAB* workspace, and the current state of the system, which is obtained from the *Stateflow* chart. Note that *MATLAB* variables that are used as inputs in *Simulink* models do not have to be constant, but can also define varying time trajectories of the input variables.

Since *Stateflow* charts are subsystems of *Simulink* models, their execution is controlled by the simulation engine of *Simulink*. *Stateflow* charts can communicate with the parent *Simulink* model either via data sharing or via events. Events are also used to trigger the execution of a *Stateflow* chart.

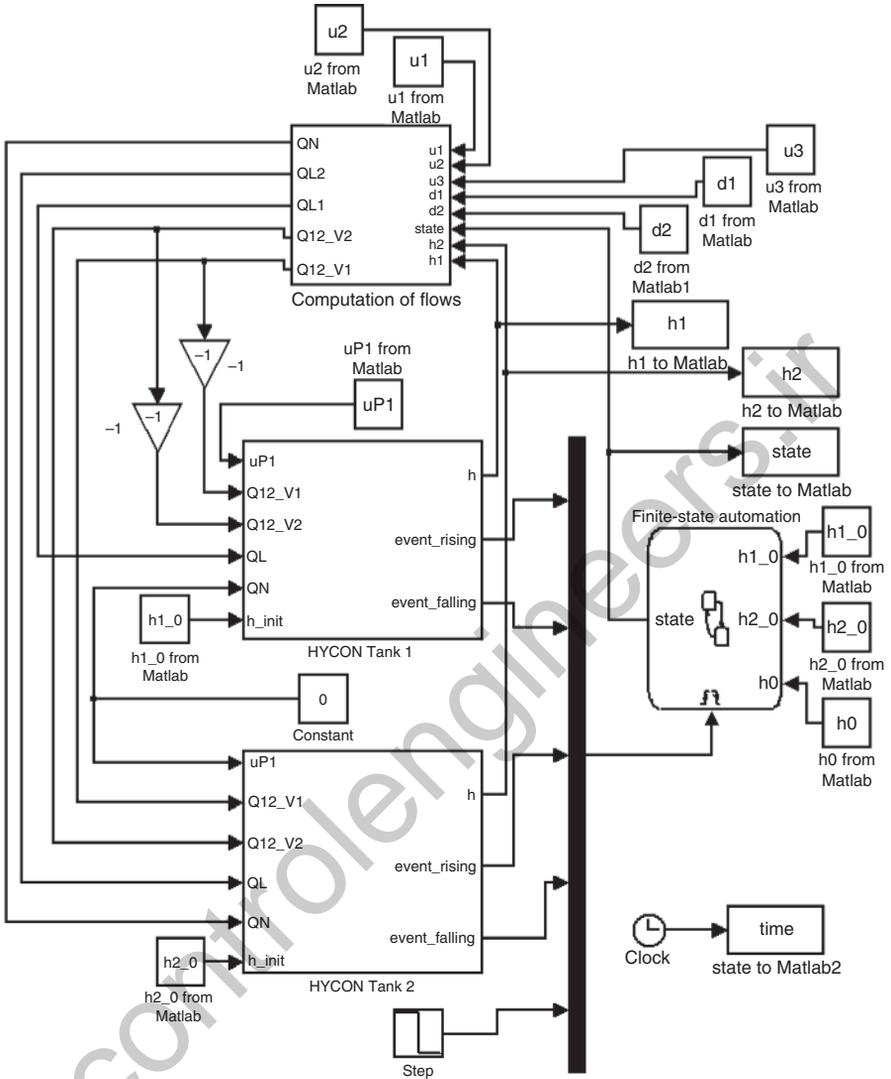


Fig. 11.2 The hierarchical *Simulink* model of the two-tank system.

The *Stateflow* implementation of the finite-state automaton that models the discrete behavior of the two-tank system is shown in Fig. 11.3. The automaton has been implemented in the Moore syntax, i.e. the active state uniquely determines the output value *state*. The interface to *Simulink* is implemented using events: the four output signals of the hit crossing blocks of the tank submodels are multiplexed into a single multidimensional signal that is transmitted to the *trigger port* of the *Stateflow* chart. Within the chart, the events *h1_rising*,

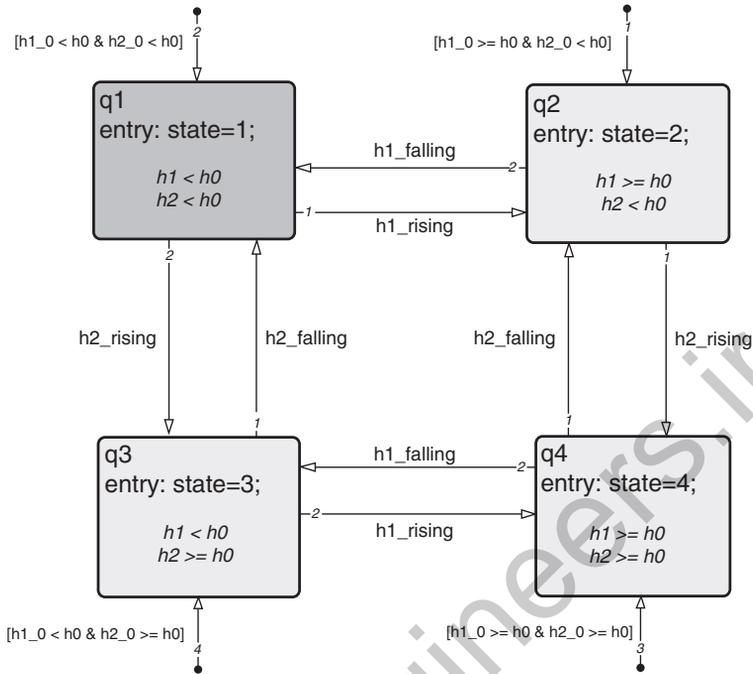


Fig. 11.3 Finite-state automaton of the two-tank system in *Stateflow*.

$h1_falling$, $h2_rising$, and $h2_falling$ are triggered if the corresponding external signal has a rising edge (i.e. changes from 0 to 1). These events achieve two feats: they trigger the execution of the chart, and they serve as transition conditions. If a transition is enabled, it is taken immediately. Upon entering a new discrete state, the chart executes an *entry action* that assigns the correct value to the output variable *state*. To correctly initialize the chart at the start of the simulation, each of the four states is equipped with a *default transition*. The conditions of these transitions are defined as logical expressions on the initial values of the tank levels and of the height h_0 that are passed to the chart as input variables. To enforce an initial execution of the chart, a fifth event is defined that is triggered by a step signal in the parent *Simulink* model (Fig. 11.2).

Only a very small subset of the *Stateflow* modeling formalism is necessary to implement this finite-state automaton. In general, *Stateflow* charts support all concepts of state charts such as superstates, AND and OR states, and history junctions. Furthermore, *Stateflow* supports several concepts that go beyond state charts, such as (simple) continuous dynamics, the inclusion of *MATLAB* functions or flow diagrams, and the definition of complex actions that can be assigned to both, the discrete states and the transitions of a chart.

The *Simulink* subsystem that models the flow rates within the two-tank system is shown in Fig. 11.4. In this model, the value of the input variable *state* is translated into the flow rates that result in the corresponding discrete mode using the *Multiport switch* block. The first input of this block serves as an index that determines which of the input signals is transmitted to the only output port. Since the representation of complex equations with the

basic *Simulink* blocks is tedious and leads to large block diagrams, the flow rate equations are implemented in *MATLAB* functions that are invoked within the model via the *MATLAB Fcn* block. □

Although *Stateflow* offers a very powerful modeling formalism for discrete-event reactive systems, it suffers from the drawback that it is integrated into the *Simulink* simulation engine which is based in the continuous-time semantic domain. Thus, *Stateflow* charts are essentially executed in a time-driven fashion. This can lead to adverse effects, such as the dependence on the time step size of the *Simulink* engine or the possibly inaccurate execution of several events that are supposed to occur at the same point in time. If, for example, the *Stateflow* chart resets the value of an *Integrator* block in *Simulink*, the *Simulink* engine performs a simulation step even if events are still waiting to be executed at the same point in time. For an illustrative example, refer to [143].

The recently introduced *MATLAB*-based toolbox *SimEvents* implements a “true” discrete-event model of computation that is independent of the *Simulink* simulation engine. With *SimEvents*, activity-based models can be developed to evaluate system parameters such as congestion, resource contention, and processing delays. *SimEvents* does not replace *Stateflow* since it was developed with a different goal in mind, but it can be used to trigger the execution of *Stateflow* at accurate event times.

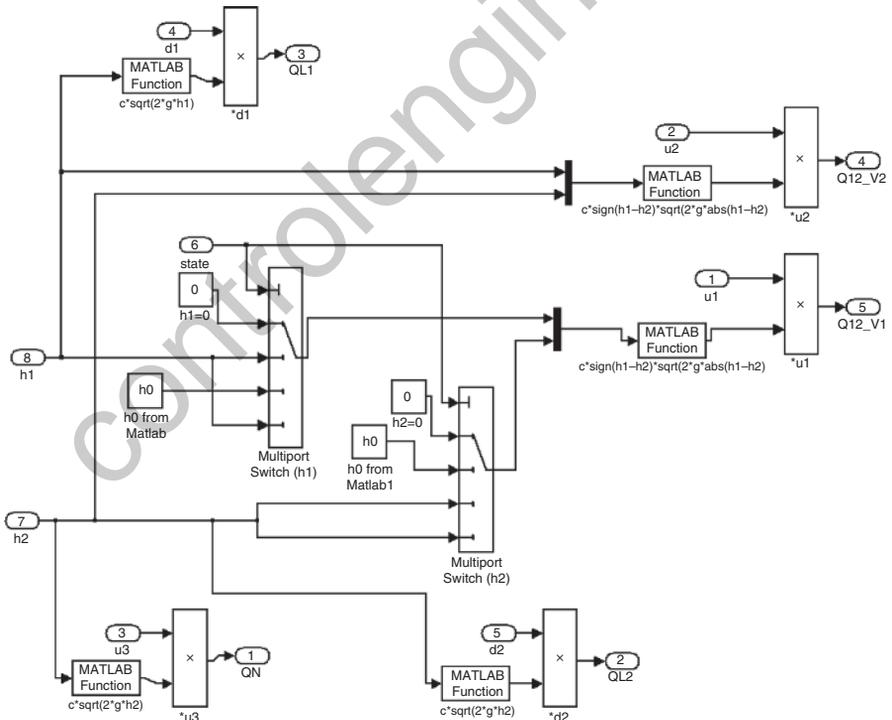


Fig. 11.4 *Simulink* subsystem for the computation of the flow rates.

The two-tank example has shown that *Simulink* models of physical systems can become very large and complicated even for simple systems. The following section shows that the noncausal modeling approach is suited much better for this task.

11.2.4 *gPROMS*

The object-oriented noncausal modeling and simulation environment *gPROMS* (*General PROcess Modeling System*) is the successor of the simulation tool *SPEEDUP* [502] and was one of the first simulation environments for general hybrid systems. It was developed with a focus on the simulation and optimization of large-scale processing systems that are modeled by (possibly discontinuous) IPDAEs (although it can in principle be used to model general large-scale hybrid systems from different domains). The textual *gPROMS* modeling language is type-safe and supports the definition of hybrid models via language elements that allow for, e.g., conditional equations or resets of state variables at discrete event times. For model composition, a graphical flow sheet editor is included, and the *gPROMS Results Management Service* (*gRMS*) provides facilities for the visualization of simulation results.

gPROMS contains an advanced engine for the symbolic preprocessing of the simulation model (which does, however, not support automatic index reduction of high-index DAE models). An outstanding feature of this engine is the support for the automatic discretization of partial differential equations so that they can be solved with standard numerical DAE solvers. Several discretization techniques are available, such as the approximation by finite differences and orthogonal collocation on finite elements. *gPROMS* comes with the *Process Model Library* (*PML*) that provides models of basic components of many processing systems, such as reactors, heat exchangers, or separation stages. Furthermore, *gPROMS* simulation models can be integrated into CAPE-OPEN-compliant tools and into *Simulink* models.

In the *gPROMS* framework, model components with autonomous switching of the continuous dynamics are represented as *MODEL* entities while sequential procedures or components that exhibit forced (i.e. external) switching are represented as *TASK* entities. This distinction reflects the structure of most processing systems: While the physical behavior of process components often exhibits only autonomous switching that stems from discontinuities in the physicochemical behavior (such as a phase change of a chemical compound), external switching is usually introduced by the interaction of the process components with the environment, e.g. with the control actions of sequential or safety controllers (see [Chapter 14](#) for examples of such systems).

The most straightforward way to model the two-tank system from [Section 1.3.1](#) in *gPROMS* is to implement it in a single *MODEL* entity. While this approach is certainly feasible, it contradicts the concepts of modularity and re-usability that are advocated by the object-oriented modeling approach. Thus, a more generic simulation model of the two-tank system will be developed in the following example that illustrates the main concepts of the noncausal object-oriented modeling approach in general and *gPROMS* in particular.

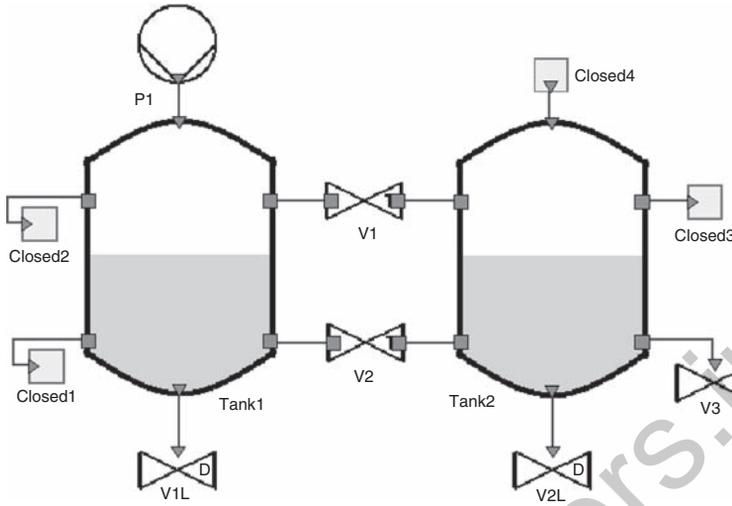


Fig. 11.5 A graphical representation of the two-tank system in the flow sheet editor of *gPROMS*.

Example 11.2 Two-tank system

The two-tank system can be decomposed into four physical components: the tanks themselves, the pump, the pipes connecting the tanks, and the drain pipes. Thus, an object-oriented approach to modeling this system is to first create generic submodels of these four components that are subsequently interconnected via well-defined interfaces to yield the overall system. The flow sheet of the resulting model is shown in Fig. 11.5. The physical components are implemented as *MODEL* entities, and the interfaces are represented as *PORT* entities in *gPROMS*. In the flow sheet, these ports are shown as squares (bi-directional ports) and triangles (inlet/outlet ports) on the boundaries of the physical components.

Since the flow rates within the two-tank system depend on the heights of the liquid levels within the tanks above the height of the connection pipe (h_0) and are computed according to Torricelli's law, first a *PORT* entity *TorriFlow* is defined that is equipped with the three variables h , h_0 , and vol_flow . An interconnection of two models via ports of type *TorriFlow* is translated into equalities of the port variables in both models.

A basic *MODEL* entity consists of three sections: a *PARAMETER* section that contains declarations of constant model parameters, a *VARIABLE* section that holds declarations of the quantities that are computed in the model equations, and an *EQUATION* section that contains the model equations. Note that since *gPROMS* is a noncausal modeling environment, the order of the equations is irrelevant. All model equations must be fulfilled at all times.

The following code fragment shows the *MODEL HYCONTank* that represents each of the tanks that are shown in Fig. 11.5. Note that *gPROMS* ignores all text in a line that follows the # symbol, and that paths in the object hierarchy are specified using a dot. For example, *flowUpperLeft.h* refers to the entity h that is encapsulated in the *PORT* entity *flowUpperLeft*.

```

# MODEL HYCONTank

PARAMETER # model parameters
A AS REAL # cross-sectional area
h0_left AS REAL DEFAULT 0 # height of upper left outlet
h0_right AS REAL DEFAULT 0 # height of upper right outlet

PORT
flowUpperLeft AS TorriFlow
flowLowerLeft AS TorriFlow
flowLowerRight AS TorriFlow
flowUpperRight AS TorriFlow
flowBottom AS TorriFlow
flowTop AS TorriFlow

VARIABLE # model variables
h AS height # state variable

EQUATION
# ensure that the height is transmitted to the ports
flowUpperLeft.h=h;
flowUpperRight.h=h;
flowLowerLeft.h=h;
flowLowerRight.h=h;
flowBottom.h=h;
flowTop.h=h;

flowUpperLeft.h0=h0_left;
flowUpperRight.h0=h0_right;
# at the bottom of the tank -> set to 0
flowLowerLeft.h0=0;
flowLowerRight.h0=0;
flowBottom.h0=0;
# set to maximum value of the variable type "height"
flowTop.h0=100000;

# the only differential equation
$h = (flowTop.vol_flow + flowUpperLeft.vol_flow +
      flowLowerLeft.vol_flow + flowLowerRight.vol_flow +
      flowUpperRight.vol_flow + flowBottom.vol_flow)/A;

```

The model contains six ports (i.e. external interfaces) of type *TorriFlow* that are declared in an additional *PORT* section and that represent the inlets and outlets at the top, at the bottom, and at the left and right sides of the tanks shown in Fig. 11.5. The cross-sectional area A and the heights $h0_left$ and $h0_right$ of the upper left and the upper right connection pipes are constant real-valued model parameters. The height parameters default to the value 0 if they are not assigned before a simulation run (see below). The only model variable (h) is of the real-valued type *height* that is defined in the *PML*. In *gPROMS*, the modeler can also define custom *variable types* to restrict the possible values to physically reasonable ranges and to assign units to variables that can be used for consistency checks.

The first 12 model equations ensure that the port variables are equal to the internal model variables and parameters. Since the flow rates and their signs are modeled in the connection

and drain pipes (see below), the only differential model equation simply sums up the flow rates from all ports. Here, the operator \dot{h} represents the time derivative of h .

The hybrid dynamics of the two-tank system is implemented in the model of the connection valve:

```
# MODEL ConnectionValve

PARAMETER # model parameters
g AS REAL DEFAULT 9.81 # gravitational constant
c AS REAL DEFAULT 3.6e-5 # valve constant

PORT # connector ports
port1 AS TorriFlow
port2 AS TorriFlow

VARIABLE # variables
u AS valve_setting

SELECTOR # discrete state
state AS (q1,q2,q3,q4) DEFAULT q4

EQUATION # model equations

CASE state OF
WHEN q1:
  port1.vol_flow = 0;
  port2.vol_flow = 0;
  SWITCH TO q2 IF port1.h >= port1.h0
    AND port2.h < port2.h0;
  SWITCH TO q3 IF port1.h < port1.h0
    AND port2.h >= port2.h0;
WHEN q2:
  port1.vol_flow = -c*SQRT(2*g*ABS(port1.h-port1.h0))*u;
  port2.vol_flow = -port1.vol_flow;
  SWITCH TO q1 IF port1.h < port1.h0
    AND port2.h < port2.h0;
  SWITCH TO q4 IF port1.h >= port1.h0
    AND port2.h >= port2.h0;
WHEN q3:
  port1.vol_flow = -port2.vol_flow;
  port2.vol_flow = -c*SQRT(2*g*ABS(port2.h-port2.h0))*u;
  SWITCH TO q1 IF port1.h < port1.h0
    AND port2.h < port2.h0;
  SWITCH TO q4 IF port1.h >= port1.h0
    AND port2.h >= port2.h0;
WHEN q4:
  port1.vol_flow = -c*SGN(port1.h-port2.h)*
    SQRT(2*g*ABS(port1.h-port2.h))*u;
  port2.vol_flow = -port1.vol_flow;
  SWITCH TO q2 IF port1.h >= port1.h0
    AND port2.h < port2.h0;
  SWITCH TO q3 IF port1.h < port1.h0
```

```

                AND port2.h >= port2.h0;
    END # end of CASE

```

In *gPROMS*, hybrid dynamics are represented by *state-transition networks* (*STNs*). In *STNs*, the discrete states represent different sets of active continuous equations, and the transitions between the discrete states are governed by logical expressions that are defined on the model variables. *gPROMS* offers two language constructs to define these *STNs*: implicit conditional equations using *IF* expressions (i.e. no discrete variable that models the discrete state is needed), and explicit conditional equations that are implemented using *CASE* expressions.

The model of the connection valve determines the flow rates between the two connected tanks by implementing the flow rate equations given in Section 1.3.1 for the variable values that are transmitted via the two *TorriFlow* ports. Here, the intrinsic mathematical *gPROMS* functions *ABS* (absolute value) and *SQRT* (square root) are used. The opening degree of the valve is represented by the variable *u* that is of the self-defined real-valued type *valve_setting*. Note that since discrete variables are not supported in *gPROMS*, *u* must be set to a discrete value manually before simulation (see below).

The discrete state of the connection valve is encoded in the *SELECTOR* variable *state* that can assume four different values (*q1*, *q2*, *q3*, or *q4*) which correspond to the discrete states shown in Section 1.3.1. The *CASE* construct that is shown in the *EQUATION* section activates different sets of equations depending on the value of *state*. Transitions between the discrete states are modeled using the *SWITCH TO <new_state> IF <transition_condition>* expression.

The continuous models for the pump and for the drain valve are implemented in a similar fashion. Two additional, very simple models *HYCONNoFlowIn* and *HYCONNoFlowOut* are created that represent the case that the inflow/outflow through a *TorriFlow* port is zero. These models are the templates for the components *Closed1–Closed4* in Fig. 11.5.

The model components can now be integrated into a more complex hierarchical model using the *gPROMS* language or via the graphical flow sheet editor. The textual equivalent of the configuration shown in Fig. 11.5 is:

```

# MODEL HYCONTanks

UNIT
  Tank1 AS HYCONTank
  Tank2 AS HYCONTank
  V1 AS ConnectionValve
  V2 AS ConnectionValve
  ...

TOPOLOGY
  Tank1.flowUpperRight = V1.port1;
  V1.port2 = Tank2.flowUpperLeft;
  Tank1.flowLowerRight = V2.port1;
  V2.port2 = Tank2.flowLowerLeft;
  ...

```

The *UNIT* section contains the declarations of all submodels of the model, and the *TOPOLOGY* section describes their interconnections (note that not all unit definitions and topology equations are shown in the above example for reasons of brevity). This model might also include additional equations, parameters, or (selector) variables and can be included as a

UNIT into an even larger model. Furthermore, the model components can now be used to create arbitrarily large networks of interconnected tanks.

Simulation activities for *MODEL* entities are defined in *PROCESS* entities using basic or user-defined *TASKS*. The following *PROCESS* entity defines a simulation scenario for the two-tank model in which the values of all input variables are held constant. The execution of this *PROCESS* generates the simulation results that are shown in [Section 1.3.1](#).

```
# PROCESS HYCONTwoTanks_simple

UNIT # models in this process
  Tanks AS HYCONTanks

SET # set parameters
  WITHIN Tanks DO
    Tank1.A := 0.0154;
    Tank1.h0_right := 0.3;
    Tank2.A := 0.0154;
    Tank2.h0_left := 0.3;
  END # end of WITHIN

ASSIGN # input variable values
  WITHIN Tanks DO
    P1.flowrate := 0.03e-3;
    V1.u := 1;
    V2.u := 1;
    V3.u := 1;
    V1L.u := 0;
    V2L.u := 0;
  END # end of WITHIN

INITIAL # initial values
  WITHIN Tanks DO
    Tank1.h = 0.25;
    Tank2.h = 0.45;
  END # end of WITHIN

SCHEDULE # simulation schedule
  SEQUENCE # simulation sequence
  CONTINUE FOR 100 # 100 seconds
  END # end of SEQUENCE
```

Like all *gPROMS* entities, a *PROCESS* contains several sections. In the *UNIT* section, the model that is to be simulated is declared, and the values of the constant model parameters are defined in the *SET* section. The *WITHIN <entity> DO ... END* construct is helpful to avoid long path names. All assignments within this construct refer to the hierarchical level that is defined in *<entity>*. All model parameters that are not *SET* are automatically assigned their default values. In the *ASSIGN* section, the values of the input variables are defined. These values can be constant or functions of time. For example, to specify a time-varying flow rate for the pump *P1*, the constant assignment shown above must be replaced by

```
P1.flowrate := 0.1e-3/2*(SIN(TIME)+1);
```

This function varies the flow rate between $0 \text{ m}^3 \text{ s}^{-1}$ and the maximum value $0.0001 \text{ m}^3 \text{ s}^{-1}$. The initial values of the state variables are defined in the *INITIAL* section.

In DAE models, not only differential but also algebraic variables can be initialized, and the *gPROMS* simulation engine automatically determines consistent initial values for the DAE system. Note that in the *SET* and *ASSIGN* sections, the assignment operator ($:=$) is used while the initial values are specified in equations ($=$) that are included into the overall simulation model, but that are only valid at $t = 0$.

Finally, the simulation sequence is defined in the *SCHEDULE* section using the elementary *TASK CONTINUE FOR* that stops the simulation when the specified time is reached. *gPROMS* provides several additional elementary *TASKS* that can be used in a simulation sequence to re-assign the input variables (task *RESET*), to change the value of a *SELECTOR* variable, i.e. to change the discrete state of a model (task *SWITCH*), to replace an input variable (which can then vary freely) with another one (task *REPLACE*), to stop the simulation (task *STOP*), and to re-assign the values of differential or algebraic model variables (task *REINITIAL*). The latter task allows to define jumps of the continuous state variables. In response to such a jump, *gPROMS* automatically re-initializes the DAE system consistently before resuming the simulation.

In *PROCESSES*, very complex sequential operation procedures can be implemented using self-defined *TASK* entities. In the following, this concept is illustrated by the extension of the two-tank system with a simple discrete level controller.

User-defined *TASK* entities consist of three sections, a *PARAMETER* section, a *VARIABLE* section in which local variables are declared, and a *SCHEDULE* section in which the sequential behavior is defined. First, a *TASK* that implements a simple level controller for a single tank is defined as:

```

# TASK LevelController

PARAMETER
  contrTank AS MODEL HYCONTank # model type
  V AS MODEL DrainValve # controlled valve
  # switching thresholds
  t_upper AS REAL
  t_lower AS REAL

SCHEDULE
  SEQUENCE
    WHILE TRUE DO
      PARALLEL
        SEQUENCE
          CONTINUE UNTIL contrTank.h >= t_upper;
        RESET
          V.u := 1;
        END # end of RESET
      END # end of first inner SEQUENCE
    SEQUENCE
      CONTINUE UNTIL contrTank.h <= t_lower;
      RESET
        V.u := 0;
      END # end of RESET
    END # end of second inner SEQUENCE
  END # end of PARALLEL
END # end of WHILE
END # end of outer SEQUENCE
    
```

In the *PARAMETER* section, the *MODEL* entities are specified on which the *TASK* operates. The level controller developed here controls the liquid level in a *HYCONTank* via a *DrainValve*. In addition, upper and lower switching thresholds are defined that indicate the maximum and minimum values of the liquid level.

The *SCHEDULE* section implements the control logic. The level controller operates in an infinite *WHILE* loop in which two concurrent operation sequences are executed. Concurrent actions can be implemented in *gPROMS* using the *PARALLEL* construct. Each sequence continuously monitors the height of the liquid in the tank and compares it with the upper and lower thresholds. If the upper threshold is exceeded, the drain valve is opened using the *RESET* task, and if the lower threshold is reached, the drain valve is closed.

The level controller for the two-tank system can now be constructed hierarchically by the concurrent execution of the two level controller *TASK*s:

```
# TASK TwoLevelControllers

PARAMETER
  contrTanks AS MODEL HYCONTanks

SCHEDULE
  PARALLEL
    LevelController
      ( contrTank IS contrTanks.Tank1,
        # tank 1 is controlled via the valve V1L
        V          IS contrTanks.V1L,
        t_upper   IS 0.5, # h_max = 0.5 m
        t_lower   IS 0.2) # h_min = 0.2 m
    LevelController
      ( contrTank IS contrTanks.Tank2,
        # tank 1 is controlled via the valve V3
        V          IS contrTanks.V3,
        t_upper   IS 0.5, # h_max = 0.5 m
        t_lower   IS 0.2) # h_min = 0.2 m
  END # end of PARALLEL
```

This task takes a two-tank model as a parameter and invokes the suitably parameterized level controller *TASK*s within a *PARALLEL* construct.

To simulate the controlled two-tank system, the *SCHEDULE* section of the *PROCESS* entity *HYCONTwoTanks_simple* that is described above must be modified slightly:

```
# PROCESS HYCONTanks_controlled
...

SCHEDULE # simulation schedule
  PARALLEL # level controllers and termination criterion

    TwoLevelControllers ( contrTanks IS Tanks )

  SEQUENCE # stop simulation after 100 seconds
    CONTINUE FOR 100
    STOP
  END # end of SEQUENCE

END # end of PARALLEL
```

Again, two sequential procedures are executed concurrently. The first sequence simply invokes the *TASK TwoLevelControllers*, and the second sequence stops the simulation after 100 seconds. □

11.2.5 Other tools and environments

20-sim The modeling and simulation package *20-sim*, which is named after its origin, the University of Twente (“*twente*”-*sim*), is a derivative of the now defunct simulation environment *TUTSIM* that was developed at the University of Twente in the 1970s, and of its successor *CAMAS*. Its modeling formalism is based on a combination of hierarchical bond graphs, block diagrams, and noncausal equations. *20-sim* provides basic facilities to model discrete-continuous systems, although this concept is not an integral part of the environment [477]. Furthermore, *20-sim* models can be exported into the *Modelica* language.

ABACUSS II/Jacobian Since the modeling and simulation tool *ABACUSS* (*Advanced Batch And Continuous Unsteady State Simulator*) and its successor *ABACUSS II* are academic derivatives of *gPROMS*, their modeling framework and language are very similar. *ABACUSS II* relies on the numerical software library *DAEPACK* (*Differential-Algebraic Equation Package*) for symbolic and numerical computations. Among others, *DAEPACK* provides advanced algorithms for the consistent initialization of DAEs, for automatic index reduction, for numerical simulation and dynamic optimization, and for the efficient integration of FORTRAN subroutines and the model export into the *CAPE-OPEN* standard. In contrast to *gPROMS*, *ABACUSS II* supports the object-oriented concept of inheritance (key word *INHERITS*), i.e. the refinement of a basic model (class) into a more specialized class. The modeling and simulation environment *Jacobian* is a commercial offspring of *ABACUSS II*.

BaSiP *BaSiP* (*Batch Simulation Package*) is a modeling and simulation environment for flexible recipe-driven chemical batch plants. The modeling language is graphical, and the *BaSiP* environment provides several graphical editors for the specification of hierarchical plant models and of hierarchical production recipes in the form of *Sequential Function Charts* (*SFCs*). The plant model can be formulated on different levels of abstraction, and the simulation engine of *BaSiP* employs algorithms that are tailored for the efficient simulation of batch plants. Furthermore, an interface to *gPROMS* is included.

Charon *Charon* is a modeling formalism for the modular specification of interacting hybrid systems. Like χ (see below), it is equipped with formally defined compositional semantics that make it very suitable for systems analysis by modular reasoning. *Charon* defines two kinds of hierarchy: architectural hierarchy that reflects the composition of distinct concurrent (i.e. parallel) processes (which are called agents), and behavioral hierarchy that reflects the composition of components that act sequentially (called modes). Communication between the model components is realized via

shared variables and via message passing, and the supported continuous dynamics are DAEs. The *Charon* tool set provides a graphical modeling language that is integrated into a graphical user interface, a simulator, and an embedded type checker.

CheckMate The *MATLAB/Simulink*-based toolbox *CheckMate* supports the modeling, the simulation, and the verification of special class of nonlinear hybrid systems, *threshold-event-driven hybrid systems (TEDHS)*. In this formalism, the continuous dynamics are modeled by ordinary differential equations for which planar switching surfaces can be defined, and the discrete dynamics are represented as finite-state machines. The models are created using a custom graphical user interface within the *MATLAB/Simulink* environment. The editor offers different *Simulink* blocks for the definition of the switched continuous dynamics while the finite-state machine is implemented in *Stateflow*. *CheckMate* offers a simulation algorithm for *TEDHS* models that is based on the *MATLAB* simulation engine.

χ (**Chi**) χ was initially developed for the simulation of manufacturing plants, but has been extended into a multi-disciplinary modular modeling formalism that integrates concepts from dynamics and control theory with concepts from computer science, in particular from process algebra. χ is a very powerful modeling formalism that supports many concepts such as continuous DAE dynamics, discrete, continuous, and algebraic variables, urgent and nonurgent actions, and stochastic operators. An outstanding feature of χ is its mathematically well-defined compositional semantics that enables formal modular reasoning about the properties of large complex χ models. Modular reasoning here means that properties of the complex overall system can be proven based on properties of the (simpler) submodels.

χ is related to the *Compositional Interchange Format (CIF)* that is described in [Chapter 12](#). However, it is grained finer: while an automaton is the smallest element in the *CIF*, in χ the elements of an automaton, such as differential-algebraic equations, assignments, and sending and receiving of information are the atomic elements.

The χ tool set provides three different simulators: a very fast simulator for a subset of χ that contains only clocks (*timed* χ), a simulator for hybrid χ models that supports variable-structure systems and that solves algebraic loops using the the mathematical computing environment *Maple*, and a simulator that can be included into *Simulink* using the *s-function* interface. In addition, formal translations into *UPPAAL* models and linear hybrid automata have been defined.

EcosimPro The *EcosimPro* tool project was initiated in 1989 by the European Space Agency (ESA) to simulate environmental control and life support systems for Hermes manned spacecraft. The multidisciplinary nature of this object-oriented noncausal modeling tool gave rise to its use in many other disciplines such as fluids, chemical, control, energy, propulsion, and others (see [Section 14.4](#) for examples). The *EcosimPro* environment provides an intuitive graphical user interface, a powerful symbolic pre-processing engine, model libraries for different areas, and facilities to model hybrid systems, such as continuous and discrete variables and language

constructs that enable the implementation of discontinuous equations, sequential behavior, and discrete events.

HyBrSim *HyBrSim (Hybrid Bond gRaph Simulator)* is a modeling and simulation tool for hybrid systems that is based on the hybrid bond graph formalism (with continuous DAE dynamics) which extends the continuous (graphical) bond graph model with switching dynamics that are represented by finite-state machines. *HyBrSim* employs an interpretative simulation approach, i.e. it does not solve a single global system of equations, but propagates the variable values through the model topology. Although this approach is relatively slow, it allows for a flexible treatment of variable-structure systems. *HyBrSim* provides a block diagram editor for model composition.

Modelica The goal of the freely available *Modelica* language specification is to provide a standardized and powerful object-oriented noncausal modeling language for multi-domain modeling. *Modelica* was (and still is) designed in a joint effort by computer scientists, dynamic model practitioners, and the developers of the object-oriented modeling languages *Allan*, *Dymola*, *NMF*, *ObjectMath*, *Omola*, *SIDOPS+*, and *Smile*.

Modelica closely follows the concepts of the object-oriented modeling paradigm and, in this respect, goes beyond most of the other object-oriented languages that are presented in this chapter: almost all objects within *Modelica* are instances of classes, and *Modelica* supports advanced object-oriented concepts such as inheritance, restricted classes or generic classes. Sequential procedures can be implemented in algorithm sections or functions and thus enable the integration of high-level discrete-event formalisms, as is e.g. done in the *DES/M* environment [514].

Besides its clear object-oriented structure, one of the major reasons for the increasing popularity of *Modelica* is the large number of free and commercial model libraries that provide model templates for many technical areas, such as thermodynamics, mechanics, control, electronics, and automotive.

Up to now, two commercial tools are available that implement the *Modelica* language: *Dymola* and *MathModelica*. In addition, an open-source tool, *OpenModelica*, is currently under development. The most mature of these tools, *Dymola*, comes with a very powerful symbolic and numerical simulation engine that e.g. supports automatic index reduction of high-index DAE systems and tearing of algebraic loops, and that provides specialized numerical solvers for large-scale DAE systems.

Omola/Omsim *Omola (Object-oriented Modeling Language)* is an object- and equation-oriented modeling language that supports mixed continuous (semi-explicit DAE) dynamics and discrete events. Similar to *Modelica*, all models are regarded as classes in *Omola*. *Omola* supports single inheritance, and the basic constructs that are used to describe discrete events enable the implementation of high-level discrete-event systems such as Petri Nets. *Omola* comes with the modeling and simulation environment *Omsim* that provides a graphical model editor, symbolic pre-processing of the model equations, and a hybrid simulation engine.

Ptolemy II/HyVisual The *Ptolemy II* project focuses on the heterogeneous modeling, simulation, and design of concurrent embedded systems. The main motivation behind *Ptolemy II* is to generate executable models that can be implemented in a large variety of computational models such as discrete-event and continuous-time systems, communicating sequential processes, or finite-state machines. *HyVisual* (*Hybrid System Visual Modeler*), which is based on the *Ptolemy II* software framework, is a block-diagram-based tool for the modeling and simulation of hierarchical continuous and hybrid dynamic systems with continuous dynamics that are modeled by ODEs.

Scilab/Scicos The development of the powerful technical computing environment *Scilab* (*Scientific laboratory*) began in 1990. It is an open-source alternative to *MATLAB* that provides a large set of tool boxes for different application domains, and the block-diagram-based simulation tool *Scicos* (*Scilab Connected Object Simulator*) is the corresponding open-source equivalent of *Simulink*. Although *Scilab/Scicos* lacks an equivalent of the *Simulink*-based *Stateflow* tool and can thus not directly represent finite-state machines, it provides specialized blocks that can be used to model hybrid systems (see [143] for examples). Like *Simulink*, *Scicos* supports the integration of external functions and programs. These can be implemented in *C*, *Fortran*, or in the *Scilab* programming language. In addition, *Scicos* directly supports a subset of the *Modelica* language.

SHIFT *SHIFT* (the name is a permutation of *HSTIF*, *Hybrid Systems Tool Interchange Format*) is a textual modeling language for dynamically reconfigurable hybrid systems with continuous ODE dynamics. It employs networks of hybrid automata in which model components may be created, interconnected, and destroyed during simulation. Thus, *SHIFT* is suited particularly for hybrid systems in which the model structure changes dynamically, such as air traffic control systems or highway systems. *SHIFT* models can be simulated in the *SHIFT* runtime environment that also provides a graphical environment for the visualization of simulation runs. Alternatively, *SHIFT* models can be compiled into stand-alone *C* programs using a dedicated compiler.

Siconos The *Siconos* (*Simulation and Control of Non-smooth Dynamic Systems*) platform was initiated within a European project of the same name and is dedicated to the modeling, simulation, analysis, and control of a special class of hybrid systems, *non-smooth dynamic systems* (*NSDS*), in which continuous dynamic systems interact with non-smooth laws. Since *Siconos* models are very abstract, they can be used to model systems from various application areas, such as mechanics, robotics, or electronics. The *Siconos* tool set is capable of simulating *NSDS* models in both, a time-driven and an event-driven fashion and provides interfaces to *MATLAB* and *Scilab*.

UPPAAL *UPPAAL* is an integrated graphical environment for modeling, simulation, verification, and controller synthesis of real-time systems and is currently the

standard tool for the analysis, simulation, and design of timed systems. *UPPAAL* models are represented as networks of timed automata that support variables and communicate via channels or shared variables. The basic *UPPAAL* tool encompasses a graphical editor for model creation, a graphical simulator, and an efficient model checker, and several variants of *UPPAAL* have been developed that provide extended functionality such as cost-optimal reachability analysis (*UPPAAL CORA*), black-box testing (*UPPAAL TRON*), and controller synthesis based on timed games (*UPPAAL TIGA*).

11.3 Dynamic optimization

11.3.1 Optimization problem

The problem formulation that is employed by many tools for the optimization of nonlinear hybrid systems is similar to that given in [Section 3.5](#), i.e. the goal is to minimize a cost function over a time interval $[t_0, t_f]$, where t_0 and t_f are the initial and the final time, and the hybrid model serves as a dynamic constraint. However, due to the requirements mentioned above, most tools support a more general formulation than that given in [Section 3.5](#). For example, the continuous dynamics in each mode can often be defined as implicit systems of differential-algebraic equations (DAEs) of the form

$$\emptyset = \mathbf{f}(\mathbf{x}(t), \dot{\mathbf{x}}(t), \mathbf{y}(t), \mathbf{u}(t), \mathbf{v}(t), \mathbf{p}),$$

or as semi-explicit systems of DAEs or the form

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{f}(\mathbf{x}(t), \mathbf{y}(t), \mathbf{u}(t), \mathbf{v}(t), \mathbf{p}), \\ \emptyset &= \mathbf{g}(\mathbf{x}(t), \mathbf{y}(t), \mathbf{u}(t), \mathbf{v}(t), \mathbf{p}). \end{aligned}$$

Here, the vectors $\mathbf{x}(t)$ represent the state variables and $\dot{\mathbf{x}}(t)$ are the corresponding time derivatives, $\mathbf{y}(t)$ are the algebraic variables, $\mathbf{u}(t)$ and $\mathbf{v}(t)$ are the continuous and the discrete inputs, and \mathbf{p} are time-invariant parameters. The cost function can then be written as:

$$\min_{\mathbf{x}(t), \mathbf{y}(t), \mathbf{u}(t), \mathbf{v}(t), \mathbf{p}, t_f} \Omega(\mathbf{x}(t), \mathbf{y}(t), \mathbf{u}(t), \mathbf{v}(t), \mathbf{p}, t_f). \quad (11.2)$$

For simplicity, the discrete modes of the hybrid system are not considered here. Many tools support the specification of additional constraints, such as equality and inequality path constraints

$$0 \leq g_p(\mathbf{x}(t), \mathbf{y}(t), \mathbf{u}(t), \mathbf{v}(t), \mathbf{p}) \quad \forall t \in [t_0, t_f], \quad (11.3)$$

which must be fulfilled over the complete time horizon $[t_0, t_f]$, initial and endpoint constraints

$$0 \leq g_i(\mathbf{x}(t_0), \mathbf{y}(t_0), \mathbf{u}(t_0), \mathbf{v}(t_0), \mathbf{p}), \quad (11.4)$$

$$0 \leq g_e(\mathbf{x}(t_f), \mathbf{y}(t_f), \mathbf{u}(t_f), \mathbf{v}(t_f), \mathbf{p}), \quad (11.5)$$

which only have to be fulfilled at t_0 and t_f , or interior-point constraints

$$0 \leq g_{ip}(\mathbf{x}(t_i), \mathbf{y}(t_i), \mathbf{u}(t_i), \mathbf{v}(t_i), \mathbf{p}), \quad (11.6)$$

which only have to be fulfilled at a particular time point $t_i \in [t_0, t_f]$.

The available optimization approaches for such dynamic optimization problems can be categorized into three classes. The first class is based on variational principles, i.e. on the solution of the optimality conditions that are derived from hybrid extensions of Pontryagin's Maximum Principle. These methods are also called indirect methods, and they are described in Section 3.5. A second branch of approaches is based on the extension of dynamic programming principles to the hybrid setting, e.g. see [115, 126, 298]. The major drawback of these approaches is that they are only applicable to relatively small hybrid systems. In consequence, most of the existing tools are based on the third class of numerical approaches which are called the direct optimization methods and which have already been applied successfully to systems with thousands of variables. These techniques have in common that they directly transform the infinite-dimensional optimization problem that is stated above into a finite-dimensional representation that is amenable to the application of (mixed-integer) nonlinear optimization algorithms.

11.3.2 Numerical optimization of nonlinear hybrid systems

While the derivation of a finite-dimensional representation of hybrid systems with linear or affine dynamics is rather straightforward as was shown in the previous chapters, the finite-dimensional reformulation of optimization problems for systems with nonlinear dynamics is more involved. The existing approaches can be divided into two categories: the first class of approaches is based on a finite-state representation of the control profiles, i.e. on control (vector) parameterization. In these sequential or single-shooting techniques, the optimization horizon is divided into time stages of varying duration, and in each stage the profiles of the continuous inputs are represented by suitable functions, such as piecewise constant functions, piecewise linear functions, or polynomials. The dynamics of the hybrid system and the cost function are evaluated by integration of the dynamic model, and an optimization algorithm, such as a mixed-integer nonlinear programming (MINLP) solver, determines optimal values for the control parameters, for the discrete inputs, for the time-invariant parameters, and for the lengths of the time intervals.

In the second class of direct approaches, the simultaneous techniques, both the control profiles and the dynamic equations are discretized. One possibility to discretize the dynamic equations is to employ a multiple-shooting approach: similar to the sequential technique, the optimization horizon is divided into stages, the continuous control profiles are represented by parameterized functions, and the dynamics are evaluated by integration. In contrast to the sequential method, the initial values of the state variables in each stage are assumed to be unknown and are chosen by the optimizer, and additional equality constraints ensure that the solution that is determined in the final iteration of the optimization algorithm fulfills the dynamic

equations of the system over the complete time horizon. In the full-discretization or collocation-based technique, the optimization problem is converted into a purely algebraic (MI)NLP problem by the approximation of the control and dynamic profiles with suitable functions, e.g. polynomials, using collocation on finite elements.

Sequential approaches lead to relatively small (MI)NLP problems and are particularly suitable for systems with few time-varying controls and many dynamic equations. However, these approaches can lead to problems if the dynamic system exhibits instabilities since in this case, it may be difficult to determine feasible solutions for given sets of control parameters. Furthermore, path constraints cannot be integrated directly into the problem formulation. In [645, 646], a sequential algorithm for the dynamic optimization of multi-stage systems is presented. In multi-stage systems, the number of time intervals and the sequence of the discrete modes of the hybrid system is fixed. In this approach, path constraints are included by the penalization of the constraint violation that is computed over the optimization horizon. A disadvantage of this approach is its restriction to systems of DAEs with an index of 1 or less. This problem is overcome by the sequential approach presented in [45, 229] that supports high-index DAEs and that employs an algorithm for constrained dynamic simulation to deal with path constraints.

Simultaneous approaches that are based on multiple shooting lead to considerably larger (MI)NLPs than sequential methods. However, a tailored solution algorithm for these problems has been developed in recent years [397, 398] that supports multi-stage systems (i.e. systems with differing continuous dynamics in the time stages) and that is sufficiently fast for real-time applications. Recently, this algorithm has been extended to handle binary controls [563] and autonomous switchings of the continuous dynamics [111].

The collocation-based simultaneous methods lead to very large but usually sparse optimization problems that can be solved efficiently using specialized large-scale NLP solvers. These methods have the advantages that path constraints can be considered directly in the problem formulation and that systems with instabilities can be handled in a well-conditioned manner. In [28], a full-discretization technique for a general class of hybrid systems is presented that yields a MINLP problem formulation. Other approaches, such as those presented in [547] and [604], reformulate the mixed-integer optimization problem into a purely continuous NLP problem using suitable continuous complementarity constraints.

To counteract the combinatorial complexity that arises in the optimization of hybrid systems with many discrete modes or discrete inputs, several approaches have been developed that decompose the optimization problem into simpler subproblems. For example, in [672] a hierarchical approach is presented that separates the computation of the optimal sequence of the discrete modes and the computation of optimal control inputs and event times. In another approach [596, 608], the possible trajectories of the discrete inputs are encoded in an acyclic graph. Optimal values for the continuous inputs are computed by embedded nonlinear optimization, the result of which is used to prune the search graph based on upper and lower cost bounds.

Many of the approaches presented above lead to MINLP formulations, and a variety of techniques is available to solve these problems. Many methods are based

on extensions of the branch-and-bound algorithms for mixed-integer linear programming (MILP) problems to the nonlinear case. In these algorithms, the discrete decisions are encoded in a tree, and in each node continuous NLP problems (with fixed or relaxed discrete variables) are solved. The tree is pruned based on lower and upper cost bounds, or on more advanced techniques based, e.g., on cutting planes that are determined from MILP subproblems. Another branch of algorithms employs an outer approximation in which the solutions of a master MILP problem and of NLPs with relaxed or fixed discrete decision variables is iterated. Here, the MILP solutions provide lower cost bounds and are used to determine promising values for the discrete variables.

While several powerful methods for the dynamic optimization of nonlinear hybrid systems and suitable optimization algorithms have been developed in the last decades, there are still many formidable challenges. For example, many optimization problems that arise in hybrid systems optimization are non-convex, and the optimization algorithms described above do not guarantee that the globally optimal solution is found for such problems. In consequence, much research effort is currently focussed on the development of global nonlinear optimization techniques. Global methods can be divided into stochastic approaches (using, e.g., evolutionary algorithms or simulated annealing techniques) and deterministic approaches that are usually based on convex relaxations of the optimization problem. In hybrid systems optimization, a stochastic approach has for example been applied in [596, 628]. In this approach, the mixed-integer nonlinear problem is solved using a tailored evolutionary algorithm. An additional advantage of stochastic algorithms is that they are not adversely affected by discontinuities in the sensitivity equations of the optimization problem. Gradient-based methods, such as SQP algorithms, require continuity of the sensitivity equations, and this problem is currently also subject to research.

11.3.3 Tools for the optimization of nonlinear hybrid systems

As already mentioned above, it is often advantageous to employ the facilities of existing modeling and simulation environments in dynamic optimization tools. One possibility to equip modeling and simulation environments with optimization facilities is to integrate the optimization algorithms as external software components. As an example, the hybrid nonlinear model-predictive control algorithm that is used to optimize the start-up of the evaporation system in Section 14.4 was integrated into the environment *EcosimPro* via its external software interface, and the optimization-based safety analysis scheme that is described in the same chapter was implemented in *MATLAB* using the *MATLAB*-based optimization package *TOMLAB* which provides a variety of high-end optimization algorithms.

Furthermore, several modeling environments directly provide facilities for the dynamic optimization using single-shooting or multiple-shooting techniques. The simulation environment *gPROMS* offers dynamic optimization facilities via the integrated tool *gOPT*. *gOPT* is described in more detail in the next section. The sequential approach described in [45, 229] was implemented in the environment *ABACUSS* and its free and commercial successors, and the *Modelica*-based tool

Dymola has recently been extended with optimization capabilities employing a sequential approach.

The multiple-shooting algorithm that is described in [397, 398] has been implemented in the optimization package *MUSCOD II*. The dynamic simulation model can be specified in the *gPROMS* language or in the programming languages *Fortran* or *C*, and *MUSCOD II* uses an efficient partially reduced sequential quadratic programming (PRSQP) optimization algorithm that enables optimization in real time, e.g. in nonlinear model-predictive control (NMPC) applications.

Several powerful environments for general algebraic mixed-integer nonlinear optimization problems are available, such as *AMPL* and *GAMS*. These environments provide intuitive modeling languages, graphical user interfaces and debugging facilities, and interfaces to a large variety of optimization algorithms. However, the tool support for the automatic discretization and subsequent optimization of dynamic systems using collocation-based methods is still very small. Recently, the *Optimica* compiler has been introduced that extends the *Modelica* specification with language constructs for the formulation of dynamic optimization problems. *Optimica* has been designed as an open architecture into which different discretization algorithms can be integrated. *Optimica* is still under development – the current implementation supports the generation of an *AMPL*-based algebraic optimization problem by automatic collocation-based discretization of continuous dynamic optimization problems.

Example 11.3 Start-up optimization of the two-tank system in *gPROMS*

The optimization tool *gOPT* that is part of the the *gPROMS* environment provides single-shooting and multiple-shooting techniques for the optimization of hybrid systems with index-1-DAE dynamics that are modeled in the *gPROMS* language. The methods of *gOPT* are based on the software package *DAEOPT* that is described in [645, 646]. *gOPT* employs different mixed-integer and continuous optimization algorithms, such as the freely available large-scale NLP solver *HQP* that is used in the multiple-shooting approach. Optimization problems with discrete decision variables are solved using an outer-approximation MINLP algorithm that is based on a single-shooting technique and that uses an SQP algorithm for the solution of the continuous subproblems. Optimization problems with purely continuous decision variables can be solved using single-shooting and multiple-shooting NLP techniques. *gOPT* supports initial and endpoint constraints as well as interior-point constraints. Path constraints can be included using suitable endpoint constraints, as is shown below. In the following, the formulation and solution of dynamic optimization problems in *gPROMS/gOPT* is illustrated using the example of a time-optimal start-up of the two-tank system. In this setting, the tanks are initially almost empty ($h_1 = h_2 = 0.01$ m), and the objective is to determine a start-up strategy that drives the heights of the liquid levels in both tanks into the range [0.5 m, 0.6 m] in minimum time.

For optimization purposes, the dynamic model that was presented above has to be modified slightly. In *gPROMS/gOPT*, the cost function value corresponds to the value of a single variable. Thus, an equation that represents the cost function must be included into the dynamic simulation model. Since the optimization goal is to minimize the start-up time, a new variable t of the self-defined real-valued type *optvar* is included into the model of the two-tank system, and the corresponding equation ensures that this variable measures the elapsed time:

```

# MODEL HYCONTanks_opt

UNIT # As before
...
Tank1 AS HYCONTank_opt # extended tank model
Tank2 AS HYCONTank_opt # extended tank model
...

VARIABLE
  t as optvar # time

TOPOLOGY # As before
...

EQUATION
  $t = 1;

```

To ensure that the liquid levels do not fall below the minimum height of 0 m and do not exceed the maximum height of the tanks $h_{\max} = 0.6$ m, suitable path constraints must be included. Since gOPT does not support path constraints, the constraints must be reformulated as endpoint constraints. This approach requires that the constraint violation is determined during simulation. Thus, the dynamic model of a single tank is extended as:

```

# MODEL HYCONTank_opt

PARAMETER # As before
...
  h_max AS REAL DEFAULT 0.6 # maximum height

PORT # As before
...

VARIABLE # Model Variables

  h AS height # state variable
  hv AS optvar # path constraint violation

EQUATION # As before
...
  # compute violation of minimal/maximal height
  $hv = MAX(0, h-h_max, -h);

```

Here, the equation for the new variable hv integrates the constraint violation over the optimization horizon $[t_0, t_f]$. Thus, the path constraints can now be implemented using endpoint equality constraints of the form $hv(t_f) = 0$.

In gPROMS/gOPT, an optimization problem is specified using a *PROCESS* entity and an *OPTIMISATION* entity. The *PROCESS* entity is used to define the model under consideration and the initial parameter and variable values while the *OPTIMISATION* entity is used to specify the number of time intervals, the cost function variable, the constraints, the control variables and their types, and the function types that are used for the parameterization of the continuous control vector. The *PROCESS* entity for the two-tank system is very similar to that of a normal simulation activity, as is shown in the following:

```

# PROCESS HYCONTwoTanks_opt

UNIT # Models in this process
    Tanks AS HYCONTanks_opt

SET # As before
    ...

ASSIGN # As before
    ...

INITIAL # initialization of the variables
    WITHIN Tanks DO
        Tank1.h = 0.01;
        Tank1.hv = 0;
        Tank2.h = 0.01;
        Tank2.hv = 0;
        t = 0;
    END

SOLUTIONPARAMETERS
    DOSolver := "CVP_SS";
    
```

In optimization problems, the *SCHEDULE* section that is needed for simulation activities is ignored because the optimizer controls the simulation. Since the optimization problem contains discrete degrees of freedom (the settings of the discrete valves V_1 , V_2 , and V_3), the single-shooting solver *CVP_SS* is employed. The *OPTIMISATION* entity can be defined via a graphical user interface or directly in the *gPROMS* language. For the two-tank system, the optimization horizon is separated into five time intervals. The degrees of freedom of the optimization are the lengths of these time intervals, the parameters that represent the continuous input u_{P1} (i.e. the flowrate through the pump $P1$), and the settings of the discrete valves V_1 , V_2 , and V_3 that are here modeled as binary piecewise constant signals. Alternatively, discrete inputs can be specified as integer values or as enumerated values. The continuous inputs can be parameterized using piecewise constant or piecewise linear functions. Note that more complex control profiles can easily be implemented by the introduction of an additional algebraic equation (e.g. a polynomial) that models the profile of the continuous input. In this case, the coefficients of this algebraic function can be chosen as the continuous degrees of freedom in the optimization. For the two-tank system, a piecewise linear profile is chosen.

Finally, endpoint inequality constraints are defined for the heights of the liquid levels as $0.5 \text{ m} \leq h_1, h_2 \leq 0.6 \text{ m}$, and interior-point constraints are specified for the time interval boundaries. These constraints ensure that the heights h_1 and h_2 remain within the range $[0 \text{ m}, 0.6 \text{ m}]$. Although these constraints are not necessary due to the path constraints that were described above, their specification can lead to an improved numerical performance since they can be handled directly by the optimizer.

Figure 11.6 shows the optimized time trajectories of the continuous and discrete control inputs and of the resulting heights of the liquid levels. The start-up is achieved in 171.9 seconds, and the optimization took 3.9 seconds on an AMD Opteron processor with 2.4 GHz. As expected, the valve V_3 is kept closed throughout the start-up, and the flow rate through the pump $P1$ is kept at its maximum value. The liquid distribution between the tanks is controlled solely using the valve V_2 . \square

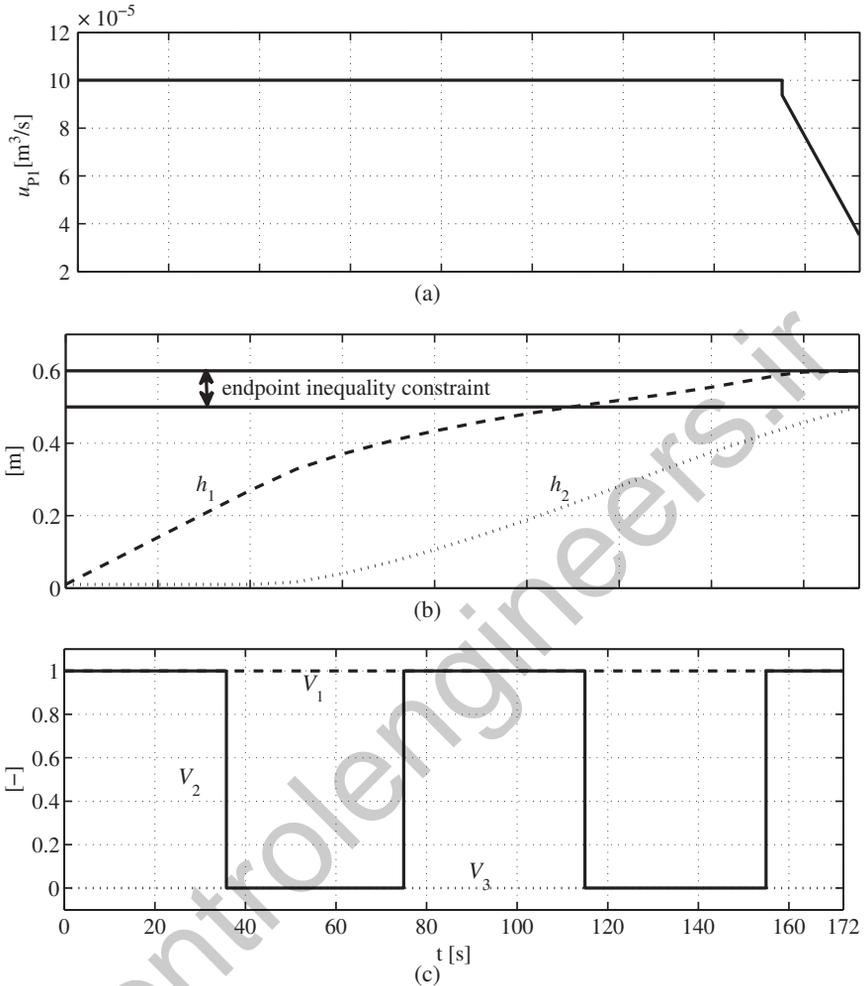


Fig. 11.6 Optimization results for the two-tank system. (a) Continuous input $pP1$. (b) Heights of the liquid levels h_1 and h_2 . (c) Discrete inputs V_1 , V_2 , and V_3 .

Bibliographical notes

The *Handbook of Dynamic System Modeling* [238] gives an excellent overview of all aspects of continuous and hybrid modeling and provides detailed introductions to several of the tools that are also presented in this chapter, such as χ , *MATLAB* with *Simulink/Stateflow/SimEvents*, and *Modelica*. The freely available book [143] investigates in detail the syntactical and semantical properties of the tool sets *MATLAB-Simulink-Stateflow*,

Modelica-Dymola, *Ptolemy II-HyVisual*, *Scicos*, *SHIFT*, and *Charon*. In addition, the modeling capabilities of these tools are analyzed and illustrated with different examples. In [477], a comparison of the support of several hybrid simulation phenomena by tool sets is presented, and numerous surveys, e.g. [44, 211, 222, 513], provide additional information on hybrid systems simulation.

More information about the current state and the major challenges of the dynamic optimization of large-scale continuous and hybrid systems can be found in [44, 46, 82, 164, 284, 285], and [221] details the current state and the future challenges of nonlinear real-time optimization in feedback control systems.

Furthermore, several online resources that are dedicated to the provision and collection of information on hybrid systems tools have been created in recent years [333, 507].

Tool-specific references The main point of reference for detailed information about a specific tool (and in many cases, for links to installation packages of the tool itself) is the tool website. Table 11.2 provides references to the websites of all tools that have been described in this chapter.

In addition, some explanatory articles and tools that have been written for virtually all tools. Some of these documents are referenced in the following:

The text [121] provides detailed information about *20-sim*, and the text book [107] gives an introduction to bond-graph-based modeling using *20-sim*. Detailed information about *DAEPACK* and *ABACUSS II* is given in [626, 627], and *BaSiP* is described in [223, 666]. The article [332] is dedicated to *Charon*, and an introduction to *CheckMate* is presented in [589]. The articles [43, 490] give an overview of *gPROMS*, and χ is described in [50]. The article [476] explains in detail the concepts behind *HyBrSim*.

Many comprehensive text books and articles are available that cover all aspects *Modelica* and provide ample information about the simulation environments *Dymola*, *MathModelica*, and *OpenModelica*. Some of these are [254, 495, 623]. Additional information about *Modelica* developments can be found in the proceedings of the *Modelica* conferences (refer to the website of *Modelica*).

The by far best documented tool set is *MATLAB* with *Simulink* and *Stateflow*. The *MATLAB* provider claims that more than 1000 *MATLAB*-related books are available that range from introductory texts on *MATLAB* [293] and *Simulink/Stateflow* [195] to more specialized books that cover virtually all engineering areas. More detailed information on *SimEvents* can be found in [148, 179].

Detailed information about *Omola/Omsim* can be found in [20, 114], and the concepts and capabilities of *Ptolemy II/HyVisual* are described in a large variety of publications such as [125, 391, 392]. The book [142] gives a comprehensive overview of modeling and simulation in *Scilab* and *Scicos*, *SHIFT* is described in detail in [210], and the concepts of *UPPAAL* are presented in [381].

The optimization framework *TOMLAB* is described in [325], and detailed information on the tool *MUSCOD II* is given in [212]. More information on *AMPL* can be found in [247], *IPOPT* is described in [655], and the *Optimica* compiler is treated in [7, 8]. In addition, [296] presents the applicability of the *Optimica* compiler using the example of a chemical reactor. In this paper, the *Modelica* model of a chemical reactor is discretized automatically, and the resulting large-scale NLP problem is solved using *IPOPT*.

Table 11.2 Online resources of modeling, simulation, and optimization tools.

Name	Provider	Licensing	Website
<i>20-sim</i>	Controllab Products B. V., Netherlands	Commercial	www.20sim.com
<i>ABACUSS II</i>	MIT, USA	Free	http://yoric.mft.edu
<i>AMPL</i>	AMPL Optimization LLC, USA	Commercial	www.ampl.com
<i>BasIP</i>	TU Dortmund, Germany	Free	www.bci.tu-dortmund.de/basip
<i>Charon</i>	University of Pennsylvania, USA	Free	http://rtg.cis.upenn.edu/mobies/charon
<i>CheckMate</i>	Carnegie Mellon University, USA	Free	www.ece.cmu.edu/webk/checkmate
<i>χ</i>	TU Eindhoven, Netherlands	Free	http://se.wtb.tue.nl/sewiki/chi
<i>Dymola</i>	Dynasim AB, Sweden	Commercial	www.dynasim.se
<i>EcosimPro</i>	EA Internacional, Spain	Commercial	www.ecosimpro.com
<i>GAMS</i>	GAMS Software GmbH, Germany	Commercial	www.gams.com
<i>gPROMS/gOPT</i>	Process Systems Enterprise Ltd, UK	Commercial	www.psenterprise.com/gproms
<i>HQP</i>	TU Ilmenau, Germany	Free	http://sourceforge.net/projects/hqp
<i>IPOPT</i>	Carnegie Mellon University, USA	Free	www.coin-or.org/Ipopt
<i>Jacobian</i>	Numerica Technology, LLC, USA	Commercial	www.numericatech.com
<i>MathModelica</i>	MathCore Engineering AB, Sweden	Commercial	www.mathcore.com/products
<i>MATLAB/Simulink/Stateflow/SimEvents</i>	The MathWorks, USA	Commercial	www.mathworks.com
<i>MUSCOD II</i>	Universitdt Heidelberg, Germany	Commercial	www.iwr.uni-heidelberg.de/agbock/RESEARCH/muscod.php
<i>Omola/Omsim</i>	Lunds Universitet, Sweden	Free	www.control.lth.se/~eace/omsim.html
<i>Optimica</i>	Lunds Universitet, Sweden	Free	www.control.lth.se/project/Langopt
<i>OpenModelica</i>	Linkpings Universitet, Sweden	Free	www.openmodelica.org
<i>Ptolemy II/HyVisual</i>	UC Berkeley, USA	Free	http://ptolemy.eecs.berkeley.edu
<i>Scilab/Scicos</i>	INRIA, France	Free	www.scilab.org
<i>Shift</i>	UC Berkeley, USA	Free	www.path.berkeley.edu/shift
<i>Siconos</i>	INRIA, France	Free	http://siconos.gforge.inria.fr
<i>TOMLAB</i>	TOMLAB Optimization	Commercial	www.tomopt.com
<i>UPPAAL</i>	Uppsala/Aalborg Univ., Sweden/Denmark	Free	www.uppaal.com

The licensing type "Free" includes all types of licenses that allow the usage free of charge at least for non-commercial purposes. Please refer to the licensing agreements for more detailed information.



controlengineers.ir

12

Interchange formats and tool integration

D. A. van Beek, M. A. Reniers, J. E. Rooda, and R. R. H. Schiffelers

To facilitate an integration of tools that have been developed separately and are based on different modeling approaches, an interchange format has been developed, which is described in this chapter together with the possible tool combinations that are enabled by this format.

Chapter contents

12.1	Overview of interchange formats for hybrid systems	page 362
12.2	Concepts in the compositional interchange format	365
12.2.1	Differential algebraic equations	365
12.2.2	Discrete, continuous, and algebraic variables	365
12.2.3	Urgency	366
12.2.4	Closed and open scopes	368
12.2.5	Input and output variables	369
12.3	Example: Bottle filling system	370

12.1 Overview of interchange formats for hybrid systems

The purpose of interchange formats is to establish interoperability of a wide range of tools by means of model transformations. In this way, the implementation of many bilateral translators between specific formalisms can be avoided as shown in Fig. 12.1 and 12.2.

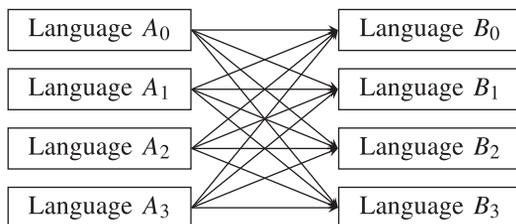


Fig. 12.1 Multiple model transformations without an interchange format.

Work on interchange formats for hybrid systems has been carried out in different projects: in the MoBIES project, the Hybrid System Interchange Format (*HSIF*) [458] is defined, and in [526] an abstract semantics of an interchange format based on the Metropolis meta model is defined (this work is a continuation of the COLUMBUS project [185]). Finally, in the HYCON network, two interchange formats have been defined: the interchange format for switched linear systems (Section 10.5) in the form of piecewise affine systems (PWAs), and the more general compositional interchange format (CIF), which is discussed in more detail in this chapter.

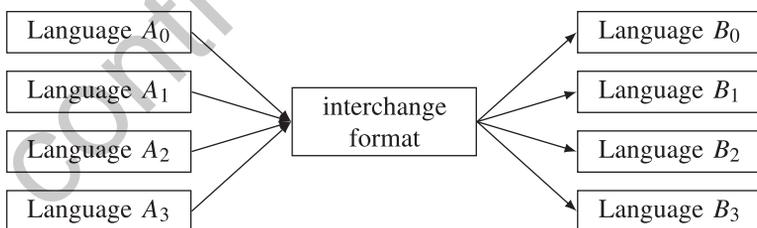


Fig. 12.2 Multiple model transformations using an interchange Format.

In *HSIF*, a network of hybrid automata is used for model representation. The network behaves as a parallel composition of its automata, without hierarchy or modules. Variables can be shared or local, and the communication mechanism is based on broadcasting of Boolean “signals,” where signals are partitioned into input and

output signals. Each signal is required to be either a global input to the network or to be modified by exactly one automaton. The semantics is defined only for “acyclic dependency graphs” with respect to the use of signals. The time dependent behavior is specified by means of ordinary differential equations (ODEs), together with algebraic relations of the form $x = f(x_1, \dots, x_n)$, and invariants. The equation $\dot{x} = 0$ is assumed for each shared variable. Circular dependencies of the algebraic equations, i.e. algebraic loops, are not allowed. To provide for “synchronous execution” of discrete steps, each automaton has a default transition. There are no algebraic variables, and there is no form of specification of urgency, such as urgency predicate or urgent action. Execution of the network is defined as an alternating sequence of continuous steps followed by discrete steps, where each discrete step of the network is the result of a sequence of discrete steps of each of the automata taken in an order that respects automata dependencies [458].

The basis of the syntax of the interchange format according to [526] is a hybrid system defined as a 9-tuple that can be composed by means of parallel composition. Parallel composition is associative. It is not commutative, however, because the equation ordering depends on the position of the hybrid systems in the composition. The structure of a hybrid system is defined by a pair of a rooted tree of components and a rooted tree of schedulers. The abstract semantics takes several implementation considerations into account, such as equation sorting, iterations that may be required for state-event detection, and iterations for reaching a fixed-point in case of algebraic loops. The semantics is defined in terms of functions and algorithms, such as *init*, *markchange*, and *solve*.

The application domain of the compositional interchange format, consists of languages and tools from computer science and from dynamics and control for modeling, simulation, analysis, controller synthesis, and verification in the area of hybrid and timed systems. The *CIF* aims to be more general than *HSIF*, and does not incorporate tool limitations, such as restrictions on circular dependencies, or restrictions on shared variables or algebraic loops, in its compositional formal semantics.

The semantics of the *CIF*, as defined in [51, 52], is different from the semantics presented in [526]. The semantics of the *CIF* aims at defining the *mathematical* meaning of models, independently of implementation aspects such as equation sorting or state-event detection. For example, the semantics defines the mathematical meaning of a switched system of equations, such as a PWA system, but an implementation may choose to implement such switching behavior with or without state-event detection. Furthermore, in the *CIF*, parallel composition is both associative and commutative. The formal semantics of the *CIF* defines all possible behaviors of a *CIF* model in terms of a hybrid transition system. The semantics defines a standard against which implementations, such as simulation or verification implementations can be evaluated. To use the *CIF* for verifications, translations from models to the interchange format, and vice versa, should preserve essential properties, that is, verification results obtained for a derived model should also be valid for the original model specified in another language.

The different languages may have different features and semantics, complicating translations between them. To keep the translations between the *CIF* and the other

languages manageable, in terms of complexity, it is important that transformations between parts of specifications within the interchange format itself can be defined. To allow such transformations, it is essential that the semantics of the *CIF* is *compositional*. Parts of a model can then be replaced by equivalent parts without changing the meaning of the model.

The purpose of the *CIF* is:

- to establish interoperability of a wide range of tools by means of model transformations to and from the *CIF* (see Fig. 12.3 for an example);
- to provide a generic modeling formalism and simulator for a wide range of hybrid systems (see Fig. 12.4 for an example).

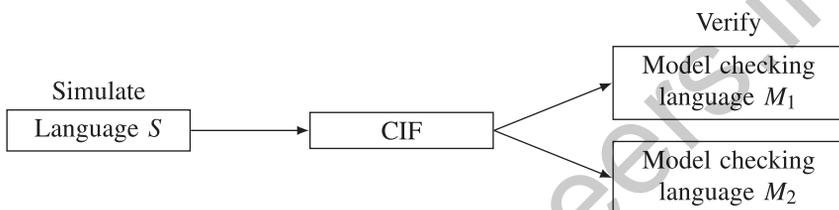


Fig. 12.3 Using *CIF* transformations for simulation and Verification.

In principle, a *CIF* model can be specified in three different formats. The first one is an abstract format, which facilitates the definition of the formal semantics. The second is a concrete format that provides user friendly syntax and can be used for modeling directly in the interchange format. Its semantics is defined by means of a translation to the abstract format. Third, a transfer format facilitates the file generation and parsing process. This format can, for example, be specified in an extensible markup language (XML) format that is supported by means of libraries in many different languages. The syntax and semantics of the abstract and concrete formats are defined in [52].

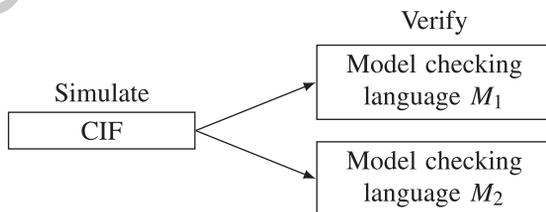


Fig. 12.4 Modeling directly in the *CIF*.

The remainder of this chapter is organized as follows: Section 12.2 discusses the concepts present in the *CIF*, and Section 12.3 presents a *CIF* model of a bottle filling system.

12.2 Concepts in the compositional interchange format

The basic element of the abstract format of the *CIF* is an *interchange automaton* with operators for parallel composition, including synchronization by means of shared labels and communication by means of shared variables and CSP channels, and operators for action hiding, variable hiding, and urgent actions. The concrete format [53] adds inputs, outputs and different forms of scoping to enable modular and hierarchical specifications. The concepts of the *CIF* are discussed in more detail in the following sections.

12.2.1 Differential algebraic equations

Modeling of physical systems, such as mechanical or chemical systems frequently leads to DAEs. Algebraic constraints can also be the result of stateless components such as proportional controllers. DAEs can be modeled and simulated using languages such as *Modelica* [623] and *EcosimPro* [216]. DAEs can be specified in the invariants of an interchange automaton, since such invariants are predicates over all variables, including the dotted variables. Flow clauses are supported for reasons of compatibility with existing hybrid automata. The reason for not enforcing a separation between invariants (over non-dotted variables) and flow clauses (over dotted variables), as in existing hybrid automata, is that such a separation is absent in the mathematical theory of dynamical systems, including control theory. In many cases, fully implicit DAEs, such as $f(\dot{x}, x, y, t) = 0$, cannot even be rewritten to a form where the algebraic constraints and the differential constraints are separated, such as the semi-explicit form $\dot{x} = g(x, y, t)$, $h(x, y, t) = 0$, where x and y are the continuous and algebraic variables, respectively. The generalized invariant allows us to consider the four expressions $x = 1 \wedge \dot{x} = 2$, $\dot{x} = 1 \wedge x = 2$, $\dot{x} = y \wedge \dot{x} = 2y \wedge y = 1$ and “false” to be equivalent (bisimilar): no behavior is possible.

The initialization clause of the interchange automaton is also defined as a predicate over all variables, including the dotted variables. This allows more general initializations than usually allowed in hybrid automata. In particular, steady state initialization, as available in *Modelica* and *EcosimPro*, is supported. For example, by defining $\dot{x} = 0$ as initialization predicate for a location with invariant $f(\dot{x}, x, y, t) = 0$, the initial state is defined as the “steady state,” that is, the solution of the set of DAEs such that all derivatives are zero: $f(0, x, y, t) = 0$.

12.2.2 Discrete, continuous, and algebraic variables

The interchange automaton defines three classes of variables: the discrete and continuous variables, and in addition the algebraic variables. Continuous variables are

the only variables for which dotted variables (derivatives) can be used in models. The values of discrete variables remain constant when model time progresses, the values of continuous variables may change according to a continuous function of time when model time progresses, and the values of algebraic variables may change according to a discontinuous function of time. The values of the discrete and continuous variables do not change in action transitions unless such changes are explicitly specified, for example by assigning a new value to such a variable. The values of algebraic variables can change arbitrarily in action transitions, unless such changes are explicitly restricted, for example by assigning a new value to such a variable.

There is a difference between the different classes of variables with respect to how the starting values of the variables in a transition relate to the resulting values of the variables of the previous transition. For a discrete or continuous variable, the starting value of a transition always equals the resulting value of the previous transition. For algebraic variables there is no such relation, because algebraic variables are not part of the state. This means that the starting value of the trajectory (the values of a variable as a function of time when time passes) of a discrete or continuous variable equals its value in the state. The starting value of the trajectory of an algebraic or dotted variable can be any value that is allowed by the invariant and flow conditions (and possible also the initial condition) of the automaton.

The state of an interchange automaton consists of the discrete and continuous variables. The values of the dotted variables and the algebraic variables are not contained in the state. The reason for this is that the state of an interchange automaton represents all information needed to determine future behavior, i.e. the state of a system makes the system's history irrelevant. The dotted and algebraic variables are not needed in the state, because their values are determined by the predicates of the interchange automaton: in particular by the initial conditions, the flow conditions and the invariants.

The different semantics of the variable classes lead to different use. In most models, the values of discrete variables are defined by assignments, whereas the values of algebraic variables are defined by invariants (equations). The values of continuous variables are usually defined by invariants or flow conditions, in combination with assignments for (re-)initialization.

In most languages that allow (implicit) DAEs, such as *Modelica*, *EcosimPro*, and *Simulink* [621], the distinction between continuous and algebraic variables is implicitly made by considering all continuous variables that do not occur differentiated as algebraic. *EcosimPro* also implicitly makes the distinction between discrete and continuous variables: variables of type real that occur differentiated are continuous (unless the variable is prefixed by the DISCR qualifier, in which case it is discrete). Variables of other types (such as integer, Boolean and string) are discrete.

12.2.3 Urgency

The concept of urgency allows the passing of time up to a certain point. There are essentially two kinds of urgency:

1. Urgency that is defined for an atomic automaton by means of one or more predicates. Such predicates can be associated to a location, or to outgoing edges of the location.
2. Urgency that is defined as an operation on a composition of one or more automata. Such an operation defines a set of actions as urgent for the composition. The operation allows the passing of time up to the point when one or more of the urgent actions can be executed.

By means of urgency, the execution of actions can be given priority over the passing of time. Urgency, however, is not used to give the execution of “urgent” actions priority over “nonurgent” actions. Urgency only restricts the passing of time until a certain point of time. When such a point of time is reached, in principle all enabled actions can be executed.

The first kind of urgency is defined in many different forms. The t_{cp} (*time can progress*) predicate [486], is a predicate over the variables of the automaton and time. The predicate is associated to a location. It allows passing of time in a location for as long as the predicate is true. Related to the t_{cp} predicate is the *stopping condition* [261], which is a predicate on the variables of the automaton, also associated to a location, and which allows passing of time in a location for as long as the stopping condition is false, or, in other words, until the time point when the stopping condition is true. *Deadline predicates* [101] and *urgency predicates* [261] are associated to the edges of an automaton. Deadline predicates allow passing of time in a location until the time point that one or more deadline predicates of the outgoing edges of the location become true. Whenever a deadline predicate of an edge becomes true, the guard associated to that edge must also be true; the deadline predicate must imply the guard. Urgency predicates are similar to deadline predicates; the only difference is that they do not have the restriction that the urgency predicate should imply the guard. Urgency predicates allow passing of time in a location until the point of time that for one or more of the outgoing edges, the guard and the urgency predicate are both true.

Restricting a t_{cp} predicate as a predicate over the variables of an automaton makes it equal to the negation of a stopping condition. Deadline predicates and urgency predicates are less expressive. They can both be expressed in terms of stopping conditions [261], or as t_{cp} predicates. For example, the stopping condition of a location corresponds to the disjunction of all deadline predicates of the outgoing edges of the location. Note that a flow condition which is false in a hybrid automaton is equivalent to a stopping condition that is true, or a t_{cp} predicate that is false. The interchange automaton format adopts the t_{cp} predicate.

In simulation languages, such as *Modelica*, *EcosimPro*, and *HyVisual* [392], usually a triggering or urgent guard semantics is used, meaning that the passing of time in a location is allowed until the time point that any of the guards of the outgoing edges becomes true. This is equivalent to a stopping condition associated to the location that is the disjunction of the guards of all outgoing edges, or to a t_{cp} predicate associated to the location that is the conjunction of the negated guards of all outgoing edges.

The second kind of urgency, as for example defined in [99] and [50], is often available with restrictions only. For example, in *HyTech* [313], edges can be defined as urgent. The composition of urgent actions is required to be well-formed: “whenever two components synchronize on a label, if one transition is urgent then the other must either be urgent, or have a jump condition expressible as a guarded command with its guard being either the predicate true or the predicate false” [311]. A second restriction is that “if there exists an urgent transition from a location v to a location v' , then for all valuations satisfying the invariant of v , an urgent transition to v' should exist.”

The timed automaton language *UPPAAL* [381] defines urgent communication by allowing the definition of channels either as urgent or nonurgent, but the semantics can be defined in terms of stopping conditions: time can pass in a location for as long as (1) none of its edges synchronize via an urgent channel, and (2) for all edges that synchronize via an urgent channel the guards remain false [638].

The interchange automaton format defines the second kind of urgency by means of the urgent action operator. Note that defining this kind of urgency by means of labeling certain edges or actions as urgent may lead to bisimulation not being a congruence for parallel composition, as described in [97]. Another example is the ASAP flag that can be attached to an edge in *HyTech*. In such cases, replacing a part of a specification by another part with the same behavior may lead to different behavior of the complete system. Straightforward translations of such languages to and from the interchange format is in principle possible if each action (or edge with a certain action) is either always urgent or never urgent in a model. If the same action (or edge with a certain action) occurs both as urgent and nonurgent, it may be necessary to eliminate parallel composition before translation to the interchange format is possible.

12.2.4 Closed and open scopes

The building blocks that support hierarchy and modularity are the closed scope $\llbracket \text{scopeDecls} :: \text{automaton} \rrbracket$ and open scope $\llbracket (\text{scopeDecls} :: \text{automaton}) \rrbracket$. Here, the double colon acts as a separator between the declarations of variables, action labels, and channels on the one hand, and the specification of the behavior on the other hand. Declarations can be internal (intern keyword) or external (extern, input or output keyword). The main difference between closed and open scopes is that open scopes may have *free* variables, that is, variables that are not bound in the open scope itself, but in an outer, more global, encompassing scope. A closed scope, on the other hand, may not contain free variables: all variables must be bound in the closed scope.

Both concepts of scoping can be found in modeling languages. The concept of open scopes is often used in different syntactic forms, such as the “for loop” used in many programming and modeling languages. It uses a local iterator that can be used in the body of the for loop together with the variables that are declared at a more global level. This local iterator is essentially a local variable declared in an open scope in which the body of the for loop is executed. The concept of closed scopes can

be found in many modeling languages, such as *Modelica*, where interfaces specify the interaction points (ports) of components explicitly, and connectors are used to connect these ports. The ports and connectors can be mapped to the external variables and connect sets, respectively, of the *CIF*.

When scopes are nested, inner scopes may declare variables using a name (identifier) that is already used in a variable declaration in an outer scope. A variable used in an automaton binds to the declaration of that variable in the smallest enclosing scope that declares the variable. The search for this variable declaration starts in the smallest enclosing scope of the automaton. If the variable declaration is not found in an open scope, the search proceeds one level up in the scope hierarchy. If it is not found in a closed scope, the model is erroneous, because closed scopes may not have free variables.

By means of connect sets, variables from different (!) closed scopes may be connected such that the connected variables refer to the same variable. The following connections are possible:

- *Hierarchical connections*: The internal and external variables of a closed scope can be connected to external variables of the contained closed automata.
- *Parallel connections*: In a parallel composition of closed scopes, the external variables of each closed scope can be connected to external variables of the other closed scope.

Variables x and y are connected if they both occur in a connect set $\{x, y\}$ or if they occur in different connect sets for which the intersection is nonempty, for example $\{x, z\}$, $\{y, z\}$. Within a connect set declaration, it is not allowed that two or more variables that are declared in the same scope are connected. Action labels and channels can be connected to other action labels and channels, respectively, in a similar way as connections between variables are specified.

Section 12.3 presents several examples of the use of open and closed scopes, together with examples of the use of connect sets, internal and external variables, and channels.

12.2.5 Input and output variables

Input variables and output variables are special classes of external variables. Input variables are by definition algebraic. For each input variable, restrictions in the form of an allowed range of values and trajectories can be assumed. The level of a tank, for instance, must have a nonnegative value, and its trajectory is usually assumed to be continuous. The assumption for an input variable of an automaton is that the automaton allows any behavior for the input variable within its allowed range of values and trajectories in action and time transitions, respectively. This corresponds to the use of input variables in the automata of [16] and [251].

Note that specification of the allowed range of values and trajectories for an input variable is currently not syntactically supported by the *CIF*. Also, the requirement that an automaton should allow behavior of each input variable within the restrictions defined for that input variable is not enforced by the semantics of the *CIF* nor by the semantics of the automata of [16] and [251]. The semantics of the *CIF* does ensure

that under the assumption that an automaton does not restrict the behavior of its own input variables, the input variables can also change arbitrarily in action and time transitions when that automaton is composed in parallel with another automaton, as long as the inputs are unconnected, or connected to other input variables only. In this way, when input variables are connected to an output variable, the behavior of the connection is completely determined by the output variable. Output variables cannot be connected to output variables of parallel automata. An output variable can be connected to an output variable of an outer block.

12.3 Example: Bottle filling system

The bottle filling system as shown in Fig. 12.5 consists of a liquid storage tank, two identical bottle filling lines, and a bottle supply [436].

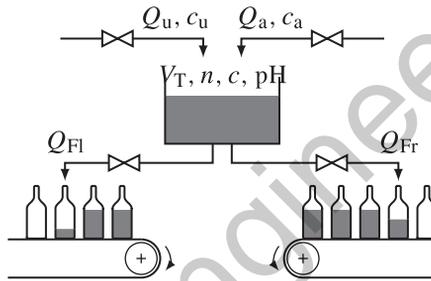


Fig. 12.5 The bottle filling system.

The bottles are filled with liquid from the storage tank. A control system keeps the volume V_T in the storage tank between 2 and 10, and the pH level (acidity) of the liquid in the storage tank between 7 and 7.1. The liquid in the storage tank slowly becomes less acidic (pH level increases). To correct this, a strong acid is dribbled into the storage tank when the acidity of the liquid becomes too low ($\text{pH} \geq 7.1$).

The acid and liquid supply processes are not modeled, since we consider the acid and liquid always to be available, and we are not interested in the amount of acid or liquid that is used.

The storage tank and the two bottle filling lines are connected by means of the variables Q_{Fl} , and Q_{Fr} , respectively. The volume of the storage tank is available in both bottle filling lines (variable V_T) to prevent filling of the bottles when the storage tank is empty.

The molar quantity and molar concentration of the acid in the storage tank are denoted by n and c , respectively, where $n = cV$. The incoming flows of liquid and acid of the liquid storage tank T are denoted by Q_u and Q_a , respectively. Acid leaves the tank in outgoing flows Q_{Fl} and Q_{Fr} . The gradual reduction of the acidity of the liquid is modeled by means of a constant K_{loss} , which leads to $\dot{n} = c_u Q_u + c_a Q_a - c Q_{Fl} - c Q_{Fr} - K_{loss} V$, where c_u and c_a denote the concentrations of acid in the

flows Q_u and Q_a . Taking into account that the units of c are in $[\text{mol}/\text{m}^3]$ instead of $[\text{mol}/\text{l}]$, the pH is given by $\text{pH} = -\log c/1000$.

In the CIF specification of the liquid storage tank, symbols Q_{seta} , Q_{setu} , c_a , c_u , and K_{loss} denote constants. The behavior of the liquid storage tank is explained as follows. Initially, the pH of the liquid in the storage tank equals 7. It is assumed that the pH level of the incoming liquid is 7 or more, since the acidity controller can only make the acidity of the storage tank increase, causing the pH to decrease. If the pH value exceeds the maximum value ($\text{pH} \geq 7.1$), the acid valve is opened ($\text{alpha}:=1$) so that acid is dribbled into the tank. Dribbling of the acid continues until the pH value comes back at 7, and the valve is closed ($\text{alpha}:=0$). In a similar way, the controller tries to keep the level of the storage tank between 2 and 10.

The behavior of the filling controller is explained as follows. When a new crate of bottles arrives, (bottles?n), where n denotes the number of bottles in a crate) the bottle volume is reset to 0, and filling a bottle is started ($(\text{VB}, \text{gamma}) := (0, 1)$). The valve switching the flow Q_F is modeled by means of the discrete variable gamma . Filling stops when the volume in the storage tank drops below 0.5 (when $\text{VT} \leq 0.5$ now do $\text{gamma}:=0$). Filling resumes when the volume in the storage tank is at least 0.7. Filling also stops when the bottle is full (when $\text{VB} > 1$ now do (gamma, n) := ($0, n-1$)).

```

model Bottle_Filling_System =
  |[ connect {tank.V, left.VT, right.VT}
           , {tank.QF1, left.QF}
           , {tank.QFr, right.QF}
           , {bs.bottles, left.bottles, right.bottles}
  :: tank:
    |[ output var V: cont real = 10
       extern var QF1, QFr: alg real
       intern var alpha, beta: disc nat = (0,0)
           ; n: cont real
           ; pH: alg real = 7
           ; c, Qa, Qu: alg real
    :: |( mode physics =
       inv dot V = Qu + Qa - QF1 - QFr
       & dot n = cu*Qu + ca*Qa - c*QF1
           - c*QFr - Kloss*V
       & n = c*V
       & pH = - log(c/1000)
       & Qa = alpha*Qseta
       & Qu = beta*Qsetu
    :: physics
    )|
  || |( mode closed = when pH >= 7.1
       now do alpha := 1 goto opened
       , opened = when pH <= 7
       now do alpha := 0 goto closed
    :: closed
    )|
  || |( mode closed = when V <= 2
       now do beta := 1 goto opened

```

```

        , opened = when V >= 10
                                now do beta := 0 goto closed
        :: closed
    )|
    ]|
|| left : Bottle_Filling_Line
|| right : Bottle_Filling_Line
|| bs : Bottle_Supply
]

automaton Bottle_Filling_Line =
|[ input var VT: real
    extern var QF: alg real
        chan bottles?: nat
    connect {QF, fp.QF},
            {VT, fc.VT},
            {fp.VB, fc.VB},
            {fp.gamma, fc.gamma},
            {bottles, fc.bottles}
:: fp : Filling_Physics
|| fc : Filling_Controller
]|

automaton Filling_Physics =
|[ input var gamma: nat
    output var VB: cont real
    extern var QF: alg real
:: |( mode m = inv dot VB = QF
        & QF = gamma*QsetF
    :: m
    )|
]|

automaton Filling_Controller =
|[ input var VB, VT: real
    output var gamma: disc nat = 0
    extern chan bottles?: nat
    intern var n: disc nat = 0
:: |( mode start =
        when n = 0 act bottles?n
            do (VB,gamma) := (0,1) goto filling
        when n > 0
            now do (VB,gamma) := (0,1) goto filling
    , filling =
        when VT <= 0.5
            now do gamma := 0 goto stopped
        when VB >= 1
            now do (gamma,n) := (0,n-1) goto start
    , stopped =

```

```

when VT >= 0.7
    now do gamma := 1 goto filling
:: start
)|
]|

automaton Bottle_Supply =
|[ extern chan bottles!: nat
  intern clock t
:: |( mode m = when t >= 2 act bottles!24
    do t:= 0 goto m

:: m
)|
]|
    
```

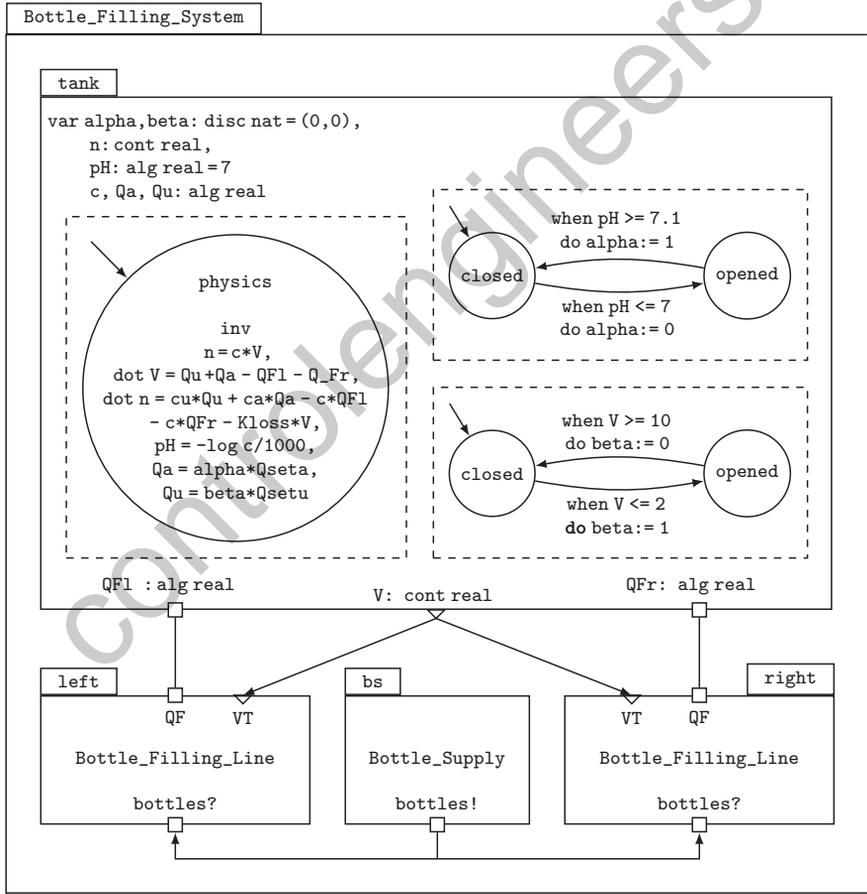


Fig. 12.6 Graphical representation of the CIF model representing the bottle filling system.

Figure 12.6 shows a graphical representation of (a part of) the *CIF* model of the bottle filling line. Solid (dashed) boxes represent closed (open) scopes, where the internal declarations are listed in the upper left corner, and the external declarations are represented as boxes (for variables) or triangles (for channels) on the borders of the box. Modes are visualized by means of circles, edges are represented as arrows between modes, and labelled with their guard, action, and update. Furthermore, urgent edges are labelled with the keyword `now`.

controlengineers.ir

Part III

Applications

controlengineers.ir

controlengineers.ir

Energy management

M. Morari, A. G. Beccuti, S. Mariéthoz, and G. Papafotiou

Energy management is an issue in many applications ranging from consumer electronics to power systems. The operation of energy management systems involves many severe hybrid phenomena, mostly due to the presence of switches. As a consequence, control and optimization play a crucial role to enable increased performance and reliability, as illustrated by two examples: the control of DC-DC converters and the model-predictive supervision of the voltage stability of power networks.

Chapter contents

13.1	Introduction to energy management	page 378
13.2	DC-DC converter control	379
13.2.1	Survey of the control methods used	379
13.2.2	System description and control model derivation	379
13.2.3	Control problem	381
13.2.4	Explicit model-predictive control	381
13.2.5	Sampled data control for robust tracking	384
13.2.6	Relaxed dynamical programming	386
13.2.7	Stabilizing control approach	388
13.2.8	Case studies	390
13.3	Supervisory hybrid model-predictive control for voltage stability of power networks	393
13.3.1	Control problem	393
13.3.2	Power network system	394
13.3.3	Emergency voltage control	397
13.3.4	Model-predictive control	397
13.3.5	Simulation results	401
13.3.6	Conclusions and future research	402
13.4	Summary and outlook	403

13.1 Introduction to energy management

Three different categories of applications can be distinguished in the area of energy management depending on the time scale at which these systems evolve:

- Power generation involves rotating machines that are used for electricity generation and for which the physical phenomena that determine the system behavior occur in the millisecond range for the magnetic part of the machine and in the range of a second for its mechanical part.
- Power transmission is done through a network and the state of the system is characterized by the voltages at the nodes and active and reactive power flows between the nodes. Such a system typically evolves at time scales varying from seconds to minutes.
- Power conversion involves power electronics devices for which the switching frequency is typically between kilohertz and megahertz.

The first category is not considered in this chapter, the focus will be on the last two. The description of the dynamical evolution of a power transmission network typically requires a large number of states (typically tens or even hundreds), while power conversion involves smaller systems with a reduced number of states (typically two to ten).

A large amount of computational power is generally available to control power systems and a relatively long sampling time is sufficient. On the other hand, power converters are usually controlled at a high switching frequency, using only a simple microcontroller or digital signal processor. As a consequence of these scale differences, the control schemes that are used for these two categories of systems are generally different in type and complexity.

Complex optimization tools are generally required for the control of power systems, while methods that are fast and simple to implement are generally used for the control of power converters. One common aspect of power transmission networks and power converters is that they feature switched elements, which make their dynamical behavior hybrid.

In [Section 13.2](#), several control schemes from optimal and hybrid control are applied to the control of a DC-DC boost converter.

In [Section 13.3](#), the application of on-line model-predictive control to the voltage control of a power system is presented.

13.2 DC-DC converter control

13.2.1 Survey of the control methods used

This section describes the high-performance control of a fixed-frequency synchronous boost (step-up) DC-DC converter. Boost converter control presents a number of challenges, starting with the switched nature of the system dynamics that directly accounts for its hybrid characteristics. Even if the classic averaging approach [451] were to be applied, the resulting expression would still be nonlinear, and the non-minimum phase behavior and input/state constraints additionally complicate the controller design process. In the recent past, we have seen the emergence of digital control [435] as an increasingly viable option for power electronics, and the availability of computational power has created interest in the investigation of alternative innovative control methods.

Four different control methodologies are presented by four research groups, identified according to their affiliations. The methods that are presented in this section have been investigated on the same benchmark problem by four research groups respectively affiliated to ETH Zürich, KTH, LTH, and SUPELEC Rennes and have been presented in [49, 471, 472]. ETH utilizes a (hybrid) piecewise affine (PWA) approximation of the converter dynamics within an explicit model-predictive control framework including duty cycle and inductor current constraints. The KTH team uses an extension of sampled-data \mathcal{H}_∞ -control theory to pulse-width modulated systems. An outer feedback loop takes care of state and control constraints and averaged sampling is used in order to achieve robust tracking. LTH employs the relaxed dynamical programming formulation from [657], where it is possible to take state and control constraints into account. The approximate optimal controller provides guaranteed robustness and stability margins. Finally, SUPELEC employs a stabilizing approach using a Lyapunov function, deduced from energetic considerations. The three first methods generate the switch control signal through a pulse width modulator (PWM), the control signal being a duty cycle bounded between 0 and 1, while the last directly controls the switch state, where the control variable is boolean.

13.2.2 System description and control model derivation

The system schematic is shown in Fig. 13.1. The converter is supplied by an unregulated DC voltage source and it provides a regulated DC voltage to a variable ohmic load. The converter comprises the inductor L and capacitor C and a switching cell composed of the MOSFET switch T_L and the diode D_H . The coil value and the minimum load current are such that the coil current does not reach zero in normal operation and the converter always operates in continuous conduction. The switching cell, therefore, only presents two modes of operation.

The control model used for synthesis is shown in Fig. 13.2. By defining $\mathbf{x}(t) = [i_\ell(t) \ v_c(t)]^T$ as the state vector, where $i_\ell(t)$ is the inductor current and $v_c(t)$ the capacitor voltage, and with a given duty cycle $d(k)$ for the k -th period, the system is described by the following pair of affine continuous-time state-space equations:

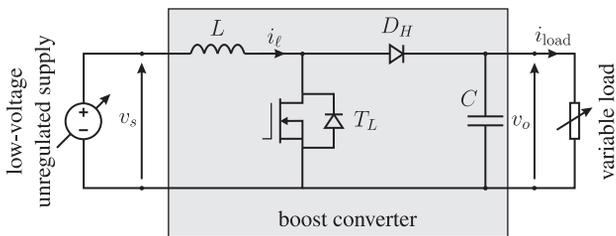


Fig. 13.1 Boost converter.

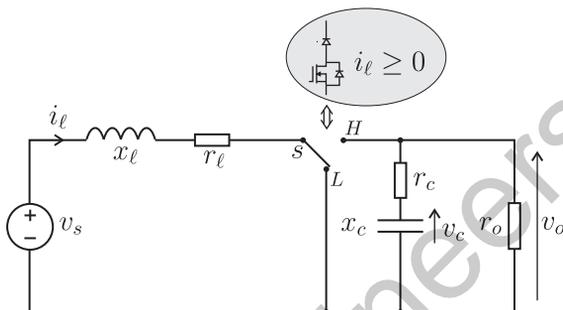


Fig. 13.2 Lumped parameter circuit used for control synthesis.

$$\dot{x}(t) = \begin{cases} \mathbf{F}_1 \mathbf{x}(t) + \mathbf{f} v_s, & kT_s \leq t < (k + d(k))T_s, \\ \mathbf{F}_2 \mathbf{x}(t) + \mathbf{f} v_s & (k + d(k))T_s \leq t < (k + 1)T_s. \end{cases} \quad (13.1)$$

where the first equation holds when s is in the L position and the second when it is in the H position, and where \mathbf{F}_1 , \mathbf{F}_2 , \mathbf{f} are given by

$$\mathbf{F}_1 = \begin{pmatrix} -\frac{r_l}{x_l} & 0 \\ 0 & -\frac{1}{x_c(r_o+r_c)} \end{pmatrix}, \quad (13.2a)$$

$$\mathbf{F}_2 = \begin{pmatrix} -\frac{1}{x_l} \left(r_l + \frac{r_o r_c}{r_o+r_c} \right) & -\frac{1}{x_l} \frac{r_o}{r_o+r_c} \\ \frac{1}{x_c} \frac{r_o}{r_o+r_c} & -\frac{1}{x_c} \frac{1}{r_o+r_c} \end{pmatrix}, \quad (13.2b)$$

$$\mathbf{f} = \begin{bmatrix} \frac{1}{x_l} \\ 0 \end{bmatrix}. \quad (13.3)$$

The output voltage $v_o(t)$ across the load r_o is expressed as a function of the states through

$$\begin{cases} v_o(t) = \mathbf{g}_1^T \mathbf{x}(t), & kT_s \leq t < (k + d(k))T_s, \\ v_o(t) = \mathbf{g}_2^T \mathbf{x}(t) & (k + d(k))T_s \leq t < (k + 1)T_s. \end{cases} \quad (13.4)$$

where

$$\mathbf{g}_1 = \left(0 \quad \frac{r_o}{r_o+r_c} \right)^T, \quad (13.5a)$$

$$\mathbf{g}_2 = \left(\frac{r_o r_c}{r_o+r_c} \quad \frac{r_o}{r_o+r_c} \right)^T. \quad (13.5b)$$

13.2.3 Control problem

The main objective is to steer the DC component of the output voltage to its reference value $v_{o,\text{ref}}$. This is achieved through an appropriate selection of the duty cycle with the exception of the method presented in Section 13.2.7, which should be kept constant during steady-state operation in order to avoid undesired phenomena (such as subharmonic oscillations) and which must be maintained in the face of measurable voltage source variations and unmeasurable load changes. Hard constraints are also present since the duty cycle is physically bounded between 0 and 1 and a limit $i_{\ell,\text{max}}$ is imposed on the inductor current as a safety measure to avoid saturation. The control problem is further complicated by the uncertainty and variation of the component values on which the controller synthesis process is based and by the nonminimum phase behavior of the output voltage with respect to the duty cycle.

Recapitulating, the goal is to develop synthesis methods subject to the following conditions:

1. Only one switch per period is allowed and the duty cycle must satisfy the constraint $d(k) \in [0, 1]$.
2. The inductor current must satisfy the constraint $i_{\ell} \leq i_{\ell,\text{max}}$, where $i_{\ell,\text{max}}$ is the pre-set bound.
3. The output voltage regulation should be maintained with an accuracy of $\pm 3\%$ unless the current-limit constraint is active.
4. The design must be robust to variations in the voltage source and load and to parametric uncertainty. The load can even be more complex than purely resistive.
5. The controller must be simple to implement and should not require excessive computation times.

Voltage regulation is well suited to achieve robustness to uncertainty and disturbances in the load and in the supply voltage, which suggests the use of output feedback. It is, however, relatively well known that the dynamical behavior can be improved if the inductor current is also present in the feedback control law. The dynamical behavior can further be improved if the supply voltage is also measured. The quantities that can be measured and used for control feedback are the source and output voltages and the current of the inductor. Some of these variables can be obtained through a state observer (inductor current) or a disturbance estimator (supply voltage or load current).

13.2.4 Explicit model-predictive control

Prediction model derivation From an implementation point of view, it is preferable that all states used in the prediction model be directly measurable. Thus, the capacitor voltage is replaced by the output voltage in the state vector which leads to

setting $\mathbf{x}(t) = [i_\ell(t) v_o(t)]^T$ by correspondingly reformulating (13.1). Additionally, to account for variations in the voltage source v_s directly, the optimal control law (to be derived) would need to be parametrized by v_s . To obviate this requirement as further explained below, the voltage source v_s is removed from the model equations by redefining the scaled state vector $\mathbf{x}'(t) = [i'_\ell(t) v'_o(t)] = [\frac{i_\ell(t)}{v_s} \frac{v_o(t)}{v_s}]$. Next, a discrete-time model is formulated by employing a sampling interval equal to the switching period T_s . The exact discrete state update is given by the nonlinear expression

$$\mathbf{x}'(k+1) = \Phi(d(k))\mathbf{x}(k) + \Gamma(d(k)), \quad (13.6)$$

where $\Phi(d(k))$ and $\Gamma(d(k))$ are matrices that depend on the duty cycle $d(k)$, calculated by integrating the converter equations from $t = kT_s$ to $t = (k+1)T_s$. This nonlinear model cannot directly be employed to derive the explicit controller and a PWA approximation of the form

$$\mathbf{x}'(k+1) = \bar{\mathbf{A}}_i \mathbf{x}'(k) + \bar{\mathbf{B}}_i d(k) + \bar{\mathbf{f}}_i, \quad (13.7a)$$

$$\text{if } d(k) \in D_i \quad i = 1, \dots, \nu, \quad (13.7b)$$

$$0 \leq d(k) \leq 1 \quad (13.7c)$$

is sought, where matrices $\bar{\mathbf{A}}_i$, $\bar{\mathbf{B}}_i$, and $\bar{\mathbf{f}}_i$ are obtained by using a direct least squares fitting approximation over each region D_i of the control input of (13.6), such that the sum of quadratic error terms (between the exact system update and the sought PWA approximation)

$$\{\Phi(d(k))\mathbf{x}(k) + \Gamma(d(k)) - (\bar{\mathbf{A}}_i \mathbf{x}(k) + \bar{\mathbf{B}}_i d(k) + \bar{\mathbf{f}}_i)\}^2 \quad (13.8)$$

is minimized over a grid of points $\mathbf{x}'(k)$ in the state space $[0, i'_{\ell, \text{lim}}] \times [0, v'_{o, \text{lim}}]$, where D_i are the ν intervals $[0, \frac{1}{\nu}]$, \dots , $[\frac{\nu-1}{\nu}, 1]$, and $i'_{\ell, \text{lim}}$, $v'_{o, \text{lim}}$ are the limit values of the scaled inductor current and output voltage over the considered range.

Optimal problem formulation The major advantage of model-predictive control (MPC) is its straight-forward design procedure [432]. Given a model of the system, including constraints, one only needs to set up an objective function that reflects the control objectives.

The control objectives are to regulate the output voltage to its reference as fast and with as little overshoot as possible, or equivalently, to minimize the absolute scaled output voltage error $v'_{o, \text{err}}(k) = |v'_o(k) - v'_{o, \text{ref}}|$, where $v'_{o, \text{ref}} = \frac{v_{o, \text{ref}}}{v_s}$. Let $\Delta d(k) = d(k) - d(k-1)$ indicate the value of the difference between two consecutive duty cycles. This term is introduced in the objective in order to reduce the presence of unwanted chattering in the input when the system has almost reached stationary conditions by penalizing any additional variations in the duty cycle. Define the penalty matrix $\mathbf{Q} = \text{diag}(q_1, q_2)$ with $q_1, q_2 \in \mathbb{R}^+$ and the vector $\varepsilon(k) = [v_{o, \text{err}}(k), \Delta d(k)]^T$. Consider the objective function

$$J(\mathbf{D}(k), \mathbf{x}(k), d(k-1)) = \sum_{n=0}^{N-1} \|\mathbf{Q} \varepsilon(k+n|k)\|_1 \quad (13.9)$$

penalizing the predicted evolution of $\varepsilon(k + l|k)$ from k over the horizon N using the 1-norm. The control input at time-instant k is then obtained by minimizing the objective function (13.9) over the sequence of duty cycles $\mathbf{D}(k) = [d(k), \dots, d(k + N - 1)]^T$ subject to the model equations and constraints (13.7a), (13.7b), (13.7c) and the current limitation $i'_\ell(k + 1) \leq i'_{\ell, \max}$, where $i'_{\ell, \max} = \frac{i_{\ell, \max}}{v_s}$;

The resulting optimization program is referred to as the constrained finite time-optimal control (CFTOC) problem.

To account for the nonminimum phase behavior of the converter, the horizon N should be chosen to be relatively long in order to capture the inverse step-response [48]. As this would increase the complexity of the problem considerably, a simple move-blocking scheme is used whereby $d(k) = d(k + 1) = \dots = d(k + N - 1)^T$ throughout the horizon.

Explicit controller synthesis Multi-parametric programming is employed to solve the optimization problem off-line for a range of parameters. In [103] it is shown how to reformulate and solve a discrete-time CFTOC problem for a PWA system as a multi-parametric program featuring the state vector as a parameter, yielding an explicit state-feedback controller. Note that the CFTOC problem is not only a function of the parameter $\mathbf{x}'(k)$, but also of the last control move $d(k - 1)$; furthermore, as it is necessary to solve the CFTOC problem for all possible values of $v'_{o, \text{ref}}$ and $i'_{\ell, \max}$, the scaled output voltage reference and inductor current maximum limit also enter the augmented state vector, which becomes five-dimensional. Again, it should be noticed that normalizing the system equations over v_s allows us to define a model independent of the voltage source and, therefore, to obtain an explicit state-feedback law that depends on one less parameter [263].

As proven in [103] the optimal state-feedback control law $d^*(k)$ is a PWA function of the (augmented) state vector defined on a polyhedral partition of the feasible (augmented) state space. As a result, such a state-feedback controller can be implemented on-line, since computing the control input amounts to determining the polyhedron in which the measured state lies and then simply evaluating the corresponding affine control law. Additionally, the derived feedback controller allows us to derive an explicit representation of the closed-loop system, for which a Lyapunov function certifying exponential stability can be sought through the method presented in [234].

External integration loop To account for load resistance variations, the derived controller is embedded within an elementary integration loop which measures the error between the output voltage and its desired values and correspondingly adjusts the reference fed into the controller, thus compensating the prediction error resulting from the fact that the derived state-feedback controller has been obtained for a time-invariant and nominal load.

13.2.5 Sampled data control for robust tracking

The control synthesis is based on a sampled-data (SD) model of the boost converter. The SD model provides a precise description of the system dynamics at the switching instances and it allows the effect of continuous time disturbances and model uncertainty to be accounted for exactly in an equivalent discrete-time model.

Within the SD framework we consider \mathcal{H}_∞ -synthesis and we, therefore, include an external disturbance w in the dynamics. The disturbance is chosen as an independent current source at the output to model disturbances and uncertainty in the load (Fig. 13.3).

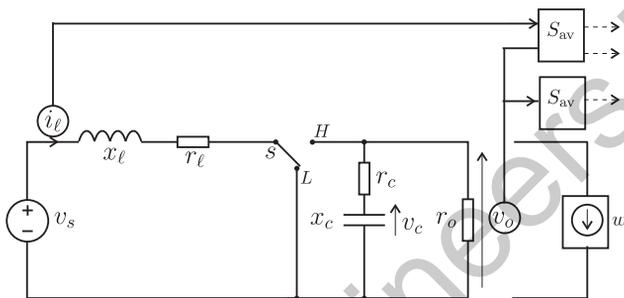


Fig. 13.3 Boost converter with load disturbance.

To derive the SD model two types of sampling are introduced. First, the ideal sampler S is defined according to $(Sf)(k) := f(kT_s)$. Second, the averaging sampler S_{av} is defined according to

$$(S_{av}f)(k) := \frac{1}{T_s} \int_{(k-1)T_s}^{kT_s} f(t) dt.$$

The quantities that are sampled are the inductor current i_ℓ and the output voltage v_o (Fig. 13.3). We define

$$\psi_1(k) = (S_{av}v_o)(k), \quad \psi_2(k) = \left(S \begin{bmatrix} i_\ell \\ v_o \end{bmatrix} \right) (k) = \begin{bmatrix} i_\ell(kT_s) \\ v_o(kT_s) \end{bmatrix}. \quad (13.10)$$

The control objective is to ensure asymptotic convergence to a nominal periodic solution (x^0, d^0) of the nominal system (i.e. the system with $w \equiv 0$) that satisfies the tracking condition

$$\lim_{k \rightarrow \infty} \frac{1}{T_s} \int_{(k-1)T_s}^{kT_s} v_o(t) dt = \int_0^{T_s} v_o^0(t) dt = v_{o,ref}, \quad (13.11)$$

where v_o^0 is the periodic output corresponding to (x^0, d^0) . We want (13.11) to be satisfied robustly in the presence of parameter uncertainties and the disturbance w , and this motivates us to introduce the integrator state

$$e_d(k) := \sum_{i=0}^{k-1} (\psi_1[i] - v_{o,\text{ref}})$$

and consider the objective of satisfying

$$\int_0^t \|\xi(t) - \xi^0(t)\|_2^2 dt + \sum_{k=0}^{\infty} q \|e_d(k)\|^2 \leq \gamma^2 \int_0^t \|w\|_2^2 dt, \quad (13.12)$$

for all $t \geq 0$ and for all solutions to the dynamics in (13.1). Here, ξ is an auxiliary output that is chosen to tune the controller and ξ^0 corresponds to the stationary periodic solution (x^0, d^0) .

To solve the problem of satisfying (13.12) subject to the dynamics in (13.1) we derive an equivalent lifting representation of the system. The lifted system has a finite dimensional discrete-time state (describing the converter state at the switching instants kT_s) and a continuous-time output that gives a complete description of the converter output (cf. [256, 257] for details).

As was mentioned in Section 13.2.3, the output voltage is nonminimum phase with respect to the duty cycle. This problem and the problem of multiple steady-state equilibria can be bypassed by formulating a current (rather than voltage) regulation problem. However, in the current regulation approach one must choose an inductor current reference that necessarily depends on the load which must be estimated.

The lifted system discussed above depends on $d(k)$ in a highly nonlinear fashion. Thus we linearize the system around d^0 and consider a quadratic approximation of the design criterion (13.12). The result is a new type of sampled-data \mathcal{H}_∞ control problem which is solved using loop shifting [257] to obtain a state feedback vector K .

The full state is not measured, but a nominal estimate of the state is obtained by linear transformation of the measured quantities i_ℓ and v_o .

The sampled-data controller is surrounded by an outer loop, which, if necessary, will adjust the duty cycle and system state. The outer loop is motivated by a number of reasons. First, the SD controller has integral action and we therefore add an anti-windup structure. If the linear feedback saturates, then the term

$$\Delta = d^0 + K(x(k) - x^0) - d_k$$

is used to modify the integrator state in a linear fashion;

$$e_d(k+1) = e_d(k) + (\psi_1(k) - v_{o,\text{ref}}) + c\Delta,$$

where $c > 0$. Second, the state constraint $i_\ell \leq i_{\ell,\text{max}}$ is not considered in the SD synthesis and needs to be dealt with by some additional control structure. We add an algorithm which determines the maximum admissible duty cycle by inverting a nonlinear system approximation. The duty cycle is saturated (if necessary) by this value, see [471] for details. Finally, the SD controller is designed for a fixed nominal input voltage. Changes in the input voltage are handled by the integrator state, but the response is made faster by using measurements of the input voltage in a feed forward fashion. We note that the outer loop remains inactive under normal operation.

13.2.6 Relaxed dynamical programming

Except for special cases, the computations required to solve a synthesis problem by means of *exact* dynamical programming are prohibitive. The only possibility is to resort to approximations. In this section, we will use the algorithms proposed by [658], which are extensions of the Relaxed Dynamic Programming techniques proposed by [407]. This choice was made for several reasons. First, important design criteria for the problem considered in this section are constraints on system variables. This can be accounted for in the method we use. Moreover, the controller we design will approximate a *stationary* optimal controller. As such, it will inherit robustness margins from the optimal controller.

Since the algorithms of [658] require the system dynamics to be affine, we need to approximate the converter dynamics (13.1) with such a system. The modeling technique presented below, which may be referred to as a robust affine approximation, is proposed in order to take into account, at the modeling stage, the switched nature of the converter and the fact that the converter is parametrized by unknown parameters (such as the load r_o).

Robust affine model approximation The exact state propagation between time kT_s and $(k+1)T_s$ is obtained by integrating (13.1):

$$\mathbf{x}((k+1)T_s) = \Phi(d(kT_s), r_o)\mathbf{x}(kT_s) + \Gamma(d(kT_s), r_o). \quad (13.13)$$

The load parameter r_o has been appended to emphasize that the matrices depend on the load. We need to approximate this nonlinear system with an affine system

$$\mathbf{x}((k+1)T_s) = \hat{\Phi}\mathbf{x}(kT_s) + \hat{\Gamma}d(kT_s) + \hat{\nu}. \quad (13.14)$$

When the model (13.13) is approximated with the model (13.14) the largest point-wise error can be expressed as

$$J = \sup \|\hat{\Phi}\mathbf{x} + \hat{\Gamma}d + \hat{\nu} - (\Phi(d, r_o)\mathbf{x} + \Gamma(d, r_o))\|$$

where the supremum is taken over $(\mathbf{x}, d, r_o) \in \mathbb{X} \times \mathbb{D} \times \mathbb{L}$, where $\mathbb{D} = [0 \ 1]$ and $\mathbb{X} = [0 \ i_{\ell, \text{lim}}] \times [0 \ v_{o, \text{lim}}]$ is the set of states on which the model should be approximated. \mathbb{L} is the set of values that the load can assume. Naturally, we would like minimize J . The robust approximation problem is to compute

$$\min J(\hat{\Phi}, \hat{\Gamma}, \hat{\nu})$$

over $(\hat{\Phi}, \hat{\Gamma}, \hat{\nu})$. Our ability to solve this problem depends on the choice of norm and the description of the set $\mathbb{X} \times \mathbb{D} \times \mathbb{L}$, the candidates are those that correspond to a finite dimensional convex optimization problem. We shall consider a simple choice. Define a finite grid of points $G \subset \mathbb{X} \times \mathbb{D} \times \mathbb{L}$, for each $\mathbf{g} = [\mathbf{x}_g^T \ d_g \ r_g] \in G$ define

$$\mathbf{b}(\mathbf{g}) = \Phi(u_g, r_g)\mathbf{x}_g + \Gamma(d_g, r_g), \text{ and } \mathbf{A}(\mathbf{g}) = \mathbf{g} \otimes \mathbf{I},$$

where \otimes denotes the Kronecker product of two matrices. If we also define $\mathbf{y} = \text{vec}([\hat{\Phi} \quad \hat{\Gamma} \quad \hat{\nu}])$, i.e. the decision variables are stacked in the vector \mathbf{y} using the vectorization operation denoted by vec , the approximation problem becomes

$$\min_{\mathbf{y}} \max_{\mathbf{g} \in G} \|\mathbf{A}(\mathbf{g})\mathbf{y} - \mathbf{b}(\mathbf{g})\|,$$

which is the same as

$$\min_{\mathbf{y}, t} t$$

$$\|\mathbf{A}(\mathbf{g})\mathbf{y} - \mathbf{b}(\mathbf{g})\| \leq t, \quad \forall \mathbf{g} \in G.$$

If the norm is either $\|\cdot\|_\infty$ or $\|\cdot\|_1$ this is an LP, if norm is $\|\cdot\|_2$ the problem is a second order cone problem. In any case, it is an easily solvable finite-dimensional convex optimization problem.

Control design To simplify notation we define $\mathbf{e}(\mathbf{x}) = (\mathbf{x}^T \mathbf{1})^T$ and $\mathbf{e}(\mathbf{x}, d) = (\mathbf{x}^T \ d \ \mathbf{1})^T$. Our goal is to synthesize a feedback controller

$$d(kT_s) = \mu(\mathbf{x}(kT_s))$$

such that the total cost $V = \sum_{k=0}^{\infty} l(\mathbf{x}(kT_s), d(kT_s))$ is approximately minimized, under the additional constraint on the inductor current, $x_1(kT_s) \leq 2.5$ and $0 \leq d(kT_s) \leq 1$. The following stage cost was used:

$$l(\mathbf{x}, d) = q_1 |v_o - v_{o,\text{ref}}| + q_2 |d(kT_s) - d((k-1)T_s)|,$$

where q_1 and q_2 are positive weights. The penalty on consecutive control values was introduced to force the duty cycle to become constant when v_o has reached the output reference. Thus, an extra state $\mathbf{x}_e(kT_s) = d((k-1)T_s)$ was introduced. We used relaxed value iteration to solve for a stationary approximate value function \hat{V} satisfying

$$\beta V^* \leq \hat{V} \leq \alpha V^*,$$

where $\beta \leq 1 \leq \alpha$ are constants and V^* is the optimal total cost function. The approximate value function is given by a max of linear functions $\hat{V} = \max_{\mathbf{p} \in P} \mathbf{p}^T \mathbf{e}(\mathbf{x})$, where P is a set of vectors. The corresponding explicit piecewise affine feedback controller is given by $\mu(\mathbf{x}) = \mathbf{L}_p(\mathbf{x})^T \mathbf{e}(\mathbf{x})$, where $\mathbf{p}(\mathbf{x}) = \text{argmax}_{\mathbf{p} \in P} \mathbf{p}^T \mathbf{e}(\mathbf{x})$. To compute the controller value $\mu(\mathbf{x})$ at a state \mathbf{x} we need to take the following steps:

1. Find $\mathbf{p} \in P$ such that $\mathbf{p}^T \mathbf{e}(\mathbf{x})$ is maximal.
2. Set $\mu(\mathbf{x}) = \mathbf{L}_p^T \mathbf{e}(\mathbf{x})$

Thus, we have to do a linear search over the table P . Consequently, it is important to keep the table P as small as possible, and for this purpose a reduction algorithm has been outlined by [658].

Finally, the errors introduced by the model approximation are handled by an outer integrator loop that adjusts the voltage reference. The integrator is activated only when the voltage v_o is sufficiently close to its reference.

13.2.7 Stabilizing control approach

The control synthesis described in this section is a stabilizing approach using a common Lyapunov function candidate. Unlike the indirect approaches presented previously that use PWM to generate the switch control signal, it directly controls the switch state using a smaller sampling period. The boolean control variables are determined so that the system is asymptotically stable. A port control Hamiltonian (PCH) formulation is used. This formulation enables taking into account considerations about the system energy. In the case of switched systems its expression is written as

$$\dot{\boldsymbol{x}} = [\boldsymbol{J}(\rho) - \boldsymbol{R}(\rho)]\boldsymbol{F}\boldsymbol{x} + \boldsymbol{G}(\rho)\boldsymbol{u}, \quad (13.15)$$

where $\boldsymbol{x} \in \mathbb{R}^n$ in this section is the state vector containing the energy variables (fluxes in the inductors and charges in the capacitors). $\rho \in \{0, 1\}^p$ is the boolean control variable. Matrix \boldsymbol{J} is skew-symmetric ($\boldsymbol{J} = -\boldsymbol{J}^T$) and corresponds to the power interconnection in the model. Matrix \boldsymbol{R} is nonnegative and corresponds to the dissipating part of the system. Matrix \boldsymbol{G} is the power input matrix and \boldsymbol{u} represents the power sources present in the system. $\boldsymbol{F} = \boldsymbol{F}^T > 0$ and, in the simple cases, it is a diagonal matrix. The vector $\boldsymbol{z} = \boldsymbol{F}\boldsymbol{x}$ represents the co-state variables (currents in the inductors and voltages on the capacitors).

In the case of power converters, the state equation is affine with respect to the boolean variables [127]. Thus, the matrices $\boldsymbol{J}(\rho)$, $\boldsymbol{R}(\rho)$ and $\boldsymbol{G}(\rho)$ can be written as

$$\begin{aligned} \boldsymbol{J}(\rho) &= \boldsymbol{J}_0 + \sum_1^p \rho_i \boldsymbol{J}_i, \quad \boldsymbol{R}(\rho) = \boldsymbol{R}_0 + \sum_1^p \rho_i \boldsymbol{R}_i, \\ \boldsymbol{G}(\rho) &= \boldsymbol{G}_0 + \sum_1^p \rho_i \boldsymbol{G}_i, \end{aligned} \quad (13.16)$$

where ρ_i are the components of the control vector ρ and p is its dimension.

The approaches in the literature which are based on Lyapunov function consider, in general, linear systems with a common equilibrium point [206, 403]. In the case of power converters, each configuration may or may not have a different equilibrium point and physical considerations enable establishing a common Lyapunov function. This function depends on the control objective, which has to be defined first. It is obtained using the same approach as with an average model where the control variable is continuous and bounded. The control objective corresponds to an admissible reference for the system which is defined by solving (13.15) for $\dot{\boldsymbol{x}} = 0$. It is a value for the co-state variable $\boldsymbol{z}_0 = \boldsymbol{F}\boldsymbol{x}_0$, which must satisfy the constraint

$$0 = (\boldsymbol{J}(\rho_0) - \boldsymbol{R}(\rho_0))\boldsymbol{z}_0 + \boldsymbol{G}(\rho_0)\boldsymbol{E}, \quad (13.17)$$

if there is a $\rho_0 \in \mathbb{R}^p$, $0 \leq \rho_{0i} \leq 1$. According to the properties of this equation and the respective dimension of \boldsymbol{x} and ρ , for one ρ_0 , the equilibrium point can be unique or not, and for ρ_0 any point of the state space can be an equilibrium point or not [128].

For a function V to be a Lyapunov function for a system in a point x_0 it must be positive everywhere except in x_0 and its derivative must always be negative. If such a control law is applied, then x will converge asymptotically toward x_0 . The candidate Lyapunov function has the following form:

$$V(x, x_0) = \frac{1}{2} (x - x_0)^T F (x - x_0). \quad (13.18)$$

Because the matrix F is unique for all the modes of the system, V is positive and continuous for every x and it is zero only in x_0 . Its derivative depends on the control variable and using (13.15) and (13.16) it can be expressed as

$$\begin{aligned} \dot{V}_\rho = & - (z - z_0)^T R(\rho) (z - z_0) \\ & + \sum_1^p (z - z_0)^T ((J_i - R_i) z_0 + g_i u) (\rho_i - \rho_{0i}). \end{aligned} \quad (13.19)$$

Because $R(\rho)$ is a nonnegative matrix, the first term is always negative. Because of $0 \leq \rho_{0i} \leq 1$, the sum can be made negative by choosing an appropriate value for each boolean ρ_i such that each product $(z - z_0)^T ((J_i - R_i) z_0 + g_i u) (\rho_i - \rho_{0i})$ is negative.

Multiple state feedback control strategies can be envisaged for attaining this goal [128]. In the following a steepest descent strategy is used as it yields better results in terms of computation time due to a simpler expression of the commutation surfaces. It consists in choosing, at each time, the value of ρ such that all the terms in the sum are negative or zero. The commutation surfaces are then p hyperplanes defined by

$$T_i = (z - z_0)^T ((J_i - R_i) z_0 + g_i u) = 0. \quad (13.20)$$

In the case of the boost converter, as there is only one control variable, the sum from expression (13.19) has only one term T , which, according to (13.20) becomes

$$T = \frac{r_o i_{\ell,0}}{r_c + r_o} (v_o - v_{o,0}) - \frac{r_o r_c i_{\ell,0} + r_o v_{o,0}}{r_c + r_o} (i_\ell - i_{\ell,0}), \quad (13.21)$$

where the admissible reference is computed by solving (13.17) for the nominal value of the output voltage $v_{o,0}$.

Because this strategy requires an infinite bandwidth a dead zone is created with the help of a parameter ϵ . This way the derivative of the Lyapunov function may take positive values for a limited amount of time. The new commutation surfaces are thus defined by $T = \epsilon$. The period and the amplitude of the oscillations around the reference depend on this parameter.

To ensure the robustness with regard to the parameter variations and to improve the start-up performance, the admissible reference is modified on-line. A new value for the current reference, i_{0n} , is computed under the following form:

$$i_{0n} = i_{\ell,0} + (v_{o,0} - v_o) \frac{k}{r_o}, \quad (13.22)$$

where k is a parameter. i_{0n} is bounded between 0 and the maximal admissible value for the current in the inductor.

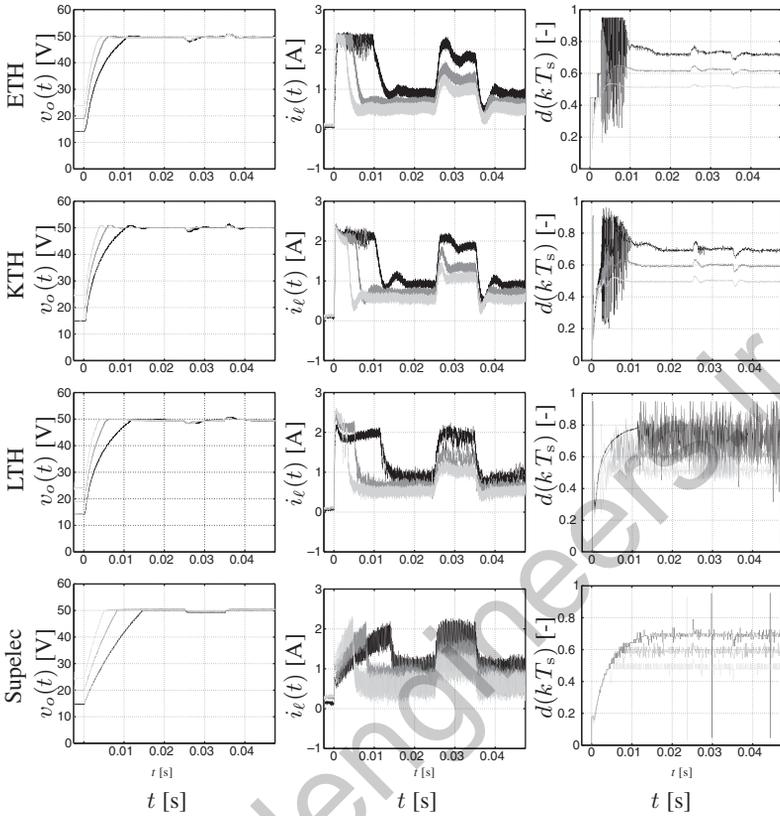


Fig. 13.4 Start-up followed by a load transient pulse (nominal 0.25 A to 0.5 A). Plots for three supply voltages are superimposed: minimum (15 V black), nominal (20 V dark gray), maximum (25 V light gray).

13.2.8 Case studies

Circuit parameters

We choose x_ℓ to be 2 mH, x_c to be 100 μF , the nominal load r_o to be 200 Ω , and the sampling and switching periods to be 50 μs . The coil current must not exceed 2.5 A in order to avoid core saturation.

Synthesis parameters for ETH method described in Section 13.2.4 The explicit controller model is derived with 3 PWA dynamics, the intervals D_i being $[0, 0.45]$, $[0.45, 0.6]$, and $[0.6, 0.9]$. The penalty matrix is chosen to be $\mathbf{Q} = \text{diag}([10, 1])$ and the prediction horizon $N = 18$.

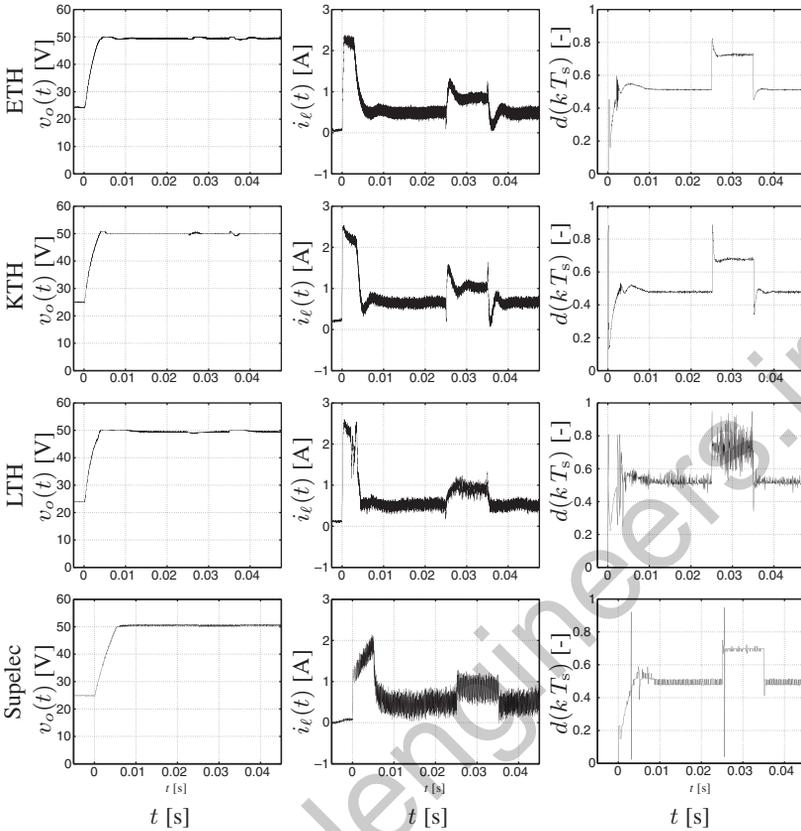


Fig. 13.5 Start-up followed by a line transient pulse from 25 V to 15 V.

Synthesis parameters for KTH method described in Section 13.2.5 The cost criterion (13.12) is considered with $\xi = [0.5, 1]$, $q = 2$ and $\gamma = 3$ to synthesize a controller with the sampled-data control for robust tracking method. As explained in Section 13.2.5, the feedback was implemented with an anti-windup structure where the gain was $c = 0.5$.

Synthesis parameters for LTH method presented in Section 13.2.6 The weights in the step-cost function were chosen as $q_1/q_2 = 1/5$. An approximate cost function \hat{V}^{38} was found after 38 value function iterations, with relaxation parameters $\beta = 0.4$ and $\alpha = 3.2$. After reduction, the size of the resulting lookup table was 130.

Synthesis parameters for SUPELEC method described in Section 13.2.7 The equation (13.21) was implemented and throughout the different scenarios a unique value was used for the ϵ parameter (fixed at $\epsilon = 5$). The sampling frequency was 120 kHz.

Experimental results The two most important tests that allow evaluating the performance of a DC-DC converter are the line transient (supply voltage variation) and the load transient (output current variation). The output voltage fluctuation must be minimal during these transients. The start-up transient is also a relevant test as constraints such as the current limit or maximum duty cycle are typically reached during this phase.

In Fig. 13.4, the performance of the controllers is evaluated at start-up for three different supply voltages ranging from the lowest possible supply voltage to the highest (low 15 V, nominal 20 V, high 25 V). If the controller is at steady state, a load variation from nominal load (0.25 A) to 0.5 A is applied from $t = 25$ ms to $t = 35$ ms. The hard current constraint ($i_\ell(t) < 2.5$ A) is respected in all cases during the startup transient. The respect of this constraint imposes a minimum time for the duration of the capacitor voltage rise time of approximately 3 ms for the highest supply voltage. All methods are close to this minimum time and the overshoot that can be observed on some plots is smaller than 3% and is mostly due to the outer integral loop that some methods feature to be robust to load variations. The magnitude of the ripple that follows the load transient varies from 1.5% to 4% depending on the methods' aggressiveness.

The stabilization approach directly control the switch. For the purpose of comparison with the other methods, the pseudo duty cycle that corresponds to the ratio of times during which the switch is on and between two switchings is displayed (instead of the real control signal which is difficult to interpret).

In Fig. 13.5, the performance of the controllers is evaluated at start-up starting from maximum voltage (25 V). When the controller is at steady state, a line voltage variation from the maximum supply voltage (25 V) to the minimum supply voltage (15 V) is applied from $t = 25$ ms to $t = 35$ ms. The ripple that is observed after the supply disturbance is less pronounced than after the load disturbance (less than 1%) and this is mostly because it is measured and accounted for by the presented control schemes.

Future work The controllers that have been presented in this section feature a current limitation and present a fast dynamical behavior. The start-up transient duration is close to the lower limit that is allowed by the maximum current constraint. The control schemes have also been selected for their reduced complexity, which makes them suitable to be executed in the short period (typically a few microseconds) required for power electronics applications when implemented using a microcontroller or a digital signal processor.

Future research work mainly focuses on further reducing the controller complexity in order to diminish the computational power requirement and increase the sampling frequency. One research direction is to investigate minimal solutions that give the desired performance.

13.3 Supervisory hybrid model-predictive control for voltage stability of power networks

13.3.1 Control problem

Huge problems in the USA and Canada [227], Italy, and the Netherlands due to power outages have shown the crucial role of a reliable operation of electricity distribution and transmission networks. A reliable and efficient operation of these networks is not only of paramount importance when the electricity system is pressed to the limits of its performance, but also under regular operating conditions. Due to deregulation in the European electricity market, the number and variety of actors increases. This number will even further increase as also large-scale industrial suppliers (co-generation (i.e. combined heat and power generation)) and small-scale individual households (via solar energy or wind energy installations) will start to feed electricity into the network [339]. With this increasing complexity faults and disturbances causing voltage instabilities are likely to occur more frequently.

In general, the behavior of power systems is characterized by complex interactions between continuous dynamics and discrete events, i.e. power systems exhibit hybrid behavior. Components such as generators and loads drive the continuous dynamic behavior. They obey physical laws, and are usually represented by coupled differential and algebraic equations. Discrete events or discrete inputs cause discrete behavior through, for example, breaking down or connecting of a transmission line, saturation effects in automatic voltage regulators and power system stabilizers, *on* or *off* switching of a generator, connecting or disconnecting of load, changing of transformer ratio settings, and connecting or disconnecting of capacitor banks; seasonal variations can also cause changes in power production capabilities as well as consumption and can modify the direction of power flows and thus cause switching behavior. The networks moreover typically span a wide range of time scales and large geographical areas.

To control such complex systems, hierarchical control in which regulation takes place at different layers based on spatial and/or temporal partitioning or decoupling is necessary [80]. The controllers at the lowest layer act directly on the actuators of the physical system. At higher layers, controllers supervise the lower layers by providing set-points or specifying constraints. The task of each layer is to steer the underlying lower layer in such a way that the performance of the physical system is optimal in some sense.

At lower layers, control is faster and more localized. More detailed models of the dynamics to be controlled can be used. Controllers at this level control only a small part of the dynamics of the overall system, without explicitly taking into account the rest of the system. Primary controllers for turbine speed and bus voltage regulation at power generators are examples of such lower-layer controllers.

At higher layers, control operates at slower time scales and larger geographical regions. The models used are more abstract and less detailed than the models used at the lower layers. The models represent the dynamics of the physical system

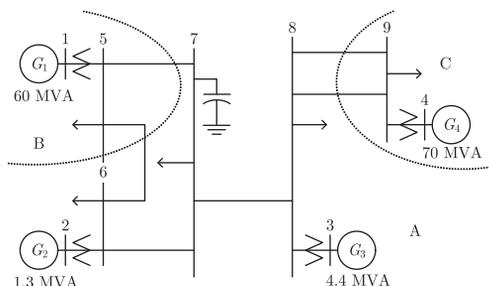


Fig. 13.6 Graphical representation of the IEEE nine-bus Anderson–Farmer network.

combined with the lower-layer controllers. Secondary controllers for voltage and frequency stability are examples of such higher-layer controllers.

In the proposed approach both continuous and discrete values are dealt with in an integrated way, i.e. a *hybrid* approach is considered.

The particular control problem under consideration is voltage stability after disturbances. After a disturbance, e.g. the outage of a transmission line, the generation and transmission network may not have sufficient capacity to provide the loads with power; voltage instability may be the result. Control actions have to be chosen that minimize negative effects of this voltage instability. Traditionally, off-line static stability studies are carried out in order to avert the occurrence of voltage instability. The proposed approach consists of an online control scheme that takes into account both the inherent temporal dynamics and that determines the most appropriate control sequence required to reach an acceptable and secure operating point. The considered scheme is used by a higher-layer controller that determines both discrete and continuous set-points for lower-layer controllers in such a way that negative effects due to voltage instability after disturbances are minimized. It is thus assumed that a lower layer that accepts set-points at discrete time steps is already present.

In Section 13.3.2 the power network and the lower layer of control are introduced. In Section 13.3.3 the voltage control problem and objectives are introduced. In Section 13.3.4 a control strategy for the higher layer based on model-predictive control is presented. Section 13.3.5 contains simulation results obtained on the considered power network.

13.3.2 Power network system

The case study under consideration is the nine-bus Anderson–Farmer network [228], depicted in Fig. 13.6.

Components of the reduced network The considered network consists of four generators G_1 , G_2 , G_3 , and G_4 (shown with their nominal apparent power ratings)

feeding the static loads at buses 5 through 9, where G_1 and G_4 and the loads connected to buses 5 and 9 are the aggregate representations for neighboring generators and loads. The synchronous machines are connected to the grid via lossless step-up transformers featuring a fixed turns ratio; a capacitor bank at node 7 provides additional reactive power to the system. The following list contains more details:

- **Generators:** Generators G_2 and G_3 represent single physical machines, whereas G_1 and G_4 denote the aggregate generators comprising several physical units. Therefore, G_2 and G_3 are described by a detailed sixth-order model [370] including the mechanical equations and the electrical transient and sub-transient dynamics, whereas G_1 and G_4 are described by second-order mechanical dynamics [370].
- **Loads:** The employed static load types comprise voltage dependent and constant impedance types [353]. The loads are described following the classical formulation in terms of active and reactive powers

$$P_h = s_h P_{0h} v_h^\alpha, \quad (13.23a)$$

$$Q_h = s_h Q_{0h} v_h^\alpha, \quad (13.23b)$$

where $h \in \{5, 6, 7, 8, 9\}$, v_h is the voltage of bus h , P_{0h} (Q_{0h}) is the active (reactive) power steady-state value at node h , and $s_h \in \{0, 0.02, \dots, 0.98\}$ per unit (p.u.) represents the discrete load shedding factor applied to a load to relieve the strain of the power demand on the system. Voltage dependent loads correspond to $\alpha = 1$ and constant impedance loads to $\alpha = 2$.

- **Capacitor bank:** The capacitor bank locally stabilizes bus voltages by injecting additional reactive power into the grid. It is represented as a (negative) purely reactive load of type (13.23b) with $\alpha = 2$ and thus describes a switched shunt capacitor.
- **Transmission lines:** The transmission lines between the buses and components transfer the power from one location to another. The lines are represented by the π model for transmission lines [370].

Primary control layer In the network there is a primary control layer that locally regulates power flows and voltage levels at the bus terminals of generators. Figure 13.7 shows a schematic representation of the local controllers' principle of operation. Feedback variables and corrective actions are depicted for each component [370]. The primary control layer consists of the following:

- **Turbine governors:** All generators feature a turbine governor (TG) controlling the mechanical power P_m acting on the shaft of the machine in order to satisfy the active power demand of the network and maintain the desired frequency $\omega_{\text{ref}} = 60$ Hz. The TG act on a time scale of tens of seconds.

- **Automatic voltage regulators:** All generators feature an automatic voltage regulator (AVR) maintaining the level of the excitation field E_{fd} in the rotor windings at the value required to keep the bus (stator) voltage close to the desired set-point. Saturation is included in the AVR to account for the maximum allowable current in the excitation system, i.e. E_{fd} has an upper limit value E_{max} and a lower limit value E_{min} . Once a machine has reached its saturation limit it cannot produce additional reactive power and can therefore no longer participate in sustaining the voltages in the network [370]. For all generators the respective AVR voltage references v_i , $i \in \{1, 2, 3, 4\}$, can be set in the range 0.9–1.1 p.u. with steps of 0.01 p.u. The AVRs act on a time scale of seconds.
- **Power system stabilizers:** Generators G_2 and G_3 feature a power system stabilizer (PSS) eliminating the presence of unwanted rotor oscillations by measuring its rotational speed ω and adding a corrective factor $v_{ref,PSS}$ to the bus terminals' voltage reference v_{ref} . Generators G_1 and G_4 feature no power system stabilizer since the faster dynamics related to the rotor oscillations are not present in the related model equations. The PSSs act on a time scale of tenths of seconds.

Controls available to a higher control layer Given the description of the network and the primary control layer, there is a number of controls available to a higher control layer in the form of set-point and reference settings. In particular the following can be adjusted:

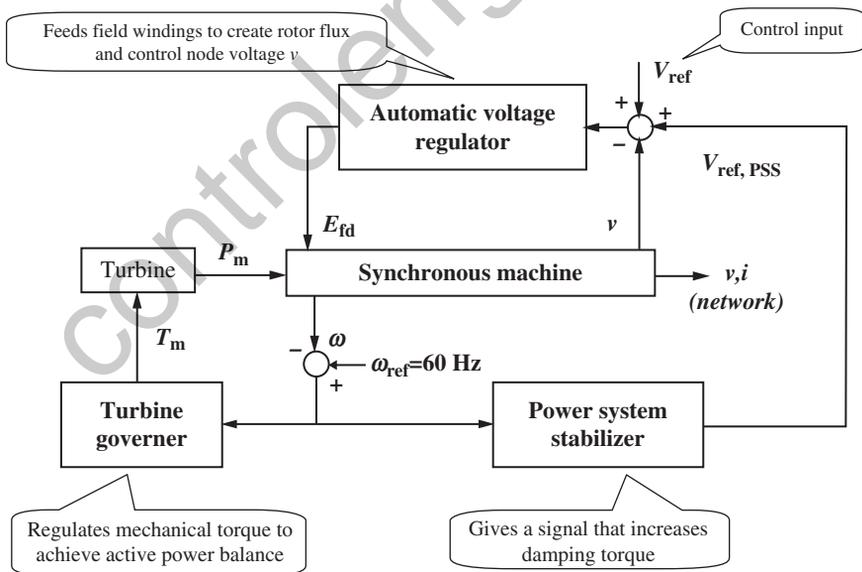


Fig. 13.7 Scheme of primary controllers at a generator.

- the voltage references for the AVRs;
- the mechanical power set-point for the TGs;
- the reference frequency for the TGs and PSSs;
- the amount of load to shed;
- the amount of capacitor banks to connect to the grid.

Depending on the particular control problem a higher-layer controller will adjust the values of these controls. In particular for the problem at hand the amount of load shed (defined by the variables s_i) and the set-points of the AVRs (defined by the variables r_i) will be taken as the available controls.

13.3.3 Emergency voltage control

A major source of power outages is voltage instability [192]. Voltage instability in general stems from the attempt of load dynamics to restore power consumption beyond the capability of the combined transmission and generation system. Typically, the capability is exceeded following the outage of one or more components in the network, such that the system cannot satisfy the load demand with the given inputs at a physically sustainable voltage profile in the network.

The control problem involves the case of emergency voltage regulation, in which the power system is initially in steady-state operation and subsequently subjected to a fault, modeled as the partial or total outage of a line. Due to the reduced transmission capacity of the network the requested load demand together with the given system configuration place the grid under an excessive amount of strain, so that corrective actions are required to avoid that the induced transients drive the system to collapse or cause unwanted and hazardous sustained oscillations. More specifically, the control objectives are the following:

1. Maintain the voltages between 0.9 and 1.1 p.u., i.e. sufficiently close to nominal values to ensure a safe operation of the system by keeping it adequately distant from low voltages, which may lead to collapse.
2. Effectively achieve a steady-state point of operation, while minimizing switching of the control inputs so that a constant and appropriate set of input values is ultimately applied to the power grid.

For this second objective, the option of load shedding is to be avoided unless absolutely necessary in order to fulfill the primary objective, as load shedding is the most disruptive countermeasure available.

13.3.4 Model-predictive control

Model-predictive control (MPC) [432] has been traditionally employed in the process industry and has shown promising performance also for a variety of other control problems [470]. The control action is obtained by minimizing an objective function at each time step over a finite horizon subject to the equations and constraints of

the employed prediction model. In a receding horizon fashion the control problem is solved at each control step.

The major advantage of MPC is its straight-forward design procedure. Given a model of the system, hard constraints can be incorporated directly as inequalities and one only needs to set up an objective function reflecting the control aim; soft constraints can also be accounted for in the objective by using penalties for violations.

Derivation of the prediction model The performance of a predictive controller relies for a large part on the accuracy of the prediction model of the system. The prediction model has to describe well how the inputs affect the system behavior. Ideally a perfect model of the system would be used. However, such a perfect model can be very complex, thus making the optimization procedure in the controller slow. Instead, an approximation is determined. If this approximation fits in a suitable form, relatively efficient optimization techniques can be used to determine the controls (e.g. linear or mixed-integer programming).

In order for the higher-layer controller to meet its control objectives, it has to be able to predict how set-point changes influence the dynamics of the network. Therefore, the controller uses a model that includes both a representation of the physical network and a representation of the primary control layer.

The network, including the primary control layer, is expressed [370] as a system of differential-algebraic equations (DAE)

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{v}), \quad (13.24a)$$

$$\mathbf{0} = \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{v}). \quad (13.24b)$$

where the state variables \boldsymbol{x} are the generator dynamic variables, \boldsymbol{u} denotes the system inputs (vectors $\boldsymbol{r}(k) = [r_1(k), \dots, r_4(k)]^T$ and $\boldsymbol{s}(k) = [s_5(k), \dots, s_9(k)]^T$) and the algebraic output variables \boldsymbol{v} are the bus voltage magnitudes. The differential equations (Eq. (13.24a)) are the synchronous machines and related primary controllers; the algebraic equations (13.24b) are the classic load flow equations.

Determining the evolution of the network given an initial state and input trajectory over the horizon thus requires the solution of this DAE. Solving DAEs in general is a complex task, in particular when dynamics of different time scales are present, as is the case for the power systems. Variable step size methods, e.g. DASSL [518], are suitable for these cases, since they automatically choose a larger step size when no fast dynamics are present, and a smaller step size when they are. These methods, however, are not particularly fast and using these inside the optimization procedure of the MPC controller results in very slow control. Therefore, such a DAE model is not directly suitable as prediction model.

Instead of taking the continuous-time DAE as prediction model, a discrete-time linearized model derived from this DAE is considered. At each discrete sampling instant kT_s the continuous-time linearization of (13.24a) and (13.24b) around $\boldsymbol{x}_0 = \boldsymbol{x}(k)$, $\boldsymbol{u}_0 = \boldsymbol{u}(k-1)$, can be written as

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{A}_c \boldsymbol{x} + \boldsymbol{B}_c \boldsymbol{u} + \boldsymbol{F}_c, \\ \boldsymbol{v} &= \boldsymbol{C}_c \boldsymbol{x} + \boldsymbol{D}_c \boldsymbol{u} + \boldsymbol{G}_c, \end{aligned}$$

where

$$\begin{aligned} \mathbf{A}_c &= \frac{\partial \mathbf{f}}{\partial \mathbf{x}} + \frac{\partial \mathbf{f}}{\partial \mathbf{v}} \left(-\frac{\partial \mathbf{g}}{\partial \mathbf{v}}\right)^{-1} \left(\frac{\partial \mathbf{g}}{\partial \mathbf{x}}\right), \quad \mathbf{B}_c = \frac{\partial \mathbf{f}}{\partial \mathbf{v}} \left(-\frac{\partial \mathbf{g}}{\partial \mathbf{v}}\right)^{-1} \frac{\partial \mathbf{g}}{\partial \mathbf{u}}, \\ \mathbf{C}_c &= \left(-\frac{\partial \mathbf{g}}{\partial \mathbf{v}}\right)^{-1} \frac{\partial \mathbf{g}}{\partial \mathbf{x}}, \quad \mathbf{D}_c = \left(-\frac{\partial \mathbf{g}}{\partial \mathbf{v}}\right)^{-1} \frac{\partial \mathbf{g}}{\partial \mathbf{u}}, \\ \mathbf{f}_c &= -\frac{\partial \mathbf{f}}{\partial \mathbf{v}} \left(-\frac{\partial \mathbf{g}}{\partial \mathbf{v}}\right)^{-1} \left(\frac{\partial \mathbf{g}}{\partial \mathbf{x}} \mathbf{x}_0 + \frac{\partial \mathbf{g}}{\partial \mathbf{v}} \mathbf{v}_0 + \frac{\partial \mathbf{g}}{\partial \mathbf{u}} \mathbf{u}_0 - \mathbf{g}(\mathbf{x}_0, \mathbf{u}_0, \mathbf{v}_0)\right) \\ &\quad - \left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}} \mathbf{x}_0 + \frac{\partial \mathbf{f}}{\partial \mathbf{v}} \mathbf{v}_0 - \mathbf{f}(\mathbf{x}_0, \mathbf{v}_0)\right), \\ \mathbf{g}_c &= -\left(-\frac{\partial \mathbf{g}}{\partial \mathbf{v}}\right)^{-1} \left(\frac{\partial \mathbf{g}}{\partial \mathbf{x}} \mathbf{x}_0 + \frac{\partial \mathbf{g}}{\partial \mathbf{v}} \mathbf{v}_0 + \frac{\partial \mathbf{g}}{\partial \mathbf{u}} \mathbf{u}_0 - \mathbf{g}(\mathbf{x}_0, \mathbf{v}_0, \mathbf{u}_0)\right) \end{aligned}$$

holds, if $\partial \mathbf{g} / \partial \mathbf{v}$ is invertible, which is typically the case for power networks. The required Jacobians can either be derived analytically or computed numerically. The latter approach is used here.

For the sake of simplicity only small variations of the variables around the linearization operating point are considered. If the variations are not small, mode changes have to be considered in the model, e.g. by using piece-wise affine or similar models.

The continuous-time linearization is discretized with the sampling interval T_s , to obtain the following control model in the affine expressions of $\mathbf{x}(k)$, $\mathbf{u}(k)$ and $\mathbf{v}(k)$

$$\mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k) + \mathbf{f}, \quad (13.25a)$$

$$\mathbf{v}(k) = \mathbf{C}\mathbf{x}(k) + \mathbf{D}\mathbf{u}(k) + \mathbf{g}, \quad (13.25b)$$

wherein k denotes the discrete time step and

$$\begin{aligned} \mathbf{A} &= e^{\mathbf{A}_c T_s}, \quad \mathbf{B} = \int_0^{T_s} e^{\mathbf{A}_c \tau} d\tau \mathbf{B}_c, \quad \mathbf{f} = \int_0^{T_s} e^{\mathbf{A}_c \tau} d\tau \mathbf{f}_c, \\ \mathbf{C} &= \mathbf{C}_c, \quad \mathbf{D} = \mathbf{D}_c, \quad \mathbf{g} = \mathbf{g}_c. \end{aligned}$$

The simulation sampling time T_s has to be chosen such that the discrete-time approximation adequately reflects the dynamics of the continuous-time linearized model.

The obtained discrete-time approximation is employed as a prediction model in the optimal control problem formulation. In this regard, the optimal control formulation must be augmented with the appropriate hard constraints on the inputs $\mathbf{u}(k) = [\mathbf{r}(k)^T \mathbf{s}(k)^T]^T$, which are physically bounded. For $\mathbf{r}(k)$ the admissible range is simply taken to be the continuous relaxation of the discrete physical values, since adjusting AVR set-points is not invasive. However, load shedding is more invasive and since it is an extremely expensive control action such an approximation might not be adequate. Therefore, for $\mathbf{s}(k)$ the control constraints are taken as the actual discrete physically feasible values, at the cost of introducing a set of integer variables in the model. The employed control model is, therefore, by necessity *hybrid* in nature.

Optimal control problem To account for the control objectives mentioned in Section 13.3.3 with their related order of importance a cost function is formulated similarly as in [264]. To maintain the voltages v_1, \dots, v_9 between 0.9 and 1.1, let the slack variables $t_j, j = 1, \dots, 9$ defined by

$$\begin{cases} 0.9 - v_j(k) \leq t_j(k), \\ -1.1 + v_j(k) \leq t_j(k), \\ 0 \leq t_j(k) \end{cases} \quad (13.26)$$

denote the amount of violation of this condition. This formulation leads to nine slack variables at each sampling instant k , grouped in the vector

$$\mathbf{t}(k) = [t_1(k), \dots, t_9(k)]^T.$$

To minimize the switching between control actions, define the variation of the manipulated variables as

$$\Delta \mathbf{u}(k) = \mathbf{u}(k) - \mathbf{u}(k-1) = [\Delta \mathbf{r}^T(k), \Delta \mathbf{s}^T(k)]^T$$

and the diagonal penalty matrices

$$\mathbf{Q}_t = \text{diag}(q_{t1}, \dots, q_{t9}), \quad \mathbf{Q}_{\Delta u} = \text{diag}(q_{\Delta u1}, \dots, q_{\Delta u9})$$

with all penalty weights in \mathbb{R}^+ and where the entries in \mathbf{Q}_t and $\mathbf{Q}_{\Delta u}$ are correlated to the corresponding ordering in $\mathbf{t}(k)$ and $\Delta \mathbf{u}(k)$. Consider now the expression for the stage cost

$$S(k) = \|\mathbf{Q}_t \mathbf{t}(k)\|_\infty + \|\mathbf{Q}_{\Delta u} \Delta \mathbf{u}(k)\|_\infty$$

and the formulation of the cost function

$$J(\mathbf{x}(k), \mathbf{u}(k-1), \mathbf{U}(k)) = \sum_{\ell=0}^{N-1} S(k+\ell|k), \quad (13.27)$$

which penalizes the predicted evolution $S(k+\ell|k)$ of $S(k)$ at step $k+\ell$ using information available at step k from time-instant k on over the finite horizon N using the ∞ -norm.

The control action at each time-instant k is obtained by minimizing the objective function (Eq. (13.27)) over the sequence of control inputs $\mathbf{U}(k) = [\mathbf{u}^T(k), \dots, \mathbf{u}^T(k+N-1)]^T$ subject to the aforementioned input constraints and to inequalities (Eq. (13.26)) for the selected prediction model (Eq. (13.25)). As a simplifying assumption the load shedding control for only the first prediction step is chosen, after which it is assumed to be constant throughout the prediction horizon. The first step of the optimal sequence $\mathbf{u}^*(k)$ thus obtained is then applied to the physical network after having rounded the AVR references to the nearest feasible value. The procedure is then repeated at the successive sampling instant $k+1$.

Since a linear objective function with linear equality and inequality constraints are employed, and since the decision variables are both continuous and discrete, the resulting optimization program amounts to a mixed-integer linear programming problem.

13.3.5 Simulation results

Scenarios

Two scenarios have been considered. Scenario 1 starts out from the system in steady-state, at 0.7 seconds the line connecting G_4 , representing the largest generation capacity in the considered grid, to bus 9 changes (possibly due to a partial fault) so that its impedance increases to 150%. Figure 13.8 shows the resulting open-loop evolution of the most important bus voltages. If no action is taken, voltages initially tend to progressively drift from the nominal region of operation until a series of sustained oscillations arises.

Scenario 2 involves a similar situation, only now the impedance increases to 400%, e.g. due to a forest fire. Figure 13.9 shows the open-loop evolution if the higher-layer controller does not provide updated set points to the lower-layer primary controllers. As can be seen the voltages quickly reach a series of fast oscillations.

Controller setup The penalty matrices are chosen such that a weight of 200 is placed on the violation of each soft constraint. The inputs are weighted with the penalty coefficients 1 or 20, respectively, for $r(k)$ and $s(k)$. The prediction horizon is $N = 8$. At each sampling instant, the linearization point is chosen by taking the current state $x(k)$ and the input applied at the preceding time instant $u(k - 1)$. The sampling interval is taken to be equal to $T_s = 0.25$ seconds.

Results Figure 13.10 depicts for Scenario 1 the evolution of the system when the proposed higher-layer MPC scheme is inserted in the feedback. As shown the controller prevents the voltages from exceeding the upper and lower bounds by acting on the reference settings of all the AVRs. No load shedding is necessary. The system subsequently enters an acceptable steady-state condition with a constant input profile.

Figure 13.11 depicts for Scenario 2 the evolution of the system with the MPC controller installed. Although the fault is significantly larger, the control prevents the voltages from crossing their limits, by providing set-points for the AVRs and shedding a minimal amount of load at node 7. After about 20 seconds the system enters a new steady-state with constant input profile.

Discussion The proposed controller works well for the studied cases, in which a rather high sampling rate of $T_s = 0.25$ seconds was taken. Indeed this rate might have to be decreased in a more realistic setting, since the system is composed of large high power components that might not allow for such a high actuation frequency. For the types of faults considered the simulations indicate that the predictions made with the linearized model are useful and that possible faults introduced due to saturation of the real system which are not modeled in the linearized system can be neglected. In fact, when the fault is smaller the sampling rate may be decreased, resulting in set-point update provided to the lower control layer less frequently. With a smaller fault, the magnitude and frequency of oscillations occurring in the system decrease in size.

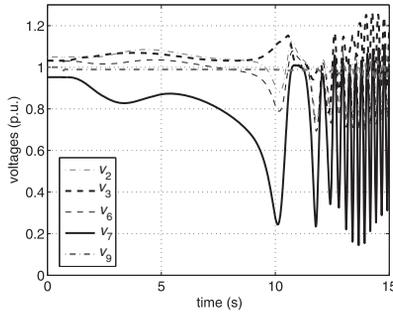


Fig. 13.8 Evolution of voltages in open loop for Scenario 1.

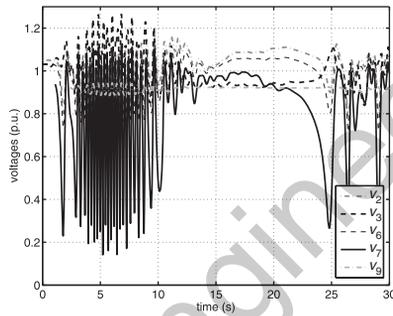


Fig. 13.9 Evolution of voltages in open loop for Scenario 2.

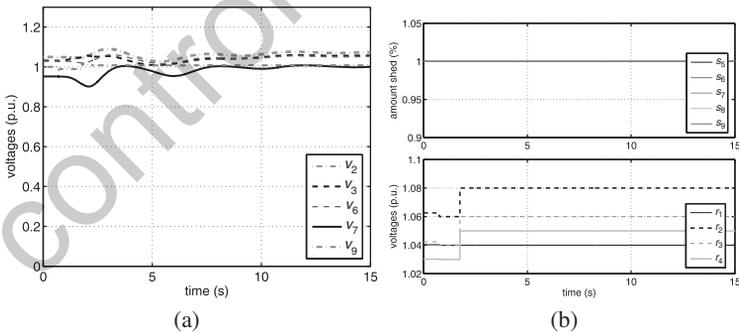


Fig. 13.10 Simulation results for Scenario 1. (a) Evolution of voltages in closed loop. (b) Control input sequence.

13.3.6 Conclusions and future research

In the present work, multi-layered control of voltage instability in a particular sub-network of a large power network has been considered. In this particular subnet-

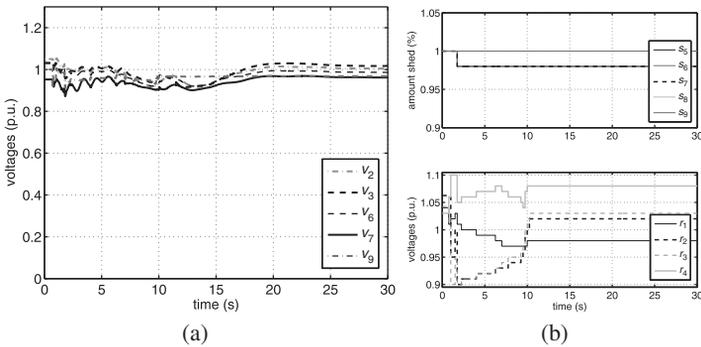


Fig. 13.11 Simulation results for Scenario 2. (a) Evolution of voltages in closed loop. (b) Control input sequence.

work a single higher-layer controller provides set-points to lower-layer controllers at discrete time steps such that the negative effects of voltage instabilities in the underlying physical system are minimized. The higher-layer controller uses a model-predictive control strategy to determine its actions. It uses a model based on a discrete-time linearized model of the continuous-time nonlinear dynamics given by a set of differential-algebraic equations. The model predicts the effects of changing set-points for the lower control layer on the evolution of network voltages. Simulations illustrate the potential of the approach.

Future research focuses on investigating the region of validity of the linearized model and if necessary replacing these with piece-wise affine models; performing simulations on a network in which the neighboring loads and generators are not aggregated, while the supervisory controller uses an aggregated model; and investigating decentralized control schemes where the local controllers of several subnetworks negotiate among them on how they should determine their actions to obtain system wide optimal performance.

13.4 Summary and outlook

Energy management involves different systems that evolve at completely different time scales. They require the use of different control methods depending on the dimension of the system and the sampling period. Power converters have extremely short sampling times and the computational power is limited, requiring simple control schemes. At the other end of the size range, in power system control, the complexity of the plant requires the use of powerful optimization tools, but sampling times are relatively longer and the computational power available is large. Several schemes for which the control law is synthesized off-line have been presented for the fast control of power converters. They include a scheme based on Explicit Model-Predictive Control with multiple linearizations, an extension of the Sampled

Data \mathcal{H}_∞ -control theory to PWM systems and relaxed dynamical programming for robust control. These methods control the converter switches through a PWM and the control input consists of one or several duty cycles. Additionally, a stabilization approach that uses a Lyapunov function deduced from energy considerations and that directly controls the switch state without requiring a PWM has been described.

An on-line optimization scheme relying on model-predictive control has been presented for minimizing the effect of network instabilities in the case of emergency voltage regulation of power grids, where due to an occurred outage the control has to react quickly to restore the system to an acceptable operating point. For other classes of problems arising in power systems control the considered sampling times can be longer, extending into the range of several seconds or even minutes.

Acknowledgment

The following have also contributed to this chapter: S. Almér, M. Bâja, J. Buisson, H. Cormerais, G. Damm, T. H. Demiray, H. Fujioka, S. Leirens, R.R. Negenborn, U. T. Jönsson, C.-Y. Kao, A. Rantzer, B. De Schutter, and A. Wernrud.

Industrial controls

S. Engell, S. Lohmann, T. Moor, C. de Prada, J. Raisch, D. Sarabia, and C. Sonntag

In this chapter, several applications of hybrid systems theory to benchmark process control problems are described, ranging from logic controller verification for an evaporation system to controller synthesis and optimization-based control of multiproduct batch plants and refrigerator systems.

Chapter contents

14.1	Introduction to process control applications	page 406
14.2	Logic control verification with application to an evaporator	407
14.2.1	The evaporator system	407
14.2.2	Task of safety analysis	408
14.2.3	Safety analysis using optimization-based state-space exploration	409
14.2.4	A semi-analytical approach to safety analysis	410
14.2.5	Evaluation of the results	412
14.3	Hierarchical control of a multiproduct batch plant	413
14.3.1	Hierarchical abstraction-based control	413
14.3.2	Hierarchical control architecture	414
14.3.3	Bottom-up design procedure	417
14.3.4	Discontinuously operated multiproduct batch plant	419
14.3.5	Hierarchical control of multiproduct batch plant	421
14.4	Model-predictive control of switched systems	425
14.4.1	Overview	425
14.4.2	Control of a supermarket refrigeration system	425
14.4.3	Start-up of a multiple-effect evaporator system	430
14.5	Summary and outlook on further industrial control applications	435

14.1 Introduction to process control applications

While continuous feedback and feedforward control is certainly crucial for the safe, economic, and ecologically benign operation of processing plants of all kinds, the dominant part of the control software for such plants handles and generates signals and events that are discrete in nature. Logic controllers supervise and filter all inputs by the operators for their admissibility, trigger process alarms and partial or complete shut-downs, and switch between control configurations. Sequential controllers govern the start-up of devices, such as pumps and compressors, and of complete units, and the execution of batch recipes as well as shut-down sequences. Even inside continuous controllers, a lot of discrete logic is present, sensors are monitored and their signals are replaced by substitute values in case of errors, the controllers are switched between different modes, anti-windup functions are realized, etc. Exception handling is a major part of all software modules of the control system.

The correct function and economic performance of a plant thus depends on the correct interaction between a vast number of logic and discrete control functions on different layers, the continuous dynamics of the plant, and the continuous or quasi-continuous controllers. While the design of continuous controllers has long been studied in the scientific literature, and very powerful methods have emerged [221], tools for the systematic analysis and synthesis of logic controllers have only recently been applied to real-world problems.

In this chapter, three different areas where the analysis and synthesis of logic controllers for continuous processing plants is of importance are addressed:

- the *verification* of the correct function of logic controllers that have to keep crucial state variables of the plant within predefined limits in case of equipment malfunctions in order to prevent emergency shutdowns (Section 14.2);
- the *synthesis* of hierarchical controllers that establish an economically optimal operation of batch plants by scheduling sequences of operations and controlling the execution of the individual processing steps (Section 14.3), based upon the theory presented in Section 6.4;
- the development of nonlinear *model-predictive controllers* for start-up sequences that involve the discrete switching of streams and for systems with switched inputs (Section 14.4).

In process control, the models of the dynamic behavior of the plant under control are usually fairly complex. All examples discussed in this chapter exhibit nonlinear dynamics, and in the case of the industrial evaporator which is analyzed in Section 14.2, a set of complex algebraic equations describes the thermodynamics of the system. Dynamic models of chemical processes typically are large nonlinear differential-algebraic systems which makes their analysis particularly challenging.

14.2 Logic control verification with application to an evaporator

14.2.1 The evaporator system

While current approaches to the safety verification of hybrid systems yield rigorous proofs for system safety, their applicability is restricted to relatively small systems. Optimization-based approaches are feasible to analyze large-scale systems. The process considered in this chapter represents a part of an industrial-scale multi-stage evaporation system that is modeled using hybrid automata.

For the analysis of the safety properties of the system, two approaches are applied. In the first approach, the hybrid system is considered only by its input/output behavior, and optimization-based techniques are used to compute worst-case scenarios. The second approach employs a semi-analytical technique which combines rigorous theorem proving that is applied to the global dynamical equations of the model with the computation of worst-case scenarios using global nonlinear optimization techniques.

Evaporation processes are widely used in the process industry to concentrate liquids in the form of suspensions, solutions, and emulsions. This is achieved by the evaporation of one or more volatile solvents from the liquid phase which leads to an enrichment of the non-volatile components.

In cooperation with Bayer Technology Services (BTS), a hybrid process model was developed in [593]. This model serves as a basis for several case studies. While the task of computing optimal start-up strategies for this system is tackled in [535, 596], this section deals with the task to analyze the safety of operation of this process.

A simplified flowsheet of the system is shown in Fig. 14.1. It consists of an evaporation vessel (A), a heat exchanger (B), a number of pumps (C), and several discretely switched valve assemblies (V_{S3} , V_{S4} , V_{VS2}) that are used to transfer liquid and vapor to and from the vessel and the heat exchanger. The term *valve assembly* refers to a redundant system of valves which can automatically circumvent a dysfunctional valve. The process is used to concentrate a liquid feed that is injected through the valve assembly V_{S3} at low temperature and consists of a non-volatile organic component (the product), an alcoholic solvent, and water. Via the valve assembly V_{VS2} , hot steam is fed to the heat exchanger, and the condensation of the steam leads to a transfer of thermal energy to the evaporator. If the liquid in the evaporator is at boiling temperature, the volatile components water and alcohol evaporate and are drained through the pipe P_V . Finally, when the product concentration within the liquid, which is measured online (Q), meets the desired purity specifications, the liquid can be drained through the valve assembly V_{S4} .

In the setting considered in the following, the process is assumed to be close to steady-state operation, and the product is continuously drained via V_{S4} . A logic controller is employed to keep the critical process variables (i.e. the liquid level L , the temperature T , and the pressure P within the evaporator) within safe bounds in the face of equipment malfunctions. The valve assemblies can only be switched discretely between two states (open/closed) by a logic controller that was designed to keep the critical process variables (i.e. the liquid level L , the temperature T , and the

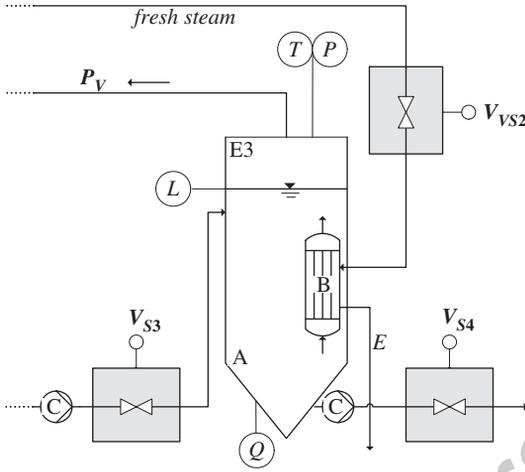


Fig. 14.1 Simplified flowsheet of the evaporator system.

Table 14.1 Strategy of the logic controller.

Controller state	V_{S3}	V_{S4}	V_{S2}
Nominal operating region	open	open	open
$L \leq L_{w,l}$	open	close	close
$L \geq L_{w,u}$	close	open	open
$T \geq T_w \wedge P \geq P_w$	open	open	close
$T \geq T_w \wedge P \geq P_w \wedge L \geq L_{w,u}$	close	open	close

pressure P within the evaporation vessel) within safe bounds. The controller receives discrete events from the plant if:

- the level, the temperature, or the pressure cross the upper warning thresholds $L_{w,u} := 90\%$, $T_w := 430$ K, and $P_w := 4.6$ bar from below
- the level crosses the lower warning threshold $L_{w,l} := 20\%$ from above; and
- all measurements have re-entered a subset of the state space designated as the nominal operating region ($L \in [30\%, 80\%]$, $T < 425$ K, $P < 4.4$ bar).

The responses of the discrete controller to the plant events are given in Table 14.1.

14.2.2 Task of safety analysis

Initially, the evaporation system is in steady-state operation, and all process variables lie within a subset of the nominal operating region, henceforth called the initial set. It is assumed that within a time interval of $[10, 300]$ seconds, two errors e_1 and

e_2 —at the time instances t_{e1} and t_{e2} —of the following types occur: a valve assembly is blocked (i.e. the controller cannot influence the valve setting) or the pipe P_V (and, thus, the vapor outflow) is obstructed. Table 14.2 gives an overview of all errors that can occur. It is assumed that an error is corrected after at most 130 seconds when the error was detected and the valve assembly has switched to a redundant line or the pipe is cleared again.

The *safety analysis task* is to verify that the logic controller keeps the system within safe bounds for at least 1000 seconds, where the safe region is defined by critical thresholds on the process measurements.

The time bound was derived from the assumption that the process is at steady-state again after at most 1000 seconds. The upper critical thresholds are given by $L_{c,u} := 100\%$, $T_c := 440$ K, and $P_c := 5$ bar, and a lower critical threshold for the liquid level is defined as $L_{c,l} := 0\%$.

Table 14.2 List of possible errors.

No.	Symbol	Explanation
1	$P_{V,c}$	P_V is obstructed
2	$V_{S3,o}$	V_{S3} is blocked in open position
3	$V_{S3,c}$	V_{S3} is blocked in closed position
4	$V_{S4,o}$	V_{S4} is blocked in open position
5	$V_{S4,c}$	V_{S4} is blocked in closed position
6	$V_{VS2,o}$	V_{VS2} is blocked in open position
7	$V_{VS2,c}$	V_{VS2} is blocked in closed position

For the described task, the plant model is implemented as a hybrid automaton. The controller, the valve assemblies, and the pipe are modeled as finite-state automata, and a set of timed automata represents the occurrence and the correction of errors. The continuous dynamics of the plant are modeled using two discrete locations with distinct systems of DAEs (each with 4 differential and 13 algebraic equations), which are chosen depending on the state of the liquid in the evaporation vessel (*nonevaporating* or *evaporating*).

14.2.3 Safety analysis using optimization-based state-space exploration

The first analysis approach employs continuous nonlinear optimization with embedded simulation of a hybrid process model. The optimizer only considers the input/output behavior of the system to drive it towards worst-case scenarios. The continuous decision variables are assembled in a vector \mathbf{d} defined as:

$$\mathbf{d} = [w_{A,0}, w_{B,0}, L_0, T_0, t_{e1}, t_{e2}]^T. \tag{14.1}$$

Here, $w_{A,0}$ (kg/kg), $w_{B,0}$ (kg/kg), L_0 (%), and T_0 (K) represent the concentrations of product and water in the liquid phase, the liquid level, and the temperature in the evaporation vessel that are used as initial values for the embedded hybrid simulation, and t_{e1} and t_{e2} are the time instances (in seconds) at which the errors e_1 and e_2 occur.

The considered optimization problems are nonlinear and nonconvex with possibly discontinuous cost functions. In addition, the embedded simulation employed in the dynamic optimization problems is computationally expensive since a simulation run of the hybrid model may take up to 1 second. To maximize the probability of finding the global optimum, the four global solvers *ego*, *rbfSolve*, *glcDirect*, and *multiMin* are employed that are part of the *MATLAB*-based optimization framework *TOMLAB* [325]. All parameters were kept at default settings.

The findings of the safety analysis using the optimization-based approach, which result from the solution of 72 optimization problems (4 optimization problems each for 18 possible error combinations) are as follows:

1. The critical threshold for the pressure is reached when P_V is obstructed and V_{VS2} is blocked in the open position. It was found that the trajectory of the pressure that was obtained for the error scenario in which the first error $P_{V,e}$ causes the system to reach the warning threshold P_w , and the second error $V_{S3,o}$ blocks the controller action, thus leading to a critical situation.
2. The warning threshold $T_w = 430$ K is not reached.
3. The warning thresholds $P_w = 4.6$ bar, $L_{w,u} = 90\%$, and $L_{w,l} = 20\%$ are only reached when the pipe P_V is obstructed. The corresponding simulation result (Fig. 14.2) shows that $L_{w,l}$ is reached at $t = 820$ s after $L_{w,u}$ was reached at $t = 110$ s. The monotonous decrease of L indicates that the system is not stabilized by the controller after reaching the nominal region. This behavior is not safety critical but clearly undesired.
4. The warning thresholds are only exceeded if the nominal controller response (Table 14.1) is prevented by an error.

14.2.4 A semi-analytical approach to safety analysis

The approach described in Section 14.2.3 is relatively easy to implement and yields good results for the evaporation system. However, it neither provides a detailed understanding of the process dynamics nor insights on how to modify the logic controller in the case that it does not meet the specifications. Furthermore, it does not yield rigorous proofs of safety properties as is for example possible by theorem proving using the equations that describe the continuous dynamics of the hybrid model.

This section introduces a semi-analytical approach to safety analysis that combines theorem-proving with optimization-based techniques and enables a deeper insight into the dynamical behavior of the controlled evaporation system. The main idea is to use process insight to determine those error scenarios that will definitely not drive the process into a critical state. These error scenarios can then be neglected

in the further investigation which simplifies the analysis problem. Since the continuous dynamics for the nonevaporating and for the evaporating case differ significantly, the two cases are treated separately.

Three different analysis techniques are employed:

1. *Theorem proving*: If explicit equations for the time derivatives of the safety-critical variables can be derived, manual theorem proving is employed to prove safety-relevant properties of the system rigorously. However, this approach is only applicable for the nonevaporating case due to the complexity of the dynamics in the evaporating case.
2. *Algebraic optimization*: In the evaporating case, the time derivatives can only be represented by an implicit system of equations. Hence, theorem proving is not applicable. In order to be able to reason about the time derivatives of the safety-critical variables, algebraic nonlinear optimization problems are solved to determine maximum or minimum values of the time derivatives. In these problems, the implicit equation systems that represent the time derivatives appear as equality constraints. However, if explicit equations for the time derivatives can be derived, an alternative formulation is used that includes these equations directly into the cost function.
3. *Optimization-based analysis using simulation*: If the results of the first two techniques hint at possibly safety-critical behaviors, the optimization-based state exploration approach is employed to verify the analysis result by simulation of the hybrid model. The cost function of the optimization problem is chosen such that the simulated trajectories for temperature, pressure, and level are driven towards a critical state estimating the worst-case evolution of the system. If the specific worst-case evolution adheres to the safety specifications, it is proved that the system is safe for the considered error scenario.

From all possible error combinations, all but six (shown in Table 14.3) could be discarded by the exclusion of noncritical error scenarios determined using the three analysis techniques mentioned above. Using the semi-analytical approach, the results of the optimization-based approach could be fully confirmed. Since the

Table 14.3 The final error combinations.

No.	Error combination
1	$P_{V,c} \wedge V_{VS2,o}$ may lead to P_c
2	$P_{V,c} \wedge V_{VS2,o}$ may lead to T_c
3-4	$P_{V,c} \wedge V_{S3,o}$ and $P_{V,c} \wedge V_{S4,c}$ may lead to $L_{c,u}$
5-6	$P_{V,c} \wedge V_{S3,c}$ and $P_{V,c} \wedge V_{S4,o}$ may lead to $L_{c,l}$

internal structure of the plant model was considered, several additional insights were obtained. For example, it might be possible that the pressure rises while the temperature falls depending on the concentration of the liquid in the evaporator. Additionally, the undesired controller response as shown in Fig. 14.2, where the level L is caused to drop from the upper to the lower warning threshold without the controller switching to the back to nominal state was detected. This is due to the fact that, by definition, the system is at nominal operation if **all** process measurements L , T , and P are within the nominal region. However, the right-hand figure, which depicts the evolution of the temperature, shows that the nominal operation region is not reached in the time range $t \in [170, 850]$. Hence, the controller does not switch back to nominal operation.

14.2.5 Evaluation of the results

The two optimization-based approaches to the safety analysis of an industrial-scale evaporation system differ with respect to the methods used and the results obtained.

The first approach adapts an input/output view of the hybrid processing system under consideration and performs an optimization-based search for worst-case scenarios based on simulations of a hybrid model. The major advantages of this approach are:

- that it is relatively easy to implement; and
- that even large-scale complex systems can be analyzed within a relatively short computation time (5–6 hours for the case study) without the need for a simplified process model.

However, this approach does not yield any safety proofs nor significant process insight that may be necessary to exactly pinpoint the design flaws of a controller.

The semi-analytical approach exploits the internal structure of the process model and employs a combination of theorem proving and optimization-based techniques. The main advantages of this approach are:

- that it provides a deep insight into the system dynamics;
- that it yields rigorous proofs of safety for a subset of the possible process behaviors; and
- that shortcomings of the controller that are not safety-critical but undesired, and situations under which the controller actions do not have the desired effect, are more likely to be revealed than in the simulation-based approach.

A drawback of the semi-analytical approach is that it is tailored specifically to the system under consideration and that it cannot be generalized easily. In addition, it requires more expert knowledge and time for its successful implementation than the brute-force exploration scheme.

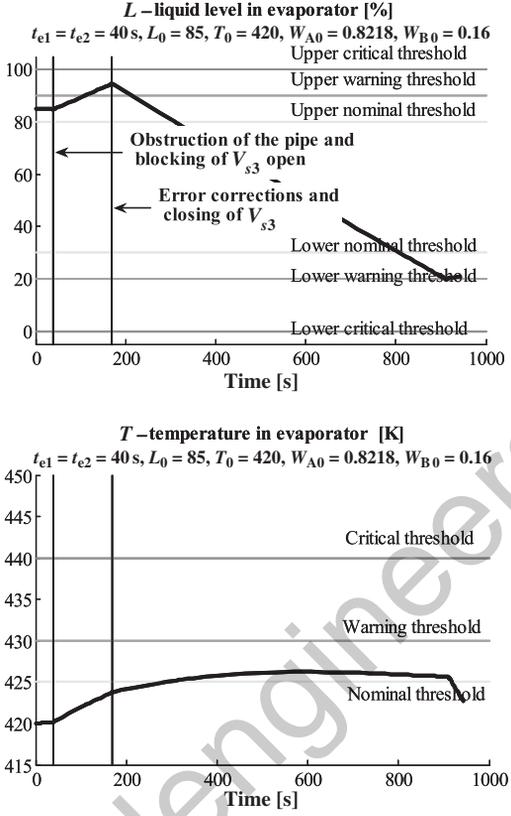


Fig. 14.2 The liquid level decreases from the upper to the lower warning level (top). The temperature remains above the nominal threshold for the period of time in question (bottom).

14.3 Hierarchical control of a multiproduct batch plant

14.3.1 Hierarchical abstraction-based control

Complexity represents a major problem in hybrid control. It may be reduced by abstraction-based control synthesis approaches that aim at replacing continuous dynamics by discrete approximations. As demonstrated in the sequel, the concept of abstractions is particularly useful in combination with hierarchical control structures.

Hierarchical decomposition of large-scale control tasks has been popular in industrial practice for a long time and, in a heuristic fashion, has been routinely applied. Unlike heuristic approaches, the hierarchical synthesis framework presented in this section guarantees that the different control layers interact properly and do indeed enforce the overall specifications for the considered plant model. It is an extension of the specific abstraction-based approach from Section 6.4.

In the context of discrete-event and hybrid systems, a number of hierarchical concepts have been discussed in the literature. Our approach has been inspired by the hierarchical DES theory developed in [667], but is technically quite different because we employ an I/O structure to adequately represent both time- and event-driven dynamics for hybrid systems. There is also a strong conceptual link to [389], where, as in [134, 508] and in our work, the preservation of fundamental properties across levels of abstraction is of prime concern. To keep exposition as simple as possible, we will focus on the case of two control layers. This, however, can be easily extended to any number of layers.

14.3.2 Hierarchical control architecture

To simplify exposition, we concentrate on the two-level control architecture shown in Fig. 14.3. Low-level control is implemented by an intermediate layer with behavior \mathfrak{B}_{im} , which communicates with the plant via low-level signals $w^L = (u^L, y^L)$ and with the high-level supervisor via high-level signals $w^H = (u^H, y^H)$. Plant and high-level supervisor behaviors are denoted by \mathfrak{B}_p^L and $\mathfrak{B}_{\text{sup}}^H$, respectively (see also Section 6.4).

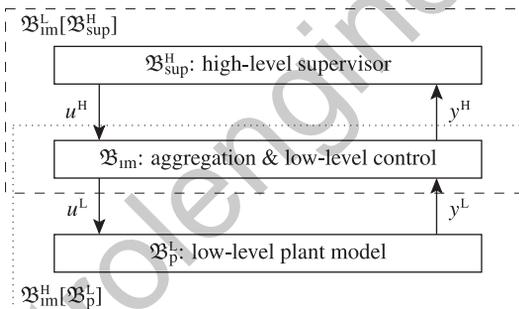


Fig. 14.3 Plant perspective (dashed) and supervisor perspective (dotted).

To make notation easier to read, all high-level signals, signal sets, and behaviors will be indicated by a sub- or superscript “H,” while low-level entities will be characterized by a sub- or superscript “L.” Apart from implementing low-level control mechanisms corresponding to high-level commands u^H , the intermediate layer \mathfrak{B}_{im} aggregates low-level measurement information y^L to provide high-level information y^H to $\mathfrak{B}_{\text{sup}}^H$. Aggregation may be both in signal space and in time, i.e. the time axis for high-level signals may be “coarser” than for low-level signals. Note that in this scenario \mathfrak{B}_{im} is a behavior on $W_H \times W_L$, where $W_H := U_H \times Y_H$ and $W_L := U_L \times Y_L$ represent the high and low-level signal sets.

From the perspective of the (low-level) plant \mathfrak{B}_p^L , interconnecting \mathfrak{B}_{im} and $\mathfrak{B}_{\text{sup}}^H$ provides the overall controller. Its external behavior is denoted by $\mathfrak{B}_{\text{im}}^L[\mathfrak{B}_{\text{sup}}^H]$ and, as indicated by the dashed box in Fig. 14.3, evolves on the low-level signal space

W_L . The behavior $\mathfrak{B}_{\text{im}}^L[\mathfrak{B}_{\text{sup}}^H]$ is given by the projection of \mathfrak{B}_{im} into $W_L^{N_0}$ with the internal high-level signal restricted to $\mathfrak{B}_{\text{sup}}^H$:

$$\mathfrak{B}_{\text{im}}^L[\mathfrak{B}_{\text{sup}}^H] := \{w^L \mid (\exists w^H \in \mathfrak{B}_{\text{sup}}^H)[(w^H, w^L) \in \mathfrak{B}_{\text{im}}]\}. \quad (14.2)$$

Clearly, we require the overall controller $\mathfrak{B}_{\text{im}}^L[\mathfrak{B}_{\text{sup}}^H]$ to be a (nontrivial) solution of the original control problem $(\mathfrak{B}_p^L, \mathfrak{B}_{\text{spec}}^L)_{\text{cp}}$, i.e. it needs to be admissible to \mathfrak{B}_p^L and has to enforce the specification $\mathfrak{B}_{\text{spec}}^L$:

$$\mathfrak{B}_p^L \cap \mathfrak{B}_{\text{im}}^L[\mathfrak{B}_{\text{sup}}^H] \subseteq \mathfrak{B}_{\text{spec}}^L. \quad (14.3)$$

We now re-examine Fig. 14.3: from the perspective of the high-level supervisor $\mathfrak{B}_{\text{sup}}^H$, interconnecting the intermediate layer \mathfrak{B}_{im} with the (low-level) plant model \mathfrak{B}_p^L provides a compound high-level plant model, which, as indicated by the dotted box in Fig. 14.3, evolves on the high-level signal set W_H . Its external behavior is denoted by $\mathfrak{B}_{\text{im}}^H[\mathfrak{B}_p^L]$ and given by

$$\mathfrak{B}_{\text{im}}^H[\mathfrak{B}_p^L] := \{w^H \mid (\exists w^L \in \mathfrak{B}_p^L)[(w^H, w^L) \in \mathfrak{B}_{\text{im}}]\}. \quad (14.4)$$

By the same argument as before, $\mathfrak{B}_{\text{sup}}^H$ is required to be admissible to the compound high-level plant model $\mathfrak{B}_{\text{im}}^H[\mathfrak{B}_p^L]$. The above discussion is summarized in the following definition:

Definition 14.1 (Two-level hierarchical solution) *The pair $(\mathfrak{B}_{\text{im}}, \mathfrak{B}_{\text{sup}}^H)_{\text{tl}}$ is a two-level hierarchical solution of the supervisory control problem*

$(\mathfrak{B}_p^L, \mathfrak{B}_{\text{spec}}^L)_{\text{cp}}$ if:

- (i) $\mathfrak{B}_p^L \cap \mathfrak{B}_{\text{im}}^L[\mathfrak{B}_{\text{sup}}^H] \subseteq \mathfrak{B}_{\text{spec}}^L$; and*
- (iia) $\mathfrak{B}_{\text{im}}^L[\mathfrak{B}_{\text{sup}}^H]$ is admissible to \mathfrak{B}_p^L ; and*
- (iib) $\mathfrak{B}_{\text{sup}}^H$ is admissible to $\mathfrak{B}_{\text{im}}^H[\mathfrak{B}_p^L]$.*

We remind the reader that \mathfrak{B}_1 is admissible to \mathfrak{B}_2 if the former is generically implementable and if \mathfrak{B}_1 and \mathfrak{B}_2 are nonconflicting. The latter is guaranteed if \mathfrak{B}_1 is complete and generically implementable and if \mathfrak{B}_2 is a complete I -behavior (Section 6.4). We will now investigate which properties of \mathfrak{B}_{im} will help to enforce admissibility conditions (iia) and (iib) in Definition 14.1. To structure the discussion, we will first address the case of uniform time scales on both signal levels.

Uniform time scales–Type I intermediate layer From Fig. 14.3 it is obvious that u^H and y^L can be interpreted as inputs of the intermediate layer \mathfrak{B}_{im} , while u^L and y^H are outputs. It is, therefore, natural to require that \mathfrak{B}_{im} is an I -behavior w.r.t. $(U_H \times Y_L, Y_H \times U_L)$. If we add the requirement that \mathfrak{B}_{im} is complete and a *strict* I -behavior, it can be shown that I - and completeness properties are passed from \mathfrak{B}_p^L to $\mathfrak{B}_{\text{im}}^H[\mathfrak{B}_p^L]$ and that completeness and generic implementability carry over from $\mathfrak{B}_{\text{sup}}^H$ to $\mathfrak{B}_{\text{im}}^L[\mathfrak{B}_{\text{sup}}^H]$ [467, 549]. This can be summarized as:

Lemma 14.1 *If the intermediate layer \mathfrak{B}_{im} is a complete strict I -behavior, if the plant model \mathfrak{B}_p^L is a complete I -behavior, and if the high-level supervisor is both complete and generically implementable, then the admissibility conditions (iia) and (iib) are satisfied.*

Multiple time scales–type II intermediate layer In many cases, high- and low-level signals will be defined on different time scales. Typically, in technical realizations, low-level signals “live” on a discrete time axis that is obtained by (fast) equidistant sampling. High-level signals mostly live on a time axis that is generated by low-level signals. A common scenario is that y^L produces events, e.g. when certain thresholds are crossed. The high-level signal y^H is then a sequence of these events. We assume that high-level commands are immediately issued after the occurrence of a high-level measurement event, hence u^H lives on the same time axis as y^H . This is illustrated in Fig. 14.4.

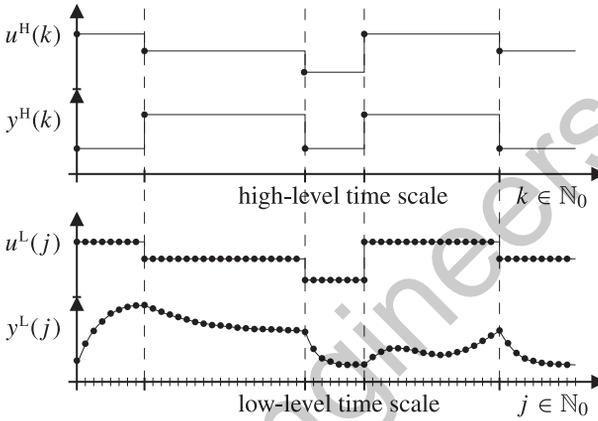


Fig. 14.4 Two time scales.

We now define the relevant class of operators mapping low-level signals $y^L \in Y_L^{\mathbb{N}_0}$ into time transformations (i.e. maps from low- to high-level time):

Definition 14.2 (Dynamic time scale) The operator $T: Y_L^{\mathbb{N}_0} \rightarrow \mathbb{N}_0^{\mathbb{N}_0}$ is said to be a dynamical time scale if:

- T is strictly causal, i.e. $\tilde{y}|_{[0,k]} = \hat{y}|_{[0,k]}$ implies $T(\tilde{y})|_{[0,k]} = T(\hat{y})|_{[0,k]}$ for all $k \in \mathbb{N}_0$, $\tilde{y}, \hat{y} \in Y_L^{\mathbb{N}_0}$; and if
- the time transformation $T(y^L): \mathbb{N}_0 \rightarrow \mathbb{N}_0$ is surjective and monotonically increasing for all $y^L \in Y_L^{\mathbb{N}_0}$.

For a fixed low-level signal y^L , the time transformation $T(y^L)$ maps low-level time $j \in \mathbb{N}_0$ to high-level time $k \in \mathbb{N}_0$. By requiring that T itself is a strictly causal operator, we ensure that at any instant of time the transformation $T(y^L)$ only depends on the strict past of y^L .

We focus on measurement aggregation operators that are causal with respect to a dynamic time scale:

Definition 14.3 (Causal operator) The operator $F: Y_L^{\mathbb{N}_0} \rightarrow Y_H^{\mathbb{N}_0}$ is said to be causal w.r.t. T if T is a dynamical time scale and if

$$\tilde{y}^L|_{[0,j]} = \hat{y}^L|_{[0,j]} \Rightarrow F(\tilde{y}^L)|_{[0,k]} = F(\hat{y}^L)|_{[0,k]} \quad (14.5)$$

for $k = T(\tilde{y}^L)(j)$ and all $j \in \mathbb{N}_0$, $\tilde{y}^L, \hat{y}^L \in Y_L^{\mathbb{N}_0}$.

We still have to link high-level control signals u^H to low-level control signals u^L . This is done via a sample-and-hold device that is triggered by the time transformation $T(y^L)$, i.e. successive high-level control actions are passed on to the lower level whenever a high-level measurement is generated. Formally, this is expressed by $u^L = u^H \circ T(y^L)$.

In summary, an intermediate layer \mathfrak{B}_{im} mediating between low-level and high-level time is completely defined by a dynamical time scale T and a measurement aggregation operator F that is causal w.r.t. T :

$$\mathfrak{B}_{\text{im}} := \{(u^H, y^H, u^L, y^L) \mid y^H = F(y^L) \text{ and } u^L = u^H \circ T(y^L)\}. \quad (14.6)$$

It can be shown that (14.6) represents a complete behavior and, like the intermediate layer discussed previously, preserves the input/output structure of the plant and generic implementability of the supervisor:

Lemma 14.2 *If \mathfrak{B}_{im} is given by (14.6), \mathfrak{B}_p^L is a complete I/- behavior w.r.t. (U_L, Y_L) , and $\mathfrak{B}_{\text{sup}}^H$ is complete and generically implementable, then the admissibility conditions (iia) and (iib) are satisfied.*

In most practical situations, we will have to combine the two types of intermediate layers discussed on the previous pages. It is intuitively clear that for arbitrary combinations of Type I and Type II layers in combination with complete I/- plant behaviors and complete and generically implementable high-level supervisors, the admissibility conditions (iia) and (iib) are satisfied (for a formal treatment, see [467]). While for \mathfrak{B}_{im} and $\mathfrak{B}_{\text{sup}}^H$, as “man-made systems,” the completeness assumption is not particularly restrictive, it may be argued that this is not the case for the plant \mathfrak{B}_p^L . However, as shown in [466], the condition that \mathfrak{B}_p^L is a complete I/- behavior can be replaced by the less restrictive condition that it can be realized by an I/S/- machine.

14.3.3 Bottom-up design procedure

It remains to discuss how to design \mathfrak{B}_{im} and $\mathfrak{B}_{\text{sup}}^H$ (subject to the above constraints) such that $\mathfrak{B}_p^L \cap \mathfrak{B}_{\text{im}}^L[\mathfrak{B}_{\text{sup}}^H] \subseteq \mathfrak{B}_{\text{spec}}^L$. We suggest an intuitive bottom-up procedure where we first design appropriate low-level control \mathfrak{B}_{im} and then proceed to find a suitable high-level supervisor $\mathfrak{B}_{\text{sup}}^H$.

In the first step, we formalize the *intended* relation between high- and low-level signals by the specification $\mathfrak{B}_{\text{spec}}^{\text{HL}} \subseteq (W_H \times W_L)^{\mathbb{N}_0}$. Hence $\mathfrak{B}_{\text{spec}}^{\text{HL}}$ denotes the set of all signal pairs (w^H, w^L) that represent the desired effect of high-level control actions on the low-level plant \mathfrak{B}_p^L and, by implication, on the high-level measurement. This specification needs to be enforced by the intermediate layer \mathfrak{B}_{im} when connected to the low-level plant \mathfrak{B}_p^L , i.e. we require

$$\{(w^H, w^L) \in \mathfrak{B}_{\text{im}} \mid w^L \in \mathfrak{B}_p^L\} \subseteq \mathfrak{B}_{\text{spec}}^{\text{HL}}. \quad (14.7)$$

Suppose we have designed \mathfrak{B}_{im} to enforce (14.7), perhaps based on classical continuous control methods appropriate for the continuous dynamics of the low-level plant. In principle, we could then proceed to design $\mathfrak{B}_{\text{sup}}$ for the compound high-level plant $\mathfrak{B}_{\text{im}}^{\text{H}}[\mathfrak{B}_{\text{p}}^{\text{L}}]$. However, from a computational point of view—particularly for hybrid systems—it is preferable to use an abstraction $\tilde{\mathfrak{B}}_{\text{p}}^{\text{H}}$ of $\mathfrak{B}_{\text{im}}^{\text{H}}[\mathfrak{B}_{\text{p}}^{\text{L}}]$ that does not explicitly depend on the low-level dynamics or the precise nature of the implemented low-level control scheme. A suitable abstraction $\tilde{\mathfrak{B}}_{\text{p}}^{\text{H}}$ is given by the projection of the specification $\mathfrak{B}_{\text{spec}}^{\text{HLL}}$ onto its high-level signal components, and a high-level specification $\tilde{\mathfrak{B}}_{\text{spec}}^{\text{H}}$ expressing $\mathfrak{B}_{\text{spec}}^{\text{L}}$ in terms of high-level signals can be easily derived from (14.7):

$$\tilde{\mathfrak{B}}_{\text{p}}^{\text{H}} := \{w^{\text{H}} \mid (\exists w^{\text{L}}) [(w^{\text{H}}, w^{\text{L}}) \in \mathfrak{B}_{\text{spec}}^{\text{HLL}}]\}, \quad (14.8)$$

$$\tilde{\mathfrak{B}}_{\text{spec}}^{\text{H}} := \{w^{\text{H}} \mid (\forall w^{\text{L}}) [(w^{\text{H}}, w^{\text{L}}) \in \mathfrak{B}_{\text{spec}}^{\text{HLL}} \Rightarrow w^{\text{L}} \in \mathfrak{B}_{\text{spec}}^{\text{L}}]\}. \quad (14.9)$$

As a consequence, the resulting high-level control problem $(\tilde{\mathfrak{B}}_{\text{p}}^{\text{H}}, \tilde{\mathfrak{B}}_{\text{spec}}^{\text{H}})_{\text{cp}}$ does *not* depend on the actual low-level plant under low-level control, $\mathfrak{B}_{\text{im}}^{\text{H}}[\mathfrak{B}_{\text{p}}^{\text{L}}]$.

It follows immediately that any solution $\mathfrak{B}_{\text{sup}}^{\text{H}}$ of the high-level control problem $(\tilde{\mathfrak{B}}_{\text{p}}^{\text{H}}, \tilde{\mathfrak{B}}_{\text{spec}}^{\text{H}})_{\text{cp}}$ will enforce the original low-level specification $\mathfrak{B}_{\text{spec}}^{\text{L}}$ when connected to $\mathfrak{B}_{\text{im}}^{\text{H}}[\mathfrak{B}_{\text{p}}^{\text{L}}]$, the plant model under low-level control:

$$\tilde{\mathfrak{B}}_{\text{p}}^{\text{H}} \cap \mathfrak{B}_{\text{sup}}^{\text{H}} \subseteq \tilde{\mathfrak{B}}_{\text{spec}}^{\text{H}} \quad \Longrightarrow \quad \mathfrak{B}_{\text{p}}^{\text{L}} \cap \mathfrak{B}_{\text{im}}^{\text{L}}[\mathfrak{B}_{\text{sup}}^{\text{H}}] \subseteq \mathfrak{B}_{\text{spec}}^{\text{L}}. \quad (14.10)$$

Hence Condition (i) from Definition 14.1 also holds, and the pair $(\mathfrak{B}_{\text{im}}, \mathfrak{B}_{\text{sup}})_{\text{tl}}$ is a *two-level hierarchical solution* of the overall control problem $(\mathfrak{B}_{\text{p}}^{\text{L}}, \mathfrak{B}_{\text{spec}}^{\text{L}})_{\text{cp}}$.

The “degree of freedom” in the proposed bottom-up approach is the specification $\mathfrak{B}_{\text{spec}}^{\text{HLL}}$. In general, its choice can be guided by the same engineering intuition that we would use in a hierarchical ad hoc design. However, unlike heuristic approaches, our method encapsulates intuition in a formal framework where we can prove that the composition of high-level controller and intermediate layer forms a valid solution of the original problem.

Minimizing the cost of closed-loop operation In addition to a “hard” specification, many applications involve minimizing a cost function. From the low-level perspective, we want to solve the worst-case optimal control problem

$$\min_{\mathfrak{B}_{\text{sup}}^{\text{L}}} \max_{w^{\text{L}}} \gamma^{\text{L}}(w^{\text{L}}) \text{ s.t. } \mathfrak{B}_{\text{sup}}^{\text{L}} \text{ solves } (\mathfrak{B}_{\text{p}}^{\text{L}}, \mathfrak{B}_{\text{spec}}^{\text{L}})_{\text{cp}}, w^{\text{L}} \in \mathfrak{B}_{\text{p}}^{\text{L}} \cap \mathfrak{B}_{\text{sup}}^{\text{L}}, \quad (14.11)$$

where $\gamma^{\text{L}}: \mathfrak{B}_{\text{p}}^{\text{L}} \cap \mathfrak{B}_{\text{spec}}^{\text{L}} \rightarrow \mathbb{R}$ is a (typically additive over time and positive) function to associate the cost $\gamma^{\text{L}}(w^{\text{L}})$ with each low-level plant signal w^{L} that satisfies the “hard” specification $\mathfrak{B}_{\text{spec}}^{\text{L}}$. Suppose that \mathfrak{B}_{im} has been designed as outlined previously. We then define a pessimistic high-level cost function

$$\gamma^{\text{H}}(w^{\text{H}}) := \max_{w^{\text{L}}} \{\gamma^{\text{L}}(w^{\text{L}}) \mid (w^{\text{H}}, w^{\text{L}}) \in \mathfrak{B}_{\text{im}}, w^{\text{L}} \in \mathfrak{B}_{\text{p}}^{\text{L}}\}, \quad (14.12)$$

and seek an optimal solution $\mathfrak{B}_{\text{sup}}^{\text{H}}$ for the high-level min-max problem

$$\min_{\mathfrak{B}_{\text{sup}}^{\text{H}}} \max_{w^{\text{H}}} \gamma^{\text{H}}(w^{\text{H}}) \text{ s.t. } \mathfrak{B}_{\text{sup}}^{\text{H}} \text{ solves } (\tilde{\mathfrak{B}}_{\text{p}}^{\text{H}}, \tilde{\mathfrak{B}}_{\text{spec}}^{\text{H}})_{\text{cp}} \text{ and } w^{\text{H}} \in \tilde{\mathfrak{B}}_{\text{p}}^{\text{H}} \cap \mathfrak{B}_{\text{sup}}^{\text{H}}. \quad (14.13)$$

Note that the overall controller, i.e. the interconnection $\mathfrak{B}_{\text{im}}^{\text{L}}[\mathfrak{B}_{\text{sup}}^{\text{H}}]$ of \mathfrak{B}_{im} and the optimal $\mathfrak{B}_{\text{sup}}^{\text{H}}$, does not necessarily form an optimal solution to the original problem (14.11). This is for two reasons:

- introducing \mathfrak{B}_{im} reduces the degrees of freedom;
- in the high-level control problem (14.13), the behavior $\mathfrak{B}_{\text{im}}^{\text{H}}[\mathfrak{B}_{\text{p}}^{\text{L}}]$ was replaced by its abstraction $\tilde{\mathfrak{B}}_{\text{p}}^{\text{H}}$, resulting in over-approximation of costs.

However, we expect problem (14.13) to be computationally tractable in situations where (14.11) is not. We emphasize that our bottom-up design method guarantees that the “hard” specification $\mathfrak{B}_{\text{spec}}^{\text{L}}$ holds.

14.3.4 Discontinuously operated multiproduct batch plant

Discontinuously operated multiproduct plants are widely used for the production of fine chemicals. In [549], we describe a specific control example, which is idealized to a certain extent but general enough to capture most of the problems that characterize multiproduct batch plants.

The plant is used to produce three kinds of color pigments, using similar production methods (Fig. 14.5): from one of the storage tanks B1, B2, or B3, solvent is pumped into either a large reactor R1 or a small reactor R2. Reactant A_i , $i = 1, 2, 3$, is added to start reaction i delivering the desired product: $A_i \xrightarrow{k_{P_i}} P_i$. It is accompanied by a parallel reaction $A_i \xrightarrow{k_{W_i}} W_i$ resulting in the waste product W_i . If, at the end of the reaction step, concentration of W_i is above a given threshold $W_{i,\text{max}}$, product quality is unacceptable and the batch is spoilt.

For the duration of the reaction, there are two control inputs: the feed rate of the reactant and the heating/cooling rate for the reactor. After the reaction is finished, the contents of the reactors are filtered through either F1, F2, or F3, and the solvent is collected in the corresponding tank B1, B2, or B3. The solvent can subsequently be fed back into either of the two reactors. If, in any of the filters, darker colors are filtered before lighter ones (say P3 before P1 or P2, and P2 before P1), an additional cleaning process between the two filtration tasks is needed, taking time t_c . The feed rates into the reactors are discrete-valued control inputs as are the decision variables (realized by discrete valve positions) that determine whether a particular reactor is emptied through a particular filter system. Heating and cooling rates for R1, R2 are continuous-valued control inputs. For simplicity, the following is assumed for the reactions involved:

- all reactions are first order;
- the volume of reactant A_i , product P_i and waste product W_i , $i = 1, 2, 3$, is negligible compared to overall reactor volume.

The latter can, therefore, be considered constant during dosing and reaction. The time constants for heating and cooling of the reactors are small compared to the reaction

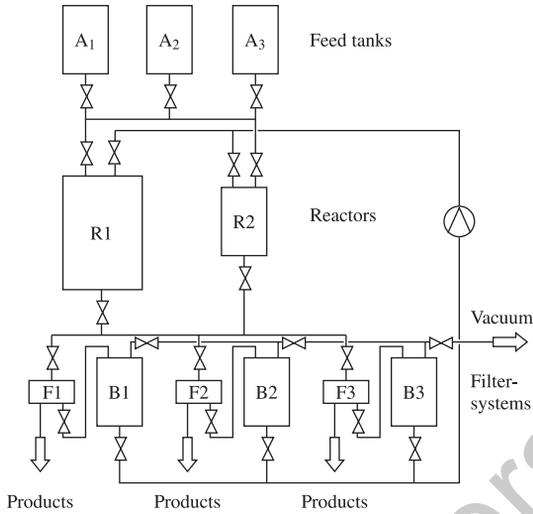


Fig. 14.5 Example plant.

time constants. The (scaled) reactor temperatures can, therefore, be considered to be the manipulated variables. With these assumptions, the model equations can be easily derived from component balances:

$$\frac{d}{dt}c_{A_i}(t) = \frac{q(t)}{V} - (k_{P_i}(t) + k_{W_i}(t))c_{A_i}(t), \quad (14.14)$$

$$\frac{d}{dt}c_{P_i}(t) = k_{P_i}(t)c_{A_i}(t), \quad (14.15)$$

$$\frac{d}{dt}c_{W_i}(t) = k_{W_i}(t)c_{A_i}(t). \quad (14.16)$$

V is the volume of the considered reactor, q the corresponding dosing rate (in kmol/h); c_{A_i} , c_{P_i} , c_{W_i} are reactant, product and waste concentration in the i -th production process (in kmol/m³), $i = 1, 2, 3$. The reaction kinetics

$$k_{P_i}(t) = k_{P_{i0}}e^{-\frac{E_{P_i}}{R\theta(t)}}, \quad k_{W_i}(t) = k_{W_{i0}}e^{-\frac{E_{W_i}}{R\theta(t)}} \quad (14.17)$$

depend on temperature θ . Defining $u(t) := k_{W_1}(t)$, $\beta_i := E_{P_i}/E_{W_1}$, $\delta_i := E_{W_i}/E_{W_1}$, $\alpha_i := k_{P_{i0}}/k_{W_1}^{\beta_i}$, and $\gamma_i := k_{W_{i0}}/k_{W_1}^{\delta_i}$, we can rewrite (14.14)–(14.16) as

$$\frac{d}{dt}c_{A_i}(t) = \frac{q(t)}{V} - (\alpha_i u(t)^{\beta_i} + \gamma_i u(t)^{\delta_i})c_{A_i}(t), \quad (14.18)$$

$$\frac{d}{dt}c_{P_i}(t) = \alpha_i u(t)^{\beta_i}c_{A_i}(t), \quad (14.19)$$

$$\frac{d}{dt}c_{W_i}(t) = \gamma_i u(t)^{\delta_i}c_{A_i}(t). \quad (14.20)$$

Note that, by definition, $\delta_1 = \gamma_1 = 1$, and that u is a strictly monotonically increasing function in θ and can, therefore, be considered as scaled temperature with unit $1/h$.

The aim is to produce the demanded product volumes with minimal operating costs while satisfying quality constraints (upper bounds for waste product concentrations) and safety constraints (upper bounds for reactor temperatures).

14.3.5 Hierarchical control of multiproduct batch plant

Low-level plant model The low-level plant model represents the continuous dynamics of filter and reaction processes in the various modes of operation. We consider low-level signals to evolve with respect to clock time, generated from a suitably small sampling period $\Delta > 0$. Note that after a reaction is finished, the respective reactor has to be emptied, i.e., its contents has to be filtered before the reactor can be reused in another production step.

Neglecting the time required to fill a reactor, there are at most two concurrent operations performed by the plant. Thus, our low-level plant model consists of two subsystems, each of which is being used for one out of three chemical reaction schemes or a subsequent filtering process. As a low-level signal space we choose $W_L = W_{L1} \times W_{L2}$, where each component corresponds to one subsystem. The possible modes of operation for the subsystem $j \in \{1, 2\}$ consist of the three chemical reactions and the filtering processes. The latter can use any nontrivial combination of the three filters. Including a filter cleaning mode and an “idle” mode, this gives a total of $3 + 2 + 7 = 12$ possible modes for each subsystem; they can be conveniently encoded as a discrete-event input u^{Dj} , $j = 1, 2$, with range $U_{Dj} = \{P1, P2, P3, \text{Clean}, \text{Idle}, F001, F010, F011, \dots, F111\}$.

While in one of the reaction modes P_i , low-level dynamics is modelled by a sampled version of the ODEs (14.18) – (14.20). The parameters are as follows: $\beta_1 = 0.5$, $\alpha_1 = 2.0 \text{ h}^{-0.5}$, $\beta_2 = 0.4$, $\alpha_2 = 2.0 \text{ h}^{-0.6}$, $\beta_3 = 0.5$, $\alpha_3 = 3.0 \text{ h}^{-0.5}$, $\delta_i = \gamma_i = 1$, $i = 1, 2, 3$; the initial concentrations at the beginning of each reaction are all zero: $c_{A_i0} = c_{P_i0} = c_{W_i0} = 0$, $i = 1, 2, 3$. The product concentrations required at the end of each reaction are $c_{P_1e} = 10 \text{ kmol/m}^3$, $c_{P_2e} = 8 \text{ kmol/m}^3$, $c_{P_3e} = 12 \text{ kmol/m}^3$, and the bounds for the waste concentrations c_{W_1} , c_{W_2} , c_{W_3} are 2 kmol/m^3 , 1.5 kmol/m^3 , and 3 kmol/m^3 . The volumes of reactor R1 and R2 are 5 m^3 and 2.5 m^3 . The (on/off) dosing signal q_j can take values in the set $\{0, 12 \text{ kmol/h}\}$, and the control signal u_j is required to “live” within the interval $[0.01 \text{ h}^{-1}, 3.0 \text{ h}^{-1}]$, where the upper bound results from safety requirements. The signal (q_j, u_j) is seen as an additional low-level input u^{Cj} with range $U_{Cj} \subseteq \mathbb{R}^2$. We assume the continuous state is measured as a plant output y^{Cj} with range $Y_{Cj} \subseteq \mathbb{R}^3$.

For filtering, an integrator models the progress of time, where the integration constant depends on the number of filters used and the volume of the respective reactor. The time to empty the smaller of the two reactors through one filter is $c_{tf} = 6 \text{ h}$. If two or three filters are being used simultaneously, this time reduces to 3 h and 2 h . For the larger reactor, filtering takes twice as long. The continuous input u^{Cj} is ignored in filtering mode.

The completion of an operation corresponds to reaching a target region in the continuous state space. This is indicated by a discrete low-level output y^{Dj} taking values in the set {Busy, Done}. The signal space of subsystem $j \in \{1, 2\}$ is then the product $W_{Lj} = U_{Lj} \times Y_{Lj}$ where $U_{Lj} = U_{Dj} \times U_{Cj}$ and $Y_{Lj} = Y_{Dj} \times Y_{Cj}$. This is illustrated in Fig. 14.6, where the two subsystems are shown merged. With the above parameters, the typical time to finish a reaction step is between 5 h and 10 h, with filtering taking at least another two hours.

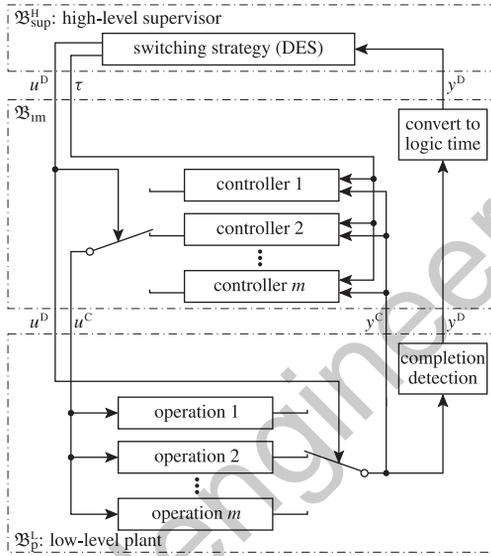


Fig. 14.6 Control architecture (subsystems merged).

Low-level specification and cost function The low-level specification $\mathfrak{B}_{\text{spec}}^L$ includes the following requirements:

- the mode of operation may only change immediately after the previous operation has been completed;
- chemical reactions and filtering alternate in each subsystem;
- each filter can only be used by one subsystem at a time;
- the filters have to be cleaned when a lighter color is to be filtered after a darker one;
- the demanded product volumes are produced.

Note that safety and quality requirements are imposed indirectly—the former via the restricted range for $u^{Cj} = (q_j, u_j)$, the latter via the completion signals y^{Dj} , which are linked to the y^{Cj} reaching their target regions. Therefore, $\mathfrak{B}_{\text{spec}}^L$ contains only discrete requirements and represents a discrete-event specification, albeit in clock

time. Formally, we have $\mathfrak{B}_{\text{spec}}^L = \mathfrak{B}_{\text{spec}}^D \times (U_C \times Y_C)^{N_0}$ for some behavior $\mathfrak{B}_{\text{spec}}^D$ over $U_D \times Y_D$.

For a finite automaton realization of $\mathfrak{B}_{\text{spec}}^D$, the state needs to keep track of the following: the current major mode of operation (5^2 possibilities for the three reactions, filtering, or cleaning in both subsystems), the current allocation of the filters (2^3 possibilities for three filters which can be allocated to either subsystem), the recent usage of the filters (3^3 possibilities for three filters, each of which could have been used for either of three products), product volumes produced so far (6^3 possibilities; this number results from the additional assumption that only integer multiples of a minimum batch size are allowed and that the maximum demand for each of the three products is known. For our example we chose 2.5 m^3 as minimum batch size and 12.5 m^3 as maximum demand). This amounts to an overall of 1.16×10^6 states.

The integral cost function γ^L only refers to the U_{C_j} components of the low-level signal. It includes energy cost (heating), material cost (feed rates), and an overhead cost depending on time spent. For a low-level signal w^L , let

$$\gamma^L(w^L) := \int_0^{T_f} (u_1(t) + \frac{q_1(t)}{20}) dt + \int_0^{T_f} (u_2(t) + \frac{q_2(t)}{20}) dt + \int_0^{T_f} 0.15 dt, \quad (14.21)$$

where T_f denotes the time when all demanded products have been delivered.

Given the low-level plant model, the specification and the cost function, one could try to solve the optimization problem (14.11) as it stands. This amounts to a nonlinear mixed discrete-continuous dynamical program with two continuous input signals (u_1, u_2), discrete input signals ($u^{D_j}, q_j, j = 1, 2$) that altogether can take $(12 \times 2)^2 = 576$ values, six continuous state variables, and a discrete state set with 1.16×10^6 elements. Over an adequate time horizon (about 50 h), we found this computationally intractable for off-the-shelf optimization software. Instead we apply the suggested hierarchical procedure.

Hierarchical design–low-level control Recall that low-level control is based on the specification $\mathfrak{B}_{\text{spec}}^{\text{HL}}$ representing the intended relation between high- and low-level signals. We introduce high-level control symbols signifying the commands “run reaction i in subsystem j such that the batch is finished at minimal cost within time $\tau_j \in T = \{1 \text{ h}, 2 \text{ h}, \dots, 10 \text{ h}\}$ ”. This is implemented by $3 \times 2 \times 10 = 60$ low-level controllers selected by $\mathfrak{B}_{\text{sup}}^{\text{H}}$ (Fig. 14.6).

Obviously, low-level controller design is local as it only refers to one reaction process and one subsystem at a time. The corresponding dynamical program has one binary input signal (q_j), one continuous input (u_j) and three continuous state variables (concentrations). It can be solved numerically by standard optimization software. As an illustration, the minimal cost $\gamma_{1,i}(\tau_1)$ for reactor R1 to produce one batch of Pi is given in Table 14.4. Obviously, low-level optimization does not depend on the demanded overall amount of products. Consequently, this design step only needs to be performed once over the life-cycle of the plant.

Hierarchical design–high-level control As indicated above, high-level control actions consist of modes from U_{D_i} and timing parameters from T . As the timing

Table 14.4 Minimal cost for reactor R1 to produce one batch.

τ_1	< 5 h	5 h	6 h	7 h	8 h	9 h	10 h
$\gamma_{1,1}$	∞	∞	3.70	3.42	3.28	3.21	3.17
$\gamma_{1,2}$	∞	2.81	2.58	2.46	2.40	2.36	2.32
$\gamma_{1,3}$	∞	∞	4.16	3.71	3.58	3.51	3.49

for the filter process and the idle operation are determined by the mode, there are $3 \times 10 + 9 = 39$ relevant high-level control actions per subsystem to be encoded in U_H . As high-level measurement, we choose the completion component from the low-level subsystems, i.e. $Y_H = \{\text{Busy}, \text{Done1}, \text{Done2}\}$. While low-level signals “live” on clock time, high-level signals evolve on logic (event-driven) time, where events are triggered by changes in the Y_{D_j} -components.

To design a high-level supervisor, we need a discrete abstraction $\tilde{\mathfrak{B}}_P^H$ of $\mathfrak{B}_{im}^H[\mathfrak{B}_P^L]$, a high-level “image” of the original specification \mathfrak{B}_{spec}^L , and a high-level cost function γ^H . As pointed out in Section 14.3.3, we can derive $\tilde{\mathfrak{B}}_P^H$ directly from \mathfrak{B}_{spec}^H . The specified external behavior of each subsystem under low-level control (w.r.t. the discrete variables u^{D_j} and y^{D_j}) can be modelled as a timed discrete-event system [110], where time is still clock time. To obtain a DES realization of an abstraction $\tilde{\mathfrak{B}}_P^H$ of $\mathfrak{B}_{im}^H[\mathfrak{B}_P^L]$, we form the synchronous product of the individual timed discrete-event systems and remove tick events (which “count” the progress of clock time) by language projection. Note that a composition of subsystems is only needed after the individual subsystems have already undergone considerable simplification.

According to (14.9), the high-level-specification $\tilde{\mathfrak{B}}_{spec}^H$ can be directly obtained from \mathfrak{B}_{spec}^D by transforming clock time to logic time. Together with the high-level abstraction $\tilde{\mathfrak{B}}_P^H$, we obtain a transition system with 17×10^6 states and an average of 13.1 relevant input events per state. Since every high-level input event corresponds to a low-level mode (chemical reaction or filtering) that will be completed at a known cost, the high-level cost function γ^H is additive over high-level time (i.e. cost per transitions). Thus, the high-level problem (14.13) can be solved using standard dynamical programming methods. On a decent desktop computer this takes about an hour. For illustration, Fig. 14.7 shows the obtained closed-loop operation to produce 12.5 m^3 , 12.5 m^3 and 7.5 m^3 of the products P1, P2, and P3, respectively. The overall cost amounts to 27.5.

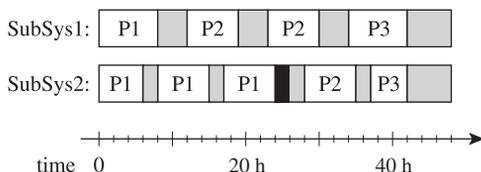


Fig. 14.7 Optimal schedule (filter processes grey, cleaning black).

This result shows that an abstraction-based control philosophy in combination with a hierarchical control architecture has the potential for reducing design complexity.

14.4 Model-predictive control of switched systems

14.4.1 Overview

The aim of this section is to show that engineered model-predictive control techniques can successfully solve industrial control problems that involve continuous and discrete inputs and switched dynamics. Two case studies will be discussed:

The first one is the control of a supermarket refrigeration system. It was provided by Danfoss, a leading manufacturer of refrigeration, air conditioning and heating systems as well as of control systems. The system is composed of a set of display cases, each with an evaporator system and a compressor rack. Here, all decision elements are discrete (on/off), and the goal is to maintain the temperature of the goods in the display cases in a certain range with minimum energy consumption and wear of the compressors.

The second case study concerns a multi-stage set of evaporators and was provided by Bayer Technology Services. The plant performs an industrial purification process where volatile components are removed from a liquid stream that contains the desired product. The control task is the optimization of the start-up procedure, i.e. to generate a control strategy that drives the evaporators from a shut-down (cold and empty) state to nominal operation in minimal time and with minimal resource consumption.

These case studies are realistic examples of systems with complex dynamics and switched inputs that cannot be handled straightforwardly by techniques for switched (piecewise affine) systems described by ODE models. While the solutions presented are problem-specific, they provide examples of a general approach to engineered solutions to complex hybrid control problems based upon hierarchical structures and a suitable parameterization of the degrees of freedom in the finite-horizon optimization problems.

For each case study, first the process is described and the goals of feedback control are formulated. Then, hybrid MPC control approaches are presented that result in optimization problems that can be solved in realistic computation times and provide good performance, as demonstrated by simulations.

14.4.2 Control of a supermarket refrigeration system

Display cases for refrigerated goods are present in all supermarkets and food stores. A schematic representation of such a process with two display cases is depicted in Fig. 14.8, and the cross section of a display case with its evaporation system is shown in Fig. 14.9.

The temperatures of the goods in the display cases must be kept within a specified range in order to preserve them for consumption. The temperatures of the air inside the display cases are used as an indirect measurement of the temperatures of the goods. The compressor rack creates a low pressure level in the suction manifold and in the evaporators which increases the rate of evaporation of the refrigerant, resulting in a transfer of heat from the walls of the display cases to the refrigerant. This is the basic mechanism for cooling the air around the refrigerated goods. The

compressed refrigerant is condensed and sent back to the inlets of the evaporators. The overall system is subject to significant disturbances, e.g. opening of display cases by customers and for refill, changes in the amount of goods placed in the display cases, and the transitions from day to night where the ambient temperature varies. In order to control the temperatures in the display cases, the inlet valves of the display cases and the state of the compressors (on/off) are used as actuators.

The process dynamics is continuous in nature. The hybrid character of the control problem results from the fact that the inlet valves of the evaporators are on/off valves and that the compressors can only be switched on or off, generating an inherent oscillatory behavior of the overall system. The process is a representative of the class of continuous or quasi-continuous processes operated with alternating on/off actions.

A full dynamic model of the system is given in [384]. The number of dynamic equations depends on the number of display cases that are modeled (four differential equations per display case, one of which is of variable structure). An additional differential equation is used to model the pressure in the suction manifold. Thus, for a system with two display cases, the model comprises nine nonlinear differential equations, and with five display cases, the model consists of 21 differential equations.

The control goal is to operate the system such that the temperatures of the air in the display cases are kept within a prescribed range ($2\text{--}5^\circ\text{C}$) with minimum energy consumption and a minimum number of switching of the manipulated variables, in spite of disturbances and the transitions from day to night. Moreover, the pressure in the suction manifold must not exceed a prescribed value.

The conventional control structure for a supermarket refrigeration system employs decentralized hysteresis temperature controllers in each evaporator that

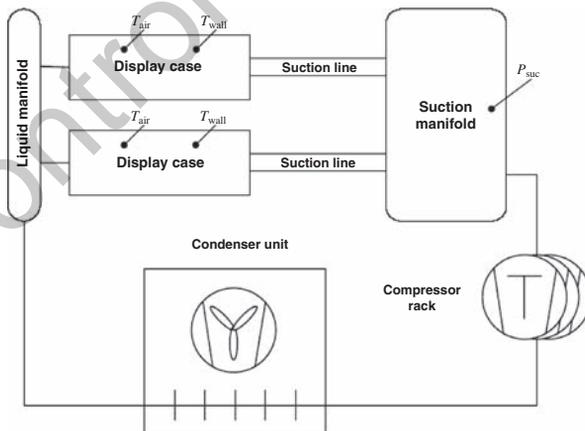


Fig. 14.8 Simplified layout of a supermarket refrigeration system.

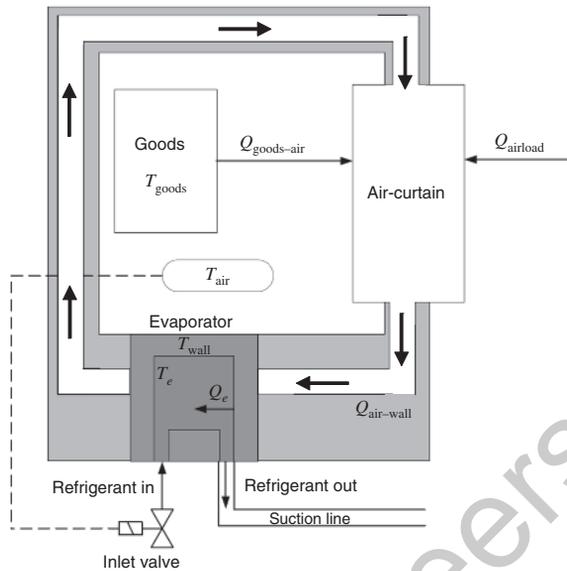


Fig. 14.9 Cross section of a refrigerated display case.

manipulate the on/off inlet valves, and PI-control of the suction pressure, with a quantifier on the controller output to determine the number of compressors in operation. This type of control, while cheap and reliable, gives rise to a phenomenon of synchronization in the operation of the display cases, i.e. the switching of the inlet valves occurs in a synchronized fashion, leading to frequent switching of the compressors which increases the wear of the compressors and creates peaks in the energy demand. This is the main problem that motivated research in more sophisticated control structures.

NMPC formulations for the supermarket refrigeration system Improving the conventional control by using a model of the hybrid system and optimization criteria for computing the control actions in a natural way leads to a finite-horizon optimization or MPC formulation. A direct solution of the hybrid problem was obtained in [383] using a MLD formulation in which binary variables represent the control actions at every sampling time, and changes in the model structure are translated into a set of linear constraints involving binary variables. If used with the nonlinear continuous process dynamics, this approach leads to a mixed-integer nonlinear programming problem that grows quickly with the considered look-ahead horizon to a size where solutions become too time-consuming.

We have, therefore, investigated two different engineered nonlinear model-predictive control (NMPC) formulations [570, 594, 595]. The “embedded logic” approach [570] is based upon two key elements:

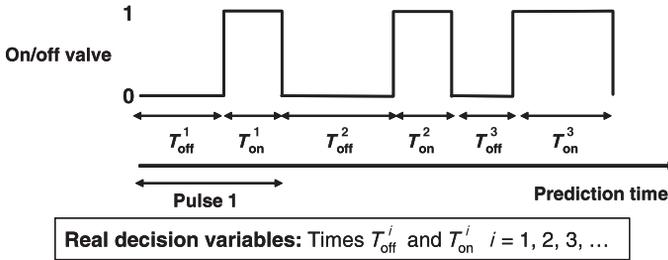


Fig. 14.10 Sequence of pulses of an on/off evaporator valve.

- An internal model is formulated in continuous time that integrates the first-principles-based model of the continuous dynamics, the logic of operation, and the binary actions. This model incorporates all discontinuities and possibly a variable structure using the formalism of the simulation environment. Using state-of-the-art simulation technology ([216], see also Chapter 11), both the values of the cost function over the prediction horizon and the values of the constraint residuals can be computed as a function of a given control policy over this horizon.
- A parameterization of the binary decision variables in terms of times of the occurrence of events is used. This parameterization is illustrated in Fig. 14.10 for the case of an evaporator inlet on/off valve. The time pattern of the valve is included in the simulation and the optimizer decides on the duration of the pulses, T_{on}^i , T_{off}^i , which are real variables. In addition, and extending the concept of control horizon of standard MPC, the pattern corresponding to the last pulse is repeated till the end of the prediction horizon, saving decision variables and imposing a certain regularity in the future behavior of the system. This parameterization has two advantages; the number of decision variables is usually smaller than in a formulation with binary variables for each sampling interval, and on the other hand, as all decision variables are continuous, the associated optimization problem can be formulated as a nonlinear programming problem (NLP), which can be solved more efficiently than a MINLP.

A similar policy can be applied to the compressor on/off actions but, for the sake of simplicity, a PI-control of the suction pressure, with a quantifier on the controller output has been maintained for the compressors in the simulation presented. The NLP optimization problem is solved in every sampling period using a sequential approach as depicted in Fig. 14.11.

In the second approach [594, 595], the same general idea is pursued, but the distribution of the decision variables is different. A hierarchical scheme is employed: the temperatures in the display cases are regulated by low-level controllers that open the inlet valves when a temperature threshold is reached. The durations of the openings and the switching strategy of the compressors are determined on a higher layer by optimization over a finite look-ahead horizon (Fig. 14.12). A branch-and-bound

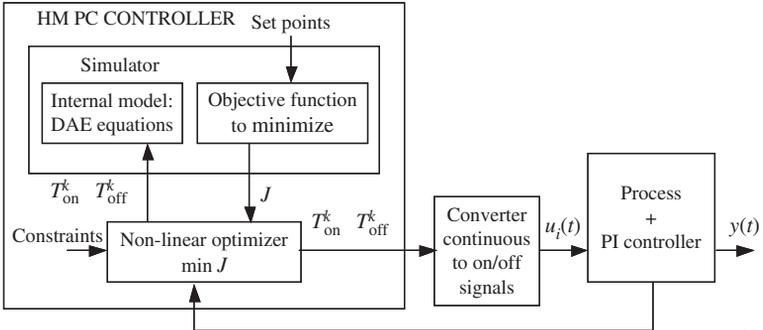


Fig. 14.11 Hybrid control implementation.

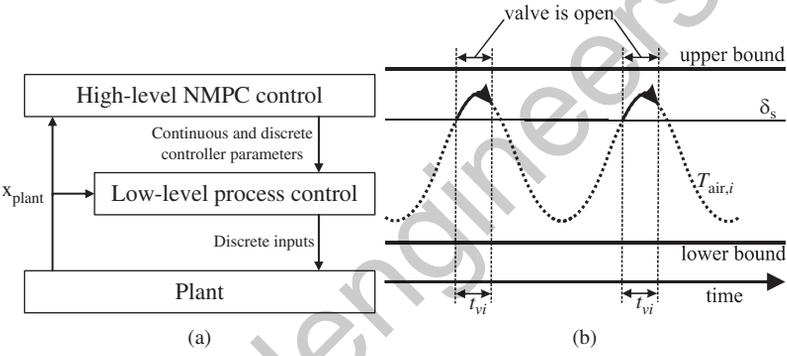


Fig. 14.12 A scheme of the hierarchical control approach (a) and the operation of the low-level temperature controllers (b).

technique is employed where patterns of possible compressor switching in the near future are investigated one-by-one, thus leading to a sequence of optimization problems with purely continuous degrees of freedom. If a pattern with no or few switches results in meeting the constraints for optimized inlet valve switching times, the search is terminated. Otherwise it is continued for a larger number of switches. The optimization horizon is divided into two parts. In the more remote future, the compressor switching is approximated by a real number of active compressors while in the near future, only integer values of the number of active compressors are considered (Fig. 14.13). This enables a larger look-ahead horizon for the optimization. An additional low-level system monitors the process and switches the compressors directly if fast and large changes in the external disturbances occur.

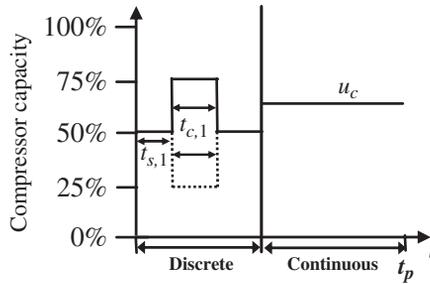


Fig. 14.13 Optimization of the switching for four compressors, shown exemplarily for a single compressor switching. t_p is the prediction horizon, $t_{s,1}$ and $t_{c,1}$ are the time points at which the compressors are switched, and u_c is a fictitious continuous compressor capacity.

Simulation results for the supermarket refrigeration system Results obtained with the first approach [570] for a system with two display cases and two compressors are shown in Fig. 14.14. The hybrid controller was tested in simulations using a cost function that penalized the deviation of the temperatures and pressure from their desired ranges. Figure 14.14 shows the simulation results of a four-hour experiment where the amount of goods in the display cases changed and a day/night transition took place after two hours. The sampling time was 1 minute, and the controller could operate in real time. On the right, the time evolution of the settings of two valves and two compressors can be seen, while the pressure, the air temperatures, and the goods temperatures appear on the left from top to bottom, showing very good behavior and no synchronization.

The second approach [594, 595] was applied to a system with five display cases and three compressors (with relative capacities of 30%, 30%, and 40%). The results are shown in Fig. 14.15. The controller and the simulation model were implemented in MATLAB, and the NLP problems were solved using the TOMLAB-based solver NPSOL [325]. The computation time per iteration of the high-level optimizer was set to 200 seconds. The controller is capable of keeping all process variables within the bounds and desynchronizes the air temperatures, which are completely synchronized initially, very quickly. Furthermore, the low-level pressure controller detects the drastic change in the external disturbances at the transition between day and night operation and switches off all compressors almost instantaneously.

14.4.3 Start-up of a multiple-effect evaporator system

The evaporator system The multi-stage evaporator case study deals with the start-up of a triple-effect evaporation system. Figure 14.16 shows a simplified flow sheet of the evaporation process. It consists of three similar evaporation stages, which are used to concentrate a feed stream (F_1) that contains a non-volatile organic component A (the product), as well as an alcoholic solvent and water. A stream of cold feed (F_1)

is injected into the first evaporator E1. The pumps (C) together with valves V_2 and V_3 are used to transfer liquid from one evaporator to the next (by controlling the flows F_2 and F_3). In normal operation, the heat supply to the first two stages is provided

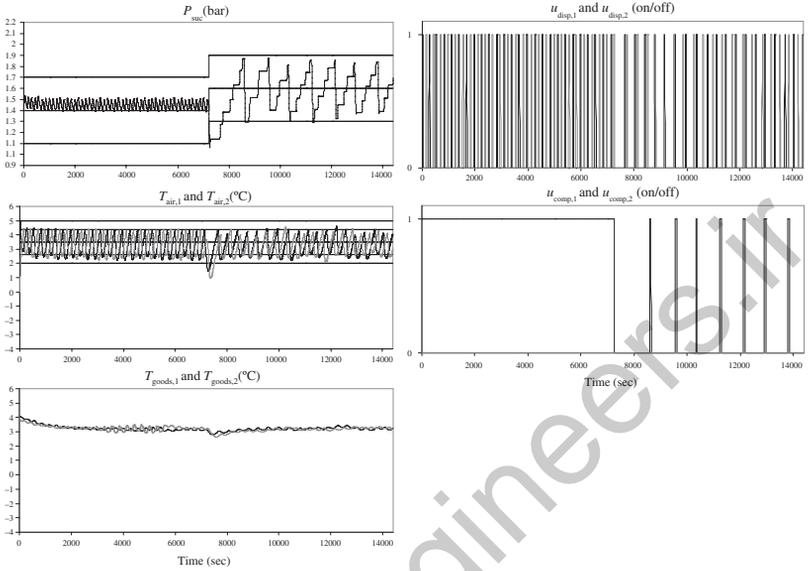


Fig. 14.14 Evolution of the main variables for a system with two display cases and two compressors using the first approach, before and after a day/night transient.

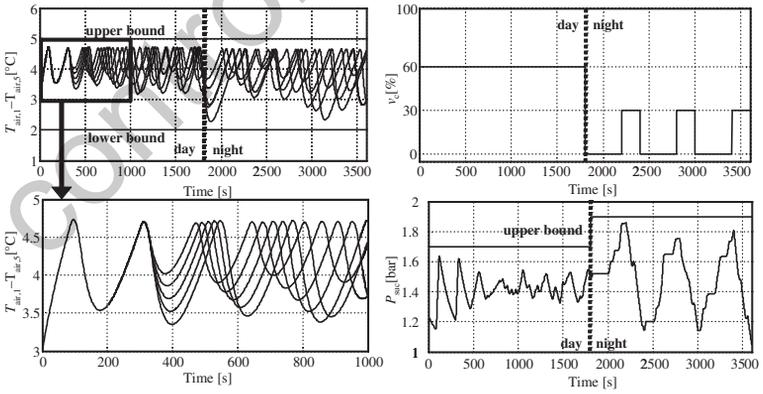


Fig. 14.15 Evolution of the main variables for a system with five display cases and three compressors using the second approach, before and after a day/night transient. The lower left figure shows an enlargement of the upper left figure.

by condensation of the hot vapor from the following evaporator in a heat exchanger. The last evaporator (E3) is heated by fresh hot steam controlled by the valve V_{V2} . The valves $V_{VS,1}$ and $V_{VS,2}$ can be used to switch between the supply of fresh steam to E1 and E2 and vapor from the following evaporator in the start-up phase. Once the liquid mixture in an evaporator starts to boil, the volatile components (water, alcohol) evaporate and can be used as heating vapor for the preceding stages. In order to establish the desired operation, the pressure profile must increase from the left to the right (i.e. $P_1 < P_2 < P_3$), since the boiling temperature of the mixture rises with an increase of pressure. The pressure in the first evaporator, P1, can be controlled by adjusting the flow rate F_C of the cooling water to the condenser CON. The final product is continuously drained from E3 if the concentration of the product *A* meets the specification.

The hybrid nature of the process is due to the fact that the process is operated using both continuous and discrete inputs: the regulation valves and the two on/off valves, $V_{VS,1}$ and $V_{VS,2}$. During transients, the process exhibits different continuous dynamics (evaporating, non-evaporating), leading to a variable-structure model. Note that the structure of the model changes also if the on/off valves are switched. The control goals are to drive the process from an empty and cold initial state to a final state where the levels are in the range of 60–64 % and the output concentration of *A* in E3 is in the range 0.8–0.84 kg/kg, while respecting constraints on the levels, the pressures, and the temperatures, and while minimizing the consumption of fresh steam and the duration of the start-up operation. A nonlinear model of the process based on first principles, as well as its description as a hybrid automaton, is given in [593].

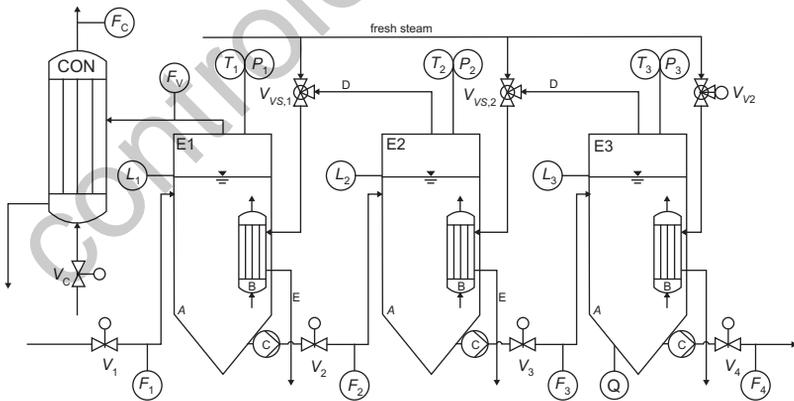


Fig. 14.16 The multi-stage evaporation system.

Computation of an optimal start-up policy The start-up of a single evaporator was optimized in [596] using a finite-horizon formulation with branch-and-bound search over the discrete variables and embedded nonlinear optimization of the continuous variables. The combinatorial complexity of the resulting problem is such that only relatively short horizons can be considered, necessitating an engineering of the cost functions based on knowledge on the global shape of the optimal trajectories.

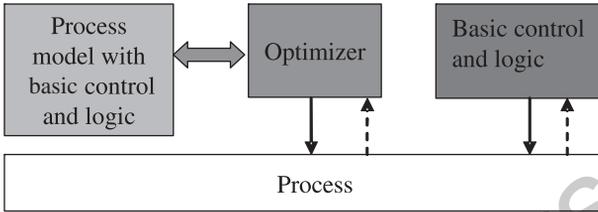


Fig. 14.17 Optimization architecture for start-up of the evaporators.

The approach taken in the optimization of the start-up of the three-stage system by [535] takes the logic steps that are used in the start-up operation according to the physical nature of the system into account, as well as the final control structure that is required for regulation of the process once the normal regime has been reached.

A key observation is that the control structure for steady-state operation supports the start-up operation and that the switching of the on/off valves $V_{VS,1}$ and $V_{VS,2}$ can be imposed when a certain pressure is reached. If this logic and continuous control structure is implemented, the number of degrees of freedom of the optimization is largely reduced. Consequently, for the optimization, logic and, basic controllers are embedded in the hybrid nonlinear model of the process, following the “logic-embedded” approach mentioned above.

The remaining degrees of freedom concern only continuous variables. Thus, the optimal start-up can be solved as a NLP problem using a sequential method.

The implementation requires a two-layer architecture as shown in Fig. 14.17. The lower layer includes PI-controllers and logic controllers that act on some process variables, while the upper layer optimizes a reduced set of key variables.

The normal operation of the evaporation system requires, at least, level control of all evaporators, pressure control in the vapor space of evaporator E1, and concentration control of product A in evaporator E3. Several alternative schematics are possible. One of them is displayed in Fig. 14.18, where the condenser is considered as a pressure boundary condition. Note that, if implemented in the initial phase, these loops will provide the right valve openings as required for the start-up. While the levels are below their set points, the valve at the liquid input of one evaporator will be open allowing the liquid to flow through the system, and when the level reaches the target, the controller keeps the level around the set point. In the same way, the output concentration controller will keep the liquid output valve of evaporator E3

closed until the product concentration reaches its target, and then it will maintain the concentration around the set point.

The only degree of freedom that is left in the system is the fresh steam supply to evaporator E3, which can be manipulated in order to optimize the transition of the station from the empty and cold state to normal operation. The set point of the vapor pressure in evaporator E1, or the switching pressure levels of valves $V_{VS,1}$ and $V_{VS,2}$, could be considered as additional degrees of freedom, but the sensibility of the solution to these variables is quite small.

The optimization problem was formulated as a dynamical optimal control problem in the NMPC framework where two performance objectives combined in a cost function with the appropriate weighting. The first one refers to minimizing the start-up time t_f , that is, the time needed to reach the target region, and the second one is to minimize the amount of fresh steam required for the start-up. The manipulated variable, the fresh steam valve opening that governs the steam flow F_b to the evaporator E3, is designed for minimizing the start-up cost taking into consideration the nonlinear dynamical model of the plant and its basic control system as well as constraints on the inputs and the outputs of the process using a suitable control vector parameterization technique and a sequential approach for computing the cost function.

A problem with the hybrid control approach is associated with the estimation of the gradients for the NLP optimization. The estimation of the gradients, or

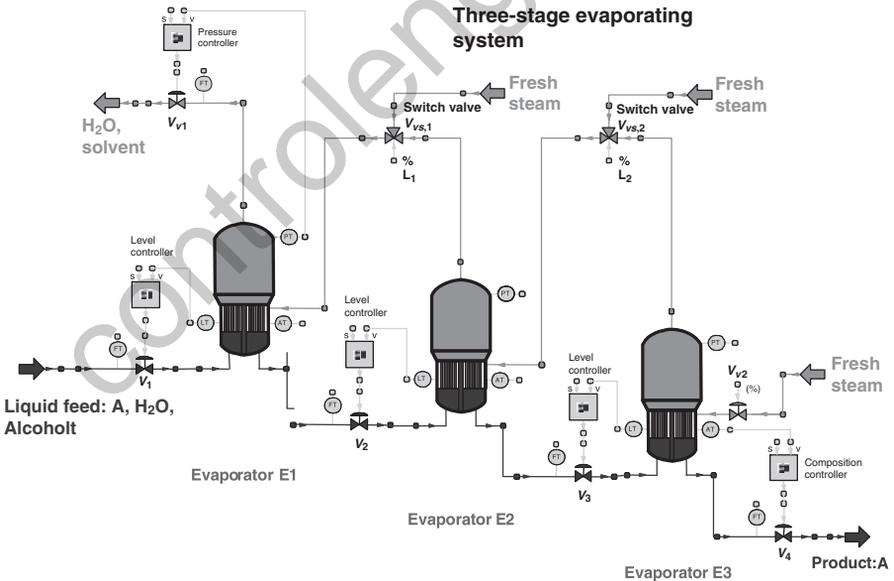


Fig. 14.18 Basic control loops of the evaporation system.

equivalently of the sensitivities with respect to the decision variables, can often be performed in spite of the fact that each time a discontinuity takes place along the prediction horizon, the sensitivities have to be reset to a new value. In problems where the discontinuities are due to the on/off nature of the manipulated variables and where the order and the number of states and of switches does not change along the prediction horizon, as in the displays cases example, the computation of the gradients can be done safely. In contrast, when the number of switches of the on/off variables is not fixed or the number of states may change, we can expect difficulties with single-shooting NLP algorithms due to the discontinuities in the gradients. Therefore, in the evaporator example, a parameterization of the decision variable (flow of fresh steam) was used that adapts the intervals where this variable is constant to the periods between events created by the underlying controllers during start-up.

Simulation results for the start-up of the evaporator system The simulation results depicted in Fig. 14.19 together with the main constraints show a good behavior. The optimal start-up takes 3 hours and all variables are maintained within their range as can be seen in the graphs. The sampling time is 2 minutes. For the computation of the cost function to be minimized, which depends on the time needed to reach normal operation conditions, the simulation stops when the target is reached. In case it is not reached within the prediction horizon length of eight hours, a penalty term is applied.

14.5 Summary and outlook on further industrial control applications

The examples dealt with in this chapter highlight the challenges faced in process control: The continuous dynamics are of high order, usually nonlinear and described frequently by sets of differential algebraic equations (DAE systems) that cannot easily be transformed to systems of ordinary differential equations. The time horizon of the evolution of the systems is quite long, but the temporal resolution has to be high because different phenomena interact and have to be controlled on different time-scales. In consequence, brute-force methods for analysis and synthesis or optimization quickly reach the limits of the available computing power because of the resulting large number of discrete variables [622].

The key to successful applications of advanced control methods in this area is the engineering of solutions: introduction of different control layers that act on different time scales, focussing on the relevant decisions and variables, and suitable parameterizations of the controlled inputs.

In this fashion, problems of realistic complexity can be solved satisfactorily using state-of-the-art numerical tools.

In essence, the formulation of verification and optimization problems in hybrid process control leads to the need to solve (large) dynamic mixed-integer nonlinear

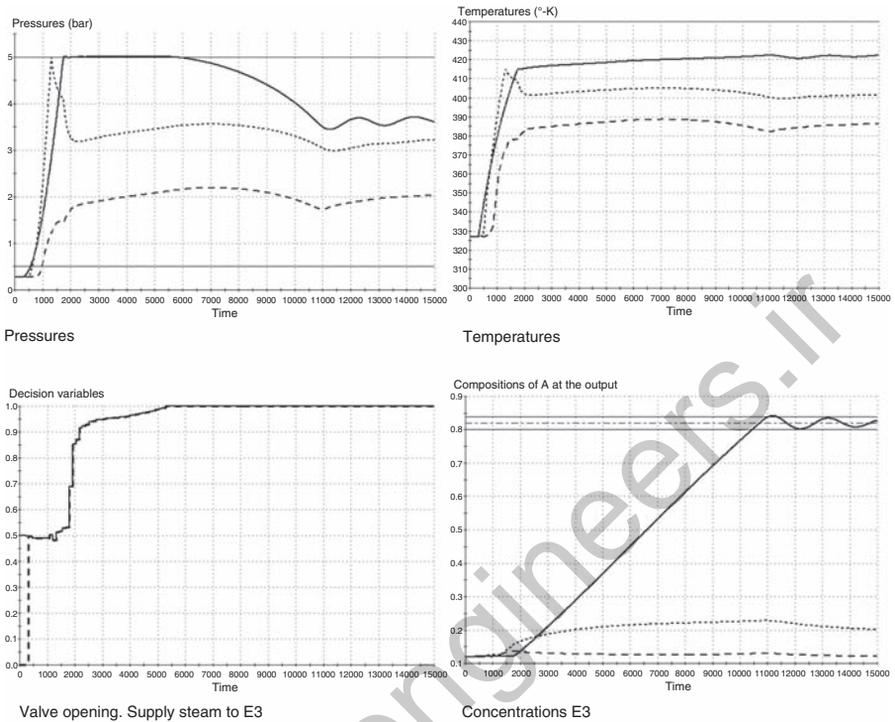


Fig. 14.19 Evolution of the main variables in the optimal start-up of the evaporator system.

optimization problems. General tools for this class of problems are not yet available, but this is becoming an area of very active research (e.g. see <http://egon.cheme.cmu.edu/ibm/page.htm>). Alternatively, the structures of dynamic optimal control problems can be exploited in order to arrive at efficient algorithms. An example of this route is the work in [563] where dynamic optimization problems with switched inputs are solved for nonlinear DAE-systems using a multiple-shooting approach. While in optimization convergence to local optima is not desirable but usually provides acceptable solutions, in verification the lack of global solutions constitutes a serious problem because then the existence of a counterexample and hence of an unsafe situation cannot be ruled out. Exact global optimization algorithms such as, e.g., *BARON* [618] can currently only handle problems of quite moderate size and fast improvements in this area seem unlikely. Thus divide-and-conquer strategies seem indispensable.

A general conceptual problem with verification in hybrid control systems is the lack of accuracy of the models that are employed in the process. Models of the system dynamics provide approximations of the behavior of the system, thus “exact” results for the model may or may not be reliable in reality. To worry too much about the exactness of the computations in this context may not be worthwhile. The derivation of (simplified) models that are faithful over-approximations of the behavior of complex processing systems thus far has turned out to be very difficult. Rather than a true/false safety result for one given model, meaningful measures of the “distance to unsafe” of the controlled system would be extremely valuable, in particular if they can be related to measures of the accuracy of the models, similar to the well-known robustness results for linear systems.

controlengineers.ir

controlengineers.ir

Automotive control

L. Benvenuti, A. Balluchi, A. Bemporad, S. Di Cairano, B. Johansson, R. Johansson, A. Sangiovanni Vincentelli, and P. Tunestål

Automotive systems offer a rich opportunity for hybrid models, controls, and tools. Beyond the traditional use of hybrid models for representing the behavior of the composition of discrete controller and continuous plants, automotive mechanical systems exhibit hybrid behavior as demonstrated in this chapter. In addition, hybrid systems can be used to capture system specifications at the highest level of abstraction and to model implementation architectures thus enabling a rich design space exploration.

Chapter contents

15.1	Introduction	page 440
15.2	Design methodologies for embedded automotive control systems	441
15.2.1	System integration	441
15.2.2	Design scenario and design flow	442
15.2.3	Control system design	445
15.3	Hybrid model-predictive control in automotive applications	452
15.3.1	Motivation	452
15.3.2	Survey of MPC applications in vehicle dynamics control	453
15.3.3	Survey of MPC applications in powertrain control	454
15.3.4	Example: Hybrid MPC of magnetic actuators	455
15.4	Control of a homogeneous charge compression ignition engine	461
15.4.1	HCCI engine concept	461
15.4.2	Control of ignition timing	463
15.4.3	Properties and model of the HCCI engine	464

15.1 Introduction

This chapter presents an application of hybrid systems that is of significant industrial interest: power-train modeling and control for automobiles.

Engine control is a challenging problem that involves many functional and non-functional requirements. The problem is to develop control algorithms and their implementation with guaranteed properties that can substantially reduce emissions and gas consumption with increased performance.

The introduction of hybrid system modeling and control was motivated by the need for verifying closed-loop systems where the plant to be controlled are continuous-time systems and the controller is a digital system. However, hybrid models are general enough to be useful in other areas of design. In particular, engine control offers a rich set of application of hybrid systems:

- The power-train itself can be represented as a hybrid system. In fact, an accurate model of a four-stroke gasoline engine has a “natural” hybrid representation:
 - Each cylinder in the engine has four discrete modes of operation corresponding to the stroke it is in (hence, its behavior is well represented by a finite state machine (FSM)).
 - The power-train delivering the torque produced by the engine can be modeled as a continuous-time process.

The continuous motion of the power-train depends on the torque generated by the engine and at the same time, it determines the timing of transitions between the modes of operation of each cylinder so that the behavior of the plant cannot be decomposed into two independent parts. In traditional design approaches, the modes of operation of the cylinders are smoothed out by using an average continuous model thus converting a mixed discrete-continuous time model into a purely continuous one. This approximation results in control actions that may perform poorly during the transients caused by the mode changes of the cylinder. Hybrid models of course do not suffer from these effects but they do require more complex control algorithm.

- Hybrid models can be extended to address problems at the boundary between control algorithm design and hw/sw implementation, since they allow the designer to capture the effects of limited resources and physical constraints on the performance of the controlled system and check the correctness of the design.
- Hybrid formalisms can be applied to represent system specifications and deploy them in an architecture of control algorithms and requirements.

This chapter is intended to increase the awareness of the automotive industry to the currently available methods, tools, and successful applications in the field.

The chapter is divided into three sections: in the first section we present a hybrid model-based design flow for embedded controllers in automotive control.

In the second section we present the use of hybrid model-predictive control (MPC) in automotive applications. MPC is an optimization approach that takes into account a cost function and constraints. Hybrid MPC provides the further capability

of employing hybrid prediction models. The application of hybrid MPC to vehicle dynamics control and to power-train control is presented.

In the third section, the behavior of the homogeneous charge compression ignition engine is described. This engine combines features of the traditional spark ignited Otto-cycle and compression ignited Diesel-cycle engines. This engine is indeed a new concept and it has great potential for substantial reduction of energy consumption and pollution. In this section, we make the case for hybrid modeling of these engines suitable for control design.

15.2 Design methodologies for embedded automotive control systems

15.2.1 System integration

Automobile functions such as engine control, gear-box control, anti-lock braking system (ABS), dashboard controller, and vehicle dynamical control (VDC) are today enabled by a distributed embedded system.

This system consists of a number of ECUs dedicated to each function yielding the so-called federated architecture. The trend is towards an integrated modular approach where the one-to-one correspondence between a function and its implementation does not hold any longer: a function may be supported by more than one ECUs and/or one ECU can support more than one function. This evolution offers clear advantages in terms of cost, extensibility and scalability but does present serious challenges.

Tier-1 suppliers responsible for the subsystems mentioned above, in general, give scant information about the details of their products causing problems for an overall understanding of the interplay of subsystems. The difficulties encountered in integrating complex parts made system integration a nightmare in the automotive industry. To mitigate the integration problems, the AUTOSAR initiative [308], promoted by leading European original equipment manufacturers (OEMs) and Tier-1 suppliers, was formed to establish an open standard for automotive electric/electronic architectures that hopefully allows plug-and-play of embedded controllers. Even in the presence of solidly established standards, the design process for embedded controllers has to be significantly improved to support the movement towards an integrated modular architecture.

Successful approaches to the design of control algorithms using hybrid system methodologies had been presented in the literature, e.g. cut-off control [33], intake throttle valve control [40], actual engaged gear identification [38], and adaptive cruise control [459].

In this section, we analyze a design flow for embedded controllers in the automotive industry with the purpose of identifying challenges and additional opportunities for hybrid system technology. In particular, in [Subsection 15.2.2](#), an overview of the typical design flow for embedded controllers adopted by the automotive industry is presented. The flow is divided into two parts: synthesis and verification/testing. We

focus on the synthesis flow as it does have the highest potential of improving substantially design productivity. In [Subsection 15.2.3](#), we present relevant problems that hybrid system technologies contributed to solve.

15.2.2 Design scenario and design flow

The ECUs interact by asynchronous communication over a communication network specifically designed for automotive applications, such as CAN ([340, 561]). Typically, an ECU implements a multirate control system composed of nested control loops, with frequency and phase drifts between fixed sampling-time actions and event-driven actions.

A typical ECU (e.g., the one demanded for engine control) may have more than one hundred I/O signals, implement up to three hundred control algorithms, and share approximately one hundred signals with the other related ECUs. The complexity of the design of automotive ECUs is further increased by very critical constraints on reliability, cost and time-to-market, and constraints on power consumption, weight, and position. As a consequence, a successful design, in which costly and time consuming re-design cycles are avoided, can only be achieved using efficient design methodologies that allow for component reuse and for evaluation of platform requirements at the early stages of the design flow [297].

Design flow The standard design flow for automotive ECUs adopted by OEMs and Tier-1 companies (subsystem suppliers) consists of two main parts: the *synthesis flow* and the *integration and testing flow*. In particular, the synthesis flow is articulated in the following steps:

1. *System specification*: Formalization of system level customer requirements; completion of under-specified requirements; abstraction at the system level of customer requirements regarding lower layers (e.g. either a control algorithm or a piece of software to be integrated in the design).
2. *Functional deployment*: Decomposition of the system into a collection of interacting subsystems. The specifications for each subsystem are derived from the overall specifications. For each subsystem, the architecture of control algorithms and their specifications are also defined.
3. *Control system*: Synthesis of each control algorithm, according to the specification defined in the previous step, and its validation.
4. *HW/SW components*: Specifications for the implementation of the control algorithms. Design of the hardware and software architectures.

The synthesis flow terminates with the development of the hardware, the software and possibly some electromechanical components.

The purpose of the integration and testing flow is the complete testing of the realization of the ECU and the verification of the compliance with the requirements. The steps taken in the integration and testing flow are as follows:

1. *HW/SW testing*: The correct realization of the hardware and software architecture is verified. This step includes testing of real-time implementation requirements, electrical power drivers, and communication.
2. *Control validation*: The correct implementation of each control algorithm with respect to the given functional description is assessed by testing either its input/output response or its closed-loop behavior.
3. *Functional integration*: The correct interaction of the implemented control algorithms is tested considering an increasing number of algorithms together, to verify that their composition exhibit the behavior defined during functional deployment.
4. *System testing*: The entire ECU is tested against system specification and the compliance with customer requirements is verified.

Platform-based design The *platform-based design* methodology proposed in [565] is a convenient framework to cast the design problems and the flow presented here. In addition, it provides concepts and techniques to maximize reuse at each design step and early verification with abstracted information from possible implementation platforms [358, 566].

A platform is a layer of abstraction that hides the *unnecessary* details of the underlying implementation and yet carries enough information about the layers below to prevent design iterations. The choice of the layers of abstraction and of the corresponding parameters are essential in the quality of the final solution of the design problem.

The basic tenets of the platform-based design methodology are:

- ‘regarding design as a “meeting-in-the-middle process”’ where successive refinements of specifications meet with abstractions of potential implementation;
- the identification of precisely defined layers where the refinement and abstraction process take place.

The layers then support designs built upon them by isolating from lower-level details but letting enough information transpire about lower levels of abstraction to allow design space exploration with a fairly accurate prediction of the properties of the final implementation. The information should be incorporated in appropriate parameters that annotate design choices at the present layer of abstraction. These layers of abstraction are called *platforms*.

In [21], the application of the platform-based design methodology to the design of powertrain control systems was described.

Derivative design Since, in the automotive industry, embedded control system design is highly dominated by the need to implement an efficient reuse to meet increasing constraints on cost and time-to-market, a *derivative design* approach is commonly adopted [440, 441]. According to this approach, every two or three years a new generation of products is conceived. The design of a new generation is intended to accommodate the specifications of customers for the near future, so that for each new customer engagement, the control algorithms as well as the electrical and mechanical components are obtained by (hopefully minor) modifications of the current product generation. When entering the design phase of a new product generation, the architecture of control algorithms as well as of their implementation should be conceived to maximize future re-use, for instance by choosing the correct granularity of partitioning. The resulting ECUs are then variants of a same originating design and ideally share the highest number of parts (e.g. algorithms, software modules, hardware parts, mechanical components).

Model-based design Another standard approach in automotive industry is the so-called *model-based design*. In this approach, specifications, functional architectures, algorithms, and implementation architectures are represented formally by models, thus allowing, at least in principle, formal analysis and automatic synthesis. Using block diagram-based modeling tools, control algorithms are designed and initial validation in off-line simulation is performed. Models of control algorithms are the basis for all subsequent development stages.

The advantages of model-based design are obvious:

- sharing models reduces the risk of mistakes and shortens the development cycles;
- design choices can be explored and evaluated much faster and more reliably;
- the result of a model-based development process is an optimized and fully tested system.

While the overall approach is quite powerful, today there is only an incomplete implementation of it in the development cycle. In fact, model-based design is widely used for the formal representation of control algorithms, using tools such as *Simulink/Stateflow* or *ASCET* by ETAS, but it is partially and superficially applied to control algorithm validation. The lack of an extensive model-based validation of the control algorithms results in major efforts in experimental validation, which is very expensive, time-consuming, and achieves only a bounded coverage of the system behavior. Due to the high cost of experimental validation, the OEMs will provide less support to Tier-1 companies for it in the future. In the rest of this section, the part of the synthesis flow regarding control design will be analyzed in detail, showing the design steps for which hybrid system techniques contributed or may significantly contribute to provide a more efficient approach.

15.2.3 Control system design

At the control system level, the algorithms to be implemented in the architecture defined at the functional level are designed. The design process for each control algorithm involves:

1. plant modeling;
2. controller synthesis;
3. fast prototyping.

Plant modeling Traditionally, control engineers adopt mean-value models to represent the behavior of automotive subsystems, which abstract from the hybrid nature. However, the need for hybrid system formalisms to model the behavior of subsystems in automotive applications is apparent in many cases. To illustrate the relevance of hybrid modeling in automotive applications, we briefly describe a hybrid model for a spark ignition engine and an automotive driveline.

Example 15.1 *Hybrid model of a four-stroke spark ignition engine.*

An accurate model of a four-stroke spark ignition engine has a natural hybrid representation because the cylinders have four modes of operation corresponding to the stroke they are in, while driveline and air dynamics are continuous-time processes. In addition, these processes interact tightly. In fact, the timing of the transitions between two phases of the cylinders is determined by the continuous motion of the driveline, which in turn depends on the torque produced by each piston.

In more detail, consider the hybrid model of the torque generation process and the driveline presented in [34]. For a given gear selection and clutch position, the driveline is described by a continuous time system whose state includes the driveline torsion angle, the crankshaft revolution speed, and the wheel revolution speed. The inputs of the model are the torque T produced by the engine and the load wheel torque T_w .

The engine torque T is given by $\sum_{i=1}^N T^i$, where T^i is the torque generated by each piston at each cycle. The profile of T^i is determined by the phases of the cylinder, the piston position, the mass of air and the mass of fuel loaded in the cylinder during the intake phase, and on the spark ignition timing.

The four-stroke engine cycle can be modeled by means of a finite-state machine (FSM) capturing the sequential nature of the behavior of the cylinders. In fact, each cylinder cycles through the following four phases:

- *Intake (I)*: The piston goes down from the *top dead center* (TDC) to the *bottom dead center* (BDC) loading the air–fuel mix present in the intake manifold.
- *Compression (C)*: The trapped mix is compressed by the piston during its upward movement from the BDC to the TDC.
- *Expansion (E)*: The combustion takes place, pushing down the piston from the TDC to the BDC.
- *Exhaust (H)*: During its upward movement, from the BDC to the TDC, the piston expels combustion exhaust gases.

However, for spark ignition engines, the torque generated by each piston is related not only to the phase of the cylinder and the air and fuel charge, but also to the spark generation process. Intuitively, spark ignition should occur exactly when the piston reaches the TDC of the compression stroke. Since the combustion process takes nonzero time to complete, then the pressure in the cylinder reaches its maximum some time after spark ignition. As a consequence, in order to achieve maximum fuel efficiency, it is convenient to produce the spark before the piston completes the compression stroke (*positive spark advance*). On the other hand, producing a spark after the piston has completed the compression phase and is in the expansion stroke (*negative spark advance*) may be used to reduce drastically (and much faster than using only the throttle valve) the value of the torque generated during the expansion run. Since spark ignition may occur either during the compression stroke or during the expansion stroke, a six-state FSM is needed to model the possible behaviors

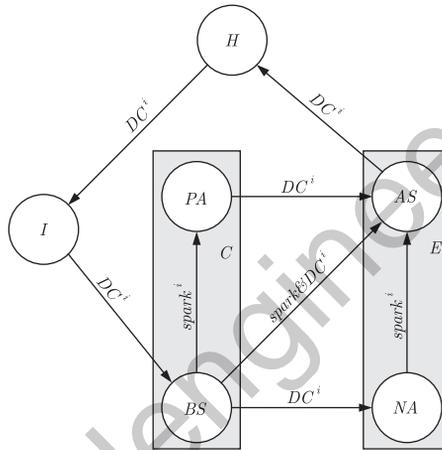


Fig. 15.1 Finite-state machine describing the behavior of the i -th cylinder.

of the cylinder. The cylinder FSM is shown in Fig. 15.1. The FSM state takes one of the following values:

- I , denoting *intake*;
- BS , denoting *before spark*: the piston is in the compression stroke and no spark has been ignited yet;
- PA , denoting *positive advance*: the piston is in the compression stroke and the spark has been ignited;
- NA , denoting *negative advance*: the piston is in the expansion stroke and the spark has not been ignited yet;
- AS , denoting *after spark*: the piston is in the expansion stroke and the spark has been ignited;
- H , denoting *exhaust*.

The cylinder FSM changes its state either when a spark is given or when a dead center (DC) is reached. The latter event depends on the continuous motion of the driveline and more precisely on the crankshaft angle, which defines the position of the piston. In turn, the crankshaft revolution speed depends on the torque T produced by the engine.

Finally, the torque produced by the cylinder depends on the air–fuel mixture loaded during the intake stroke. Since the air–fuel mixture is loaded in the cylinder during the intake stroke while the torque generation starts after the spark is ignited, then there is a delay between the time at which the mixture is loaded and the time at which the corresponding active torque is generated. This delay can be modeled by means of another discrete-event model that is synchronized with the FSM transitions. The overall model of the torque generation process for a single cylinder consists then of four communicating submodels:

- an FSM, modeling the four-stroke engine cycle and the spark generation process;
- a discrete model describing the discrete delay on the active torque generation; and
- two continuous-time systems, modeling the air intake process and the profile of the generated torque. □

Example 15.2 *Driveline*

A second very interesting automotive subsystem rich of discrete-continuous interactions is the driveline [37]. An accurate model of the driveline has a natural hybrid representation because of the discontinuities due to clutch and the gear on the continuous motion of the driveline.

The clutch can be modeled as a hybrid system with three discrete states: *Locked*, *Slipping*, and *Unlocked*. In Fig. 15.2 an FSM representing the clutch is depicted. When the clutch is *Locked* the clutch plate and the flywheel are rigidly connected by static friction, so that their inertias are collected in a single first-order dynamics.

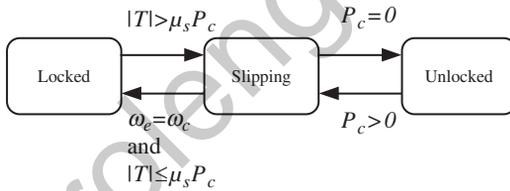


Fig. 15.2 Hybrid model of the clutch.

The highest coupling torque T_{\max} before incurring in clutch slipping corresponds to the maximum static friction torque, which is a function of the pressure P_c between the clutch plate and the flywheel, i.e. $T_{\max} = \mu_s P_c$. When the transmitted torque T exceeds T_{\max} , the system enters the state *Slipping*, where the clutch plate and the flywheel are no longer strictly connected but they slip one on the other. In this case the coupling torque is due to dynamical friction and it is a function of the sliding speed. If the transmitted torque decreases, then the clutch returns in the state *Locked*, while if $P_c = 0$ the clutch enters the state *Unlocked*. In the state *Unlocked* the crankshaft is completely decoupled from the rest of the driveline and the two systems follow independent dynamics.

For some applications, e.g. actual engaged gear identification [38], it is useful to collect in a single state the clutch *Unlocked* and *Slipping* states. In such cases, the overall system can be described by a hybrid system with seven discrete states and four continuous-state variables.

The FSM describing the discrete dynamics of the model is depicted in Fig. 15.3. The discrete state q_i , for $i = 1, \dots, 5$, correspond to i -th gear engaged and clutch locked.

The location q_{RG} models reverse gear engaged and the location q_N corresponds to the case of open driveline (idle gear and/or clutch open) or slipping clutch. The continuous state variables are the driveline torsion angle α , the crankshaft revolution speed ω_e , the clutch plate revolution speed ω_c , and the wheel revolution speed ω_w . When the clutch is locked, then $\omega_e = \omega_c$, so that the continuous behavior of the driveline can be described by a third-order linear system with parameters depending on the selected gear. The hybrid model has as inputs the position of the gear lever

$$\text{lever} \in \{1, 2, 3, 4, 5, RG, N\}$$

and the torque generated by the engine while the connection pressure of the clutch plates P_c and the load wheel torque T_w are considered as disturbances. A more detailed driveline hybrid model, with 6048 discrete state combinations and 12 continuous state variables, is presented in [37]. In addition to the clutch and the gear, the proposed hybrid model describes the discontinuities in the driveline due to engine suspension, elastic torsional characteristic, tires, frictions and backlashes. This very detailed driveline hybrid model exhibits a behavior very close to the physical driveline and has been developed to be used for control algorithm validation. □

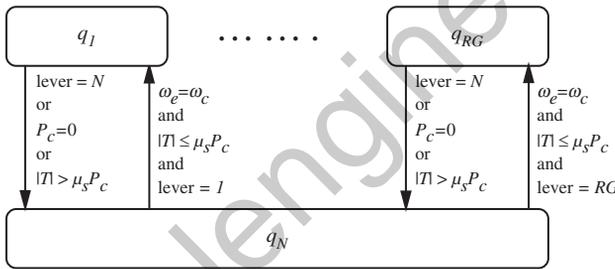


Fig. 15.3 Hybrid model of the driveline.

Controller synthesis Control algorithms are often characterized by many operation modes, that are conceived to cover the entire life-time of the product: starting from in-factory operations before car installation, configuration, first power-on, power-on, functioning, power-off, connection to diagnostic tools, and so on. During normal functioning, control strategies can be either in one of the nominal operation modes or in some recovery mode. A significant number of algorithms are dedicated to the computation of switching conditions between modes and controller initializations.

A short and by no-means exhaustive list of control actions for which hybrid system design is particularly interesting is as follows: fuel injection, spark ignition, throttle valve control (especially with stepper motor), electromechanical intake/exhaust valve control, engine start-up and stroke detection, crankshaft sensor

management, VGT and EGR actuation (hysteresis management), emission control (cold start-up, lambda on/off sensor feedback), longitudinal oscillations control (backlash and elasticity discontinuities), gear-box control (servo-actuation in traditional gear shift systems), cruise control and adaptive cruise control, diagnostic algorithms (signals and functionalities on-line monitoring), and algorithms for fault-tolerance, safety, and recovery (degraded mode activation).

Diagnostic algorithms represent a large part of the strategies implemented in automotive ECUs. For engine control, the implementation of diagnosis algorithms is enforced by legislation: OBDII (on-board diagnosis II) in USA and EOBD (European On Board Diagnosis) in the EU. In general, these requirements specify that every fault, malfunction or simple component degradation that leads to pollutant emissions over given thresholds should be diagnosed and signaled to the driver. This requirement has a significant impact on ECU design, since it implies the development of many on-line diagnostic algorithms [168].

Both specifications and accurate models of the plant are often hybrid in automotive applications but the methodology currently adopted for algorithm development is rather crude and can be summarized as follows: the continuous functionalities to be implemented in the controller are designed based on mean-value models of the plant, with some *ad hoc* solutions to manage hybrid system issues (such as synchronization with event-based behaviors); if the resulting behavior is not satisfactory under some specific conditions, then the controller is modified to detect critical behaviors and operate consequently (introducing further control switching). Nevertheless hybrid system techniques can significantly contribute to the improvement of control algorithm design as illustrated by the two following algorithms regarding cut-off control and identification of the actual engaged gear.

Cut-off control A quite critical driveability objective is the control of longitudinal oscillations of the car when fast engine torque variations are requested by the driver (tip-in and tip-out). Roughly speaking, the control consists of active damping of powertrain oscillations.

The problem is particularly challenging when the engine is not equipped with electronic throttle valve, since in this case only fuel injection and spark ignition controls can be used for engine torque modulation to achieve the desired damping of the oscillations. Most of the proposed approaches are based on mean-value continuous-time models of the torque generation. As a consequence, since the torque generation process has a discrete behavior, the implementation of such control strategies on a real engine may result in very poor closed-loop performance and may give rise to unpredicted unpleasant behaviors. On the contrary, a design based on a hybrid model of the engine allows to develop control laws for which closed-loop performances are guaranteed.

A hybrid approach to the design of a longitudinal oscillation damping control during tip-out was presented in [33]. The control problem arises when the driver, by releasing the fuel pedal, requests no torque to the engine. In this case, an obvious strategy to minimize fuel consumption and emissions is to shut fuel injection, an operation called cut-off. However, cutting off fuel injection as soon as the gas pedal is

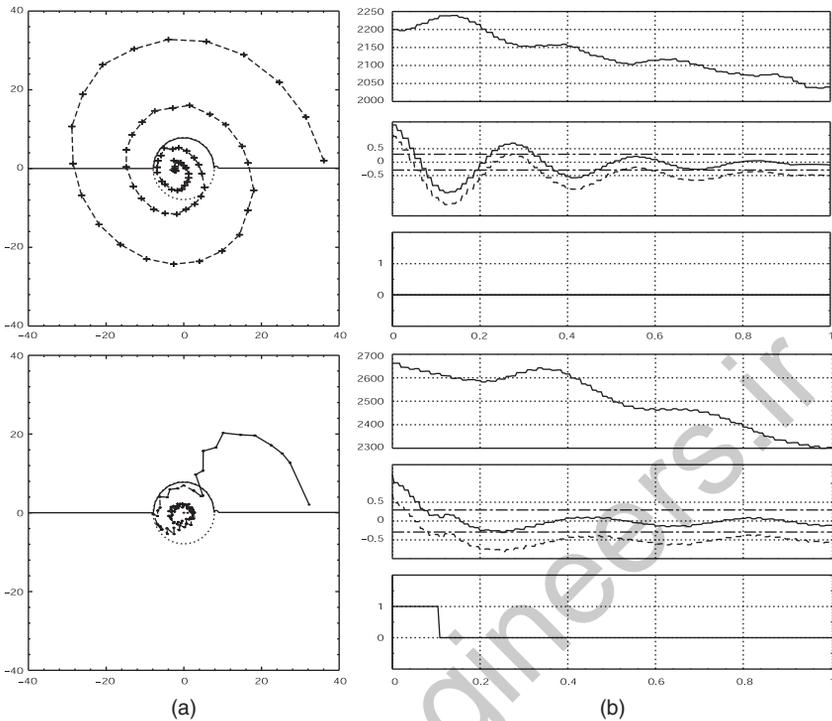


Fig. 15.4 Evolutions of the oscillating modes (a) and engine speed, accelerations, and injection signal profiles (b), in an uncontrolled cut-off (top) and with the proposed hybrid control strategy (bottom).

released, causes a sudden torque reduction that may result in unpleasant oscillations compromising driving comfort. A more complex control action involves modulating the engine torque from the present value to the value corresponding to cut-off in an attempt to prevent oscillations. This control policy is implemented by slowing down air flow decay and, when air quantity is below a threshold, reducing fuel injection gradually to zero.

As it is often the case, heuristic rule-based controls need extensive tuning, yield satisfactory solutions only in a limited range of operations, and are hardly optimal with respect to the emissions and fuel consumption. In particular, if air reduction is too slow, when the driver releases the gas pedal and presses the clutch pedal to change gear, engine speed raises for a while, thus causing a definite reduction in passengers' comfort. Moreover, if air and/or fuel reduction is too fast, oscillations take place anyway.

The hybrid control algorithm presented in [33] is able to steer the evolution of the system to the fuel cut-off condition, minimizing the amplitude of the undesired oscillations. Since a hybrid model of the engine has been considered during the design, the algorithm acts on fuel injection and spark ignition once per engine cycle

for each cylinder taking into account synchronization and actuators' delay. The hybrid approach adopted for synthesis of the cut-off control algorithm guarantees the correctness of the behavior when applied to the real plant.

The proposed cut-off control strategy was tested at Magneti-Marelli Engine Control Division on a commercial car, a 16-valve 1400 cc engine car. The experiment was carried out driving the car in the test ring and measuring the important parameters and variables that determine the performance of the control strategy. In Fig. 15.4 the performance achieved by the proposed hybrid cut-off strategy is compared with an instantaneous uncontrolled cut-off operation. On the left the evolution of the oscillating modes \hat{x} are reported. The effectiveness of the proposed controller is apparent from the evolution of the vehicle acceleration and engine speed reported on the right.

Actual engaged gear identification As a second example, consider the problem of on-line identification of the actual engaged gear. Engine control strategies achieving high performance and efficient emissions control depend critically on such identification algorithm. In fact, the knowledge of the actual engaged gear is necessary in engine torque control to compensate the equivalent inertia of the vehicle on the crankshaft and, for diesel engines, it is very important to improve emissions control. In cars equipped with manual gear shift, this information is not directly available and, at present, it is deduced by comparing the revolution speed of the wheels with the revolution speed of the crankshaft. However, since both of them are affected by oscillations due to the elasticity of the driveline and the tires, then this approach implies large time delays in the identification and may produce significant identification errors.

The identification algorithm presented in [38] is based on a hybrid model of the driveline (see the driveline model previously presented) where the engaged gear and connection clutch state are represented as discrete states. The design problem is then formulated as the identification of the discrete state of the driveline hybrid model and an on-line identification algorithm is obtained by applying the methodology for hybrid observer design proposed in [35]. The algorithm is able to detect a change in the continuous time dynamics and identify the discrete state of the driveline hybrid model by processing the crankshaft and wheel revolution speed measurements and an estimate of the engine torque mean-value.

On-line identification is achieved by the generation of appropriate residual signals, one for each gear, which vanish when the corresponding gear is engaged and the clutch is locked.

Figure 15.5 shows the results on actual engaged gear identification obtained in Magneti Marelli Powertrain using an Opel Astra equipped with a diesel engine and a robotized gearbox SeleSpeed. For the validation of the identification algorithm, the estimated engaged gear is compared to the signal on actual engaged gear provided by the control unit of the robotized gearbox. The algorithm was tested on several maneuvers for a total of 250 gear engagements and it was able to identify correctly the actual engaged gear within 250 msec, as requested by specification. It proved to be remarkably robust with respect to parameter uncertainties (e.g. vehicle inertia) and time-varying unknown disturbances (e.g. wheel torque and road slope).

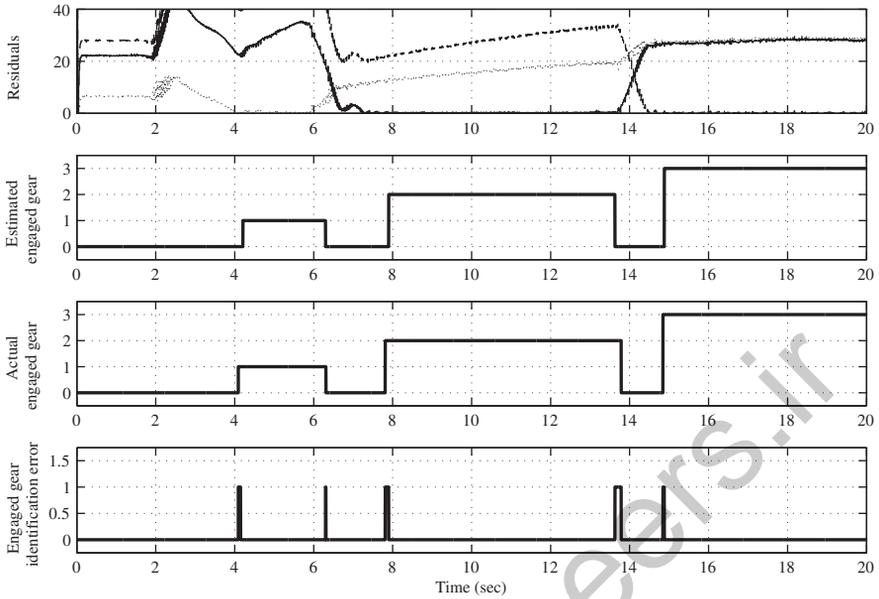


Fig. 15.5 Gear identification with experimental data. The maneuver starts with car at rest, clutch open and first gear engaged ($q = q_N = 0$). After a clutch slipping phase ($q = q_N = 0$), the clutch is locked ($q = q_1 = 1$); later, second gear ($q = q_2 = 2$) and then third gear ($q = q_3 = 3$) are engaged, passing through idle and slipping ($q = q_N = 0$).

15.3 Hybrid model-predictive control in automotive applications

15.3.1 Motivation

Automotive control problems are characterized by fast dynamics, constraints on manipulated and control variables, performance criteria, and the need for low-complexity control software. Model-predictive control (MPC) accounts for a cost function to be optimized, for constraints on system inputs and outputs. Furthermore, explicit MPC synthesis strategies convert the controller into an equivalent piecewise affine feedback law, which can be quickly evaluated by a simple control code, hence allowing the use of MPC for fast systems, and the implementation in simple electronic control units.

Hybrid model-predictive control provides the further capability of employing hybrid prediction models in MPC. Hybrid behaviors appear in automotive systems in the form of impulsive effects such as impacts [72], different operating modes [106], and discrete elements such as quantized inputs [268]. Furthermore, hybrid models with mode-dependent linear continuous dynamics provide a better approximation of nonlinear dynamics compared to linear models [137]. In the automotive domain,

hybrid model-predictive control has been successfully applied to vehicle dynamics control and to powertrain control.

15.3.2 Survey of MPC applications in vehicle dynamics control

The main contributions involve traction control, adaptive cruise control, and semi-active suspensions. In [106] the problem of traction control has been considered. Two modes for traction torque on the tire τ_t have been identified from experimental data:

$$\tau_t(\Delta\omega, \mu) = k_{i1}\Delta\omega + k_{i2}\mu + k_{i3}, \quad \text{if } h_i\Delta\omega + k_i\mu \leq \ell_i, \quad i = 1, 2,$$

where $\Delta\omega$ is the wheel slip (the difference between normalized engine and vehicle speed) and μ is the road adhesion coefficient. As a result, the vehicle dynamics is formulated as a hybrid system with two modes, that is used for prediction in a hybrid MPC controller. Experiments on a test vehicle were performed by executing an explicit MPC controller consisting of 504 regions on a 266-MHz Pentium laptop connected to the car, showing very good performance and robustness.

In the cruise control problem, the vehicle velocity is regulated to track a user defined reference. In adaptive cruise control, the controller accounts also for the interaction with other on-road vehicles, ensuring a safety distance. In [459] the adaptive cruise control is formulated as a multi-objective problem, in which one objective is to track the reference velocity, the other is to maintain a safety distance. The second objective becomes active only when another vehicle is close enough,

$$J_d(d(k)) = \begin{cases} 0 & \text{if } d(k) \geq d_{\text{safe}}, \\ w_d(d(k) - d_{\text{safe}})^2 & \text{otherwise,} \end{cases}$$

where J_d is the cost related to the distance $d(t)$ to the nearest vehicle, d_{safe} is the safety distance above where the distance cost is null and w_d is a positive weight. The hybrid character of this control problem appears in the performance criterion, and hence hybrid model-predictive control techniques are employed.

Another approach for adaptive cruise control was presented in [187] where a leader-follower problem was considered. In this case an MPC controller regulates the follower so that it keeps a reference separation distance from the leading vehicle. The vehicle dynamics have been represented by a piecewise affine model, modeling nonlinear aerodynamical friction and gear-dependent traction force. The minimum safety distance is enforced as a hard constraint:

$$x_{\text{leader}} - x_{\text{follower}} \geq d_{\text{safe}},$$

where x_{leader} is the leading car position and x_{follower} is the following car position. This application has been also used to compare the performance and the complexity of different hybrid model-predictive control strategies.

In [267] a hybrid MPC approach has been proposed for control of semi-active suspensions. As opposed to standard (passive) suspensions, active suspensions are

devices in which an active controller is used to improve the dynamical response of the suspension. In semi-active suspensions a controller is still present, but is passive, that is, it can only regulate the damping coefficient. Even though a semi-active suspension is significantly simpler and less expensive than an active suspension, the capability of modifying the parameters results in an increased performance in terms of driver's feeling and vehicle handling with respect to passive suspensions. In [267] the employed dynamical model is linear (quarter car model), and the hybrid systems framework is necessary to model the passivity constraint

$$F_{\text{cmp}}v \geq 0 \leftrightarrow \begin{cases} F_{\text{cmp}} \leq 0 & \text{if } v \leq 0, \\ F_{\text{cmp}} \geq 0 & \text{if } v > 0, \end{cases}$$

where F_{cmp} is the compression force applied by the controller to the suspension, and v is the velocity of extension of the suspension. The passivity constraint limits the resulting controller to only be able to dissipate energy, and hence the whole suspension system is semi-active.

15.3.3 Survey of MPC applications in powertrain control

In powertrain control, hybrid model-predictive control has been applied for airpath control in diesel engines, control of electronic throttles, control of magnetic actuators, and control of direct injection stratified charge engines.

In [494] the airpath of turbocharged diesel engines has been controlled by means of a switched model-predictive controller. The airpath dynamics is controlled by means of exhaust gas recirculation valves (EGR) and variable geometry turbines (VGT) and it can impressively reduce the emissions of nitrogen oxides (NO_x) and particulate matter. The approach presented in [494] is based on partitioning the engine speed and the injected fuel into intervals \mathcal{I}_s^i , $i = 1, \dots, n_s$, \mathcal{I}_f^j , $j = 1, \dots, n_f$, and to compute a linear approximation of the airpath dynamics in each of the $n_s \times n_f$ resulting regions. A linear MPC controller is designed for each one of these regions, and during the control cycle, depending on the current engine conditions, the appropriate controller is activated according to the scheme

$$u(k) = u_{\text{MPC}}^{(ij)}(x(k), r(k), u(k-1)), \quad \text{if } \text{speed} \in \mathcal{I}_s^i, \text{ fuel} \in \mathcal{I}_f^j.$$

In [644] hybrid model-predictive control is applied to an electronic throttle. The electronic throttle is one of the fundamental components in *drive-by-wire* vehicles, and it must be controlled to guarantee high tracking performance, which implies fuel consumption and emissions reduction, while ensuring safety constraints on the system components. In [644] the authors formulate a piecewise affine model of the nonlinearities in the throttle dynamics, such as friction, and the spring characteristics

$$F_{\text{sp}} = -\theta_h x, \quad \text{if } x \in \mathcal{I}_h,$$

where x is the spring displacement from the neutral position, F_{sp} is the spring force, and \mathcal{I}_h and θ_h , $h = 1, \dots, n_x$ are intervals that partition the operating range of x . The

obtained piecewise affine prediction model is used for time-optimal model-predictive control to achieve high tracking performance and safety constraints satisfaction.

In [137] the authors consider the control of magnetic actuators that are in automotive applications are used for instance in semi-active suspensions and in fuel delivery systems. Magnetic actuators are devices in which a moving cursor is displaced using the magnetic force generated by a voltage-controlled coil. The control objective is to maximize the tracking performance of a reference signal and to reduce the excitation of manipulated variables. Meanwhile, several constraints have to be enforced that avoid impacts of the cursor with the coil, and that bounds the manipulated variables within safe ranges. The magnetic actuator dynamics are nonlinear, and in [137] the authors have proposed an hybridized model exploited for hybrid model-predictive control. The hybrid MPC is shown to perform better with respect to a linear MPC implementation. This example application will be discussed in detail in Section 15.3.4. The approach was extended in [72] to handle actuators composed of multiple interacting masses, whose model represents a good example of hybrid dynamics.

Hybrid MPC has been applied to direct injection stratified charge (DISI) engines in [268]. DISI engines have two operation modes: homogeneous charge and stratified charge. Homogenous charge is the standard vehicle operation mode. The stratified charge mode is activated by late fuel injection. Stratified charge mode reduces fuel consumption, but it can be only enabled at low to medium engine speeds and loads and it must be periodically disabled to purge a lean NO_x trap. In [268] hybrid MPC is used to select the discrete operation mode concurrently with the continuous inputs. The resulting vector of manipulated variables is

$$\mathbf{u}^T = [W_{\text{th}} \ W_f \ \delta \ \rho],$$

where W_{th} is the airflow through the throttle, W_f is the fuel flow, δ is the spark timing, and ρ is the mode, where $\rho = 1$ is homogenous charge and $\rho = 0$ is stratified charge. The model-predictive controller achieves fuel consumption reduction, while ensuring torque, air-to-fuel ratio, and spark timing tracking performance and safety constraints on airflow rate, air-to-fuel ratio, and spark timing.

15.3.4 Example: Hybrid MPC of magnetic actuators

As an illustrative example of hybrid model-predictive control in automotive application we discuss the control of a magnetic actuator which can be potentially used in fuel delivery systems. The magnetically actuated mass-spring-damper system, whose schematics are shown in Fig. 15.6, is a heterogenous system composed by a mechanical subsystem and an electromagnetic subsystem that influence each other. A mass m [kg] moves linearly within a bounded region under the effect of a controlled magnetic force F [N] generated by a coil placed at one of the boundary of the region. Additional forces acting on the mass are generated by a spring and a damper. The overall equations defining the system are

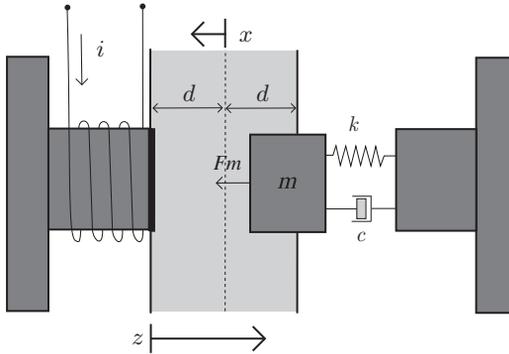


Fig. 15.6 Schematics of a magnetically actuated mass spring damper system.

$$m\ddot{x} = F - c\dot{x} - kx, \quad (15.1a)$$

$$\dot{\lambda} = V - Ri, \quad (15.1b)$$

$$\lambda = \frac{2k_a i}{k_b + z}, \quad (15.1c)$$

$$F = \frac{k_a i^2}{(z + k_b)^2} = \frac{\lambda^2}{4k_a}, \quad (15.1d)$$

$$z = d - x. \quad (15.1e)$$

Equation (15.1a) represents the dynamics of the mass position x [m] under the effect of the external force F , of a spring with stiffness k [N/m] and of a damper with coefficient c [N·s/m]. Equation (15.1b) is Faraday's law for a resistive circuit with resistance R [Ω], subject to magnetic flux variations, where the applied voltage V [V] is the control input. The relation between the magnetic flux λ [V·s] and current i [A] is defined by (15.1c), where k_a , k_b are constants, while (15.1d) defines the magnetic force either as a function of the current or as a function of the magnetic flux. Finally, (15.1e) defines the relation between position coordinates in the mechanical (x) and in the electromagnetic (z) subsystem.

The magnetically actuated mass spring damper is subject to several constraints related to physical limits and performance. The constraint

$$-d \leq x \leq d \quad [\text{m}], \quad (15.2)$$

where $d = 4 \times 10^{-3}$, prevents the moving mass from penetrating the coil or the symmetric stop at the other end and avoids undesirable bouncing with consequent wear of the parts.

The *soft-landing* constraint

$$-\varepsilon - \beta(d - x) \leq \dot{x} \leq \varepsilon + \beta(d - x) \quad [\text{m/s}], \quad (15.3)$$

where ε and β are constants avoids high velocities when the mass is positioned against the coil with ($x = d$). This reduces collision wear and prevents excessive

disturbances to the electrical current. When the mass is at the contact position ($x = d$), the velocity is constrained in $[-\varepsilon, \varepsilon]$, $\varepsilon = 0.1\text{m/s}$, a range which is progressively relaxed with rate β as the mass moves away from the coil.

The current in the circuit cannot be negative and, as a consequence of (15.1d), the magnetic force is able to only attract the mass

$$i \geq 0 \quad [\text{A}], \quad (15.4a)$$

$$F \geq 0 \quad [\text{N}]. \quad (15.4b)$$

In addition, the constraint on the voltage $0 \leq V \leq V_{\max}$ [V] is enforced to take into account the physical limits and the safety of operation of the electrical circuit.

Decoupled hybrid MPC architecture Because of the nonlinearity in (15.1) and of the constraints, nontrivial control techniques are needed to meet the requested specifications. As the dynamics of the electrical subsystem are much faster than the mechanical ones, the control problem can be decoupled by designing an inner-loop controller acting only on the electrical subsystem, and an MPC controller based on the reduced system model

$$\ddot{x} = -\frac{c}{m}\dot{x} - \frac{k}{m}x + \frac{F}{m}, \quad (15.5)$$

where the position (x) and the velocity (\dot{x}) of the mass are the state components and the magnetic force F is the controlled input, subject to constraints (15.2), (15.3), (15.4b), and

$$F \leq k_a \frac{i_{\max}^2}{(d + k_b - x)^2}. \quad (15.6)$$

Constraint (15.6) defines an upper bound on the available force, related to the maximum available current i_{\max} . The value i_{\max} is computed from (15.1b) in static conditions, given the maximum voltage V_{\max} . The set (15.6) in the (x, F) -space is the nonconvex hypograph of a convex function. The force command generated by the MPC is used as reference by the inner-loop controller.

Thus, the control system architecture operates as follows:

- the MPC controller computes the force profile r_F , based (15.5) and on the mass reference position r_x ;
- the current reference profile r_i computed from r_F is used as the reference by the inner-loop controller to regulate the electromagnetic subsystem;
- the inner-loop controller actuates the voltage V to make the current i in the electromagnetic subsystem track r_i ;
- the current i generates the actual force F .

A simple way to make the current i track the desired reference r_i given the nonlinear dynamics (15.1b), (15.1c), (15.1d) is to design a controller so that the

closed-loop current dynamics become linear first-order dynamics with a stable pole $p_i = -\beta$ and steady-state gain γ/β . For the considered actuator dynamics (15.1), an approach to obtain this is by feedback linearization. The feedback linearization controller gives good performance in the nominal case, but it is not very robust. Alternative designs may be used to increase robustness.

The overall idea behind the proposed control architecture is to exploit a nonlinear controller to obtain a linearization of the dynamics, and to use a model-predictive controller to meet specifications and performance. In the MPC prediction model we assume that the dynamics of the electrical subsystem in closed-loop with the inner-loop controller are infinitely fast, hence $F = r_F$. Thus, we will refer to F as the output of the MPC controller.

A hybrid MPC controller is needed to account for the nonconvex constraint (15.6). The nonlinear constraint is approximated by a piecewise affine function so that the prediction model is formulated as a piecewise affine system used in the hybrid MPC algorithm.

Actuator model We approximate the upper bound of (15.6) as a piecewise affine function of the position $\bar{F}(x) = r_i x + q_i$, if $x \in [\bar{x}_i, \bar{x}_{i+1})$, $i = 0 \dots \ell - 1$, where $\bar{x}_i < \bar{x}_{i+1}$. The points $\{\bar{x}_i\}_{i=1}^{\ell-1}$ are the function breakpoints and define the borders between regions in which $\bar{F}(x)$ has different affine terms. Next, we introduce $\ell - 1$ binary variables $\delta_1, \dots, \delta_{\ell-1} \in \{0, 1\}$ defined by the logical conditions

$$[\delta_i = 1] \leftrightarrow [x \leq \bar{x}_i], \quad i = 1, \dots, \ell - 1 \quad (15.7)$$

and $\ell - 1$ continuous variables $z_1, \dots, z_{\ell-1} \in \mathbb{R}$ defined by

$$z_i = \begin{cases} (r_{i-1} - r_i)x + (q_{i-1} - q_i) & \text{if } \delta_i = 1 \\ 0 & \text{otherwise} \end{cases} \quad i = 1, \dots, \ell - 2^*, \quad (15.8a)$$

$$z_{\ell-1} = \begin{cases} r_{\ell-2}x + q_{\ell-2} & \text{if } \delta_{\ell-1} = 1 \\ r_{\ell-1}x + q_{\ell-1} & \text{otherwise.} \end{cases} \quad (15.8b)$$

Then, the piecewise affine approximation is

$$\bar{F}(x) = \sum_{i=1}^{\ell-1} z_i. \quad (15.9)$$

Relations (15.7), (15.8c), (15.9) can be embedded into an MLD system, using for instance the modeling language *HYSDEL* [632], along with the logical constraints

$$[\delta_i = 1] \rightarrow [\delta_{i+1} = 1], \quad i = 1, \dots, \ell_i - 1 \quad (15.10)$$

which are included to largely simplify the complexity of the MPC optimization problem associated with the MLD model.

We consider here $\ell = 3$, and, as a consequence, two δ and two z auxiliary variables have been introduced and (15.6) is approximated as

$$u \leq z_1 + z_2. \quad (15.11)$$

Model (15.5), (15.7), (15.8c), (15.11), and the mechanical subsystem model (15.5) sampled with $T_s = 0.5$ ms can be modelled in *HYSDEL* to get the equivalent MLD model

$$\xi(k+1) = \mathbf{A}\xi(k) + \mathbf{B}_1 u(k) + \mathbf{B}_2 \delta(k) + \mathbf{B}_3 \mathbf{z}(k), \quad (15.12a)$$

$$\mathbf{y}(k) = \mathbf{C}\xi(k) + \mathbf{D}_1 u(k) + \mathbf{D}_2 \delta(k) + \mathbf{D}_3 \mathbf{z}(k), \quad (15.12b)$$

$$\mathbf{E}_2 \delta(k) + \mathbf{E}_3 \mathbf{z}(k) \leq \mathbf{E}_1 \mathbf{u}(k) + \mathbf{E}_4 \xi(k) + \mathbf{E}_5, \quad (15.12c)$$

with $\xi^T = [x \ \dot{x}]$, $\mathbf{y}^T = [x \ \beta x + \dot{x} \ -\beta x + \dot{x}]$.

Control design The second and the third output are used to enforce (15.3) as output constraints. The hybrid MPC optimization problem is formulated as

$$\min_{\mathbf{U}(k)} \|\xi(N|k) - \mathbf{r}_\xi(k)\|_2^{\mathbf{Q}_N} + \sum_{i=0}^{N-1} \|\xi(i|k) - \mathbf{r}_\xi(k)\|_2^{\mathbf{Q}_\xi} + \|u(i|k)\|_2^{\mathbf{Q}_u} \quad (15.13a)$$

$$\text{s.t. : MLD dynamics (15.2),} \quad (15.13b)$$

$$\mathbf{y}_{\min} \leq \mathbf{y}(i|k) \leq \mathbf{y}_{\max}, \quad i = 1, \dots, N, \quad (15.13c)$$

$$u_{\min} \leq u(i|k) \leq u_{\max}, \quad i = 0, \dots, N-1, \quad (15.13d)$$

$$\xi(0|k) = \xi(k), \quad (15.13e)$$

where $\|a\|_2^{\mathbf{Q}} = a^T \mathbf{Q} a$, $\mathbf{U}(k) = \{u(i|k)\}_{i=0}^{N-1}$, $\mathbf{r}_\xi^T = [r_x \ 0]$ and r_x is an externally generated reference, (15.13c) models (15.2) and (15.3), and (15.13d) models (15.4b). Here, we choose

$$\mathbf{Q}_\xi = \mathbf{Q}_N = \begin{pmatrix} 2 \cdot 10^6 & 0 \\ 0 & 0 \end{pmatrix}, \quad \mathbf{Q}_u = 10^{-7}, \quad N = 3,$$

$$\mathbf{y}_{\min} = \begin{pmatrix} -4 \cdot 10^{-3} \\ -\infty \\ -10.2 \end{pmatrix}, \quad \mathbf{y}_{\max} = \begin{pmatrix} 4 \cdot 10^{-3} \\ 10.2 \\ \infty \end{pmatrix}, \quad u_{\min} = 0, \quad u_{\max} = +\infty,$$

and the approximation of (15.6), which is embedded into the MLD model, are always enforced as hard constraints.

Results In Fig. 15.7 the behavior of the system in closed-loop with the hybrid MPC controller (15.13) is reported. The dashed line shows the behavior of the system in closed loop with a linear controller that enforces (15.6) by a cascaded saturator instead of in the MPC. The performance improvements are evident in terms of reduced overshoot and settling time.

Due to the fast sampling time and to the implementation in a simple microcontroller, the MPC controller cannot be executed in its on-line form, where an optimization problem has to be solved at every control cycle. Instead, the explicit hybrid feedback law [10] is computed. The resulting controller is automatically generated as a C-code through the *Hybrid Toolbox* [57] and has an average execution time of 0.025 ms and a worst case of 0.3 ms on a 2-GHz Pentium-M, with 1 Gb RAM. The worst case number of operations to be executed is of about 12 000 multiplications 9000 sums, and 4000 comparisons per control cycle. By an architecture simulator

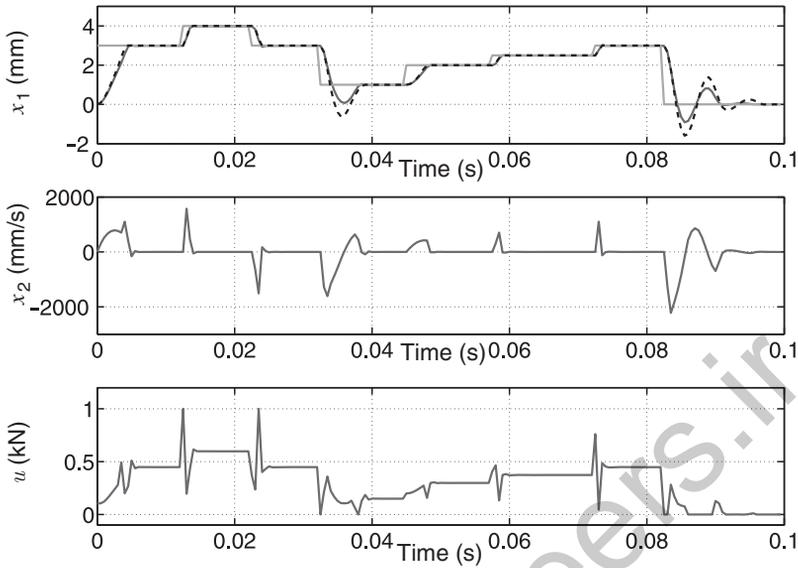


Fig. 15.7 Simulation of the system in closed-loop with the hybrid MPC. The dashed is the position trajectory obtained by a saturated linear MPC [137].

this has been evaluated to be feasible for microcontrollers commonly used in automotive operations within the prescribed sampling period.

Evaluation of the results The automotive control problems considered in this section are relatively simple (few states and control variables), yet they exhibit hybrid features and require small sampling times. Therefore, despite their apparent simplicity, they are quite challenging from a control viewpoint. Hybrid modeling based on *HYSDEL* or piecewise affine models is a quite convenient approach to come up with a compact dynamical model of the open-loop process and its operating and dynamical constraints, capturing the most relevant aspects of the problem.

Regarding the implementation, the hybrid MPC controllers are more computationally demanding than standard techniques used in automotive applications, such as feedforward (open-loop) control or PID controllers. The MPC controllers can certainly take advantage of the increased amount of computational power embedded in the electronic control units, and ad-hoc computation reduction techniques can be applied to the MPC controller. Also, the controller complexity can often be tuned by appropriate selection of the prediction model and of the prediction horizon until a desired trade-off between performance and hardware cost is found. For instance in [137] it has been shown that for different MPC controllers for magnetic actuators, the number of computations is feasible for different classes of automotive hardware.

15.4 Control of a homogeneous charge compression ignition engine

The homogeneous charge compression ignition (HCCI) engine with its excellent potential for combining low exhaust emissions with high efficiency gained substantial interest towards the end of the twentieth century [171]. The HCCI engine combines features of the traditional spark ignited (SI) Otto-cycle and compression ignited (CI) Diesel-cycle engines to a new engine concept.

15.4.1 HCCI engine concept

HCCI operation can be two-stroke or four-stroke, and the first studies [489, 493] were performed on two-stroke engines. Later studies [171, 614] on four-stroke engines show that high efficiency can be combined with low NO_x emissions for HCCI engines running with a high compression ratio and lean operation. This section will focus exclusively on the four-stroke version of the HCCI engine.

HCCI cycle The four-stroke HCCI cycle can be described by its four strokes: *intake*, *compression*, *expansion*, and *exhaust*. During the intake stroke, a more or less homogeneous mix of fuel and air is inducted into the cylinder. During the compression stroke, this charge is compressed by the upward motion of the piston. Towards the end of the compression stroke, temperature and pressure have reached levels where pre-combustion reactions start to take place. Somewhere near the TDC (top dead center), actual combustion starts (Fig. 15.8). During the initial part of the expansion stroke, the bulk of combustion takes place during the course of a few crank-angle degrees. During the rest of the expansion stroke, the high pressure caused by combustion forces the piston down towards the bottom dead center (BDC). In the exhaust stroke, the upward motion of the piston forces the exhaust gas to leave the cylinder through the exhaust valve. Hence, the coexistence of physics-based continuous-time dynamics with aperiodic four-stroke engine operation determines a clearcut hybrid system.

Ignition An HCCI engine, unlike to SI and Diesel-cycle CI engines, has no direct means for controlling ignition timing. The SI engine has spark timing, and the Diesel-cycle CI engine has the start of fuel injection, which both directly control the onset of combustion. However, for an HCCI engine, ignition timing is dictated by the conditions of the charge and the cylinder walls at the time when the intake valve closes. It can only be controlled indirectly through adjustments in the cylinder charge preparation. This is one of the biggest challenges with practical implementation of HCCI engine technology. The following paragraphs will describe the most important variables that control ignition timing for an HCCI engine.

The temperature of the air when it enters the cylinder has a large influence on the charge temperature towards the end of the compression stroke. With a compression ratio of 18:1, a change in intake temperature by 30 K will result in a change in temperature at TDC by almost 100 K. Since temperature is a very important factor

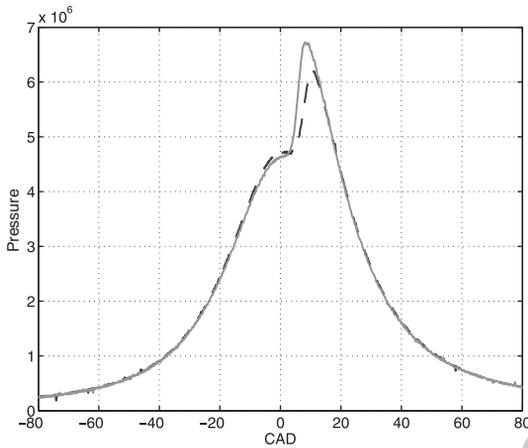


Fig. 15.8 Pressure trace from experiment at inlet temperature $T_{in} = 373$ K, fuel rate $Q_{in} = 1400$ J/cycle, for engine speed $n = 1200$ rpm (—) or $n = 1500$ rpm (---), respectively.

in auto-ignition, an increase in intake temperature will have a very strong advancing influence on ignition timing.

The portion of the exhaust gas that is not expelled during the exhaust stroke and which is called the residual gas, is particularly important for HCCI operation. The thermal energy provided by the residual gas contributes to heating the charge of the following cycle, and affects the crank angle at which ignition takes place. On an engine with variable valve timing, the residual gas fraction can be controlled by early closing of the exhaust valve, which will trap a larger amount of exhaust gas in the cylinder for the following cycle. It is necessary to remember though that exhaust gas also acts as a diluent, and thereby slows down the combustion chemistry. This will tend to retard ignition timing, and with a very high residual-gas fraction this effect will dominate.

Closely related to residual gases is the exhaust gas recirculation (EGR). This refers to exhaust gas that is routed back from the exhaust manifold to the intake manifold. Combined with an EGR cooler, this can be used for diluting the charge and thus lowering the reaction rate. An increase in EGR rate will retard ignition timing.

Another important factor is the cylinder wall temperature. Hot cylinder walls will heat the charge throughout the intake and compression strokes, and will advance ignition timing.

The fuel–air equivalence ratio affects both fuel concentration and oxygen concentration. However, since HCCI engines operate lean, the equivalence ratio has a stronger influence on fuel concentration than on oxygen concentration. The dominating effect of increasing the equivalence ratio, thus, is an increase in fuel concentration,

which will result in a higher reaction rate. Thus, increasing the equivalence ratio serves to advance ignition timing.

Another possible way to control ignition timing is by changing the fuel composition. Addition of a second fuel with higher reactivity will serve to advance ignition timing. Examples are the addition of hydrogen to natural gas and *n*-heptane to iso-octane.

A variable compression ratio provides an effective means of controlling the temperature towards the end of the compression stroke. A higher compression ratio increases the charge temperature near the TDC, and tends to advance ignition timing.

Charge stratification, which means inhomogeneous charge distribution can be used to locally increase the equivalence ratio, and thus the reaction rate, in order to advance the ignition timing. Charge stratification can be achieved through late fuel injection. As a drawback locally high temperatures, result which cause an increase in NO_x production.

Evidently, there are many variables that affect ignition timing, but they all do so in non-trivial ways. Furthermore, many of the variables affect each other as well. Actually, some variables are affected by ignition timing itself, for example the cylinder wall temperature increases with advanced ignition timing. When ignition timing is advanced, the peak cylinder temperature increases which, in turn, causes an increase in cylinder wall temperature. It follows that ignition timing is very sensitive to operating conditions.

15.4.2 Control of ignition timing

It is evident from above that ignition control is much more challenging for an HCCI engine than for an SI or Diesel-cycle CI engine. The most readily available means of controlling ignition timing is by adjusting the fuel composition. This does not require any novel mechanical design like variable valve timing or variable compression ratio. It merely requires a doubling of the port fuel injection system.

Selection of the feedback structure The sensitivity of ignition timing to operating conditions does not allow an open-loop solution in the form of a look-up table. Furthermore, the system becomes unstable for some operating conditions at high load. Thus, closed-loop control is an absolute necessity, which poses the question of which variables should be fed back.

Cylinder pressure is the natural choice of the control variable, since ignition is an in-cylinder phenomenon (Fig. 15.8). What characteristic of the cylinder pressure trace reflects when combustion takes place, though?

The crank angle of maximum pressure gives some information about when the bulk of combustion is taking place, but for combustion timing before or near the TDC, this angle tends to gravitate towards TDC due to its dependence on volume in the ideal gas law. Furthermore, for very late combustion timing, the pressure maximum from compression dominates the one from combustion. Another problem is that the maximum is flat, which makes the detection of combustion ambiguous.

The crank angle of maximum pressure derivative suffers from the same problems as the crank angle of maximum pressure in addition to the inherent noise problems with numerical differentiation. Another possibility is to search for the inflection point, where the pressure trace transitions are from negative to a positive second derivative due to the onset of combustion. This also suffers from the problems with numerical differentiation.

It turns out that a first-principles analysis based on the pressure measurements and heat release analysis provides a very robust source of feedback. Heat release analysis applies the first law of thermodynamics to the combustion chamber during the entire combustion event in order to estimate the rate at which chemical energy is converted to thermal energy. If no adjustments are made for heat transfer or flow into and out of crevices, the net heat release is obtained. Integration with respect to the crank angle yields the cumulative heat release, which roughly reflects the mass fraction burned.

Combustion in an HCCI engine is usually very fast. The mass fraction burned usually goes from 10% to 90% in about 5 crank angle degrees, which means that the crank angle of 50% heat release, CA50, provides a very accurate measure of when combustion is taking place. In the following, CA50, combustion timing, and ignition timing will be used interchangeably to denote the crank angle of 50% heat release.

15.4.3 Properties and model of the HCCI engine

Two fuel injectors and one cylinder pressure sensor per cylinder allows separate control loops for each cylinder. Thus, the control structure indicated in Fig. 15.9 can be used for the combustion timing control of each cylinder. Fuel octane number is a measure of a fuel's resistance to auto-ignition, and can be used as the control input for combustion timing control of an HCCI engine cylinder. When using a mixture of iso-octane and *n*-heptane, the octane number is, by definition, the percentage of iso-octane.

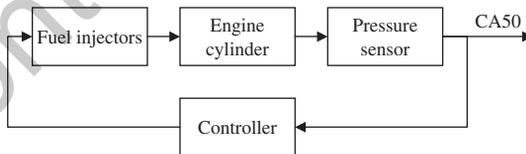


Fig. 15.9 Control structure for combustion timing control. The CA50 based on cylinder pressure measurements provides feedback about combustion timing, and octane rating provides a control input.

Processing cylinder pressure measurements Cylinder pressure measurements are normally performed with either piezoelectric elements combined with charge

amplifiers or with fiber-optical sensors. Both methods fail to measure the DC component of the cylinder pressure. A thermodynamically based method of estimating the DC component is detailed in [637], and amounts to estimating an initial pressure and a measurement offset based on pressure measurements during the compression stroke. It is essential to select the crank-angle interval for estimation between the intake-valve closing and the start of combustion for the thermodynamical assumptions to hold.

The cylinder pressure measurements p_m can be decomposed into the actual pressure p and a sensor offset Δp :

$$p_m = p + \Delta p. \tag{15.14}$$

The real pressure can be modeled with polytropic compression

$$p = CV^{-\kappa}, \tag{15.15}$$

where V is the combustion chamber volume κ is the polytropic exponent, and C depends on the initial pressure according to:

$$C = p_0 V_0^\kappa. \tag{15.16}$$

In [637], a method of estimating the polytropic exponent is also provided. In cases where the polytropic exponent is thought to vary significantly from cycle to cycle, this method can be used. Intake temperature and fuel composition as well as the equivalence ratio affect the polytropic exponent.

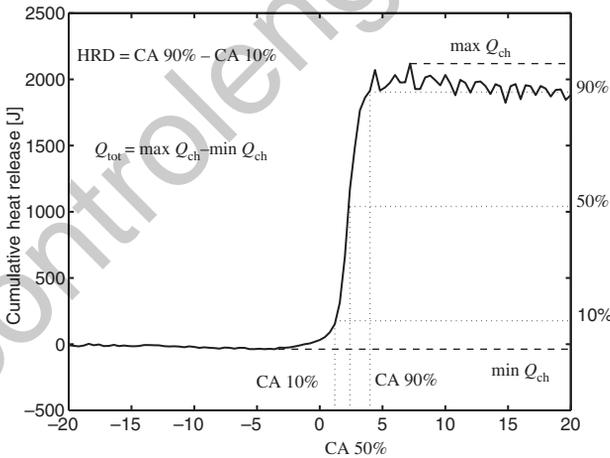


Fig. 15.10 Definitions of some heat-release based cycle parameters.

Heat release analysis An analysis of the combustion chamber, based on the first law of thermodynamics, relates the rate at which chemical energy is converted to

thermal energy to the pressure in the combustion chamber. This type of analysis is conventionally called *heat release analysis*, and can be used to determine when combustion is taking place. This term stems from the simplification that is normally done, in which the charge composition is assumed to be constant and the increase in internal energy is interpreted as heat. If the actual heat transfer to the cylinder walls as well as crevice flow is neglected, (15.17) relates heat release to cylinder pressure:

$$\delta Q_{\text{ch}} = \frac{c_v}{R} V dp + \frac{c_p}{R} p dV. \quad (15.17)$$

This equation is integrated over a crank angle interval which includes the whole combustion event. The parameters c_v and c_p are the specific heats at constant volume and pressure, respectively, and technically depend on temperature. However, if the only objective is to determine combustion timing, they can be assumed to be constant. R is the universal gas constant.

The result of the integration of the heat release equation is the cumulative net heat release as a function of crank angle $Q_{\text{ch}}(\alpha)$. A typical heat release trace is plotted in Fig. 15.10 together with some definitions. The most important definition in this context is CA50, the crank angle of 50% heat release. Since combustion is very fast, CA50 can be used as a robust source of feedback for combustion timing.

HCCI engine The sensitivity of CA50 to changes in fuel octane number varies by orders of magnitude for different operating points (Fig. 15.11). Each line in the plot represents a specific intake temperature and load. Within each line, the fuel octane number has been varied to achieve an interval of combustion timings. The strong nonlinearity of the plant makes a linear controller unsuitable for the task. The situation can be remedied, however, if the sensitivity is mapped over the multi-dimensional space of operating conditions. This map can be used for gain-scheduling the otherwise linear controller. In [492], a multivariable function is fitted to measure-

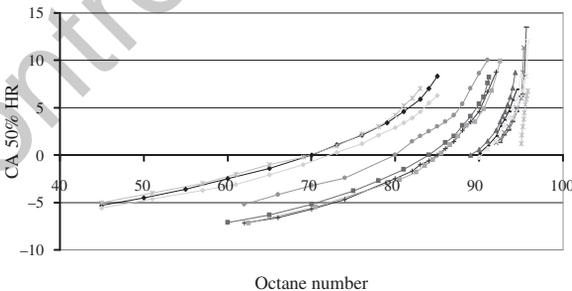


Fig. 15.11 Combustion timing versus fuel octane number for various operating points. Measurements on a Scania D12 six-cylinder engine converted for HCCI operation. Octane number varied through fueling with a variable mixture of iso-octane and n -heptane.

ments of the sensitivity of CA50 to changes in octane number for a multitude of operating conditions. In order to get a simple, computationally inexpensive model, the sensitivity is modeled as a product of functions of one variable each. This approach is entirely empirical, but yields a sensitivity model with acceptable residuals (within 3%). The variables that are included in this model are engine speed, inlet air temperature, fuel octane number, fuel mass per cycle, and CA50. A later model revision includes inlet pressure as well.

Figure 15.12 shows the octane number component of the sensitivity function. The sensitivity model is used for scaling the controller gains, which implicitly assumes that the dynamical behavior of the plant is independent of the operating point. Only the DC gain of the plant changes.

A minimum requirement of physical modeling is the explanation of the nature of the in-cylinder pressure traces (Fig. 15.8) where adiabatic compression combines with fuel-dependent auto-ignition [83]. In previous work, modeling choices involved aspects of chemical kinetics, cycle-to-cycle coupling, in-cylinder concentrations of reactants, wall temperature dynamics, pressure dynamics, and auto-ignition timing. Modeling details fell into categories of single-zone model, multi-zone models, multi-dimensional computational fluid dynamics (CFD) models, sometimes combined with exosystem simulation on the form of stochastic disturbances, load modeling, sensor modeling [74].

For each of the approaches investigated, the modeling was based on an open system first law analysis, with steady state compressible flow relations used to model the mass flow through the intake and exhaust valves. The model includes nine states: the temperature T , the concentrations of propane [C_3H_8], oxygen [O_2], nitrogen [N_2], carbon dioxide [CO_2], water [H_2O], the mass in the exhaust manifold m_e , the internal energy of the product gases in the exhaust u_e and carbon monoxide [CO], the crank angle θ , the cylinder volume V , and for HCCI, an integrated Arrhenius rate to capture ignition timing [74]. Moreover, Shaver *et al.* [581] singled out six distinct stages in modeling of HCCI engine operation: induction, compression, combustion, expansion, exhaust and residence in the exhaust manifold. As both physical aspects and operational aspects require attention, hybrid modeling turns out to be instrumental for HCCI engine control [74].

Open-loop stability An interesting phenomenon that appears in some regions of the operating space of an HCCI engine is open-loop instability. This phenomenon results when wall temperature effects dominate the ignition dynamics. Figure 15.13 shows the open-loop behavior at a stable and an unstable operating point, respectively. All control inputs are held constant in both cases.

The cause of instability is the positive thermal feedback provided through the interaction between ignition timing and cylinder wall temperature. A small increase in cylinder wall temperature results in a hotter cylinder charge, which advances ignition timing. Advanced ignition timing, however, results in higher gas temperature and more heat transfer to the walls, thus higher wall temperature. The reversed case is a small drop in cylinder wall temperature, which results in cooler cylinder charge. Ignition timing is retarded, which results in lower gas temperature, which in turn

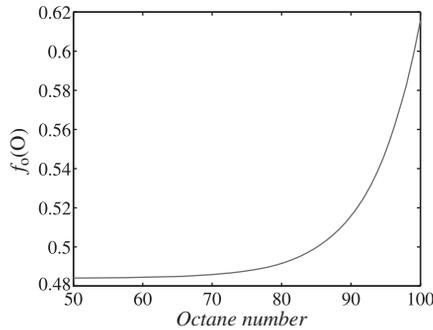


Fig. 15.12 The octane number component of the sensitivity function.

reduces the heat transfer to the walls, and thus the wall temperature. It is evident that operating points where this effect dominates are unstable.

The positive feedback mentioned above is always present, but not all operating points are unstable (Fig. 15.13). The stabilizing negative feedback is closely related to the destabilizing positive feedback. Early ignition leads to high peak temperature and heat transfer, but this results in lower gas temperature towards the end of the cycle, which means both colder residual gas and a larger amount of it. This reduces the reactivity of the charge for the next cycle, and retards ignition timing. The opposite holds for late ignition. Thus, the residual gas provides the stabilizing negative feedback.

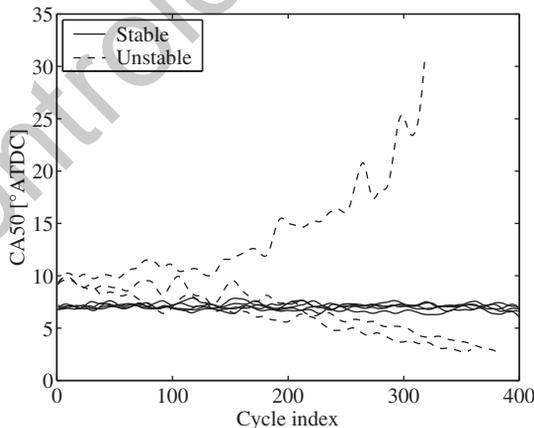


Fig. 15.13 Repeated open-loop operation at one *stable* and one *unstable* operating point.

Bibliographical notes

As for stable operation, combustion phasing control design requires appropriate models and system output variables usable for feedback control. Recently, mode-transition operation and control of Diesel-HCCI and SI-HCCI engines and other hybrid control aspects have received attention [582].

In order to avoid difficult conditions of HCCI operation at high loads, mode transition control is relevant and, thus, hybrid modeling for switching between SI and HCCI (or Diesel-HCCI) combustion mode [582].

Recently, successful model-based HCCI engine control using conservation principles in hybrid modeling was reported [85]. Hence, hybrid modeling of HCCI engines suitable for control design is a key issue for progress in this area.

controlengineers.ir

controlengineers.ir

16

Networked control

M. D. Di Benedetto, A. Bicchi, A. D’Innocenzo, K. H. Johansson, A. Robertsson, F. Santucci, U. Tiberi, and A. Tzes

Control loops which are closed over a digital communication network became a topic of intensive research in the recent years. This chapter surveys the main problems to be solved and show how hybrid systems theory can help to solve them. The chapter ends with a case study that was inspired by a practical application of hybrid systems methods in an ore mine.

Chapter contents

16.1	Introduction to distributed control applications and networked control systems	page 472
16.2	Algorithm and methods in distributed processing	474
16.2.1	Coordination of multi-agents system	474
16.2.2	Consensus problems for teams of mobile agents	480
16.2.3	Distributed Kalman filtering using weighted averaging	484
16.3	Adaptive resource management and networking for control applications	487
16.3.1	Networked control systems: hybrid modeling issues	487
16.3.2	Controller synthesis	489
16.3.3	Modeling of adaptive transmission behaviors in control over wireless	490
16.3.4	Safety problem with wireless networks and the hybrid model	491
16.4	A case study on networked control: the mining ventilation process	495
16.4.1	Description of the mining ventilation process	495
16.4.2	Today’s control architecture	496
16.4.3	Proposed wireless control architecture	497
16.5	Conclusions and open problems	499

16.1 Introduction to distributed control applications and networked control systems

In this chapter, we deal with control over networks, i.e. with control implementations where the control actions and decisions are taken based on measurement, decisions, and actuations that take place in a distributed environment. The control agents may rely upon a centralized facility that coordinates and optimizes the overall control strategy and on a shared communication resource like a bus (*distributed control*) or may be acting only on local information and on data exchanged with neighboring nodes (*decentralized control*). There are obviously pros and cons for each strategy. Decentralized control systems have the following characteristics [283]:

- There is no central control node.
- There is no common communication facility; communication is point-to-point.
- The global network topology is unknown to the nodes, which are only aware of their neighborhood.

These features yield interesting properties:

- The system is scalable: there are no limits imposed by centralized computing power or global communication bandwidth.
- The system is robust and fault tolerant, because it supports dynamical changes of the network topology and losses of nodes.
- The system is easy to program and operate, because modular techniques can be used.

An example of decentralized control is *cooperative control*, where several agents collaborate to solve a problem. Cooperative control is of particular interest in industrial control and in defense applications where a single agent is clearly unable to deliver the performance that is typical of these environments.

There are interesting challenges to overcome in distributed and decentralized systems, called networked systems, which are not typically seen in traditional control [116, 400, 624, 654, 681, 682]. Phenomena such as packet loss, quantization noise and variable delays must be taken into consideration since the operation of the system depends critically on the communication properties [454]. These phenomena are always present but appear with more determining effect when communication is entirely or in part wirelessly.

Networked control systems characterized by fixed delays can be analyzed using ordinary sampled-data control theory, since they can be considered as linear time-invariant systems. For variable delays, different methods of analysis have been proposed. They can be cast into a stochastic framework, where variable delay is treated as a random variable with known probability distribution [488]. Alternatively, we can use a deterministic framework where lower and upper bounds for the delay are considered [454]. In the former case, stochastic control techniques are relevant, while in the latter, robust control offers a potential solution [330, 488]. Synthesis of systems with time varying delays has not been fully explored although considered to some extent in the literature.

A problem of considerable importance in networked systems is optimal resource utilization. In fact, besides being imperfect, communication and computation resources may be scarce in terms of both communication bandwidth and execution time. In this case, control techniques can be used to determine the optimal use of the resources.

There are well-known static techniques to access communication bandwidth, such as TDMA and FDMA. These schemes require schedulability analysis that relies on worst-case behavior that may be unknown or overly conservative. A more effective technique may be to allocate communication resources dynamically using control techniques that can be used to guarantee quality of service (QoS) constraints that involve consumed power and response time [500]

The importance of considering jointly communication and control system design was addressed in seminal papers by Goldsmith [409, 410] for the case of sensors networks: communication parameters need to be properly chosen to optimize the performance of control systems with feedback loops over wireless communication links. In addition, these papers stressed that the overall performance of a control system degrades with the introduction of MAC protocols that introduce additional delays over the ones due to the physical aspects of communications faults (e.g., random delays, packet loss, etc.). Further, these papers showed that optimization of throughput and packet loss requires cross layer design. The need of cross layer design is due to the dependence of different communication parameters influencing control system performance on the link, the MAC layer protocol and routing algorithms [409]. In addition, network traffic has to be considered when solving the design problem.

In Section 16.1 we discuss bounds and limitation among multi-agent processing. In Section 16.2, we discuss a coordination of mobile multi-agent problem, the consensus problem and distributed Kalman filtering estimation. Section 16.3 concerns adaptive resource management over a networked control system. Finally, in Section 16.4 we propose a case study on networked control system, namely a mining ventilation problem. As a connection to Chapter 3, we will show that distributed and networked systems often require hybrid models (Section 16.2.1 and 16.3.3), since they include continuous and discrete dynamics. For this reason, methodological results for the analysis of properties like reachability, safety, deadlock, liveness etc. are of direct application to distributed and networked control problems.

Performance bounds and limitations in multi-agent processing When involved in the study of consensus and coordination problems, and in the study of identification problems for agents evolving in networks, both questions can be phrased as questions concerning long products of matrices.

In the consensus and coordination problems, agents with identical dynamics are distributed in the plane or in space, they exchange limited information and they aim to achieve consensus or coordination. Let X_i represent the information state of agent i . For example, the information state can be the position or velocity of the agent, the state of the local controller, etc. The consensus problem is that of choosing a control law for every agent such that the information state X_i of all the agents converge to the

same value. The problem we consider here is to determine how the communication influences control performance, namely the rate of convergence to consensus.

The related identification problem for agents evolving in networks is partly motivated by practical questions for sensor networks. We consider now an agent that moves on an edge-colored directed graph and we attempt to determine the position of the agent in the graph after a sufficient long observation of the traversed edge colors. Graphs for which such an identification can always be performed are said to be *cruisable*. In a *cruisable* graph an agent is able to determine his position in the graph for all observations of length larger than some T . Some graphs are not *cruisable* in this way but in a weaker sense when, even though the agent is not able to determine his position at all time, he is able to do so infinitely often. We have exactly characterized both *cruisability* properties and shown that they can be checked in time polynomial in the dimension of the problem. The design question of assigning colors to a graph so as to make it *cruisable* remains an unsolved challenging question.

In mathematical terms both the consensus and the identification problems can be formulated as questions on products of matrices taken from a given finite set. These sets of matrices either represents different communication topologies, or different color patterns in the graph. In future research we will further investigate how these problems connect to each other and how general properties on long products of matrices can be exploited for these particular applications.

16.2 Algorithm and methods in distributed processing

16.2.1 Coordination of multi-agents system

In this section, hybrid control methodologies for distributed coordination and processing in multi-agent systems are considered. The development of new flexible manufacturing and control techniques for future industrial automation and robotics is the main driving factor of this study, though the research problem is quite generic and relevant in many other cases of distributed control and networked synchronization. The considered scenario is rather general and comprises a set of autonomous agents that are interconnected through a wireless network, and have to cooperate in order to accomplish either a common task or specific missions with different objectives. Furthermore, it is required that no centralized decision making authority is exploited, thus each agent must decide on its own motion only based on locally available information, such as its own state and goal, and limited data exchanged via the finite bandwidth wireless channel with neighbouring agents. Moreover, agents are assumed to be collaborative, though non-cooperation among them (e.g. due to malfunction or tampering) will be considered. Furthermore, the communication topology is either given by a fixed graph or varying over-time depending on agents' motion.

One of the main aspects in controlling multi-agent systems is the problem of the design, verification and implementation of decentralized algorithms for conflict management of networked multi-agent mobile systems. Decentralized approaches require that each vehicle plans its own trajectory based only on information limited

to neighbouring vehicles. A completely decentralized and safe control strategy has been developed for a very large and dynamically changing number of autonomous vehicles moving in a shared environment. In the considered framework, each vehicle is assumed to have a different task to accomplish in terms of reaching a target configuration. However, since the agents act only on local information, global properties of a decentralized control policy are often hard to establish. In particular the two fundamental properties of decentralized control strategies are liveness and safety for which a formal definition has been provided and described in the following. Furthermore, conditions under which both properties are satisfied have been provided. Unfortunately, formal verification that such conditions are sufficient to ensure liveness appears to be overwhelmingly complex. We therefore assessed the correctness of the conjecture in probability through the analysis of the results of a large number of randomized experiments. Only main results are reported in this section, whilst further details can be found in [249] and [497].

Problem formulation Let us consider n mobile agents moving on the plane at constant speed, along paths with bounded curvature. Let the configuration of the i -th agent be specified by $g_i \in SE(2)$, the group of rigid body transformation on the plane. In coordinates, the configuration of the i -th agent is given by the triple $g_i = (x_i, y_i, \theta_i)$, where x_i and y_i specify the coordinates of a reference point on the agent's body with respect to an orthogonal fixed reference frame, and the heading θ_i is the angle formed by a longitudinal axis on the agent's body with the $y = 0$ axis.

Each agent enters the environment at the initial configuration $g_i(0) = g_{0,i} \in SE(2)$, and it is assigned a target configuration $g_{f,i} \in SE(2)$. The agents move along a continuous path $g_i : \mathbb{R} \rightarrow SE(2)$ according to the model

$$\begin{aligned}
 \dot{x}_i(k) &= v_i \cos(\theta_i(k)), \\
 \dot{y}_i(k) &= v_i \sin(\theta_i(k)), \\
 \dot{\theta}_i(k) &= \omega_i(k),
 \end{aligned} \tag{16.1}$$

where $\omega_i : \mathbb{R} \rightarrow [-\frac{1}{R_C}, \frac{1}{R_C}]$ is a bounded signed curvature control signal. Without loss of generality, we can scale the control ω_i down to range $[-1, 1]$ by considering $R_C = 1$, and assume the linear velocity v_i to be constant and equal to 1 for each agent.

A *collision* between two agents occurs at time t_c , if the agents become closer than a specified safety distance d_s . Hence, associating to each agent a *safety disc* of radius $R_S = d_s/2$ centered at the agent position, a collision occurs whenever two safety discs overlap.

A control policy π is said *spatially decentralized* if it can be written as a function of the configurations of the agents that are nearer than a given alert distance d_a from the computing agent.

Below, we describe the proposed spatially decentralized cooperative policy for collision avoidance that is referred to as *generalized roundabout policy* and has been introduced in [249].

Reserved disc The proposed policy is based on the concept of *reserved disc*, over which each active agent claims exclusive ownership. Given the agent configuration g , the associated reserved disc has radius $R = 1 + R_S$, is centered in $(x^c, y^c) = (x + \sin(\theta), y - \cos(\theta))$, and inherits the agent’s heading θ (see Fig. 16.1). The configuration $g^c = (x^c, y^c, \theta^c)$ of the reserved disc has the dynamics $\dot{g}^c = ((1 + \omega) \cos \theta^c, (1 + \omega) \sin \theta^c, \omega)$. Notice that when the agent has control $\omega_i = -1$, corresponding to a maximum curvature radius clockwise turn, the center of the associated reserved disc is fixed, see Fig. 16.1. Hence the reserved disc can be stopped at any time, by setting $\omega = -1$ and it can be moved in any direction, provided one waits long enough for the heading θ to reach the appropriate value.

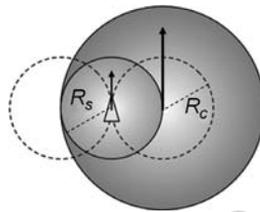


Fig. 16.1 The reserved disc of a nonholonomic vehicle with bounded angular velocity.

Constraints A sufficient condition that ensures safety is that the interiors of reserved disks are disjoint at all times since they always contain the agent’s safety discs. If the reserved disk of agent i is in contact with the reserved disks of agents with indices in $J_i \subset \{1, \dots, n\}$, the motion of the agents is constrained as follows

$$\dot{x}_i^c(x_i^c - x_j^c) + \dot{y}_i^c(y_i^c - y_j^c) \geq 0, \quad \forall j \in J_i. \quad (16.2)$$

In other words, the velocity of the i -th reserved disk is constrained to remain in the convex cone Θ , namely *admissible cone*, determined by the intersection of a number of closed half-planes (16.2). Θ can be computed assuming that each agent is aware of the configuration of all agents within an alert distance $d_a = 4 + d_s$. Hence, the amount of information needed by each agent to compute Θ is bounded and independent from the number of agents in the system: in fact, at each instant, the maximum number of agents with distance less than d_a from the considered agent is six.

Holding As previously mentioned, setting $\omega = -1$ causes an immediate stop of an agent’s reserved disk’s motion. We will say that when $\omega = -1$, the agent is in the *hold* state.

Moving in a free space In an obstacle-free environment, an agent can accomplish the task of reaching an assigned final configuration g_f , starting from g_0 , switching between the *hold* state and the *straight* state associated to the control $\omega = 0$ of

the agent. The switching policy can be summarized as follows. Let Δ_f be the vector from the center of the reserved disc g_c to the center g_{cf} of the reserved disc associated with the final configuration. Furthermore, let $\phi : \mathbb{R}^2 \setminus 0 \rightarrow S^1$ be a function returning the polar angle of a vector. Whenever the heading θ of the agent is equal to $\phi(\Delta_f)$ the agent switches its control to $\omega = 0$ and moves straight toward the final configuration until $g_c \equiv g_{cf}$. At this point the agent switches its control to $\omega = -1$ until the target configuration is reached.

Avoiding collision As already mentioned in the properties of the reserved region motions, by switching in the `hold` mode, the reserved region stops. Hence, each agent can switch to this mode whenever its heading does not belong to the admissible cone generated by possible contacts between reserved discs, i.e. $\theta \notin \Theta^-$.

Stationary obstacle If the path of the reserved disk to its position at the target is blocked by another reserved disk, a possible course of action is represented by rolling in a pre-specified direction (in our case, the *positive* direction) on the boundary of the blocking disk. In order to roll on such disk, without violating safety constraints, the control input must be set to $\omega = (1 + R_S)^{-1}$ as soon as the heading of the agent is equal to the value of the counterclockwise direction boundary of the admissible cone, namely $\theta = \max(\Theta^-)$, as illustrated in Fig. 16.2. We refer to this mode as the `roll` state. While the above condition on θ is not true the agent remains in the `hold` state (i.e. $\omega = 0$).

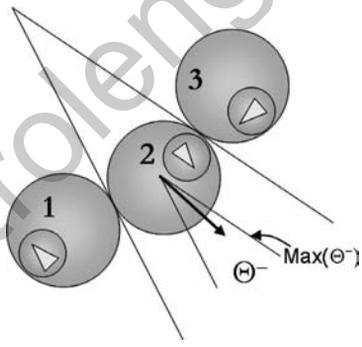


Fig. 16.2 The set of allowable directions in which the center of the i -th reserved disk can move generated by the contact with reserved discs of vehicles j , m , and k respectively.

Moving obstacle In general, the reserved disk of an agent will not necessarily remain stationary while an agent is rolling on it. While it can be recognized that the interiors of the reserved disks of two or more agents executing the described maneuver will always remain disjoint, it is possible that contact between two agents is lost unexpectedly. In this case, we introduce a new state, which we call `roll2`, in which the agents turns in the positive direction at the maximum rate, i.e., $\omega = +1$,

unless this violates the constraints. The rationale for such a behavior is to attempt to recover contact with the former neighbor, and to exploit the maximum turn rate when possible. The `roll2` state can only be entered if the previous state was `roll`. The agent is forced to exit from the `roll2` state after at most time 2π .

Generalized roundabout policy: We are now ready to state our policy for cooperative, decentralized, conflict resolution; we call it generalized roundabout (GR) policy. The policy followed by each vehicle is based on four distinct *modes of operation*, each assigning a constant value to the control input ω . As a consequence, the closed-loop behavior of an individual agent can be modeled as a hybrid system.

The states of the hybrid systems are 4 and correspond to constant inputs $\omega_{\text{roll}} = (1 + R_S)^{-1}$, $\omega_{\text{roll2}} = +1$, $\omega_{\text{hold}} = -1$, and $\omega_{\text{straight}} = 0$, respectively.

The map Φ_{GR} that describes the agents’ dynamic in each node of the system, is derived from (16.1), substituting the appropriate value for ω , based on the discrete mode, and by the clock rate $\dot{\tau} = 1$ (needed only in the `roll2` state).

We do not explicitly write down the GR policy and its transition relations, guards, and invariants, but we refer the reader to Fig. 16.3, which should provide the necessary detail in a clearer fashion.

The multiple-vehicle system (\mathcal{S}_{GR}) we are considering is the parallel composition of n agents of the hybrid system described above. We do not define the operation of parallel composition here; see, e.g., [429] for details.

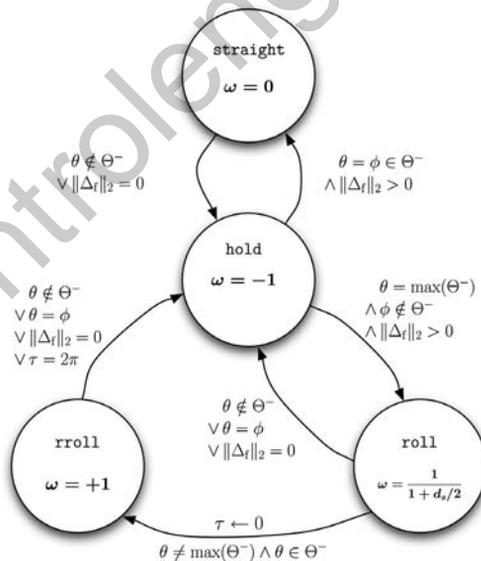


Fig. 16.3 A hybrid automaton describing the Generalized Roundabout policy.

Analysis of the policy The policy described above has provided effective solutions for large-scale problems, such as, e.g., the 70-agents conflict resolution illustrated in Fig. 16.4.

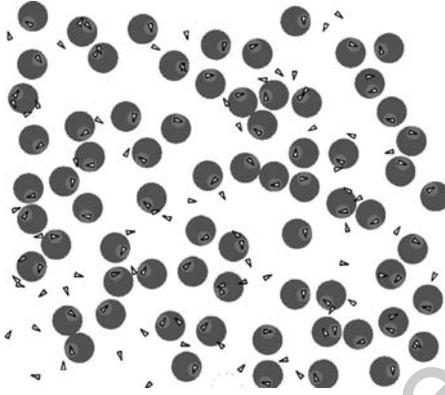


Fig. 16.4 A conflict resolution problem with 70 agents in narrow space, for which the proposed policy provides a correct solution. Initial configurations are identified by the presence of gray circles, indicating their reserved discs.

In the following, we investigate methods to systematically assess conditions under which the policy is applicable and provides solutions which are guaranteed to be collision-free (i.e. *safe* and to ultimately lead all agents to their goals avoiding stalls (i.e. *nonblocking* or *live*).

For a given policy π , the following definitions are used:

- an initial configuration set is *safe* for the policy π if there exists a set of target configurations such that application of π leads to a collision;
- a target configuration set is *nonblocking* for the policy π if there exists a set of initial configurations from which the application of π allows each agent to reach the final configuration.

The following propositions have been proved:

Proposition 16.1 *If the initial configuration set is safe the hybrid system (\mathcal{S}_{GR}) is well defined.*

Proposition 16.2 *An initial configuration set is safe if the associated reserved disks are pair-wise disjoint.*

The analysis of the liveness (or nonblocking) property is more complex, and hinges upon the definition of a condition concerning the separation of reserved discs associated with target configurations. Let G_f^c denote the set of configurations of the reserved discs corresponding to G_f , and P_f^c be the set of their center coordinates.

Sparsity condition For all $(x, y) \in \mathbb{R}^2$ and for $m = 2, \dots, n$,

$$\text{card}\{(x_{f,i}^c, y_{f,i}^c) \in P_f^c : \|(x_{f,i}^c, y_{f,i}^c) - (x, y)\|_2 < \rho(m)\} < m, \quad (16.3)$$

where

$$\rho(m) = \begin{cases} 2(1 + R_S) & \text{for } m \leq 4, \\ (1 + \cot(\frac{\pi}{m}))(1 + R_S) & \text{for } m \geq 4. \end{cases} \quad (16.4)$$

In other words, any circle of radius $\rho(m)$, with $1 < m \leq n$, can contain at most $m - 1$ reserved disk centers of targets.

Proposition 16.3 *If a target configuration set G_f violates the sparsity condition, then G_f is blocking.*

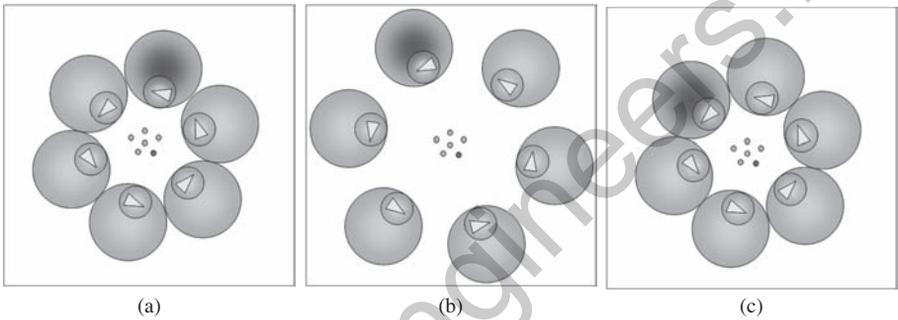


Fig. 16.5 Livelock-generating conditions for the GR policy with $\hat{m} = 6$.

An example of livelock-generating conditions is illustrated in Fig. 16.5.

In conclusion, we have proved that the sparsity of target configurations is a necessary condition that rules out the possibility of blocking executions of the GR policy. Since a proof of sufficiency appears to be very complex, we have adopted a probabilistic approach, based on [620] that exploits a large number of exact simulations. Refer to [497] and [498] for further detail.

16.2.2 Consensus problems for teams of mobile agents

In this section, we summarize the results of a case study on multi-agent coordination, with particular reference to cooperative and distributed estimation. The system model includes a switching communication topology among the agents, and time-varying measures. We describe a consensus algorithm, which enables the design of decentralized optimal solutions. In particular, the consensus of the estimates is obtained while minimizing the variance of the estimation error in each agent. The proposed algorithm is distributed and each agent computes the estimates without any central coordination.

Problem formulation Consider a system with N mobile agents communicating through wireless sensor nodes. Suppose each node samples a common time-varying scalar signal $d(k)$:

$$u_i(k) = d(k) + v_i(k), \quad k = 0, 1, \dots, \quad i = 1, \dots, N,$$

where $v_i(k)$ is zero-mean white noise. We collect measurements and noise variables in vectors, $\mathbf{u}(k) = [u_1(k), \dots, u_N(k)]^T$ and $\mathbf{v}(k) = [v_1(k), \dots, v_N(k)]^T$, so that

$$\mathbf{u}(k) = \mathbf{d}(k)\mathbf{1} + \mathbf{v}(k), \quad k = 0, 1, \dots,$$

where $\mathbf{1} = (1, \dots, 1)^T$. The covariance matrix of $\mathbf{v}(k)$ is supposed to be diagonal $\Sigma = \sigma^2 I$. Note that the additive noise can be rapidly averaged out only if nodes communicate measurements and estimates. Indeed, reducing the sampling time, and averaging on the larger number of samples, would not be beneficial, since the measurements may exhibit a correlated noise. This is the case, for example, of received signal strength measures [290].

Let us now introduce a model for the wireless communication network. For the time-varying undirected graph

$$\mathcal{G}(k) = (\mathcal{V}, \mathcal{E}(k)), \quad k = 0, 1, \dots,$$

the vertices $\mathcal{V} = \{1, \dots, N\}$ represent the fixed set of network nodes and $\mathcal{E}(k) \subset \mathcal{V} \times \mathcal{V}$ the edges present at time k . The packet losses of the wireless channels are thus modeled through an on/off binary random variable, having average p , indicating whether there is a successful communication between two nodes or not at sampling time k . Let $\mathcal{G}^S = (\mathcal{V}, \mathcal{E}^S)$ denote the graph sum of $\mathcal{G}(k)$, $k = 0, 1, \dots$. Specifically, we denote \mathcal{G}^S the nominal graph and suppose that it is connected. The neighbors of node $i \in \mathcal{V}$ at time k is denoted

$$\mathcal{N}_i(k) = \{j \in \mathcal{V} : (j, i) \in \mathcal{E}(k)\},$$

and the corresponding notation for \mathcal{G}^S is \mathcal{N}_i^S . For all $k = 0, 1, \dots$ and $i = 1, \dots, N$, we suppose that $(i, i) \in \mathcal{E}(k)$. Note that, when there are not packet losses, $\mathcal{G}(k) = \mathcal{G}^S$, and $\mathcal{N}(k)_i = \mathcal{N}_i^S$, for all $k = 0, 1, \dots$.

In the consensus algorithm, node i computes an estimate x_i of d by taking a linear combination of neighboring estimates and measures, i.e.,

$$x_i(k+1) = \sum_{j \in \mathcal{N}_i(k)} k_{ij}(k)x_j(k) + \sum_{j \in \mathcal{N}_i(k)} h_{ij}(k)u_j(k). \quad (16.5)$$

For node i , the algorithm is initialized with $x_j(0) = u_i(0)$ for all $j \in \mathcal{N}_i^0$. In vector notation, we have

$$\mathbf{x}(k+1) = \mathbf{K}(k)\mathbf{x}(k) + \mathbf{H}(k)\mathbf{u}(k). \quad (16.6)$$

The matrices $\mathbf{K}(k)$ and $\mathbf{H}(k)$ can be interpreted as the adjacency matrices of two weighted time-varying graphs compatible with $\mathcal{G}(k)$. The main problem can now be

stated as follows: Given a WSN modeled by a time-varying graph $\mathcal{G}(k)$, find matrices $\mathbf{K}(k)$ and $\mathbf{H}(k)$ compatible with $\mathcal{G}(k)$, such that $\mathbf{x}(k)$ is a minimum variance estimate of $\mathbf{d}(k)$ and $\mathbf{K}(k)$ and $\mathbf{H}(k)$ can be computed from local information.

Remark 16.1 Notice that in this problem we assume the sensor to be static, i.e. they are not mobile robots. Thus in this context the consensus is not in the physical position of the sensors, but on their state variables.

Distributed estimation We show in this section how the problem posed above can be solved by first presenting the corresponding centralized solution. The dynamics of the estimation error is given by

$$\begin{aligned} \mathbf{e}(k+1) &= \mathbf{x}(k+1) - \mathbf{d}(k+1)\mathbf{1} = \mathbf{K}(k)\mathbf{e}(k) \\ &\quad + (\mathbf{K}(k) + \mathbf{H}(k))\mathbf{d}(k)\mathbf{1} - \mathbf{d}(k+1)\mathbf{1} + \mathbf{H}(k)\mathbf{v}(k). \end{aligned}$$

By taking the expected value with respect to $\mathbf{v}(k)$, we obtain the following result, where $\gamma_{\max(\cdot)}$ denote the largest singular value (see [601] for a proof):

Proposition 16.4 *If for all $k = 0, 1, \dots$ it holds that $\gamma_{\max}(\mathbf{K}(k)) < 1$,*

$$(\mathbf{K}(k) + \mathbf{H}(k))\mathbf{1} = \mathbf{1}, \quad (16.7)$$

and $|\mathbf{d}(k+1) - \mathbf{d}(k)| < \delta$, then $\mathbf{e}(k) = \mathbf{x}(k) - \mathbf{d}(k)\mathbf{1}$ converges in mean to a hypercube centered in the origin with side length proportional to δ .

The proposition suggests two conditions on the matrices $\mathbf{K}(k)$ and $\mathbf{H}(k)$. The remaining degree of freedom will be used to minimize the variance of the estimates.

Let us introduce the covariance matrix $\mathbf{P}(k) = \mathbb{E}\{\mathbf{e}(k)\mathbf{e}^T(k)\}$ and note that

$$\mathbf{P}(k+1) \approx \mathbf{K}(k)\mathbf{P}(k)\mathbf{K}(k)^T + \sigma^2\mathbf{H}(k)\mathbf{H}(k)^T. \quad (16.8)$$

where the approximation is done assuming that $\mathbf{d}(k)$ is a slowly varying signal, or equivalently that $|\mathbf{d}(k+1) - \mathbf{d}(k)| < \delta \ll 1$

We would like to choose $\mathbf{K}(k)$ and $\mathbf{H}(k)$ such that the spectral norm of $\mathbf{P}(k+1)$ is minimized. A centralized solution can be obtained through a semi-definite program [601], but that solution is not interesting here since our goal is to find a distributed algorithm. In the following, we present a distributed algorithm by first characterizing the cases without packet losses, and then expanding the analysis to the case with losses.

Let M_i denote the number of neighbors of node i for the nominal communication graph, i.e., M_i is the cardinality of $\mathcal{N}_i^S = \{i_1, \dots, i_{M_i}\}$. Collect the estimation errors available at node i in the vector $\varepsilon_i(k) \in \mathbb{R}^{M_i}$. The elements of ε_i are ordered according to the node indices:

$$\varepsilon_i(k) = (e_{i_1}(k), \dots, e_{i_{M_i}}(k))^T, \quad i_1 < \dots < i_{M_i}.$$

Similarly, we introduce vectors $\kappa_i(k), \eta_i(k) \in \mathbb{R}^{M_i}$ corresponding to the nonzero elements of row i of the matrices $K(k)$ and $H(k)$, respectively, and ordered according to node indices. It follows from (16.8) that the variance of $e_i(k)$ at $k+1$ can be evaluated as

$$\mathbb{E}\{e_i(k+1)^2\} = \kappa_i^T(k) \Gamma_i(k) \kappa_i(k) + \sigma^2 \eta_i^T(k) \eta_i(k),$$

where $\Gamma_i(k) = \mathbb{E}\{\varepsilon_i(k) \varepsilon_i(k)^T\}$. To minimize the variance of the estimation error in each node, we propose that $\kappa_i(k)$ and $\eta_i(k)$ are chosen to minimize this expression. To obtain consensus and convergence as well, the following optimization problem should be solved at each time k and in each node i :

$$\min_{\kappa_i(k), \eta_i(k)} \quad \kappa(k)_i^T \Gamma_i(k) \kappa_i(k) + \sigma^2 \eta_i(k)^T \eta_i(k) \quad (16.9)$$

$$\text{s.t.} \quad (\kappa_i(k) + \eta_i(k))^T \mathbf{1} = 1 \quad (16.10)$$

$$\kappa_i(k) \in \Theta_i, k = 0, 1, \dots, \quad (16.11)$$

where

$$\Theta_i = \left\{ (\theta_1, \dots, \theta_{M_i}) : \sum_{j=1}^{M_i} \theta_j^2 < 1/2 \text{ and } \sum_{j=1}^{M_i} |\theta_j| < 1 \right\}. \quad (16.12)$$

The new optimization problem is tractable since the cost function (16.9) and the constraint (16.11) are convex and the constraint (16.10) is linear. Furthermore, if the constraint (16.11) is relaxed, then it is easy to show that we can even find an explicit solution from the dual formulation of the optimization problem:

$$\kappa_i(k) = \frac{\Gamma_i(k)^{-1} \mathbf{1}}{\sigma^{-2} M_i + \mathbf{1}^T \Gamma_i(k)^{-1} \mathbf{1}} \quad (16.13)$$

$$\eta_i(k) = \frac{\mathbf{1}}{M_i + \sigma^2 \mathbf{1}^T \Gamma_i(k)^{-1} \mathbf{1}}. \quad (16.14)$$

Note that the covariance matrix $\Gamma_i(k)$ needs to be estimated, since it depends on the unknowns d and v . In the implementation, this estimate is computed simply by a low-pass recursive filter, as further discussed in next section.

The case with lossy links requires only a slight modification to the scheme proposed above. In this case $M_i(k)$ is a function of time and interpreted as the cardinality of $\mathcal{N}_i(k)$ (instead of \mathcal{N}_i^S). If node i does not receive estimates $x_j(k)$ from some node $j \in \mathcal{N}_i(k-1)$, then $e_j(k)$ should simply be removed from the definition of $\varepsilon_i(k-1)$ to form $\varepsilon_i(k)$ and the covariance matrix $\Gamma_i(k-1)$ is punctured accordingly to obtain $\Gamma_i(k)$. If node i receives estimates $x_j(k)$ from some new node $j \in \mathcal{N}_i(k-1)$, then $e_j(k)$ should simply be added to the definition of $\varepsilon_i(k)$. The covariance matrix $\Gamma_i(k)$ is expanded from $\Gamma_i(k-1)$ by adding the covariance elements i, j and j, i , which are set to the maximum of the elements of $\Gamma_i(k-1)$. In both cases of establishment of new connections, or lost of existing ones, the optimal vectors $\kappa_i(k)$ and $\eta_i(k)$ can be readily calculated as in (16.13) and (16.14). The estimate $x_i(k+1)$ is then obtained from (16.5).

16.2.3 Distributed Kalman filtering using weighted averaging

As in the consensus problem described above, the objective is to estimate a number of time varying physical quantities or states. The time variation is modeled as a linear time-invariant system subject to noise. The problem can be solved for a fixed communication topology using a weighted averaging approach. The fixed communication topology allows for most of the computations to be done off-line. The on-line computations consist only of the standard stationary Kalman filter equation plus one additional step, where estimates from neighbours are used.

Aim and generalities We assume that there's no centralized computation center but the full state estimation is done with a decentralized algorithm. Then we assume that communications take place only between neighbors. Furthermore to reduce the bandwidth consumption of individual nodes, estimates instead of measurements are transmitted. A new estimate is formed as a weighted average of the neighboring estimates. The weights are optimized to yield a small estimation error covariance in stationarity. We also take account of graphs with possible loops.

Problem formulation Consider the following discrete-time linear system:

$$\mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{v}(k), \quad (16.15)$$

where $\mathbf{x}(k) \in \mathbb{R}^n$ is the state of the system and $\mathbf{v}(k) \in \mathbb{R}^n$ is a stochastic disturbance. The disturbance is assumed to be a white zero mean Gaussian process with covariance defined below.

The process is observed by N agents each with some processing and communication capability. The agents are labeled $i = 1, 2, \dots, N$ and form the set V . The communication topology is modeled as a graph $G = (V, E)$, where the edge (i, j) is in E if and only if node i and node j can exchange messages. The nodes to which a node communicates are called neighbors and are contained in the set N_i . Note that node i is also included in the set N_i .

Each node observes the process (16.15) by a measurement $\mathbf{y}_i(k) \in \mathbb{R}^{p_i}$ of the following form:

$$\mathbf{y}_i(k) = \mathbf{C}_i\mathbf{x}(k) + \mathbf{e}_i(k), \quad (16.16)$$

where $\mathbf{e}_i(k) \in \mathbb{R}^{p_i}$ is a white zero mean Gaussian process. The measurement and process disturbances are correlated according to

$$E \begin{bmatrix} \mathbf{v}(k) \\ \mathbf{e}_1(k) \\ \vdots \\ \mathbf{e}_N(k) \end{bmatrix} \begin{bmatrix} \mathbf{v}(l) \\ \mathbf{e}_1(l) \\ \vdots \\ \mathbf{e}_N(l) \end{bmatrix}^T = \begin{bmatrix} R_v & 0 & \dots & 0 \\ 0 & R_{e11} & \dots & R_{e1N} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & R_{eN1} & \dots & R_{eNN} \end{bmatrix} \delta_{kl} \quad (16.17)$$

where $\delta_{kl} = 1$ only if $k = l$. Note that this is a heterogeneous setup where each agent is allowed to take measurements of arbitrary size and precision. Further the disturbances acting on the measurements are allowed to be correlated.

Each node is only allowed to communicate with its neighbors and only once between each measurement. Further the only assumption made on the graph structure is that it has to be connected, other assumptions such as requiring it to be loop free is not necessary. This setup is somewhat different from the setup used in for example distributed control problems where each node in the graph also has dynamics associated with it. The reader should think of the problem studied here as for example a network of sensors trying to estimate the position of an external object they observe.

The goal is to make sure the every node in the network has a good estimate $\hat{x}_i(n)$ of the state $x(n)$.

Sharing estimates The first problem that occurs is that estimates are not independent, as they contain the same process noise, and possibly also the same measurement information. To optimally combine two estimates the mutual information must be subtracted.

For a graph with loops, two nodes can not compute the mutual information by just using local information.

To solve the problem for a general communication topology a global off-line method will be used. Instead of subtracting the mutual information, the estimates are weighed so that the covariance of the merged estimate is minimized. This approach will not give the optimal solution, but is applicable to graphs with loops.

On-line computations The algorithm consists of the two traditional estimation steps, measurement update and prediction together with an additional step where the nodes communicate and merge estimates. We will refer to an estimate after measurement update as local and after the communication step as regional.

1. *Measurement update*: The local estimate $\hat{x}_i^{\text{local}}(k|k)$ is formed by the predicted regional estimate $\hat{x}_i^{\text{reg}}(k|k-1)$ and the local measurement $y_i(k)$:

$$\hat{x}_i^{\text{local}}(k|k) = \hat{x}_i^{\text{reg}}(k|k-1) + \mathbf{K}_i[y_i(k) - \mathbf{C}_i\hat{x}_i^{\text{reg}}(k|k-1)], \quad (16.18)$$

where \mathbf{K}_i is computed off-line. The predicted estimate at time zero is defined as $\hat{x}_i^{\text{reg}}(0|-1) = \hat{x}_0$, where \hat{x}_0 is the initial estimate of $x(0)$.

2. *Merging*: First the agents exchange their estimates over the communication channel. This communication is assumed to be error and delay free. The merged estimate $\hat{x}_i^{\text{reg}}(k|k)$ in node i is defined as a linear combination of the estimates in the neighboring nodes N_i .

$$\hat{x}_i^{\text{reg}}(k|k) = \sum_{j \in N_i} \mathbf{W}_{ij} \hat{x}_j^{\text{local}}(k|k) \quad (16.19)$$

The weighting matrices \mathbf{W}_{ij} are computed off-line by the procedure described in [Section 16.2.3](#).

3. *Prediction*: Because the measurement and process noises are independent, the prediction step only includes

$$\hat{\mathbf{x}}_i^{\text{reg}}(k+1|k) = \mathbf{A}\hat{\mathbf{x}}_i^{\text{reg}}(k|k). \quad (16.20)$$

Off-line computations To be able to execute the steps described above the parameters $\mathbf{K}_i \in \mathbb{R}^{n \times p_i}$ and $\mathbf{W}_{ij} \in \mathbb{R}^{n \times n}$ must be chosen. In this section an iterative algorithm will be developed for this purpose. First let the estimation error in node i be defined as

$$\tilde{\mathbf{x}}_i(k) = \mathbf{x}(k) - \hat{\mathbf{x}}_i(k) \quad (16.21)$$

with covariance

$$\mathbf{P}_{ij}(k) = \mathbb{E}\tilde{\mathbf{x}}_i(k)\tilde{\mathbf{x}}_j(k)^T. \quad (16.22)$$

Now note that the estimation error covariance (16.22) after step 1 above can be written as

$$\mathbf{P}_{ij}^{\text{local}}(k|k) = (\mathbf{I} - \mathbf{K}_i(k)\mathbf{C}_i)\mathbf{P}_{ij}^{\text{reg}}(k|k-1)(\mathbf{I} - \mathbf{K}_j(k)\mathbf{C}_j)^T + \mathbf{K}_i\mathbf{R}_{eij}\mathbf{K}_j^T. \quad (16.23)$$

with $\mathbf{P}_{ij}^{\text{reg}}(0| - 1) = \mathbf{P}_0$ where \mathbf{P}_0 is the initial estimation error covariance. Minimizing $\mathbf{P}_{ii}^{\text{local}}$ with respect to $\mathbf{K}_i(k)$ gives

$$\mathbf{K}_i(k) = \mathbf{P}_{ii}^{\text{reg}}(k|k-1)\mathbf{C}_i^T(\mathbf{R}_{eii} + \mathbf{C}_i\mathbf{P}_{ii}^{\text{reg}}(k|k-1)\mathbf{C}_i^T)^{-1}. \quad (16.24)$$

The goal of the weight selection is to minimize the estimation error covariance matrix in each node in steady state. That is, to minimize $\mathbf{P}_{ii}^{\text{reg}}(k|k)$ for all i given the constraints (16.26) and (16.27) as k approaches infinity.

To simplify notation the time indexes will be dropped in the following part. The optimization problem for each node can thus be posed as

$$\min_{\mathbf{W}_i} \mathbf{P}_{ii}^{\text{reg}}, \quad (16.25)$$

subject to

$$\sum_{j \in N_i} \mathbf{W}_{ij}(k) = \mathbf{I}_{n \times n} \quad (16.26)$$

and

$$\mathbf{W}_{ij}(k) = \emptyset \text{ if } (i, j) \notin \mathcal{E}, \quad (16.27)$$

where $\mathbf{W}_i = [\mathbf{W}_{i1} \dots \mathbf{W}_{iN}]$.

16.3 Adaptive resource management and networking for control applications

The goal of this section is to derive adaptive frameworks for the joint optimization of control performance and network operations in time-varying link and networking contexts. Two specific and complementary items are considered: how to handle uncertain delays (a major quality of service metric) in a control loop, and how to model the interaction of application requirements (e.g. safety) with explicit components of a wireless link (e.g. modulation format and transmission power).

In this context a constrained finite time optimal controller (CFTOC) for networked control systems (NCS) is presented for the case where: (a) the induced network delays are uncertain and bounded, and (b) there are constraints in the magnitude of the control action and the outputs/states of the system. Based on the assumption that the round-trip latency time varies within known bounds, the transitions of the random delays $d(k)$ are modelled as a finite-state Markov process and are embedded in the system model, resulting in a polytopic uncertain system.

If a static feedback law is employed (which is not the case here), it is then allowed to lump the delays $\tau_s(k)$ and $\tau_a(k)$ into one “lumped” delay $\tau_t(k) = \tau_s(k) + \tau_a(k)$. Since the delay $\tau_s(k)$ can be handled assuming “time-stamps” in the packets arriving from the sensor to the controller [487, 488], our approach focuses on the case of controller-to-actuator delay ($\tau_a(k)$). Our current research effort focuses on time-varying (uncertain) delays with known bounds.

16.3.1 Networked control systems: hybrid modeling issues

Focusing on the time varying delay $\tau_a(k)$ in the actuation path, we consider the following system model with the following input delays and input constraints to describe the dynamics of the class of NCS examined in this work:

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t - \tau), \quad \mathbf{y}(t) = \mathbf{C}\mathbf{x}(t), \quad (16.28)$$

$$\mathbf{u}_{\min} \leq \mathbf{u}(t) \leq \mathbf{u}_{\max}, \quad \mathbf{y}_{\min} \leq \mathbf{y}(t) \leq \mathbf{y}_{\max}, \quad (16.29)$$

with $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{u} \in \mathbb{R}^m$. In the current approach the network-induced delays are considered to be greater than one sampling period as shown for example in the recent work [499], where the unknown, time-varying time delay is decomposed into two parts: one fixed part which is unknown and is an integer multiple of the sampling time; and a second part which is randomly varying but bounded by one sampling time. The transitions of the random delays $d(k)$ could be modelled as a finite-state Markov process [671]. In this case the probabilistic notions of stability can be given by

$$\text{Prob} [d(k + 1) = j \mid d(k) = i] = p_{ij}, \quad (16.30)$$

where $0 \leq i, j \leq D$. This model is quite general and the communication package loss in the network can be naturally modeled as explained below. The assumption

here is that the controller will always use the most recent data. Thus if we have $x(k - d(k))$ at step k , but there is no new information coming at step $k + 1$ (data could be lost or there is a longer delay), then we at least have $x(k - d(k))$ available for feedback. So the delay $d(k)$ can increase at most by 1 each step, and so we constrain:

$$\text{Prob}[d(k + 1) > d(k) + 1] = 0. \quad (16.31)$$

Suppose now that the delay τ is constant and exactly known and smaller than h , $\tau < h$. Then, following [24], the discretization of (16.28) is quite straightforward by realizing that between the sampling instances kh and $(k + 1)h$, the control action coming out of the event-driven (zero-order) hold device, changes from $u(kh - h)$ into $u(kh)$ at the instant $kh + \tau$. The system becomes

$$\mathbf{x}(kh + h) = \Phi \mathbf{x}(kh) + \Gamma_0 \mathbf{u}(kh - dh + h) + \Gamma_1 \mathbf{u}(kh - dh), \quad (16.32)$$

where

$$\Gamma_1 = \int_{h-\tau}^h \exp(\mathbf{A}r) \mathbf{B} dr, \quad (16.33)$$

$$\Gamma_0 = \int_0^{h-\tau} \exp(\mathbf{A}r) \mathbf{B} dr. \quad (16.34)$$

Complete model with uncertain delay and input constraints Suppose that the delay is unknown but bounded by $\tau_{\min} \leq \tau \leq \tau_{\max}$. Following a procedure similar to the one described in ([624]), the system's nominal model and the corresponding control synthesis is intentionally based on the choice of the average delay $\tau_{\text{avg}} = (\tau_{\min} + \tau_{\max})/2$ as the nominal value of the uncertain delay. The actual uncertain delay can now be modelled as

$$\tau = \tau_{\text{avg}} + \alpha \tau_{\max} \delta, \quad \text{with } \|\delta\| \leq 1 \quad (16.35)$$

where the auxiliary constant variable α is defined as $\alpha \triangleq 0.5 (\tau_{\max} - \tau_{\min})$. The nominal and the uncertain part of the plant model can now be derived. From now on we shall use the simpler notation τ_a instead of τ_{avg} since there will be no confusion with the controller-to-actuation delay.

It is possible to decompose the matrices Γ_0, Γ_1 into a constant-known part $\Gamma_0(\tau_a), \Gamma_1(\tau_a)$ and an uncertain part $\Delta \Gamma_0(\tau), \Delta \Gamma_1(\tau)$. It results in

$$\Gamma_0(\tau_a) = \int_0^{h-\tau_a} \exp(\mathbf{A}r) \mathbf{B} dr, \quad (16.36)$$

$$\Delta\Gamma_0(\tau) = \exp(\mathbf{A}(h - \tau_a)) \int_0^{\tau_a - \tau} \exp(\mathbf{A}\lambda) \mathbf{B} d\lambda, \quad (16.37)$$

$$\Gamma_1(\tau_a) = -\Gamma_0(\tau_a) + \int_0^h \exp(\mathbf{A}r) \mathbf{B} dr, \quad (16.38)$$

$$\Delta\Gamma_1(\tau) = -\Delta\Gamma_0(\tau). \quad (16.39)$$

Equations (16.36), (16.38) define the computable “nominal plant,” whereas (16.37), (16.39) define the (polytopic) uncertainty. The dynamics of the system can then be written as:

$$\begin{aligned} \mathbf{z}(k+1) &= \mathbf{A}_d(\tau) \mathbf{z}_k + \mathbf{B}_d(\tau) \mathbf{u}_k \\ &= (\mathbf{A}_d(\tau_a) + \Delta\mathbf{A}_d(\tau)) \mathbf{z}_k + (\mathbf{B}_d(\tau_a) + \Delta\mathbf{B}_d(\tau)) \mathbf{u}_k, \end{aligned} \quad (16.40)$$

with:

$$\Delta\mathbf{A}_d(\tau) = \begin{bmatrix} 0_{n \times n} & \Delta\Gamma_1(\tau) \\ 0_{m \times n} & 0_{m \times m} \end{bmatrix}, \quad (16.41)$$

and

$$\Delta\mathbf{B}_d(\tau) = \begin{bmatrix} \Delta\Gamma_0(\tau) \\ 0_m \end{bmatrix}. \quad (16.42)$$

Now we can use the norm bounds of $\Delta\mathbf{A}_d(\tau)$, $\Delta\mathbf{B}_d(\tau)$ to construct the two vertices of the polytope and proceed with the control design. Adding the input and output constraints, the uncertain system in (16.40) becomes a constrained polytopic uncertain system, which can be handled using the CFTOC machinery [68, 374].

16.3.2 Controller synthesis

Assume a discrete-time (possibly polytopic uncertain) system

$$\mathbf{x}(k+1) = \mathbf{A}_i \mathbf{x}(k) + \mathbf{B}_i \mathbf{u}(k), \quad (16.43)$$

with state (output) and input constraints expressed via the following “guard functions”:

$$\begin{bmatrix} \mathbf{x}(k) \\ \mathbf{u}(k) \end{bmatrix} \in \mathcal{P} = \{H_i \mathbf{x} + J_i \mathbf{u} \leq K_i, i = 1, \dots, M\}. \quad (16.44)$$

The CFTOC (“multi-parametric”) approach consists in computing the optimizer vector $U_N = \{\mathbf{u}'_0, \dots, \mathbf{u}'_{N-1}\}'$, which minimizes the following cost function:

$$J_N(\mathbf{x}_0) = \min_{U_N} \left[\|P\mathbf{x}_N\|_l + \sum_{i=0}^{N-1} (\|\mathbf{R}\mathbf{u}_k\|_l + \|\mathbf{Q}\mathbf{x}_k\|_l) \right], \quad (16.45)$$

subject to the linear system dynamics and state/input constraints defined before. The cost (16.45) may be linear (e.g. $l \in \{1, \infty\}$) or quadratic (e.g. $l = 2$) depending on

the vector norm employed. \mathbf{x}_0 is the currently available sample of the state vector, while \mathbf{x}_k with $k = 0, \dots, N - 1$ are the predicted values of the state vector through (16.43) starting from $\mathbf{x}_0 = \mathbf{x}(0)$ and applying the input sequence U_N . Note the clear distinction made between the current actual state $\mathbf{x}(k)$ and the variable \mathbf{x}_k used in the cost function. (Analogous comments hold for the system’s input $\mathbf{u}(k)$ and the k -th optimization variable \mathbf{u}_k .) Moreover, N is the prediction horizon, \mathbf{Q} , \mathbf{R} , and \mathbf{P} are the full column rank weighting matrices on the corresponding optimization variables, i.e. predicted states, control effort, and the desired final state, respectively. The predicted final state \mathbf{x}_N is usually supposed to belong to a predefined set X_{set} a choice typically dictated by stability and feasibility requirements especially when CFTOC is implemented in a receding horizon fashion.

For a given initial state $\mathbf{x}(0)$, problem (16.43), (16.44), (16.45) can be solved as a linear (LP) or quadratic (QP) Program for linear or quadratic cost objectives respectively. It is well known [68, 102, 280, 374] that the “multi-parametric” CFTOC optimizer is a continuous piecewise affine state feedback of the following form:

$$\mathbf{u}(k) = \mathbf{F}_j \mathbf{x}(k) + \mathbf{G}_j, \text{ if } \mathbf{x}(k) \in R_j, \quad (16.46)$$

defined over convex polyhedra R_j —henceforth referred to as “regions”—which are also generated by the CFTOC algorithm. The algorithm additionally provides the feasibility set $X_f \subseteq \mathbb{R}^n$ which is the set of all initial states $\mathbf{x}(0)$ for which the CFTOC problem is feasible, i.e. $X_f = \{\mathbf{x}(0) \in \mathbb{R}^n \mid \exists (\mathbf{u}_0, \dots, \mathbf{u}_{N-1}) \in \mathbb{R}^{Nm}, \mathbf{x}(k) \in X, \mathbf{u}(k-1) \in U, \forall k \in \{1, \dots, N\}\}$.

The previous state regulation cost function (16.45) can be easily modified into an output (set-point) regulation problem using $\|\mathbf{Q}_y (\mathbf{y}_{ref} - \mathbf{y}_k)\|$ instead of $\|\mathbf{Q}\mathbf{x}_k\|$, with \mathbf{Q}_y being the output weighting matrix (instead of the state-weighting matrix \mathbf{Q}).

Notice that even though the computations of the multi-parametric control law are carried out off-line, they quickly become prohibitive for larger problems. This is not only due to the high complexity of the multi-parametric programs involved, but mainly because of the exponential number of transitions between regions which can occur when a controller is computed in a dynamic programming fashion [104]. Thus the number of the controller’s regions is not only a measure of the controller’s complexity but also affects directly its on-line implementation in the form of a look-up table.

16.3.3 Modeling of adaptive transmission behaviors in control over wireless

Introduction In this section our aim is to develop a hybrid dynamical model for resource allocation over wireless links, with particular attention to wireless sensor and actuator networks used in control applications. In this context we try to exploit the theoretical framework related to the platform-based design (PBD) methodology [580]. According to the PBD principles, we first map control specifications to communication network specifications involving a set of values of the communication parameters such as quantization noise, coding, modulation scheme, and power level.

Then, a communication scheme is chosen according to the communications specifications so that the control specifications are satisfied. To do so, we use a hybrid system formalism that models the dynamical behavior of the communication scheme to capture the mixed continuous-discrete characteristics of the configuration parameters.

Model We consider a basic feedback digital control scheme, where the controller and the plant exchange signals by means of a time-varying wireless link. We take into account a control scheme where the controller C and the plant P share information via a wireless link; information coming from the controller is at first quantized through a uniform quantizer and then sent via a wireless link; information coming from the plant is at first sampled, then quantized through a uniform quantizer, and finally sent via the wireless link to the controller.

We associate to P the corresponding exponential discretization P_T with sampling time $T > 0$ (e.g. see [139]):

$$\begin{aligned} \mathbf{x}(t+1) &= \mathbf{A}_d \mathbf{x}(t) + \mathbf{B}_d \mathbf{u}(t), \\ \mathbf{x}(t) &\in \mathbb{R}^n, \mathbf{u}(t) \in \mathbb{R}^m, t \geq 0. \end{aligned}$$

16.3.4 Safety problem with wireless networks and the hybrid model

The controller C is a digital controller. The communication system is characterized by the sampling time T , the quantization threshold M , the quantization width δ , the modulation scheme mod(e.g. PAM, PSK, FSK), the transmission power p , the distance between the plant and the controller r , and the channel disturbance n .

We can state the problem as follows: given a plant P and a set of good states $\Omega \subset \mathbb{R}^n$ within which the state x of the plant P should evolve, find

- a digital controller C ;
- a communication system; and
- the set of all initial states in Ω .

such that the closed-loop system satisfies the safety requirements, i.e.

$$x(t) \in \Omega, \forall t \geq 0.$$

We solve this problem by applying the platform based design methodology and thus breaking the original controller synthesis problem into two main steps:

1. We consider the synthesis of the controller C subject to bounded disturbances that capture the non-idealities of the communication channels.
2. Then we design the communication system so that the disturbance bounds are never violated and to do so we use the hybrid system formalism.

We assume that a *corrupted packet is not discarded* and nonidealities coming from the communication system are modeled as additive continuous disturbances d_1 and d_2 and delays τ_1 and τ_2 in data transmission, as shown in Fig. 16.6. We assume that

$$d_1 \in D_1, d_2 \in D_2, \tau_1 \in [0, \tau_{1,\max}], \tau_2 \in [0, \tau_{2,\max}], \quad (16.47)$$

where $D_1 \subset \mathbb{R}^m$ and $D_2 \subset \mathbb{R}^n$ are compact sets and $0 \in D_1$ and $0 \in D_2$. Note that $D_1 = D_1(T, M, \delta, \text{mod}, p, r, n)$ and $D_2 = D_2(T, M, \delta, \text{mod}, p, r, n)$

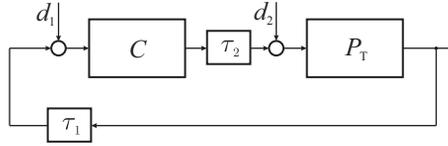


Fig. 16.6 Control scheme.

It should be noticed that we only carry out our analysis for the uplink since the downlink can be similarly treated. Moreover, indicating as W_{uplink} the space generated by combinations of the communication parameters, our aim becomes the association between the set D_1 and the set $\Phi(D_1) \subset W_{\text{uplink}}$, i.e. we want to define, for a given set of disturbances D_1 for the value of d_1 , the set of communication parameters $(T, M, \delta, \text{mod}, p, r, n)$ that guarantee $d_1 \in D_1$.

For the sake of simplicity, we assume here that:

- variables T, M are fixed a priori;
- variables δ, mod can have a finite set of values;
- $D_1 = [0, D_{1,\max}]$ is mono-dimensional.

Then, we need to define a function $\Phi_{T,M,\delta,\text{mod}}$ from D_1 to the space generated by the triple (p, r, n) for each combination of the parameters T, M, δ, mod . Note that the number of these combinations is finite.

The function Φ maps the set of allowed disturbance values $d_1 \in D_1$ to a subset of W_{uplink} of allowed communication parameters. The range of $\Phi_{T,M,\delta,\text{mod}}(D_1)$ is a projection of $\Phi(D_1)$ for fixed parameters T, M, δ, mod on the three-dimensional space with coordinates (p, r, n) .

If an m -PAM modulation is assumed it is possible to demonstrate that the function Φ can be explicitated as follows:

$$\Phi_{T,M,\delta,m\text{-PAM}}(D_1) = \left\{ (p, r, n) : |n(r(t), t)| \leq \frac{p(2z - 1)}{2m} \right\},$$

where z depends on the disturbance bound $D_{1,\max}$, the quantization width δ , and the quantization threshold M . This equation then translates the specifications given at the controller-plant level to specifications at the communication system level.

Since we found out relations between the sets D_i and the communication parameters, our problem translates into finding suitable communication parameters for which the robust safety problem can be solved.

We underline that the set of feasible solutions to our control problem is specified by means of a set of admissible *static* configurations of the communication parameters $(T, M, \delta, \text{mod}, p, r, n)$. In a real wireless channel, parameters as the distance

r between the controller and the plant, and the disturbance n introduced by the channel, are time-varying. Nevertheless, it is still possible to apply a static approach to a time-varying scenario, by restricting the maximum distance and disturbance over time

$$\max_t(n(r(t), t)),$$

thus guaranteeing that the control algorithm works. However, using the “strongest” configuration would lead to a waste of energy. We propose an adaptive configuration of the communication parameters to minimize the transmission power.

We assume that the sampling time T and the quantization threshold M are fixed, and the tunable parameters are as follows:

1. *quantization width* δ can be selected in a finite set of admissible values: $\Delta = \{\delta_1, \dots, \delta_{|\Delta|}\}$;
2. *modulation scheme* Mod can be selected in a finite set of admissible values: $\text{MOD} = \{2^1 - \text{PAM}, 2^2 - \text{PAM}, \dots, 2^{|\text{MOD}|} - \text{PAM}\}$;
3. *power level* p can be selected in a finite set of admissible values

$$P = \{p_1, \dots, p_{|P|}\}.$$

Furthermore, we assume that the evolution of the *distance* r between the plant and the controller is described by the (autonomous) dynamics $\dot{r} = f(r)$.

We use now an hybrid formalism in order to give a rigorous formulation of an adaptive communication system. This allows us to formally define properties of an adaptive communication system (e.g. the feasibility and probability of “out-of-service”).

Recalling that a generic hybrid system can be described by the tuple

$$\mathcal{H} = (Q \times X, Y, S_q, \Sigma, E),$$

we define the set Q as the set of all tunable parameters in the transmission scheme, namely:

$$Q := \Delta \times P \times \text{MOD}.$$

A discrete state $q \in Q$ is a triple (δ, mod, p) that defines the current configuration of the communication system. We define Σ as the finite set of configuration commands to the quantizer and the physical layer (configuration of (δ, mod, p)), and the set of edges $E \subseteq Q \times \Sigma \times Q$ such that each arc models an admissible switching of the communication scheme, and can be triggered by the discrete control input $\sigma \in \Sigma$. For instance, the physical layer can accept configuration commands to set the current transmission power p by means of either increments and decrements or power level value. In the first case, the set of arcs E connects only states associated to neighbor values of power, while in the second case the graph will be fully connected.

The continuous state $(n, r) \in \mathbb{R}^2$ is given by the disturbance introduced by the channel n and by the distance r between the transmitter and the receiver. The dynamics of r is given by the differential equation $\dot{r} = f(r)$. The observed continuous

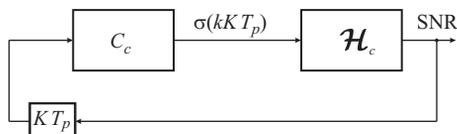


Fig. 16.7 Controller of the communication parameters.

output $y \in Y$ is the signal-to-noise ratio (SNR), which depends on the current hybrid state (namely the current communication configuration, the communication channel status, and the distance) as follows:

$$\text{SNR} = g_{(\delta, \text{mod}, p)}(n, r, t).$$

The current discrete state of \mathcal{H}_c , i.e. the current communication system configuration, is determined by the discrete input $\sigma \in \Sigma$ generated by the controller of the communication system C_c . This controller receives as input the continuous output of \mathcal{H}_c , as shown in Fig. 16.7.

We assume that the controller C_c is not able to generate a control at each time instant $t \in \mathbb{R}^+$: we assume here that a command σ can be generated only after the transmission of K packets, namely at time instants kKT_p , where $k \in \mathbb{N}$ and T_p is the transmission time of a packet. Thus, C_c controls the discrete dynamics of \mathcal{H}_c generating output symbols according to a strategy modeled by the function $\gamma : Y \rightarrow \Sigma \cup \{\varepsilon\}$, where $\sigma(kKT_p) = \gamma(y(kKT_p))$, $k \in \mathbb{N}$. When the controller does not generate any symbol (no control command), $\gamma(kKT_p)$ is equal to the empty string ε .

Does a control strategy γ of the communication configuration exist for any disturbance n introduced by the channel and motion $\dot{r} = f(r)$ of the transmitter/receiver, such that the communication parameters belong for each time instant to the set $\Phi(D_1)$? That is, does a control strategy exist such that the discrete state $q = (\delta, \text{mod}, p)$ and the continuous state (n, r) of \mathcal{H}_c both belong to $\Phi(D_1)$?

This is a safety problem on \mathcal{H}_c , and Γ is the set of all control strategies γ on the system \mathcal{H}_c : we define a function $\Psi : \Gamma \rightarrow Q \times X$ that associates to a control strategy the safe set that can be guaranteed on the state space $Q \times X$ of \mathcal{H}_c . Note that, by definition, $Q \times X = (\Delta \times \text{MOD} \times P) \times (\mathbb{R} \times \mathbb{R})$ and $W_{\text{uplink}} = T \times M \times \Delta \times \text{MOD} \times P \times \mathbb{R} \times \mathbb{R}$ with T, M fixed. Thus, following the “meet-in-the-middle” argument of PBD [580], the space of solutions \mathcal{S} of our control problem on a time-varying wireless channel is defined as follows:

$$\mathcal{S} := \{\gamma : \Phi(D_1) \subseteq \Psi(\gamma)\}.$$

Once the space of solutions is defined, it is possible to search for a specific strategy γ^* that minimizes a desired cost function, e.g. minimizes the transmission power p .

16.4 A case study on networked control: the mining ventilation process

16.4.1 Description of the mining ventilation process

In this section we provide a description of a mine ventilation process. The main processes associated to ore extraction are drilling and blasting, ore transportation, and ore crushing. One supporting process is the ventilation in tunnels, clearly needed for the oxygen supply of the personnel and for the combustion process of vehicles (e.g. loader and dump trucks involved in ore transportation). The ventilation is achieved by a turbine and a heater on the surface and a vertical ventilation shaft (primary system), operated on a clockwise basis. Air pumped in from the surface is usually heated (in winter time at least), to avoid it to cause freezing down in the mine; moreover, it is necessary to cool down air at high depths (more than 1100 meters) because of the geothermal effect. From the primary ventilation shaft, a system of fans at each depth level of the mine pumps fresh air to the extraction rooms via tarpaulin tubes (secondary system): the secondary system is currently controlled based on manual demand by personnel entering a room. Bad quality air naturally flows because of pressure gradient from the extraction rooms back into the decline (which is a spiraling tunnel down) and to the exhaust ventilation shaft, which is similar but separate from the primary ventilation shaft.

At the endpoints of tarpaulin tubes, manual clips allow unused rooms to be isolated from ventilation. For these reasons, the secondary system is a typical hybrid environment, where the continuous variables are given by gas concentrations and air-flows, and the discrete variables are given by the number of trucks and the status of clips. It is interesting to note that ventilation represents about 50–60% of the power consumption cost for the ore extraction process. As we will point out in the next chapter, the current ventilation control system is either poor or non-existing. Moreover, a continuous monitoring of air quality is absent. The amount of air pumped in the rooms is controlled in open loop, and safety is guaranteed with a huge margin. It is clear that investigating automatic control solutions is of great industrial interest: because of heating and cooling down of air at different depth levels, the amount of pumped air can be minimized to save energy consumption.

After all accessible ore has been retrieved from a mine level, the extraction rooms are filled and a new level further down along the decline is bored. All equipments, including the ventilation, has to be moved and re-configured in the new level. Hence mining is like a mobile process industry. Obviously there is a clear economic benefit in fast re-commissioning of the ventilation control system. In this context, the idea of having wireless access to the sensors as well as fan control inputs is quite natural. Moreover, wiring is unfeasible in the extraction rooms because of blasting and drilling operations. In fact, electric power cables (220, 400, and 1000 V) and lighting are usually available up to the entrance of the rooms.

Wireless networks can be used also for improving the efficiency of other processes that are of importance in operating a mine, e.g. equipment (trucks and

ventilation system) maintenance, people and equipment localization, voice communication, and security via video surveillance.

The overall objective of the mining ventilation control system is to provide good air quality in the extraction rooms. For a future wireless system supporting the ventilation control systems, it is also desirable to increase safety by using the wireless system for personal communication and localization. We specify the objective as follows:

- Control air quality (O_2 , NO_x and CO_x) in the extraction rooms at suitable levels.

This is suitable to fulfill a cascade control configuration with the following two objectives:

- Regulate turbine and heater to provide suitable air flow pressure at the ventilation fans in the tunnels.
- Regulate ventilation fans to ensure air quality in extraction rooms.

An additional system objective is to obtain:

- Safety through wireless networking for personal communication and localization.

It should be noted that today's control architecture does not enable the fulfillment of these objectives, since there is no automatic control: in fact, maximum ventilation power is used during ore extraction. It provides no continuous monitoring of the air quality, it uses no wireless sensing and hence does not allow for a localization system to be easily implemented. The proposed wireless control architecture strives for fulfilling all the objectives listed above.

16.4.2 Today's control architecture

The ventilation is achieved by a turbine and a heater, operated on a clockwise basis, and a system of fans, controlled based on demand of air flow in different parts of the mine thanks to frequency converters. The inflow from the turbine has to provide air to fans that can be located very far from the ground (up to a kilometer deep) and that is usually heated (in winter time at least), to avoid it causing freezing down in the mine. The fresh air is then carried to the extraction rooms thanks to tarpaulin tubes connected to the fans, which pump the air from the vertical shaft. The actual automation is operated in a clock-wise way for the turbine and heater (i.e. they are run at maximum speed for a preset number of hours) and based on the vehicles demand combined with a timer for the fans. This airflow demand is determined based on messages sent by the drivers to the control room using walkie-talkies, indicating their position and which fan needs to be set to its maximum speed, as described in Fig. 16.8. An additional safety system triggers the fan high-speed operation based on a motion detector placed at the room's entrance.

To summarize, the actual control architecture is characterized by:

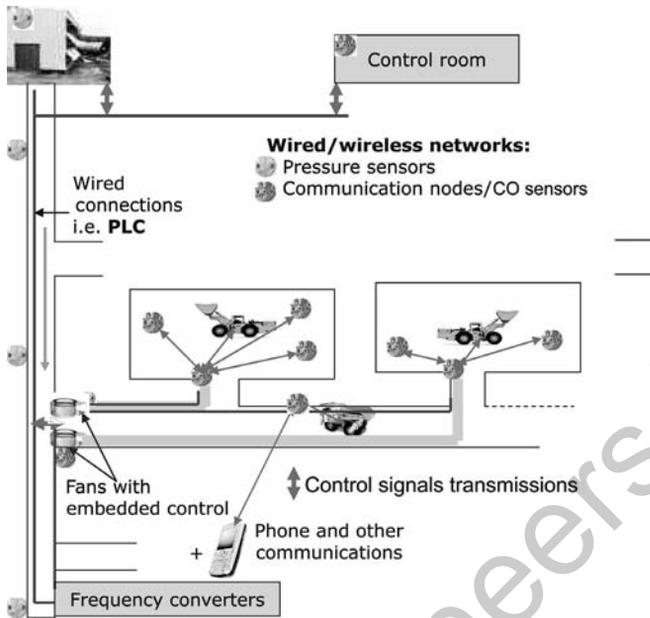


Fig. 16.9 System description.

simple, due to the limited computation capabilities, while we can develop more advanced control strategies for the algorithms run in the control room.

Based on this control approach, the ventilation system can be described in two parts: one fixed installation, which is the primary air supply from the ground via a vertical shaft, and the secondary system, a mobile network of fans located underground. This distinction results in the block diagram description proposed in Fig. 16.10, where both the physical (air flow) and communication (wired and

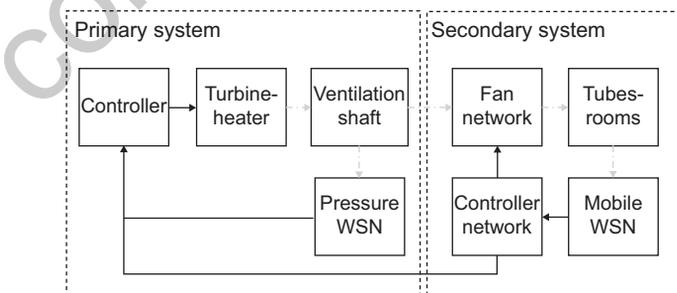


Fig. 16.10 Block diagram description of the control setup, including communications (—) and air flows (- · -).

wireless) interconnections are presented. This description includes the existing systems as well as key elements of the advanced technological solutions expected in this project.

16.5 Conclusions and open problems

In this chapter, we addressed topics in control over networks, i.e. control implementations where the control actions and decisions are taken based on measurement, decisions, and actuations that take place in a distributed environment. We discussed bounds and limitation among multi-agent processing, coordination of mobile multi agents problem, the consensus problem, and distributed Kalman filtering estimation. Moreover, we proposed some scenarios on adaptive resource management over a networked control system. We finally proposed a mining ventilation problem as a real case study on networked control system. We showed that distributed and networked systems require hybrid systems analysis techniques, since they include continuous and discrete dynamics.

However, many open problems yield difficult industrial implementation of wireless networked control systems. An interesting research line is providing novel theoretical results in the integration of control theory (stabilization and controller performance guarantee), communication theory (real communication and scheduling protocols-specifications) and theory of computation (computational complexity analysis). Given a multi-hop wireless network, a physical plant, and a CPU that implements a control algorithm, it is not trivial to integrate these different components into the same mathematical framework. Such a theoretical framework would allow analysis of optimization problems for minimizing power consumption of the network, while preserving properties of the closed-loop control system such as (practical) stability, minimum control performance, safety, etc.

controlengineers.ir

Solar air conditioning – a benchmark for hybrid systems control

E. F. Camacho and D. Zambrano

By considering a solar air conditioning plant, the typical steps for analyzing and controlling hybrid systems under practical circumstances are described in this chapter.

Chapter contents

17.1	Plant description	page	502
17.1.1	Main components		502
17.1.2	Signals for controlling the plant		504
17.1.3	Operating modes of the process		504
17.2	Plant model		505
17.3	Hybrid control		506
17.3.1	Control objectives		506
17.3.2	Hybrid control algorithm		507
17.3.3	Experimental results and discussion		508

17.1 Plant description

The present chapter describes the application and the implementation of a hybrid control scheme of a solar air conditioning plant. The conditioning plant considered is a hybrid system characterized by a variable configuration, with discrete and continuous variables, and components that change their dynamics according to the conditions under which the plant operates. Section 17.1 describes the solar air conditioning plant. Section 17.2 shows briefly the hybrid modeling of the plant. Section 17.3.1 describes the control requirements to operate the plant. Section 17.3.2 develops a hybrid control strategies for the operation of the plant. Section 17.3.3 shows the experimental results and discusses them.

17.1.1 Main components

The solar air conditioning plant considered in this chapter is located in Seville (Spain) and is used to cool down the laboratories of the Department of System Engineering and Automation of the University of Seville. It consists of a solar field producing hot water that feeds into an absorption machine, generating chilled water and injecting it into the air distribution system, which has a cooling power of 35 kW.

Figure 17.1 offers a general scheme of the plant, and shows its main components: the solar subsystem, composed of a set of flat solar collectors; the accumulation subsystem, composed of two tanks storing hot water; and the cooling machine. There also exist an auxiliary gas-fired heater that can supply energy would solar radiation not be enough, and a thermal load subsystem composed of a heat pump that allows testing for different load profiles.

The hybrid nature of the plant stems from the possible use of two different energy sources, namely solar and gas, which can be either combined or used independently. In addition, thermal energy coming from a storage tank can be added to the system. The plant can be reconfigured on-line, by simply manipulating valves and pumps (on/off) that allow selecting the required energy supply components. The main characteristic of the plant are described below.

Solar subsystem The primary source of energy is solar radiation, which is used by the solar collectors to increase the temperature of the circulating water. The solar field is composed of 151m^2 of flat collectors, which work in the range of 60°C to 100°C and supply a nominal power of 50 kW.

Accumulation subsystem The accumulation subsystem is composed of two tanks, each of 2500L, working in parallel. This system acts as a buffer, storing hot water, which is used in transient situations where solar radiation does not allow the desired temperature to be obtained at the end of the hot water circuit.

Auxiliary energy subsystem As a complement to the solar energy supplied by the field, an auxiliary energy system consisting of a gas-fired heater of 68 kW can be used when solar radiation is not enough. This heater transfers additional thermal

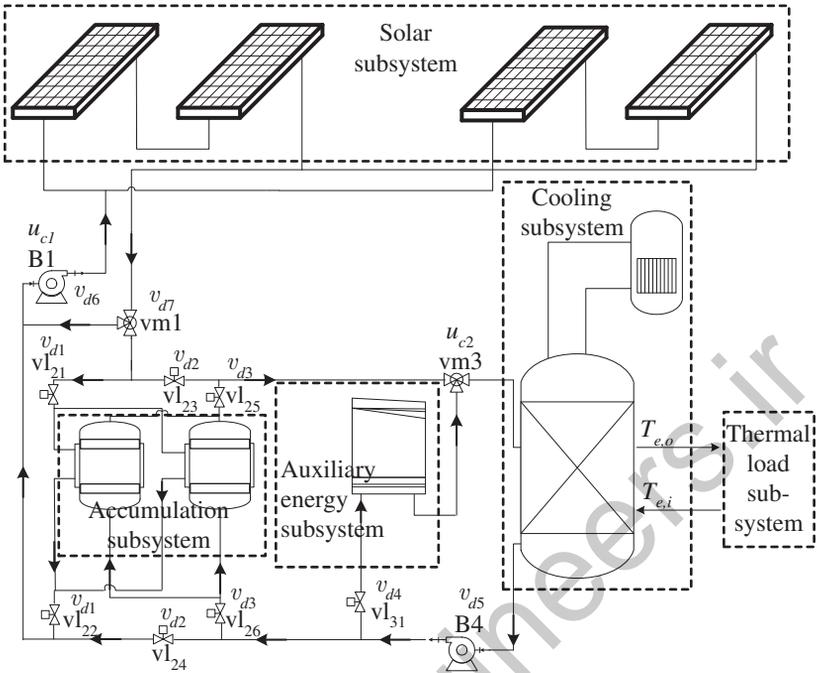


Fig. 17.1 Plant scheme.

energy to the water coming from the collectors. The existing heater has a built-in on/off controller, which makes its outlet temperature rather oscillatory.

Cooling subsystem The cooling subsystem is an absorption machine that uses as cooling fluids water and a water solution of lithium bromide ($H_2O-LiBr$). The correct operation of the machine requires its inlet temperature to be within a specific range, for chilled water to be produced. The machine has four different circuits: an evaporator, a generator, a condenser, and an absorber, where the energy exchanges for the production of chilled water take place. A refrigeration tower is used to evacuate the heat.

Thermal load subsystem This subsystem is composed by a heat exchanger and a heat pump of 54 kW. The subsystem has a PI controller that regulates the output temperature of the primary circuit of the heat exchanger. This value can be used to fix several input temperature profiles of the evaporator.

17.1.2 Signals for controlling the plant

The continuous variables are the pump speed of B1 (u_{c1}) and the mix valve position of vm3 (u_{c2}).

The logic manipulated variables are the open/close valves and ON/OFF pumps:

- v_{l21}/v_{l22} (v_{d1}) allows the connection of the solar collectors with the tanks;
- v_{l23}/v_{l24} (v_{d2}) allows the connection of the solar collectors with the absorption machine/gas heater;
- v_{l25}/v_{l26} (v_{d3}) allows the connection of the tanks with the absorption machine/gas heater;
- v_{l31} (v_{d4}) allows the circulation of water through the gas heater;
- B4 (v_{d5}) allows the pumping of water from the absorption machine (constant speed);
- B1 (v_{d6}) allows the pumping of water through solar collectors (variable speed);
- vm1 (v_{d7}) allows the circulation of water to the solar collectors or to the other components.

The main measured variables are the following:

Temperatures

- Solar collectors inlet and output temperatures.
- Tanks inlet and output temperatures.
- Gas heater inlet and output temperatures.
- Generator inlet and output temperatures of the absorption machine.
- Evaporator output temperature of the absorption machine.

Flows

- Flow through the solar collectors.
- Flow through the generator and condenser of the absorption machine.
- Flow through the gas heater.

The following measurable disturbances occur:

- solar irradiation;
- environmental temperature;
- flow through the evaporator of the absorption machine;
- evaporator input temperature of the absorption machine.

17.1.3 Operating modes of the process

In its daily operation, the plant may follow 13 operating modes. A logic variable " l_i " is associated to each operating mode, where $l_i = 1$ holds if the i -th operating mode is active. The operating modes are selected by means of on/off pumps and open/close valves:

1. *Recirculation*: The water of the tanks flows through the solar collectors, in order to increase its temperature.
2. *Loading the tanks with hot water*: The water in the solar collectors flows through the tanks. When too little solar irradiation is available, and if its temperature is adequate, the stored hot water is used.
3. *Using the solar collectors*: The water, heated in the solar collectors, feeds into the generator circuit of the absorption machine.
4. *Using the solar collectors and gas heater*: The water is heated in the solar collectors and the gas heater. The gas heater is used when the absorption machine input temperature is inadequate. In any case, its use should be avoided.
5. *Using the gas heater*: The water is heated in the gas heater, and then it flows through the absorption machine.
6. *Using the tanks and gas heater*: The water of the absorption machine is given by the tanks and gas heater. This operating mode is used when both the absorption machine inlet temperature and solar irradiation are lows. It is preferable to avoid having the gas heater too.
7. *Using tanks*: When solar radiation is low, the heat stored in the tanks is used to operate the absorption machine.
8. *Loading the tanks and using the gas heater*: The tanks are loaded with heated water by the solar collectors. The water of the absorption machine is given by the gas heater.
9. *Recirculation and using the gas heater*: The water is re-circulated through the solar collectors. It is the gas heater only that supplies water to the absorption machine.
10. *Using the solar collectors and loading tank*: The water from the solar collectors is divided between the tanks and the absorption machine.
11. *Using the solar collectors and gas heater, and loading tanks*: The water from the solar collectors is divided between the tanks and the absorption machine. The solar collectors and the gas heater supply water into the absorption machine.
12. *Using the tanks and gas heater to feed the absorption machine, and recirculation*: The water re-circulates into the solar collectors. The tanks and the gas heater supply water into the absorption machine.
13. *Using the tanks to feed the absorption machine, and recirculation*: The water from the tanks circulates through the generator circuit. The water circulates back into the solar collectors.

17.2 Plant model

The modeling of the plant is described in [678]. A finite-state machine has been used to represent the various operating modes of the process, as shown in [677]. The transition conditions are expressed as a function of the set of discrete manipulated variables \mathcal{V} , $v \in \mathbb{Z}$, as is shown in (17.1), to switch to the operating mode n_1 . The logic variables associated to each operating mode are used to model the output temperatures, and to select inlet temperatures and flows, and they must satisfy (17.2):

$$c1 = \neg v_{d1} \wedge \neg v_{d2} \wedge \neg v_{d3} \wedge \neg v_{d4} \wedge \neg v_{d5} \wedge v_{d6} \wedge \neg v_{d7}, \quad (17.1)$$

$$\sum_{i=1}^r l_i = 1. \quad (17.2)$$

The temperatures have been modeled using energy balances, as well as considering its behavior with and without the water flow. They are written as a mixed logical dynamical model. The general structure used is

$$C \frac{d}{dt} T_{j,\text{out}} = k_{1,j} M_j (T_{j,\text{in}} - T_{j,\text{out}}) l_i + k_{2,j} (1 - l_i) (T_{\text{env}} - T_{j,\text{out}}), \quad (17.3)$$

$$M_j = m_{1,j}(\mathbf{u}_c) l_1 + m_{2,j}(\mathbf{u}_c) l_2 + \dots + m_{r,j}(\mathbf{u}_c) l_r, \quad (17.4)$$

$$T_{j,\text{in}} = \tau_{1,j}(T_{j,\text{out}}, M_j) l_1 + \tau_{2,j}(T_{j,\text{out}}, M_j) l_2 + \dots + \tau_{r,j}(T_{j,\text{out}}, M_j) l_r,$$

where T is temperature, M is flow, C , k_1 and k_2 are parameters of the models, “in” and “out” sub-indexes denote input and output respectively, “env” refers to environmental temperature, j indicates the subsystem that is used (i.e. solar collectors, accumulators, gas heater, generator, and evaporator), and $\mathbf{u}_c = [u_{c1}, u_{c2}]$ is the continuous input vector, $\mathbf{u}_c \in \mathbb{R}$.

The flows are modeled as static polynomial functions m_i , $i = 1, \dots, r$ with $r = 13$ operating modes, where the inputs are the manipulated continuous variables (17.4). The inlet temperatures are selected according to the logic variable associated to each operating mode, and are calculated using the functions τ in (17.5).

17.3 Hybrid control

17.3.1 Control objectives

The main control objective is to supply chilled water to the air distribution system, according to the demanded temperature. The water gets cool when the absorption machine is working. Experimental tests suggested that the generator inlet temperature must be set at temperatures higher than 78°C and lower than 90°C. The water can be warmed up using the following energy sources: solar energy from the solar collectors, the stored energy in the tanks, the auxiliary energy provided by the gas heater, or by an optimal combination of the mentioned sources. The auxiliary energy (gas) consumption should be minimized, for environmental and economical reasons. Another control objective relates to the storage of energy in the tanks at the end day, given that the solar plant works daily.

Since the primary source of energy of the plant, i.e. solar radiation, cannot be manipulated, it has to be treated as a measurable disturbance. This implies that the control system must be able to keep the cooling machine working at the desired operating point. This is achieved by keeping the machine inlet water temperature at the given set-point.

A hybrid control strategy is needed to satisfy the control objectives of the plant. The operating mode must be set in such a way as to keep the absorption machine

working, and to preferably use solar energy. Doing so, it would achieve both its cooling effect and its economic saving purposes, since the auxiliary energy would not be used.

17.3.2 Hybrid control algorithm

Several renowned research groups participated in a HYCON's benchmark exercise carried out over a solar air conditioning plant. Each research group had direct access to many experimental tests oriented to both identification and control. Various proposals were put forward. The University of Dortmund presented a supervisory control scheme based on insights gained from a thorough analysis of the energetic and dynamical features of the system. The scheme allowed selecting among a set of operating modes, and each operating mode used a table to set the continuous manipulated variables. The University of Valladolid instead designed a hybrid control. This was based on a model-predictive control that incorporated an internal model with embedded logic control, used to transform the hybrid problem into a continuous-nonlinear one. The University of Siena worked on a hybrid model-predictive controller, whereas the University of Seville used a conventional flow chart to select the plant setting. Finally, the University of Los Andes designed a hierarchical control scheme based on an integer optimization problem, using variable weights that depended upon the current environmental conditions and the actual state of the plant.

A heuristic predictive logic controller (HPLoC) was designed in the proposal of the University of Los Andes, based on a model-predictive control (MPC) and on the knowledge of the process. The controller was divided into two levels. The operating mode of the solar plant was obtained using an HPLoC in the higher level. The continuous variables were controlled using an MPC for each operating mode in the lower level. These control schemes all took into account the disturbances in the prediction models.

HPLoC formulation The system has multiple operating modes n_i ($i = 1, \dots, r$) and is composed by D components, $D = \{d_1, d_2, \dots, d_p\}$. The configuration n_i of the D components must be selected by means of minimizing an objective function J in a prediction horizon N . An index $\delta(d_j)$ with $j = 1, \dots, p$ is defined for each component, and it depends on its main variables. In the case of the solar plant, they are expressed in function of the temperatures and the solar irradiation. For each operating mode n_i , a valuation function θ_{n_i} is defined depending on the indexes $\delta(d_j)$.

The problem to solve is an integer optimization problem. The objective function J is defined as shown in (17.5). The Θ vector has the θ_{n_i} elements, and \mathbf{W} is a weighting matrix. The decision vector \mathbf{n} is a binary vector, $\mathbf{n} \in \mathbb{Z}$. The problem is subject to a set of constraints meant to avoid overlapping between the operating modes.

$$J = \Theta^T \mathbf{W} \mathbf{n} \quad (17.5)$$

The optimization problem is defined as in (17.6):

$$\begin{aligned}
 \min_{\mathbf{n} \in \mathbb{Z}} J & & (17.6) \\
 \text{s.t. } \mathbf{A}_C \mathbf{n} & \leq \mathbf{b}_C,
 \end{aligned}$$

where \mathbf{n} is a vector of the logic variables associated to the operating modes, and \mathbf{A}_C and \mathbf{b}_C are matrices of adequate dimension that avoid the overlap between the various operating modes.

The weighting matrix \mathbf{W} is a diagonal matrix, and its elements are calculated using the analytic hierarchy process (AHP). The expert criteria are given by the first row of the Saaty matrix

$$\mathbf{A}(1, 2, \dots, r) = [a_{1,1} \quad a_{1,2} \quad \dots \quad a_{1,r}]. \quad (17.7)$$

The Saaty matrix has a particular structure, in that it is consistent and reciprocal. The maximum eigenvalue of the Saaty matrix, λ_{\max} , is calculated. Next, the eigenvector corresponding to the maximum eigenvalue is obtained, ω_i . The vector is normalized in order to avoid scaling problems.

The expert criteria a_i ,

$$a_{1,i} = g_i f_i + h_i (\neg f_i) \quad \text{for } i = 1, \dots, r, \quad (17.8)$$

$$h_i \ll g_i, \quad (17.9)$$

have been defined weighting the logic conditions f_i . They are defined by each operating mode, and the weighting g_i is selected according to the comparison between operating modes.

The logic conditions f_i indicate for each operating mode when that operating mode can be used, according to the current state of the environmental conditions and the temperatures. Therefore, f_i takes a binary value equal to 1 if the operating mode n_i can be used while satisfying the control requirements. Else f_i is equal to zero. As f_i takes a binary value, and since the $a_{1,1}$ element must be equal to 1, a h_i value is added when the logic condition f_i is zero. When f_i is zero, the expert criterion takes a lower value than when it is true, and therefore the h_i coefficient must be greater than g_i . The first row of the matrix \mathbf{A} is normalized using the value of the $a_{1,1}$ element before completing the other rows of the matrix \mathbf{A} .

17.3.3 Experimental results and discussion

The control algorithm has been programmed in *MATLAB*. An *OPC (OLE for Process Control)* server has been used for the implementation of the controller over the experimental plant. A server-client scheme was implemented between the *SCADA* system of the plant (*Simatic IT*) and the *MATLAB* environment. The sample time was 4 seconds, which allowed for a quick update of the measured disturbances. One advantage of the algorithm implemented is its short execution time, due to its being an integer optimization problem and its reduced number of variables.

Figure 17.2 shows the settings of the plant during the test. The cooling demand began at 11:00. Before this time, the water circulated through the various circuits to raise the temperature of pipelines. As the solar irradiation was lower than 200 W/m^2 , the plant was set to follow those operating modes using auxiliary energy (n_9 and n_{12}).

When the solar irradiation was high, at about 12:30, the plant was set for a short period to operate according to mode n_4 . Since the solar irradiation was lower than 400 W/m^2 , the plant was set to use auxiliary energy again.

After 13:30, the sky got quickly cloudy, and therefore the plant was set to follow those operating modes using solar energy (n_{10} , for example). At end of the test, the sky got cleared up and the plant was set again to using solar energy only and energy stored in the accumulators. When the cooling demand ended at 17:30, the plant was set to switch between operating modes n_1 and n_2 . Figure 17.3 shows the evaporator output and inlet temperature. The evaporator inlet temperature was near of 18°C , and the chilled water temperature around 15°C . Therefore, the cooling demand was satisfied despite of the adverse weather conditions.

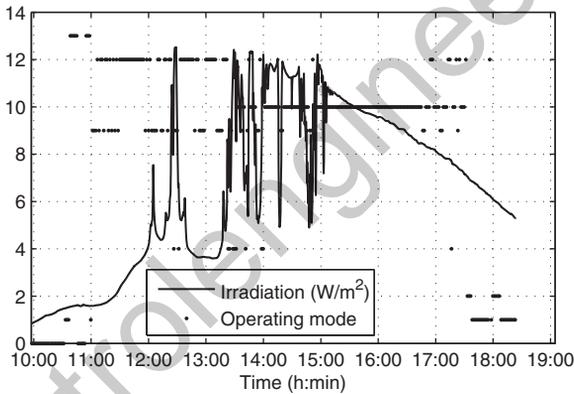


Fig. 17.2 Solar irradiation and operating mode of the plant during the test.

Bibliographical notes

Analytic hierarchy process (AHP) is a technique that allows the criteria weights to be found by means of pairwise comparisons. The decision-maker is asked to give the criteria relative importance by comparing them two by two [562].

The phrase model-predictive control (MPC) designates a range of control methods. These make explicit use of a model of the process in order to obtain the control signal by minimizing an objective function [140].

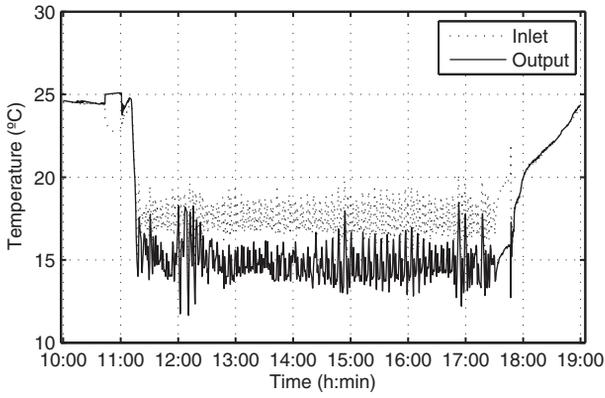


Fig. 17.3 Evaporator inlet and output temperatures.

controlengineers.ir

References

- [1] A. Abate, M. Prandini, L. Lygeros, and S. Sastry. Probabilistic reachability and safety for controlled discrete time stochastic hybrid systems. *Automatica*, **44**(11):2724–2734, 2008.
- [2] A. Abate, M. Prandini, L. Lygeros, and S. Sastry. Approximation of general stochastic hybrid systems by switching diffusions with random hybrid jumps. In M. Egerstedt and B. Mishra, eds., *Hybrid Systems: Computation and Control*, pp. 4–17. Springer-Verlag, Berlin, 2008.
- [3] V. Acary and B. Brogliato. *Numerical Methods for Nonsmooth Dynamical Systems. Applications in Mechanics and Electronics*. Springer-Verlag, Berlin, 2008.
- [4] A. A. Agrachev and D. Liberzon. Lie-algebraic stability criteria for switched systems. *SIAM J. Control Optim.*, **40**(1):253–269, 2001.
- [5] A. A. Agrachev and Y. L. Sachkov. *Control Theory from the Geometric Viewpoint*. Springer-Verlag, Berlin, 2004.
- [6] M. A. Aizerman and E. S. Pyatnitskii. Fundamentals of the theory of discontinuous dynamical systems. I, II. *Automatika i telemekhanika*, **7**: 33–47 and **8**: 39–61, 1974 (in Russian).
- [7] J. Akesson. Languages and tools for optimization of large-scale systems. Unpublished PhD thesis, Lund, Sweden, Department of Automatic Control, Lund University, 2007.
- [8] J. Akesson. Optimica—an extension of modelica supporting dynamic optimization. In *Proc. 6th Int. Modelica Conf.*, pp. 57–66, Bielefeld, Germany, 2008.
- [9] A. Alessandri and P. Coletta. Design of Luenberger observers for a class of hybrid linear systems. In M. D. Di Benedetto and A. L. Sangiovanni-Vincentelli, eds., *Hybrid Systems: Computation and Control*, pp. 7–18. Springer-Verlag, Berlin, 2001.
- [10] A. Alessio and A. Bemporad. Feasible mode enumeration and cost comparison for explicit quadratic model predictive control of hybrid systems. In *2nd IFAC Conf. on Analysis and Design of Hybrid Systems*, pp. 302–308, Alghero, Italy, 2006.

- [11] A. Alessio and A. Bemporad. A survey on explicit model predictive control. In *Proc. Int. Workshop on Assessment and Future Directions of Nonlinear Model Predictive Control*, Pavia, Italy, 2008.
- [12] E. Allen Emerson. Temporal and modal logic. In *Handbook of Theoretical Computer Science, volume B: Formal Models and Semantics*, pp. 995–1072. Elsevier, Amsterdam, 1990.
- [13] R. Alur and D. L. Dill. A theory of timed automata. *Theor. Comp. Sci.*, **126**(2): 183–235, 1994.
- [14] R. Alur and T. A. Henzinger. Real-time logics: complexity and expressiveness. *Information and Computation*, **104**(1):35–77, 1993.
- [15] R. Alur, C. Courcoubetis, T. A. Henzinger, and P. H. Ho. Hybrid automata: an algorithmic approach to the specification and verification of hybrid systems. In *Hybrid Systems*, pp. 209–229. Springer-Verlag, Berlin, 1993.
- [16] R. Alur, C. Courcoubetis, N. Halbwachs, *et al.* The algorithmic analysis of hybrid systems. *Theor. Comp. Sci.*, **138**(1):3–34, 1995.
- [17] R. Alur, T. Henzinger, G. Lafferriere, and G. J. Pappas. Discrete abstractions of hybrid systems. *Proc. IEEE*, **88**(2):971–984, 2000.
- [18] E. Amaldi and M. Mattavelli. The MIN PFS problem and piecewise linear model estimation. *Discrete Appl. Math.*, **118**(1–2):115–143, 2002.
- [19] A. D. Ames, A. Abate, and S. Sastry. Sufficient conditions for the existence of Zeno behavior. In *Proc. 44th IEEE Conf. Decision and Control*, pp. 696–701, Seville, Spain, 2005.
- [20] M. Andersson. Object-oriented modeling and simulation of hybrid systems. Unpublished PhD thesis, Lund Institute of Technology, Lund, Sweden, 1994.
- [21] M. Antoniotti, A. Balluchi, L. Benvenuti, *et al.* A top-down constraints-driven design methodology for powertrain control systems. In *Proc. Global Powertrain Congress: Emissions, Testing and Controls*, pp. 74–84, Detroit, MI, 1998.
- [22] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Trans. Signal Processing*, **50**(2):174–188, 2002.
- [23] D. Arzelier, D. Peaucelle, C. Farges, and J. Daafouz. Robust analysis and synthesis of linear polytopic discrete-time periodic systems via LMIs. In *Proc. 44th IEEE Conf. Decision and Control*, Seville, Spain, 2005.
- [24] K. J. Aström and B. Wittenmark. *Computer Controlled Systems*. Prentice-Hall, Englewood Cliffs, NJ, 1997.
- [25] J. P. Aubin and A. Cellina. *Differential Inclusions*. Springer-Verlag, New-York, 1984.
- [26] J. P. Aubin and A. Cellina. *Differential Inclusions: Set-Valued Maps and Viability Theory*. Springer-Verlag, Berlin, 1984.
- [27] J. P. Aubin, J. Lygeros, M. Quincampoix, S. Sastry, and N. Seube. Impulse differential inclusions: a viability approach to hybrid systems. *IEEE Trans. Automat. Contr.*, **47**(1):2–20, 2002.

- [28] M. P. Avraam, N. Shah, and C. C. Pantelides. Modelling and optimisation of general hybrid systems in the continuous time domain. *Comp. Chem. Eng.*, **22**(1):221–228, 1998.
- [29] S. I. Azuma, M. Egerstedt, and Y. Wardi. Output-based optimal timing control of switched systems. In J. Hespanha, ed., *Hybrid Systems: Computation and Control*, pp. 64–78. Springer-Verlag, Santa Barbara, CA, 2006.
- [30] M. Babaali and M. Egerstedt. Observability for switched linear systems. In O. Maler and A. Pnueli, eds., *Hybrid Systems: Computation and Control*. Springer-Verlag, Berlin, 2004.
- [31] M. Babaali and G. J. Pappas. Observability of switched linear systems in continuous time. In M. Morari and L. Thiele, eds., *Hybrid Systems: Computation and Control*, pp. 103–117. Springer-Verlag, Berlin, 2005.
- [32] J. Baillieul. Feedback designs in information-based control. In *Proc. Workshop Stochastic Theory and Control*, pp. 35–57, Lawrence, Kansas, 2001.
- [33] A. Balluchi, M. D. Di Benedetto, C. Pinello, C. Rossi, and A. L. Sangiovanni-Vincentelli. Hybrid control in automotive applications: the cut-off control. *Automatica*, **35**(3):519–535, 1999.
- [34] A. Balluchi, L. Benvenuti, M. D. Di Benedetto, C. Pinello, and A. L. Sangiovanni-Vincentelli. Automotive engine control and hybrid systems: challenges and opportunities. *Proc. IEEE*, **88**(7):888–912, 2000.
- [35] A. Balluchi, L. Benvenuti, M. D. Di Benedetto, and A. L. Sangiovanni-Vincentelli. Design of observers for hybrid systems. In C. J. Tomlin and J. R. Greenstreet, eds., *Hybrid Systems: Computation and Control*, pp. 76–89. Springer-Verlag, Stanford, CA, 2002.
- [36] A. Balluchi, L. Benvenuti, M. D. Di Benedetto, and A. L. Sangiovanni-Vincentelli. Observability for hybrid systems. In *Proc. 42th IEEE Conf. Decision and Control*, pp. 1159–1164, Maui, USA, 2003.
- [37] A. Balluchi, L. Benvenuti, C. Lemma, P. Murrieri, and A. L. Sangiovanni-Vincentelli. Hybrid models of an automotive driveline. Tech. report, PARADES, Rome, Italy, 2004.
- [38] A. Balluchi, L. Benvenuti, C. Lemma, A. L. Sangiovanni-Vincentelli, and G. Serra. Actual engaged gear identification: a hybrid observer approach. In *Proc. 16th IFAC World Congress*, Prague, Czech Republic, 2005.
- [39] M. Baotic. An efficient algorithm for multi-parametric quadratic programming. Technical Report AUT02-04, Automatic Control Laboratory, ETH Zurich, Switzerland, 2002.
- [40] M. Baotic, M. Vasak, M. Morari, and N. Peric. Hybrid theory based optimal control of electronic throttle. In *Proc. 2003 American Control Conf.*, pp. 5209–5214, Denver, CO, 2003.
- [41] G. I. Bara, J. Daafouz, F. Kratz, and J. Ragot. Parameter dependent state observer design for affine LPV systems. *Int. J. Control*, **74**(16):1601–1611, 2001.
- [42] T. P. Barnett and D. W. Pierce. When will Lake Mead go dry? *J. Water Resour. Res.*, **44**, 2008.

- [43] P. I. Barton. The modelling and simulation of combined discrete/continuous processes. Unpublished PhD thesis, Imperial College of Science, Technology and Medicine, 1992.
- [44] P. I. Barton and C. K. Lee. Modeling, simulation, sensitivity analysis, and optimization of hybrid systems. *ACM Trans. Model. Comp. Simul.*, **12**(4): 256–289, 2002.
- [45] P. I. Barton, R. J. Allgor, W. F. Feehery, and S. Galan. Dynamic optimization in a discontinuous world. *Ind. Eng. Chem. Res.*, **37**(3):966–981, 1998.
- [46] P. I. Barton, C. K. Lee, and M. Yunt. Optimization of Hybrid Systems. *Comp. Chem. Eng.*, **30**(10–12):1576–1589, 2006.
- [47] G. Basile and G. Marro. *Controlled and Conditioned Invariants in Linear System Theory*. Prentice Hall, Englewood Cliffs, NJ, 1992.
- [48] A. G. Beccuti, G. Papafotiou, R. Frasca, and M. Morari. Explicit hybrid model predictive control of the dc-dc boost converter. In *Proc. IEEE Power Electronics Specialist Conf.*, Orlando, FL, 2007.
- [49] A. G. Beccuti, G. Papafotiou, M. Morari, *et al.* Hybrid control techniques for switched-mode DC-DC Converters, part ii: The step-up topology. In *Proc. 2007 American Control Conf.*, New York, 2007.
- [50] D. A. van Beek, K. L. Man, M. A. Reniers, J. E. Rooda, and R. R. H. Schiffelers. Syntax and consistent equation semantics of hybrid chi. *J. Logic Algebraic Program.*, **68**(1–2):129–210, 2006.
- [51] D. A. van Beek, M. A. Reniers, J. E. Rooda, and R. R. H. Schiffelers. Foundations of an interchange format for hybrid systems. In A. Bemporad, A. Bicchi, and G. Butazzo, eds., *Hybrid Systems: Computation and Control*, pp. 587–600. Springer-Verlag, Pisa, Italy, 2007.
- [52] D. A. van Beek, M. A. Reniers, J. E. Rooda, and R. R. H. Schiffelers. Revised hybrid system interchange format. Technical Report HYCON Deliverable D3.6.3, HYCON NoE, 2007.
- [53] D. A. van Beek, M. A. Reniers, J. E. Rooda, and R. R. H. Schiffelers. Concrete syntax and semantics of the compositional interchange format for hybrid systems. In *Proc. 17th IFAC World Congress*, Seoul, Korea, 2008.
- [54] G. Behrmann, K. G. Larsen, J. Pearson, C. Weise, and W. Yi. Efficient timed reachability analysis using clock difference diagrams. In *Computer Aided Verification*, pp. 341–353. Springer-Verlag, Berlin, 1999.
- [55] A. Bemporad. An efficient technique for translating mixed logical dynamical systems into piecewise affine systems. In *Proc. 41th IEEE Conf. Decision and Control*, pp. 1970–1975, Las Vegas, NV, 2002.
- [56] A. Bemporad. Efficient conversion of mixed logical dynamical systems into an equivalent piecewise affine form. *IEEE Trans. Automat. Contr.*, **49**(5): 832–838, 2004.
- [57] A. Bemporad. Hybrid toolbox—user’s guide, 2004. Available at: <http://www.dii.unisi.it/hybrid/toolbox>.
- [58] A. Bemporad. Model-based predictive control design: new trends and tools. In *Proc. 45th IEEE Conf. Decision and Control*, pp. 6678–6683, San Diego, CA, 2006.

- [59] A. Bemporad and S. Di Cairano. Optimal control of discrete hybrid stochastic automata. In M. Morari and L. Thiele, eds., *Hybrid Systems: Computation and Control*, pp. 151–167. Springer-Verlag, Berlin, 2005.
- [60] A. Bemporad and N. Giorgetti. Logic-based methods for optimal control of hybrid systems. *IEEE Trans. Automat. Contr.*, **51**(6):963–976, 2006.
- [61] A. Bemporad and D. Mignone. *MIQP.M: A MATLAB function for solving mixed integer quadratic programs*, 2000. Available at: <http://www.dii.unisi.it/~hybrid/tools/miqp>.
- [62] A. Bemporad and M. Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, **35**(3):407–427, 1999.
- [63] A. Bemporad, F. Borrelli, and M. Morari. Piecewise linear optimal controllers for hybrid systems. In *Proc. 2000 American Control Conf.*, pp. 1190–1194, Chicago, IL, 2000.
- [64] A. Bemporad, F. Borrelli, and M. Morari. Explicit solution of LP-based model predictive control. In *Proc. 39th IEEE Conf. Decision and Control*, Sydney, Australia, 2000.
- [65] A. Bemporad, G. Ferrari-Trecate, and M. Morari. Observability and controllability of piecewise affine and hybrid systems. *IEEE Trans. Automat. Contr.*, **45**(10):1864–1876, 2000.
- [66] A. Bemporad, F. D. Torrisi, and M. Morari. Discrete-time hybrid modeling and verification of the batch evaporator process benchmark. *Eur. J. Control*, **7**(4):382–399, 2001.
- [67] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, **38**(1):3–20, 2002.
- [68] A. Bemporad, F. Borrelli, and M. Morari. Min-max control of constrained uncertain discrete-time linear systems. *IEEE Trans. Automat. Contr.*, **48**(9):1600–1606, 2003.
- [69] A. Bemporad, M. Morari, and N. L. Ricker. *Model Predictive Control Toolbox for MATLAB—User’s Guide*. The Mathworks, Inc., 2004. Available at: <http://www.mathworks.com/access/helpdesk/help/toolbox/mpc/>.
- [70] A. Bemporad, A. Garulli, S. Paoletti, and A. Vicino. A bounded-error approach to piecewise affine system identification. *IEEE Trans. Automat. Contr.*, **50**(10):1567–1580, 2005.
- [71] A. Bemporad, S. Di Cairano, and J. Júlvez. Event-based model predictive control and verification of integral continuous-time hybrid automata. In J. P. Hespanha and A. Tiwari, eds., *Hybrid Systems: Computation and Control*, pp. 93–107. Springer-Verlag, Santa Barbara, CA, 2006.
- [72] A. Bemporad, S. Di Cairano, I. V. Kolmanovsky, and D. Hrovat. Hybrid modeling and control of a multibody magnetic actuator for automotive applications. In *Proc. 46th IEEE Conf. Decision and Control*, pp. 5270–5275, New Orleans, LA, 2007.
- [73] J. Bengtsson and W. Yi. Timed automata: semantics, algorithms and tools. In J. Desel, W. Reisig, and G. Rozenberg, eds., *Lectures on Concurrency and Petri Nets*, pp. 87–124. Springer-Verlag, Berlin, 2004.

- [74] J. Bengtsson, P. Strandh, R. Johansson, P. Tunestål, and B. Johansson. Hybrid modeling of homogeneous charge compression ignition (HCCI) engine dynamics. *Int. J. Control*, **80**(11):1814–1848, 2007.
- [75] K. P. Bennett and O. L. Mangasarian. Multicategory discrimination via linear programming. *Optim. Methods Software*, **3**:27–39, 1994.
- [76] A. Bensoussan and J. L. Menaldi. Stochastic hybrid control. *J. Math. Anal. Appl.*, **249**(1):261–288, 2000.
- [77] M. A. Berger and Y. Wang. Bounded semigroups of matrices. *Linear Algebr. Appl.*, **166**:21–27, 1992.
- [78] A. Bernard and A. El Kharroubi. Regulations de processus dans le premier ‘orthant’ de \mathbb{R}^n . *Comptes Rendus Acad. Sci. Paris Ser. I: Math.*, **309**:371–375, 1989.
- [79] M. Di Bernardo, C. Budd, A. R. Champneys, *et al.* Bifurcations in nonsmooth dynamical systems. *SIAM Rev.*, **50**(4): 629–701, 2008.
- [80] J. Bernussou and A. Titli. *Interconnected Dynamical Systems: Stability, Decomposition and Decentralisation*. North-Holland Publishing Company, Amsterdam, The Netherlands, 1982.
- [81] D. P. Bertsekas and S. E. Shreve. *Stochastic Optimal Control: The Discrete Time case*. Athena Scientific, Belmont, MA, 1996.
- [82] L. Biegler and I. Grossmann. Retrospective on optimization. *Comp. Chem. Eng.*, **28**:1169–1192, 2004.
- [83] E. Y. Bitar, H. J. Schock, and A. K. Oppenheim. Model for control of combustion in a piston engine. *SAE Technical Papers*, p/n 2006-01-0401, 2006.
- [84] M. Blanke, M. Kinnaert, J. Lunze, and M. Staroswiecki. *Diagnosis and Fault-Tolerant Control*. Springer-Verlag, Heidelberg, 2006.
- [85] D. Blom, M. Karlsson, K. Ekholm, P. Tunestal, and R. Johansson. HCCI engine modeling and control using conservation principles. *SAE Technical Papers 2008-01-0789*, 2008. Also in *Homogeneous Charge Compression Ignition (HCCI) Combustion*, SAE Special Publication SAE SP-2182, SAE, Warrendale, PA, 2008.
- [86] H. A. P. Blom and J. Lygeros (eds.). *Stochastic Hybrid Systems: Theory and Safety Critical Applications*. Springer-Verlag, Berlin, 2006.
- [87] V. D. Blondel and J. N. Tsitsiklis. NP-hardness of some linear control design problems. *SIAM J. Control Optim.*, **35**(6):2118–2127, 1997.
- [88] V. D. Blondel and V. Canterini. Undecidable problems for probabilistic automata of fixed dimension. *Theor. Comp. Syst.*, **36**(3):231–245, 2003.
- [89] V. D. Blondel and Y. Nesterov. Computationally efficient approximations of the joint spectral radius. *SIAM J. Matrix Anal.*, **27**(1):256–272, 2005.
- [90] V. D. Blondel and J. N. Tsitsiklis. The boundedness of all products of a pair of matrices is undecidable. *Syst. Control Lett.*, **41**(2):135–140, 2000.
- [91] V. D. Blondel and J. N. Tsitsiklis. A survey of computational complexity results in systems and control. *Automatica*, **36**(9):1249–1274, 2000.
- [92] V. D. Blondel and J. N. Tsitsiklis. Complexity of stability and controllability of elementary hybrid systems. *Automatica*, **35**(3):479–490, 1999.

- [93] V. D. Blondel, O. Bournez, P. Koiran, and J. N. Tsitsiklis. The stability of saturated linear dynamical systems is undecidable. *J. Comp. Syst. Sci.*, **62**(3): 442–462, 2001.
- [94] V. D. Blondel, J. Theys, and A. A. Vladimirov. An elementary counterexample to the finiteness conjecture. *SIAM J. Matrix Anal. Appl.*, **24**(4):963–970, 2003.
- [95] V. D. Blondel, R. Jungers, and V. Protasov. On the complexity of computing the capacity of codes that avoid forbidden difference patterns. *IEEE Trans. Inf. Theor.*, **52**(11):5122–5127, 2006.
- [96] M. Boccadoro, Y. Wardi, M. Egerstedt, and E. Verriest. Optimal control of switching surfaces in hybrid systems. *Discrete Event Dyn. Syst.*, **15**(4): 433–448, 2005.
- [97] H. C. Bohnenkamp, P. R. D’Argenio, H. Hermanns, and J. P. Katoen. MoDeST: a compositional modeling formalism for real-time and stochastic systems. Technical Report TR-CTIT-04-46, University of Twente, Centre for Telematics and Information Technology, 2004.
- [98] G. Böker and J. Lunze. Stability and performance of switching Kalman filters. *Int. J. Control*, **75**(16–17):1269–1281, 2002.
- [99] T. Bolognesi and F. Lucidi. Timed process algebras with urgent interactions and a unique powerful binary operator. In J. W. de Bakker, C. Huizing, W. P. de Roever, and G. Rozenberg, eds., *Real-Time: Theory in Practice*, pp. 124–148. Springer-Verlag, Mook, The Netherlands, 1991.
- [100] P. Bolzern and W. Spinelli. Quadratic stabilization of a switched affine system about a nonequilibrium point. In *Proc. 2004 American Control Conf.*, Boston, MA, 2004.
- [101] S. Bornot and J. Sifakis. An algebraic framework for urgency. *Inform. Comp.*, **163**(1):172–202, 2000.
- [102] F. Borrelli. Discrete time constrained optimal control. Unpublished PhD thesis, Swiss Federal Institute of Technology, ETH, Zurich, 2002.
- [103] F. Borrelli. *Constrained Optimal Control of Linear and Hybrid Systems*, Springer-Verlag, Berlin, 2003.
- [104] F. Borrelli, M. Baotić, A. Bemporad, and M. Morari. An efficient algorithm for computing the state feedback optimal control law for discrete time hybrid systems. In *Proc. 2003 American Control Conf.*, Denver, CO, 2003.
- [105] F. Borrelli, M. Baotić, A. Bemporad, and M. Morari. Dynamic programming for constrained optimal control of discrete-time linear hybrid systems. *Automatica*, **41**(10):1709–1721, 2005.
- [106] F. Borrelli, A. Bemporad, M. Fodor, and D. Hrovat. An MPC/hybrid system approach to traction control. *IEEE Trans. Control Syst. Technol.*, **14**(3): 541–552, 2006.
- [107] W. Borutzky. *Bond Graphs—A Methodology for Modelling Multidisciplinary Dynamic Systems*. SCS Publishing House, 2004.
- [108] T. Bousch and J. Mairesse. Asymptotic height optimization for topical IFS, Tetris heaps and the finiteness conjecture. *J. Amer. Math. Soc.*, **15**:77–111, 2002.

- [109] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. *Linear Matrix Inequalities in System and Control Theory*. SIAM, Philadelphia, PA, 1994.
- [110] B. A. Brandin and W. M. Wonham. Supervisory control of timed discrete-event systems. *IEEE Trans. Automat. Contr.*, **39**(2):329–342, 1994.
- [111] U. Brandt-Pollmann. Numerical solution of optimal control problems with implicitly defined discontinuities with applications in engineering. Unpublished PhD thesis, Ruprecht-Karls-Universität, Heidelberg, Germany, 2004.
- [112] M. S. Branicky. Introduction to hybrid systems. In D. Hristu-Varsakelis and W. S. Levine, eds., *Handbook of Networked and Embedded Control Systems*, pp. 91–116. Birkhäuser, Boston, MA, 2005.
- [113] M. S. Branicky. Multiple Lyapunov theory and other analysis tools for switched and hybrid systems. *IEEE Trans. Automat. Control*, **43**(4):475–482, 1998.
- [114] M. S. Branicky and S. E. Mattsson. Simulation of hybrid systems in Omola/Omsim. In *Proc. 7th IFAC Symp. Computer Aided Control Systems Design*, Gent, Belgium, pp. 3–8, 1997.
- [115] M. S. Branicky, V. S. Borkar, and S. K. Mitter. A Unified Framework for Hybrid Control: Model and Optimal Control theory. *IEEE Trans. Automat. Control*, **43**(1):31–45, 1998.
- [116] M. S. Branicky, S. Phillips, and W. Zhang. Stability of networked control systems: Explicit analysis of delay. *Proc. 2000 American Control Conf.*, Chicago, IL, pp. 2352–2357, 2000.
- [117] E. J. Bredensteiner and K. P. Bennett. Multicategory Classification by Support Vector Machines. *Comput. Optim. Appl.*, **12**(1–3):53–79, 1999.
- [118] A. Bressan and B. Piccoli. *Introduction to the Mathematical Theory of Control*, vol. 2. AIMS, Springfield, USA, 2007.
- [119] R. Brockett and D. Liberzon. Quantized feedback stabilization of linear systems. *IEEE Trans. Automat. Contr.*, **45**(7):1279–1289, 2000.
- [120] R. W. Brockett. Hybrid models for motion control systems. In H. L. Trentelman and J. C. Willems, eds., *Progress in Systems and Control Theory, Essays on Control: Perspectives in the Theory and its Applications*, vol. 14, pp. 29–53. Birkhäuser, Boston, 1993.
- [121] J. F. Broenink. 20-sim software for hierarchical bond-graph/block-diagram models. *Simul. Pract. Theor.*, **7**:481–492, 1999.
- [122] B. Brogliato. Some perspectives on the analysis and control of complementarity systems. *IEEE Trans. Automat. Control*, **48**(6):918–935, 2003.
- [123] B. Brogliato. *Nonsmooth Impact Mechanics. Models, Dynamics and Control*. Springer-Verlag, London, 1996.
- [124] B. Brogliato, A. Daniilidis, C. Lemaréchal, and V. Acary. On the equivalence between complementarity systems, projected systems and differential inclusions. *Syst. Control Lett.* **55**(1):45–51, 2006.
- [125] C. Brooks, A. Cataldo, E. A. Lee, J. Liu, S. Neuendorffer, and H. Zheng. HyVisual: a hybrid system visual modeler. Technical Memorandum UCB/ERL M05/24, EECS Department, University of California, Berkeley, CA, 2005.

- [126] M. Broucke, M. D. Di Benedetto, S. Di Gennaro, and A. L. Sangiovanni-Vincentelli. Theory of Optimal Control Using Bisimulations. In N. Lynch and B. H. Krogh, eds., *Hybrid Systems: Computation and Control*, pp. 89–102. Springer-Verlag, Berlin, 2000.
- [127] J. Buisson, H. Cormerais, and P. Y. Richard. Analysis of the bond graph model of hybrid physical systems with ideal switches. *J. Syst. Control Eng.*, **216**(11): 47–72, 2002.
- [128] J. Buisson, H. Cormerais, and P. Y. Richard. On the stabilization of switching electrical power converters. In *Hybrid Systems: Computation and Control*, Zürich, Switzerland, 2005.
- [129] M. Bujorianu and J. Lygeros. Toward a general theory of stochastic hybrid systems. In H. A. P. Blom and J. Lygeros, eds., *Stochastic Hybrid Systems: Theory and Safety Applications*, pp. 3–30. Springer-Verlag, Berlin, 2006.
- [130] M. L. Bujorianu. Extended stochastic hybrid systems and their reachability problem. In R. Alur and G. Pappas, eds., *Hybrid Systems: Computation and Control*, pp. 234–249. Springer-Verlag, Berlin, 2004.
- [131] M. L. Bujorianu and J. Lygeros. Reachability questions in piecewise deterministic markov processes. In O. Maler and A. Pnueli, eds., *Hybrid Systems: Computation and Control*, pp. 126–140. Springer-Verlag, Berlin, 2003.
- [132] F. Bullo and D. Liberzon. Quantized control via locational optimization. *IEEE Trans. Automat. Control*, **51**(1):2–13, 2006.
- [133] C. Cai, A. R. Teel, and R. Goebel. Smooth Lyapunov functions for hybrid systems, part i: existence is equivalent to robustness. *IEEE Trans. Automat. Control*, **52**(7):1264–1277, 2007.
- [134] P. E. Caines and Y. J. Wei. Hierarchical hybrid control systems: a lattice theoretic formulation. *IEEE Trans. Automat. Control*, **43**(4):501–508, 1998.
- [135] P. E. Caines, F. H. Clarke, X. Lui, and R. B. Vinter. A maximum principle for hybrid optimal control problems with pathwise state constraints. *Proc. 45th IEEE Conf. Decision and Control*, San Diego, CA, pp. 4821–4825, 2006.
- [136] S. Di Cairano and A. Bemporad. An equivalence results between linear hybrid automata and piecewise affine systems. In *Proc. 45th IEEE Conf. Decision and Control*, pp. 2631–2636, San Diego, CA, 2006.
- [137] S. Di Cairano, A. Bemporad, I. Kolmanovsky, and D. Hrovat. Model predictive control of magnetically actuated mass spring dampers for automotive applications. *Int. J. Control*, **80**(11):1701–1716, 2007.
- [138] S. Di Cairano, M. Lazar, A. Bemporad, and W. P. M. H. Heemels. A control Lyapunov approach to predictive control of hybrid systems. In M. Egerstedt and B. Mishra, eds., *Hybrid Systems: Computation and Control*, pp. 130–143. Springer-Verlag, Berlin, 2008.
- [139] F. M. Callier and C. A. Desoer. *Linear System Theory*. Springer-Verlag, New York, 1991.
- [140] E. F. Camacho and C. Bordons. *Model Predictive Control*, 2nd edn. Springer-Verlag, London, 2004.
- [141] S. L. Campbell and C. W. Gear. The index of general nonlinear DAEs. *Numerische Mathematik*, **72**(2):173–196, 1994.

- [142] S. L. Campbell, J. P. Chancelier, and R. Nikoukhah. *Modeling and Simulation in Scilab/Scicos*. Springer-Verlag, Berlin, 2005.
- [143] L. P. Carloni, R. Passerone, A. Pinto, and A. L. Sangiovanni-Vincentelli. Languages and tools for hybrid systems design. *Foundations and Trends in Electronic Design Automation*, **1**(1–2):1–193, 2006.
- [144] C. G. Cassandras. *Discrete Event Systems: Modeling and Performance Analysis*. Richard D. Irwin, Inc., Burr Ridge, IL, 1993.
- [145] C. G. Cassandras and J. Lygeros (eds.). *Stochastic Hybrid Systems*. Taylor & Francis CRC Press, Boca Raton, FL, 2007.
- [146] C. G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*. Kluwer, Dordrecht, 1999.
- [147] C. G. Cassandras, D. L. Pepyne, and Y. Wardi. Optimal control of a class of hybrid systems. *IEEE Trans. Automat. Control*, **46**(3):398–415, 2001.
- [148] C. G. Cassandras, M. I. Clune, and P. J. Mosterman. Hybrid System Simulation with SimEvents. In *Proc. 2nd IFAC Conf. on Analysis and Design of Hybrid Systems*, Alghero, Italy, pp. 136–141, 2006.
- [149] M. K. Çamlıbel. Complementarity methods in the analysis of piecewise linear dynamical systems. Unpublished PhD thesis, Dissertation Series of Center for Economic Research, Tilburg University, The Netherlands, 2001.
- [150] M. K. Çamlıbel. Popov–Belevitch–Hautus type tests for the controllability of linear complementarity systems. *Syst. Control Lett.*, **56**(5):381–387, 2007.
- [151] M. K. Çamlıbel and J. M. Schumacher. Existence and uniqueness of solutions for a class of piecewise linear dynamical systems. *Linear Algebr. Appl.*, **351**(2):147–184, 2002.
- [152] M. K. Çamlıbel and J. M. Schumacher. On the Zeno behavior of linear complementarity systems. In *Proc. 40th IEEE Conf. Decision and Control*, pp. 346–351, Orlando, FL, 2001.
- [153] M. K. Çamlıbel, W. P. M. H. Heemels, and J. M. Schumacher. Well-posedness of a class of linear networks with ideal diodes. In *Proc. 14th Int. Symp. Mathematical Theory of Networks and Systems*, Perpignan, France, 2000.
- [154] M. K. Çamlıbel, W. P. M. H. Heemels, and J. M. Schumacher. Consistency of a time-stepping method for a class of piecewise linear networks. *IEEE Trans. Circuits Syst. I*, **49**(3):349–357, 2002.
- [155] M. K. Çamlıbel, W. P. M. H. Heemels, and J. M. Schumacher. On linear passive complementarity systems. *Eur. J. Control*, **8**(3):220–237, 2002.
- [156] M. K. Çamlıbel, W. P. M. H. Heemels, and J. M. Schumacher. Stability and controllability of planar bimodal complementarity systems. In *Proc. 42th IEEE Conf. Decision and Control*, Maui, USA, 2003.
- [157] M. K. Çamlıbel, W. P. M. H. Heemels, A. J. van der Schaft, and J. M. Schumacher. Switched networks and complementarity. *IEEE Trans. Circuits Syst. I*, **50**(8):1036–1046, 2003.
- [158] M. K. Çamlıbel, L. Iannelli, and F. Vasca. Modelling switching power converters as complementarity systems. In *Proc. 43th IEEE Conf. Decision and Control*, pp. 2328–2333, Paradise Islands, Bahamas, 2004.

- [159] M. K. Çamlıbel, J. S. Pang, and J. Shen. Conewise linear systems: non-zenoness and observability. *SIAM J. Control Optim.*, **45**(5):1769–1800, 2006.
- [160] M. K. Çamlıbel, W. P. M. H. Heemels, and J. M. Schumacher. Algebraic necessary and sufficient conditions for the controllability of conewise linear systems. *IEEE Trans. Automat. Control*, **53**(3):762–774, 2008.
- [161] M. K. Çamlıbel, L. Iannelli, and F. Vasca. Passivity and complementarity. Under review for publication in *Mathematical Programming-A*, 2009.
- [162] M. K. Çamlıbel, J. S. Pang, J. M. Schumacher, and J. Shen. Directional differentiability of the trajectories of linear passive complementarity systems. In preparation, 2009.
- [163] F. Ceragioli and C. De Persis. Discontinuous stabilization of nonlinear systems: quantized and switching controls. *Syst. Control Lett.*, **56**(7–8):461–473, 2007.
- [164] B. Chachuat, A. B. Singer, and P. I. Barton. Global methods for dynamic optimization and mixed-integer dynamic optimization. *Indus. Eng. Chem. Res.*, **45**(25):8373–8392, 2006.
- [165] V. Chandru and J. N. Hooker. *Optimization Methods for Logical Inference*. Wiley-Interscience, New York, 1999.
- [166] Z. Chaochen, C. A. R. Hoare, and A. P. Ravn. A calculus of durations. *Inform. Processing Lett.*, **40**(5):269–276, 1991.
- [167] C. Chase, J. Serrano, and P. J. Ramadge. Periodicity and chaos from switched flow systems: contrasting examples of discretely controlled systems. *IEEE Trans. Automat. Control*, **38**(1):70–83, 1993.
- [168] J. Chen and R. J. Patton. *Robust Model-Based Fault Diagnosis for Dynamic Systems*. Kluwer, Dordrecht, 1999.
- [169] M. Chen, C. R. Zhu, and G. Feng. Linear-matrix-inequality-based approach to H_∞ controller synthesis of uncertain continuous-time piecewise linear systems. *IEE Proc. Control Theory Appl.*, **151**(3):295–300, 2004.
- [170] Y. Cho, C. G. Cassandras, and D. L. Pepyne. Forward decomposition algorithms for optimal control of a class of hybrid systems. *Int. J. Robust Nonlin. Control*, **11**(5):497–513, 2001.
- [171] M. Christensen, P. Einewall, and B. Johansson. Homogeneous charge compression ignition (HCCI) using isoctane, ethanol and natural Gas—a comparison to spark ignition operation. SAE Technical Papers 972874, 1997.
- [172] L. O. Chua and T. S. Parker. *Practical Numerical Algorithms for Chaotic Systems*. Springer-Verlag, Berlin, 1989.
- [173] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. The MIT Press, Cambridge, MA, 2002.
- [174] F. H. Clarke and R. B. Vinter. Optimal multiprocesses. *SIAM J. Control Optim.*, **27**(5):1072–1091, 1989.
- [175] F. H. Clarke and R. B. Vinter. Applications of optimal multiprocesses. *SIAM J. Control Optim.*, **27**(5):1048–1071, 1989.
- [176] F. H. Clarke, Y. S. Ledyaev, E. D. Sontag, and A. I. Subbotin. Asymptotic controllability implies feedback stabilization. *IEEE Trans. Automat. Control*, **42**(10):1394–1407, 1997.

- [177] F. H. Clarke, Y. S. Ledyaev, and R. J. Stern. Asymptotic stability and smooth Lyapunov functions. *J. Diff. Eq.*, **149**(1):69–114, 1998.
- [178] F. H. Clarke, Yu. S. Ledyaev, R. J. Stern, and P. R. Wolenski. *Nonsmooth Analysis and Control Theory*. Springer-Verlag, Berlin, 1998.
- [179] M. I. Clune, P. J. Mosterman, and C. G. Cassandras. Discrete Event and Hybrid System Simulation with SimEvents. In *Proc. 8th Int. Workshop on Discrete Event Systems*, pp. 386–387, Michigan, USA, 2006.
- [180] CO-LaN. The CAPE-OPEN laboratories network, 2008. Available at: <http://www.colan.org>.
- [181] D. Collela and D. Heil. Characterization of scaling functions. I. Continuous solutions. *SIAM J. Matrix Anal. Appl.*, **15**:496–518, 1994.
- [182] P. Collins. Continuity and computability of reachable sets. *Theor. Comp. Sci.*, **341**:162–195, 2005.
- [183] P. Collins and J. H. van Schuppen. Observability of piecewise-affine hybrid systems. In R. Alur and G. J. Pappas, eds., *Hybrid Systems: Computation and Control*, pp. 265–279. Springer-Verlag, Berlin, 2004.
- [184] P. Collins, L. Habets, A. Kuut, M. Nool, M. Petreczky, and J. H. van Schuppen. ConPAHS—a software package for control of piecewise-affine hybrid system. In *Proc. Computer Aided Control System Design*, pp. 76–81. IEEE Press, 2006.
- [185] Columbus IST project, 2006. Available at: <http://www.columbus.gr>.
- [186] H. Cormerais, P. Y. Richard, C. Morvan, and J. Buisson. A generic passivity based control for multicellular serial converters. In *Proc. 16th IFAC World Congress*, Prague, Czech Republic, 2005.
- [187] D. Corona and B. De Schutter. Adaptive cruise control for a SMART car: a comparison benchmark for MPC-PWA control methods. *IEEE Trans. Control Syst. Technol.*, **16**(2):365–372, 2008.
- [188] J. Cortes. Discontinuous dynamical systems. *IEEE Control Syst. Magazine*, **28**(3): 36–73, 2008.
- [189] R. W. Cottle, J. S. Pang, and R. E. Stone. *The Linear Complementarity Problem*. Academic Press, Boston, MA, 1992.
- [190] J. E. R. Cury, B. A. Krogh, and T. Niinomi. Synthesis of supervisory controllers for hybrid systems based on approximating automata. *IEEE Trans. Automat. Control*, **43**(4):564–568, 1998.
- [191] C. R. Cutler and B. L. Ramaker. Dynamic matrix control—a computer control algorithm. In *Proc. 86th AIChE National Meeting*, Houston, TX, 1979.
- [192] T. van Cutsem and C. Vournas. *Voltage Stability of Electric Power Systems*. Kluwer, Dordrecht, 1998.
- [193] J. Daafouz and J. Bernussou. Parameter dependent Lyapunov functions for discrete time systems with time varying parametric uncertainties. *Syst. Control Lett.*, **43**(5):355–359, 2001.
- [194] J. Daafouz, P. Riedinger, and C. Iung. Stability analysis and control synthesis for switched systems: a switched Lyapunov function approach. *IEEE Trans. Automat. Control*, **47**(11):1883–1887, 2002.

- [195] J. B. Dabney and T. L. Harman. *Mastering Simulink*. Prentice Hall, Harlow, 2003.
- [196] H. Dankowicz and P. T. Piiroinen. Exploiting discontinuities for stabilization of recurrent motions. *Dynamical Syst.*, **17**(4):317–342, 2002.
- [197] C. D’Apice, M. Garavello, R. Manzo, and B. Piccoli. Hybrid optimal control: case study of a car with gears. *Int. J. Control*, **76**(13):1272–1284, 2003.
- [198] Dash Associates. *XPRESS-MP User Guide*, 2004. Available at: <http://www.dashoptimization.com>.
- [199] I. Daubechies and J. C. Lagarias. Corrigendum/addendum to: sets of matrices all infinite products of which converge. *Linear Algebr. Appl.*, **327**(1–3):69–83, 2001.
- [200] I. Daubechies and J. C. Lagarias. Sets of matrices all infinite products of which converge. *Linear Algebr. Appl.*, **161**:227–263, 1992.
- [201] R. David and H. Alla. Petri nets for modelling of dynamic systems—a survey. *Automatica*, **30**(2):175–202, 1994.
- [202] J. Davies. *Specification and Proof in Real-time CSP*. Cambridge University Press, Cambridge, 1993.
- [203] M. H. A. Davis. *Markov Processes and Optimization*. Chapman & Hall, London, 1993.
- [204] C. Daws, R. Langerak, and J. W. Polderman. Decision algorithm for the stability of planar switching linear systems. In *Proc. 18th Int. Symp. Mathematical Theory of Networks and Systems*, Blacksburg, VA, 2008.
- [205] W. P. Dayawansa and C. F. Martin. A converse Lyapunov theorem for a class of dynamical systems which undergo switching. *IEEE Trans. Automat. Control*, **44**(4):751–760, 1999.
- [206] R. A. Decarlo, M. S. Branicky, S. Pettersson, and B. Lennartson. Perspectives and results on the stability and stabilizability of hybrid systems. *IEEE Trans. Automat. Control*, **88**(7):1069–1082, 2000.
- [207] D. F. Delchamps. Stabilizing a linear system with quantized state feedback. *IEEE Trans. Automat. Control*, **35**(8):916–924, 1990.
- [208] J. C. Delvenne. An optimal quantized feedback strategy for scalar linear systems. *IEEE Trans. Automat. Control*, **51**(2):298–303, 2006.
- [209] A. Deshpande. Control of hybrid systems. Unpublished PhD thesis, Department of Electrical Engineering, University of California, Berkeley, CA, 1994.
- [210] A. Deshpande, A. Göllü, and P. Varaiya. SHIFT: a formalism and a programming language for dynamic networks of hybrid automata. In *Hybrid Systems: Computation and Control*, pp. 113–133. Springer-Verlag, Berlin, 1997.
- [211] Y. Dessouky and C. A. Roberts. A review and classification of combined simulation. *Comput. Industrial Eng.*, **32**(2):251–264, 1997.
- [212] M. Diehl, D. B. Leineweber, and A. Schäfer. *MUSCOD-II User’s Manual*, University of Heidelberg, Germany, 2001.
- [213] I. A. Digailova and A. B. Kurzhanski. Reachability analysis under control-dependent stochastic noise. In *Proc. 16th IFAC World Congress*, p/n FR-A13-TO/1, Prague, Czech Republic, 2005.

- [214] A. D’Innocenzo, M. D. Di Benedetto, and S. Di Gennaro. Observability of hybrid automata by abstraction. In J. Hespanha and A. Tiwari, eds., *Hybrid Systems: Computation and Control*, pp. 169–183. Springer-Verlag, Berlin, 2006.
- [215] V. Donde and I. A. Hiskens. Shooting methods for locating grazing phenomena in hybrid systems. *Int. J. Bifurcation Chaos*, **16**(3):671–692, 2006.
- [216] EA Internacional. EcosimPro user manual, 2008. Available at: <http://www.ecosimpro.com>.
- [217] B. C. Eaves and C. E. Lemke. Equivalence of LCP and PLS. *Math. Oper. Res.*, **6**(4):475–484, 1981.
- [218] M. Egerstedt, Y. Wardi, and H. Axelsson. Transition-time optimization for switched-mode dynamical systems. *IEEE Trans. Automat. Control*, **51**(1): 110–115, 2006.
- [219] N. Elia and S. Mitter. Stabilization of linear systems with limited information. *IEEE Trans. Automat. Control*, **46**(9):1384–1400, 2001.
- [220] H. Elmqvist. A structured model language for large continuous systems. Unpublished PhD thesis, Department of Autom. Control, Lund Institute of Technology, Lund, Sweden, 1978.
- [221] S. Engell. Feedback control for optimal process operation. *J. Process Control*, **17**(3):203–219, 2007.
- [222] S. Engell, S. Kowalewski, C. Schulz, and O. Stursberg. Continuous-discrete interactions in chemical processing plants. *Proc. IEEE*, **88**(7):1050–1068, 2000.
- [223] S. Engell, M. Fritz, and K. Wöllhaf. Object-oriented modeling and simulation of batch plants. In D. T. Leondes, ed., *Computer-Integrated Manufacturing, Vol. II*, Chapter 6. CRC Press, Boca Raton, CA, 2001.
- [224] S. Engell, G. Frehse, and E. Schnieder (eds.). *Modelling, Analysis, and Design of Hybrid Systems*. Springer-Verlag, Berlin, 2002.
- [225] F. Fagnani and S. Zampieri. Stability analysis and synthesis for scalar linear systems with a quantized feedback. *IEEE Trans. Automat. Control*, **48**(9): 1569–1584, 2003.
- [226] F. Fagnani and S. Zampieri. Quantized stabilization of linear systems: complexity versus performance. *IEEE Trans. Automat. Control*, **49**(9):1534–1548, 2004.
- [227] P. Fairley. The unruly power grid. *IEEE Spectrum*, **41**(8):22–27, 2004.
- [228] R. G. Farmer and P. M. Anderson. *Series Compensation of Power Systems*. PBLSH, Inc., Encinitas, CA, 1996.
- [229] W. F. Feehery and P. I. Barton. Dynamic optimization with state variable path constraints. *Comput. Chem. Eng.*, **22**(9):1241–1256, 1998.
- [230] E. Feron, P. Apkarian, and P. Gahinet. Analysis and synthesis of robust control systems via parameter-dependent Lyapunov functions. *IEEE Trans. Automat. Control*, **41**(7):1041–1046, 1996.
- [231] G. Ferrari-Trecate. Hybrid identification toolbox (HIT), 2005. Available at: http://sisdin.unipv.it/lab/personale/pers_hp/ferrari/HIT_toolbox.html.

- [232] G. Ferrari-Trecate and M. Muselli. Single-linkage clustering for optimal classification in piecewise affine regression. In *Proc. 1st IFAC Conf. Analysis and Design of Hybrid Systems*, Saint-Malo, France, 2003.
- [233] G. Ferrari-Trecate and M. Schinkel. Conditions of optimal classification for piecewise affine regression. In O. Maler and A. Pnueli, eds., *Hybrid Systems: Computation and Control*, pp. 188–202. Springer-Verlag, Berlin, 2003.
- [234] G. Ferrari-Trecate, F. A. Cuzzola, D. Mignone, and M. Morari. Analysis of discrete time piecewise affine and hybrid systems. *Automatica*, **38**(12): 2139–2146, 2002.
- [235] G. Ferrari-Trecate, D. Mignone, and M. Morari. Moving horizon estimation for hybrid systems. *IEEE Trans. Automat. Control*, **47**(10):1663–1676, 2002.
- [236] G. Ferrari-Trecate, M. Muselli, D. Liberati, and M. Morari. A clustering technique for the identification of piecewise affine systems. *Automatica*, **39**(2): 205–217, 2003.
- [237] A. F. Filippov. *Differential Equations with Discontinuous Righthand Sides*. Kluwer, Dordrecht, 1988.
- [238] P. A. Fishwick (ed.). *Handbook of Dynamic System Modeling*. Chapman & Hall/CRC Taylor & Francis Group, 2007.
- [239] G. Flandin. XML Tree—an XML parser for MATLAB, 2002. Available at: <http://www.artefact.tk/software/matlab/xml/>.
- [240] D. Flieller, P. Riedinger, and J. P. Louis. Computation and stability of limit cycles in hybrid systems. *Nonlinear Anal.*, **64**(2):352–367, 2006.
- [241] A. Flores-Tlacuahuac and L. T. Biegler. Simultaneous mixed-integer dynamic optimization for integrated design and control. *Comput. Chem. Eng.*, **31**(5–6): 588–600, 2007.
- [242] D. Förstner and J. Lunze. Fault detection of a diesel injection system by qualitative modelling. In *Workshop on Advances in Automotive Control*, pp. 263–269, Karlsruhe, Germany, 2001.
- [243] D. Förstner and J. Lunze. Discrete-event models of quantised systems for diagnosis. *Int. J. Control*, **74**(7):690–700, 2001.
- [244] D. Förstner and J. Lunze. Qualitative modeling of a power stage for diagnosis. In *13th Int. Workshop Qualitative Reasoning*, pp. 105–112, Loch Awe, Scotland, 1999.
- [245] D. Förstner, M. Jung, and J. Lunze. A discrete-event model of asynchronous quantised systems. *Automatica*, **38**(8):1277–1286, 2002.
- [246] D. Förstner, R. Weber, and J. Lunze. Diagnose eines Diesel-Einspritzsystems mit ereignisdiskreten Modellen. *Automatisierungstechnische Praxis*, **45**:73–79, 2003.
- [247] R. Fourer, D. M. Gay, and B. W. Kernighan. *AMPL: A Modeling Language for Mathematical Programming*. Duxbury Press/Brooks Cole Publishing Company, Princeton, NJ, 2002.
- [248] A. L. Fradkov and A. Y. Progronsky. *Introduction to Control of Oscillations and Chaos*. World Scientific, 1998.
- [249] E. Frazzoli, L. Pallottino, V. G. Scordio, and A. Bicchi. Decentralized cooperative conflict resolution for multiple nonholonomic vehicles. *Proc.*

- AIAA Guidance, Navigation and Control Conf.*, p/n 2005-6048, San Francisco, CA, 2005.
- [250] G. Frehse. PHAVer: Algorithmic verification of hybrid systems past HyTech. *Int. J. Software Tools Tech. Transfer (STTT)*, **10**(3):263–279, 2008.
- [251] G. Frehse. Compositional verification of hybrid systems using simulation relations. Unpublished PhD thesis, Radboud University Nijmegen, 2004.
- [252] G. Frehse, Z. Han, and B. H. Krogh. Assume-guarantee reasoning for hybrid I/O-automata by over-approximation of continuous interaction. In *Proc. 43th IEEE Conf. Decision and Control*, Paradise Islands, Bahamas, 2004.
- [253] G. Frehse, S. Kumar Jha, and B. H. Krogh. A counterexample-guided approach to parameter synthesis for linear hybrid automata. In M. Egerstedt and B. Mishra, eds., *Hybrid Systems: Computation and Control*. Springer-Verlag, Berlin, 2008.
- [254] P. Fritzson. *Principles of Object-Oriented Modeling and Simulation with Modelica 2.1*. Wiley Interscience /IEEE Press, 2004.
- [255] M. Fu and L. Xie. The sector bound approach to quantized feedback control. *IEEE Trans. Automat. Control*, **50**(11):1698–1711, 2005.
- [256] H. Fujioka, C. Y. Kao, S. Almér, and U. Jönsson. Sampled-data \mathcal{H}_∞ control design for a class of PWM systems. In *Proc. 44th IEEE Conf. Decision and Control*, pp. 4499–4504, Seville, Spain, 2005.
- [257] H. Fujioka, S. Almèr, U. Jönsson, and C. Y. Kao. Control synthesis for a class of PWM systems for robust tracking and \mathcal{H}_∞ performance. In *Proc. 45th IEEE Conf. Decision and Control*, San Diego, CA, 2006.
- [258] G. Fung and O. L. Mangasarian. Proximal support vector machine classifiers. In F. Provost and R. Srikant, eds., *Proc. KDD-2001: Knowledge Discovery and Data Mining*, pp. 77–86, San Francisco, CA, 2001. Association for Computing Machinery.
- [259] M. Garavello and B. Piccoli. Hybrid necessary principle. *SIAM J. Control Optim.*, **43**(5):1867–1887, 2005.
- [260] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, San Francisco, CA, 1979.
- [261] B. Gebremichael and F. Vaandrager. Specifying urgency in timed I/O automata. In *3rd IEEE Conf. Software Engineering and Formal Methods*, pp. 64–74, 2005.
- [262] J. C. Geromel and P. Colaneri. Stability and stabilization of discrete time switched systems. *Int. J. Control*, **79**(7):719–728, 2006.
- [263] T. Geyer. Low complexity model predictive control in power electronics and power systems. Unpublished PhD thesis, ETH Zurich, Zurich, Switzerland, 2005.
- [264] T. Geyer, M. Larsson, and M. Morari. Hybrid emergency voltage control in power systems. In *Proc. 2003 European Control Conf.*, Cambridge, UK, 2003.
- [265] T. Geyer, F. D. Torrisi, and M. Morari. Efficient mode enumeration of compositional hybrid models. In A. Pnueli and O. Maler, eds., *Hybrid Systems: Computation and Control*, pp. 216–232. Springer-Verlag, Berlin, 2003.

- [266] M. K. Ghosh, A. Arapostathis, and S. I. Marcus. Ergodic control of switching diffusions. *SIAM J. Control Optim.*, **35**(6):1952–1988, 1997.
- [267] N. Giorgetti, A. Bemporad, E. H. Tseng, and D. Hrovat. Hybrid model predictive control application towards optimal semi-active suspension. *Int. J. Control*, **79**(5):521–533, 2006.
- [268] N. Giorgetti, G. Ripaccioli, A. Bemporad, I. V. Kolmanovsky, and D. Hrovat. Hybrid model predictive control of direct injection stratified charge engines. *IEEE/ASME Trans. Mechatronics*, **11**(5):499–506, 2006.
- [269] A. Girard. Computation and stability analysis of limit cycles in piecewise linear hybrid systems. In *Proc. 1st IFAC Conf. Analysis and Design of Hybrid Systems*, pp. 181–186, Saint-Malo, France, 2003.
- [270] A. Girard. Reachability of uncertain linear systems using zonotopes. In M. Morari and L. Thiele, eds., *Hybrid Systems: Computation and Control*, pp. 291–305. Springer-Verlag, Zurich, 2005.
- [271] R. Goebel and A. R. Teel. Solutions to hybrid inclusions via set and graphical convergence with stability theory applications. *Automatica*, **42**(4):573–587, 2006.
- [272] R. Goebel and A. R. Teel. Zeno behavior in homogeneous hybrid systems. In *Proc. 47th IEEE Conf. Decision and Control*, pp. 2758–2763, Cancun, Mexico, 2008.
- [273] R. Goebel, J. Hespanha, A. R. Teel, C. Cai, and R. Sanfelice. Hybrid systems: generalized solutions and robust stability. In *IFAC Symp. Nonlinear Control Systems*, pp. 1–12, Stuttgart, Germany, 2004.
- [274] J. M. Goncalves, A. Megretski, and M. A. Dahleh. Global analysis of piecewise linear systems using impact maps and quadratic surface Lyapunov functions. *IEEE Trans. Automat. Control*, **48**(12):2089–2106, 2003.
- [275] J. M. Goncalves and T. M. Yi. Drosophila circadian rhythms: stability robustness analysis and model reduction. In *Proc. 16th Int. Symp. Mathematical Theory of Networks and Systems*, p. 256, Leuven, Belgium, 2004.
- [276] P. Grieder and M. Morari. Complexity reduction of receding horizon control. In *Proc. 42th IEEE Conf. Decision and Control*, pp. 3179–3190, Maui, USA, 2003.
- [277] P. Grieder, F. Borrelli, F. D. Torrisi, and M. Morari. Computation of the constrained infinite time linear quadratic regulator. In *Proc. 2003 American Control Conf.*, pp. 4711–4716, Denver, CO, 2003.
- [278] P. Grieder, M. Lüthi, P. Parillo, and M. Morari. Stability and feasibility of receding horizon control. In *Proc. 2003 European Control Conf.*, p. 117, Cambridge, UK, 2003.
- [279] P. Grieder, P. Parillo, and M. Morari. Robust receding horizon control. In *Proc. 42th IEEE Conf. Decision and Control*, pp. 941–946, Maui, USA, 2003.
- [280] P. Grieder, F. Borelli, F. Torrisi, and M. Morari. Computation of the constrained infinite time linear quadratic regulator. *Automatica*, **40**(4):701–701, 2004.

- [281] P. Grieder, M. Kvasnica, M. Baotić, and M. Morari. Stabilizing low complexity feedback control of constrained piecewise affine system. *Automatica*, **41**(10):1683–1694, 2005.
- [282] G. Gripenberg. Computing the joint spectral radius. *Linear Algebr. Appl.*, **234**:43–60, 1996.
- [283] B. Grocholsky, H. Durrant-White, and P. Gibbens. An information theoretic approach to decentralized control of multiple autonomous flight vehicles. *Sensor fusion and decentralized control in robotic systems III*, **4196**:348–359, 2000.
- [284] I. E. Grossmann. Review of nonlinear mixed-integer and disjunctive programming techniques. *Optimization and Engineering*, **3**(3):227–252, 2002.
- [285] I. E. Grossmann and L. T. Biegler. Future perspective on optimization. *Comp. Chem. Eng.*, **28**(8):1193–1218, 2004.
- [286] L. Grüne and O. Junge. A set-oriented approach to optimal feedback stabilization. *Syst. Control Lett.*, **54**(2):169–180, 2005.
- [287] J. Gu, P. W. Purdom, J. Franco, and B. Wah. Algorithms for the satisfiability (SAT) problem: a survey. *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, **35**:19–151, 1997.
- [288] H. Gueguen, M. A. Lefebvre, O. Nasri, and J. Zaytoon. Safety verification and reachability analysis for hybrid systems. In *Proc. 17th IFAC World Congress*, pp. 8949–8959, Seoul, Korea, 2008.
- [289] N. Guglielmi and M. Zennaro. On the zero-stability of variable stepsize multistep methods: the spectral radius approach. *Numer. Math.*, **88**(3):445–458, 2001.
- [290] F. Gustafsson and F. Gunnarsson. Mobile positioning using wireless networks. *IEEE Signal Proc. Magazine*, **22**(4):41–53, 2005.
- [291] L. C. G. J. M. Habets and J. H. van Schuppen. A controllability result for piecewise-linear hybrid systems. In *Proc. 2001 European Control Conf.*, Porto, Portugal, 2001.
- [292] W. M. Haddad, V. S. Chellaboina, and S. G. Nersesov. *Impulsive and Hybrid Dynamical Systems*. Princeton University Press, Princeton, NJ, 2007.
- [293] B. Hahn and D. Valentine. *Essential MATLAB for Engineers and Scientists* 3rd edn. Butterworth Heinemann Oxford, 2007.
- [294] N. Halbwachs, Y. E. Proy, and P. Roumanoff. Verification of real-time systems using linear relation analysis. *Formal Methods in System Design*, **11**(2): 157–185, 1997.
- [295] D. Harel. Statecharts: a visual formalism for complex systems. *Science of Computer Programming*, **8**(8):231–274, 1987.
- [296] S. Haugwitz, J. Akesson, and P. Hagander. Dynamic optimization of a plate reactor start-up supported by Modelica-based code generation software. In *Proc. 8th Int. Symp. Dynamics and Control of Process Systems*, pp. 99–104, Cancun, Mexico, 2007.
- [297] B. S. Heck, L. M. Wills, and G. J. Vachtsevanos. Software technology for implementing reusable, distributed control systems. *IEEE Control Sys. Magazine*, **23**(1):21–35, 2003.

- [298] S. Hedlund and A. Rantzer. Convex dynamic programming for hybrid systems. *IEEE Trans. Automat. Control*, **47**(9):1536–1540, 2002.
- [299] S. Hedlund and A. Rantzer. Optimal control of hybrid systems. In *Proc. 38th IEEE Conf. Decision and Control*, pp. 3972–3977, Phoenix, AZ, 1999.
- [300] W. P. M. H. Heemels. Linear complementarity systems: a study in hybrid dynamics. Unpublished PhD thesis, Dept. of Electrical Engineering, Eindhoven University of Technology, Eindhoven, The Netherlands, 1999.
- [301] W. P. M. H. Heemels and S. Weiland. Input-to-state stability and interconnections of discontinuous dynamical systems. *Automatica*, 2009. To appear.
- [302] W. P. M. H. Heemels, J. M. Schumacher, and S. Weiland. The rational complementarity problem. *Linear Algebr. Appl.*, **294**(1–3):93–135, 1999.
- [303] W. P. M. H. Heemels, J. M. Schumacher, and S. Weiland. Projected dynamical systems in a complementarity formalism. *Operations Res. Lett.*, **27**(2):83–91, 2000.
- [304] W. P. M. H. Heemels, J. M. Schumacher, and S. Weiland. Linear complementarity systems. *SIAM J. Appl. Math.*, **60**(4):1234–1269, 2000.
- [305] W. P. M. H. Heemels, B. De Schutter, and A. Bemporad. Equivalence of hybrid dynamical models. *Automatica*, **37**(7):1085–1091, 2001.
- [306] W. P. M. H. Heemels, M. K. Çamlıbel, and J. M. Schumacher. On the dynamic analysis of piecewise-linear networks. *IEEE Trans. Circuits Syst. I*, **49**(3):315–327, 2002.
- [307] W. P. M. H. Heemels, M. K. Çamlıbel, A. J. van der Schaft, and J. M. Schumacher. Modelling, well-posedness, and stability of switched electrical networks. In O. Maler and A. Pnueli, eds., *Hybrid Systems: Computation and Control*, pp. 249–266. Springer-Verlag, Berlin, 2003.
- [308] H. Heinecke, K. P. Schnelle, H. Fennel, *et al.* AUTomotive Open System ARchitecture—an industry-wide initiative to manage the complexity of emerging automotive E/E-architectures. In *Proc. Convergence 2004*, Detroit, MI, 2004.
- [309] T. A. Henzinger. The theory of hybrid automata. In *Proc. 11th Annual IEEE Symp. LICS*, pp. 278–292. IEEE Computer Society Press, New Brunswick, NJ, 1996.
- [310] T. A. Henzinger, P. H. Ho, and H. Wong-Toi. HYTECH: the next generation. In *Proc. 16th IEEE RTSS*, pp. 56–65. IEEE Computer Society, 1995.
- [311] T. A. Henzinger, P. H. Ho, and H. Wong-Toi. A user guide to HYTECH. In E. Brinksma, R. Cleaveland, K. G. Larsen, T. Margaria, and B. Steffen, eds., *Tools and Algorithms for Construction and Analysis of Systems, First Int. Workshop*, pp. 41–71, Aarhus, Denmark. Springer-Verlag, Berlin, 1995.
- [312] T. A. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya. What’s decidable about hybrid automata. In *Proc. 27th Annual Symp. Theory of Computing*, pp. 373–382. ACM Press, Las Vegas, NV, 1995.
- [313] T. A. Henzinger, P. H. Ho, and H. Wong-Toi. HYTECH: a model checker for hybrid systems. *Int. J. Software Tools Technol. Transfer*, **1**(1–2):110–122, 1997.

- [314] T. A. Henzinger, P. H. Ho, and H. Wong-Toi. Algorithmic analysis of nonlinear hybrid systems. *IEEE Trans. Automat. Control*, **43**(4):540–554, 1998.
- [315] T. A. Henzinger, P. W. Kopke, A. Puria, and P. Varaiya. What’s decidable about hybrid automata? *J. Comp. Syst. Sci.*, **57**(1):94–124, 1998.
- [316] J. P. Hespanha. A model for stochastic hybrid systems with application to communication networks. *Nonlinear Anal.*, **62**(8):1353–1383, 2005.
- [317] J. P. Hespanha and A. S. Morse. Stabilization of nonholonomic integrators via logic-based switching. *Automatica*, **35**(3):385–393, 1999.
- [318] J. P. Hespanha and S. A. Morse. Stability of switched systems with average dwell-time. In *Proc. 38th IEEE Conf. Decision and Control*, pp. 2655–2660, Phoenix, AZ, 1999.
- [319] J. P. Hespanha, D. Liberzon, and A. S. Morse. Towards the supervisory control of uncertain nonholonomic systems. In *Proc. 1999 American Control Conf.*, pp. 3520–3524, San Diego, CA, 1999.
- [320] J. P. Hespanha, D. Liberzon, and A. R. Teel. Lyapunov conditions for input-to-state stability of impulsive systems. *Automatica*, **44**(11):2735–2744, 2008.
- [321] I. A. Hiskens. Stability of limit cycles in hybrid systems. In *Proc. 34th IEEE Hawaii Int. Conf. System Sciences*, 6pp., Maui, USA, 2001.
- [322] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, New York, 1985.
- [323] T. Hodrus and E. Münz. Hybride Phänomene in zeitdiskreter Darstellung. *Automatisierungstechnik*, **51**:574–582, 2005.
- [324] L. E. Holloway, B. H. Krogh, and A. Giua. A survey of petri net methods for controlled discrete event systems. *Discrete Event Dynam. Syst.: Theor. Appl.*, **7**(2):151–190, 1997.
- [325] K. Holmström, A. O. Göran, and M. M. Edvall. User’s guide for TOMLAB, 2007. Available at: <http://www.tomopt.com>.
- [326] D. Hristu-Varsakelis and W. S. Levine (eds.). *Handbook of Networked and Embedded Control Systems*. Birkhäuser, Boston, MA, 2005.
- [327] C. S. Hsu. *Cell-to-Cell Mapping: A Method of Global Analysis for Nonlinear Systems*. Springer-Verlag, New York, 1987.
- [328] J. Hu and M. Prandini. Aircraft conflict detection: a method for computing the probability of conflict based on Markov chain approximation. In *Proc. 2003 European Control Conf.*, p. 378, Cambridge, UK, 2003.
- [329] J. Hu, J. Lygeros, and S. Sastry. Towards a theory of stochastic hybrid systems. In N. Lynch and B. H. Krogh, eds., *Hybrid Systems: Computation and Control*, pp. 160–173. Springer-Verlag, Berlin, 2000.
- [330] Y.-P. Huang and K. Zhou. Robust stability and performance of uncertain delay systems with structured uncertainties, In *Proc. 39th IEEE Conf. Decision and Control*, pp. 1509–1514, Sydney, Australia, 2000.
- [331] T. Huerlimann. *Reference Manual for the LPL Modeling Language, Version 4.4.2*. Departement for Informatics, Université de Fribourg, Switzerland, 2001. Available at: <http://www2-iiuf.unifr.ch/tcs/lpl/TonyHome.htm>.

- [332] Y. Hur and I. Lee. Distributed simulation of multi-agent hybrid systems. In *Proc. IEEE Int. Symp. Object-Oriented Real-time Distributed Computing*, pp. 356–364, Magdeburg, Germany, 2002.
- [333] HYCON. HYCON tool repository and demonstrator site. European Embedded Control Institute (EECI), Gif-sur-Yvette, France / Technische Universität Dortmund, Germany, 2008. Available at: <http://wp3.hycon.bci.tu-dortmund.de>.
- [334] ILOG, Inc. *CPLEX 11.0 User Manual*. Gentilly Cedex, France, 2008.
- [335] J. I. Imura and A. J. van der Schaft. Characterization of well-posedness of piecewise linear systems. *IEEE Trans. Automat. Control*, **45**(9):1600–1619, 2000.
- [336] J. I. Imura and A. J. van der Schaft. Well-posedness of a class of dynamically interconnected systems. In *Proc. 38th IEEE Conf. Decision and Control*, pp. 3031–3036, Phoenix, AZ, 1999.
- [337] A. Ingimundarson, C. Ocampo-Martinez, and A. Bemporad. Model predictive control of hybrid systems based on mode-switching constraints. In *Proc. 46th IEEE Conf. Decision and Control*, pp. 5265–5269, New Orleans, LA, 2007.
- [338] H. Ishii and B. A. Francis. Stabilizing a linear systems by switching control with dwell time. *IEEE Trans. Automat. Control*, **47**(12):1962–1973, 2002.
- [339] N. Jenkins, R. Allan, P. Crossley, D. Kirschen, and G. Strbac. *Embedded Generation*. TJ International Ltd., Padstow, UK, 2000.
- [340] K. H. Johansson, M. Törngren, and L. Nielsen. Vehicle applications of controller area network. In D. Hristu-Varsakelis and W. S. Levine, eds., *Handbook of Networked and Embedded Control Systems*, pp. 741–766. Birkhäuser, Boston, MA, 2005.
- [341] K. J. Johansson, M. Egerstedt, J. Lygeros, and S. Sastry. On the regularization of zeno hybrid automata. *Syst. Control Lett.*, **38**(3):141–150, 1999.
- [342] M. Johansson. *Piecewise Linear Control Systems*. Springer-Verlag, Berlin, 2003.
- [343] M. Johansson and A. Rantzer. Computation of piecewise quadratic Lyapunov functions for hybrid systems. *IEEE Trans. Automat. Control*, **43**(4):555–559, 1998.
- [344] R. Johansson and A. Rantzer (eds.). *Nonlinear and Hybrid Systems in Automotive Control*. Springer-Verlag, Berlin, 2002.
- [345] A. Lj. Juloski and S. Weiland. A Bayesian approach to the identification of piecewise linear output error models. In *Proc. 14th IFAC Symp. System Identification*, pp. 374–379, Newcastle, Australia, 2006.
- [346] A. Lj. Juloski, W. P. M. H. Heemels, and G. Ferrari-Trecate. Data-based hybrid modelling of the component placement process in pick-and-place machines. *Control Eng. Prac.*, **12**(10):1241–1252, 2004.
- [347] A. Lj. Juloski, W. P. M. H. Heemels, G. Ferrari-Trecate, R. Vidal, S. Paoletti, and J. H. G. Niessen. Comparison of four procedures for the identification of hybrid systems. In M. Morari and L. Thiele, eds., *Hybrid Systems: Computation and Control*, pp. 354–369. Springer-Verlag, Berlin, 2005.

- [348] A. Lj. Juloski, S. Wieland, and W. P. M. H. Heemels. A Bayesian approach to identification of hybrid systems. *IEEE Trans. Automat. Control*, **50**(10): 1520–1533, 2005.
- [349] A. Lj. Juloski, S. Paoletti, and J. Roll. Recent techniques for the identification of piecewise affine and hybrid systems. In L. Menini, L. Zaccarian, and C. T. Abdallah, eds., *Current Trends in Nonlinear Systems and Control*, pp. 77–97. Birkäuser, Boston, MA, 2006.
- [350] A. Lj. Juloski, W. P. M. H. Heemels, and S. Weiland. Observer design for a class of piecewise linear systems. *Int. J. Robust Nonlinear Control*, **17**(15): 1387–1404, 2007.
- [351] R. Jungers and V. D. Blondel. On the finiteness property for rational matrices. *Linear Algebr. Appl.*, **428**(10):2283–2295, 2008.
- [352] R. Jungers, V. Protasov, and V. Blondel. Efficient algorithms for deciding the type of growth of products of integer matrices. *Linear Algebr. Appl.*, **428**(10): 2296–2311, 2008.
- [353] D. Karlsson and D. J. Hill. Modelling and identification of nonlinear dynamic loads in power systems. *IEEE Trans. Power Systems*, **9**:157–163, 1994.
- [354] J. P. Katoen. Stochastic Model Checking. In C. G. Cassandras and J. Lygeros, eds., *Stochastic Hybrid Systems*, pp. 79–106. Taylor & Francis Group/CRC Press, London, 2006.
- [355] C. M. Kellett and A. R. Teel. Smooth Lyapunov functions and robustness of stability for difference inclusions. *Syst. Control Lett.*, **52**(5):395–405, 2004.
- [356] E. C. Kerrigan and J. M. Maciejowski. Robustly stable feedback min-max model predictive control. In *Proc. 2003 American Control Conf.*, Denver, CO, 2003.
- [357] E. C. Kerrigan and D. Q. Mayne. Optimal control of constrained, piecewise affine systems with bounded disturbances. In *Proc. 41th IEEE Conf. Decision and Control*, Las Vegas, NV, 2002.
- [358] K. Keutzer, S. Malik, R. Newton, J. Rabaey, and A. L. Sangiovanni-Vincentelli. System level design: orthogonalization of concerns and platform-based design. *IEEE Trans. Computer-Aided Des.*, **19**(12):1523–1543, 2000.
- [359] H. K. Khalil. *Nonlinear Systems*. Prentice Hall, Upper Saddle River, NJ, 2001.
- [360] C. King and R. Shorten. A singularity test for the existence of common quadratic Lyapunov functions for pairs of stable LTI systems. In *Proc. 2004 American Control Conf.*, pp. 3881–3884, Boston, MA, 2004.
- [361] P. Koiran, M. Cosnard, and M. Garzon. Computability properties of low-dimensional dynamical systems. *Theor. Comp. Sci.*, **132**:113–128, 1994.
- [362] M. Kokar. On consistent symbolic representations of general dynamic systems. *IEEE Trans. Automat. Control*, **40**(8):1231–1242, 1995.
- [363] X. Koutsoukos, P. J. Antsaklis, J. A. Stiver, and M. D. Lemmon. Supervisory control of hybrid systems. *Proc. IEEE*, **88**(7):1026–1049, 2000.
- [364] S. Kowalewski, S. Engell, J. Preußig, and O. Stursberg. Verification of logic controllers for continuous plants using timed condition/event-system models. *Automatica*, **35**(3):505–518, 1999.

- [365] V. S. Kozyakin. Algebraic unsolvability of problem of absolute stability of desynchronized systems. *Automation and Remote Control*, **51**(4):754–759, 1990.
- [366] B. H. Krogh and A. Chutinan. Computing polyhedral approximations to flow pipes for dynamic systems. In *Proc. 37th IEEE Conf. Decision and Control*, vol. 2, pp. 2089–2094, Tampa, FL, 1998.
- [367] J. Krystul and A. Bagchi. Approximation of first passage time of switching diffusions. In *Proc. 16th Int. Symp. Mathematical Theory of Networks and Systems*, Leuven, Belgium, 2004.
- [368] J. Krystul, H. A. P. Blom, and A. Bagchi. Stochastic differential equations on hybrid state spaces. In C. G. Cassandras and J. Lygeros, eds., *Stochastic Hybrid Systems*, pp. 15–45. CRC Press, Boca Raton, FL, 2006.
- [369] S. Kumar Jha, B. H. Krogh, J. E. Weimer, and E. M. Clarke. Reachability for linear hybrid automata using iterative relaxation abstraction. In *Hybrid Systems: Computation and Control*, pp. 287–300. Springer-Verlag, Pisa, Italy, 2007.
- [370] P. Kundur. *Power System Stability and Control*. McGraw-Hill, New York, 1994.
- [371] A. B. Kurzhanski and P. Varaiya. Ellipsoidal techniques for reachability analysis of discrete-time linear systems. *Commun. Inform. Syst.*, **6**(3):179–192, 2006.
- [372] H. J. Kushner and P. G. Dupuis. *Numerical Methods for Stochastic Control Problems in Continuous Time*. Springer-Verlag, New York, 2001.
- [373] M. Kvasnica. Efficient software tools for control and analysis of hybrid systems. Unpublished PhD thesis, ETH Zurich, Zurich, Switzerland, 2008.
- [374] M. Kvasnica, P. Grieder, M. Baotic, and M. Morari. Multi-parametric toolbox (MPT). In R. Alur and G. J. Pappas, eds., *Hybrid Systems: Computation and Control*, pp. 448–462. Springer-Verlag, Berlin, 2004.
- [375] M. Kvasnica, P. Grieder, and M. Baotic. *Multi Parametric Toolbox (MPT)*, 2006. Available at: <http://control.ee.ethz.ch/~mpt/>.
- [376] G. Lafferriere, G. J. Pappas, and S. Sastry. O-minimal hybrid systems. *Math. Control Signals Syst.*, **13**(1):1–21, 2000.
- [377] J. C. Lagarias and Y. Wang. The finiteness conjecture for the generalized spectral radius of a set of matrices. *Linear Algebr. Appl.*, **214**:17–42, 1995.
- [378] R. Langerak and J. W. Polderman. Tools for stability of switching linear systems: gain Automata and delay compensation. In *Proc. 44th IEEE Conf. Decision and Control*, pp. 4867–4872, Sevilla, Spain, 2005.
- [379] R. Langerak, J. W. Polderman, and T. Krilavičius. Stability analysis for hybrid automata using conservative gains. In *Proc. 1st IFAC Conf. Analysis and Design of Hybrid Systems*, pp. 337–342, Saint Malo, France, 2003.
- [380] K. Larsen and Y. Wang. Time-abstracted bisimulation: implicit specifications and decidability. *Inform. Comp.*, **134**(2):75–101, 1997.
- [381] K. G. Larsen, P. Petterson, and W. Yi. UPPAAL in a Nutshell. *Int. J. Software Tools Technol. Transfer*, **1**(1):134–152, 1997.

- [382] K. G. Larsen, B. Steffen, and C. Weise. Continuous modelling of real time and hybrid systems: from concepts to tools. *Software Tools Technol. Transfer*, **1**(1–2):64–85, 1997.
- [383] L. F. S. Larsen, T. Geyer, and M. Morari. Hybrid MPC in supermarket refrigeration systems. In *Proc. 16th IFAC World Congress*, Prague, Czech Republic, 2005.
- [384] L. F. S. Larsen, R. Izadi-Zamanabadi, and R. Wisniewski. Supermarket refrigeration system—benchmark for hybrid system control. In *Proc. 2007 European Control Conf.*, pp. 113–120, Kos, Greece, 2007.
- [385] M. Lazar. Model predictive control of hybrid systems: stability and robustness. Unpublished PhD thesis, Eindhoven University of Technology, Eindhoven, The Netherlands, 2006.
- [386] M. Lazar, W. P. M. H. Heemels, S. Weiland, and A. Bemporad. On the stability of 2-norm based model predictive control of constrained PWA systems. In *Proc. 2005 American Control Conf.*, pp. 575–580, Portland, OR, 2005.
- [387] M. Lazar, W. P. M. H. Heemels, S. Weiland, A. Bemporad, and O. Pastravanu. Infinity norms as Lyapunov functions for model predictive control of constrained PWA systems. In M. Morari and L. Thiele, eds., *Hybrid Systems: Computation and Control*, pp. 417–432. Springer-Verlag, Berlin, 2005.
- [388] M. Lazar, W. P. M. H. Heemels, S. Weiland, and A. Bemporad. Stabilizing model predictive control of hybrid systems. *IEEE Trans. Automat. Control*, **51**(11):1813–1818, 2006.
- [389] R. J. Leduc, B. A. Brandin, W. M. Wonham, and M. Lawford. Hierarchical interface-based supervisory control. In *Proc. 40th IEEE Conf. Decision and Control*, pp. 4116–4121, Orlando, FL, 2001.
- [390] Y. S. Ledyaeu and E. D. Sontag. A Lyapunov characterization of robust stabilization. *Nonlinear Anal.*, **37**(7):813–840, 1999.
- [391] E. A. Lee. Overview of the Ptolemy project. Technical Memorandum No. UCB/ERL M03/25, EECS Department, University of California, Berkeley, CA, 2003.
- [392] E. A. Lee and H. Zheng. Operational semantics of hybrid systems. In M. Morari and L. Thiele, eds., *Hybrid Systems: Computation and Control*, pp. 25–53. Springer-Verlag, Berlin, 2005.
- [393] J. W. Lee and G. E. Dullerud. Uniformly stabilizing sets of switching sequences for switched linear systems. *IEEE Trans. Automat. Control*, **52**(5): 868–874, 2007.
- [394] K. K. Lee and A. Arapostathis. On the controllability of piece-wise linear hypersurface systems. *Syst. Control Lett.*, **9**:89–96, 1987.
- [395] D. M. W. Leenaerts and W. M. G. van Bokhoven. *Piecewise Linear Modelling and Analysis*. Kluwer, Dordrecht, 1998.
- [396] M. A. Lefebvre and H. Gueguen. Hybrid abstraction of affine systems. *Non-linear Anal.*, **65**(6):1150–1167, 2006.
- [397] D. B. Leineweber, A. Schäfer, H. G. Bock, and J. P. Schlöder. An efficient multiple shooting based reduced SQP strategy for large-scale dynamic process

- optimization—part I: theoretical aspects. *Comp. Chem. Eng.*, **27**(2):157–166, 2003.
- [398] D. B. Leineweber, A. Schäfer, H. G. Bock, and J. P. Schlöder. An efficient multiple shooting based reduced SQP strategy for large-scale dynamic process optimization—part II: software aspects and applications. *Comp. Chem. Eng.*, **27**(2):167–174, 2003.
- [399] X. Li, S. K. Jha, and L. Bu. Towards an efficient path-oriented tool for bounded reachability analysis of linear hybrid systems using linear programming. In *Proc. Workshop on Bounded Model Checking*, pp. 57–70, Seattle, WA, 2006.
- [400] F. L. Lian. Analysis, modeling and control of networked control systems. Unpublished PhD thesis, University of Michigan, MI, 2001.
- [401] D. Liberzon. *Switching in Systems and Control*. Birkhäuser, Boston, MA, 2003.
- [402] D. Liberzon and J. Hespanha. Stabilization of nonlinear systems with limited information feedback. *IEEE Trans. Automat. Control*, **50**(6):910–915, 2005.
- [403] D. Liberzon and A. S. Morse. Basic problems in stability and design of switched systems. *IEEE Control Syst. Magazine*, **19**(5):59–70, 1999.
- [404] D. Liberzon and D. Nesić. Input-to-state stabilization of linear systems with quantized state measurements. *IEEE Trans. Automat. Control*, **52**(5):767–781, 2007.
- [405] D. Liberzon, J. P. Hespanha, and A. S. Morse. Stability of switched linear systems: a Lie-algebraic condition. *Syst. Control Lett.*, **37**(3):117–122, 1999.
- [406] G. Lichtenberg, J. Lunze, R. Scheuring, and J. Schröder. Prozeßdiagnose mittels qualitativer Modelle am Beispiel eines Wasserstoffverdichters. *Automatisierungstechnik*, **47**:101–109, 1999.
- [407] B. Lincoln. Dynamic programming and time-varying delay systems. Unpublished PhD thesis, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden, 2003.
- [408] Q. Ling and M. D. Lemmon. Stability of quantized control systems under dynamic bit Assignment. *IEEE Trans. Automat. Control*, **50**(5):734–740, 2005.
- [409] X. Liu and A. Goldsmith. Wireless network design for distributed control. In *Proc. 43th IEEE Conf. Decision and Control*, pp. 2823–2829, Paradise Islands, Bahamas, 2004.
- [410] X. Liu and A. Goldsmith. Wireless medium access control in network control system. In *Proc. 43th IEEE Conf. Decision and Control*, vol. 3, pp. 2823–2829, Paradise Islands, Bahamas, 2004.
- [411] L. Ljung. *System Identification: Theory for the User*. Prentice-Hall, Upper Saddle River, NJ, 1999.
- [412] J. Löfberg. YALMIP: A toolbox for modeling and optimization in MATLAB. In *Proc. CACSD Conf.*, pp. 284–289, Taipei, Taiwan, 2004.
- [413] P. Lötstedt. Mechanical systems of rigid bodies subject to unilateral constraints. *SIAM J. Appl. Math.*, **42**(2):281–296, 1982.

- [414] J. Lunze. The state partitioning problem of quantized systems. In V. D. Blondel, ed., *Unsolved Problems in Mathematical Systems and Control Theory*, pp. 134–139. Princeton University Press, Princeton, NJ, 2004.
- [415] J. Lunze. Qualitative modelling of linear dynamical systems with quantized state measurements. *Automatica*, **30**(3):417–431, 1994.
- [416] J. Lunze. On the markov property of quantized state measurement sequences. *Automatica*, **34**(11):1439–1444, 1998.
- [417] J. Lunze. A timed discrete-event abstraction of continuous-variable systems. *Int. J. Control*, **72**(13):1147–1164, 1999.
- [418] J. Lunze. Diagnosis of quantised systems based on a timed discrete-event model. *IEEE Trans. Syst., Man Cybernetics*, **SMC-30**(3):322–335, 2000.
- [419] J. Lunze and B. Nixdorf. Representation of hybrid systems by means of stochastic automata. *Math. Comp. Modeling Dynam. Syst.*, **7**:383–422, 2001.
- [420] J. Lunze and J. Schröder. State observation and diagnosis of discrete-event systems described by stochastic automata. *Discrete Event Dynam. Syst. Theor. Appl.*, **11**(4):319–369, 2001.
- [421] J. Lunze and J. Schröder. Connections between qualitative dynamical models and the Frobenius-Perron operator. *Automatisierungstechnik*, **51**:471–479, 2003.
- [422] J. Lunze and J. Schröder. Process diagnosis based on a discrete-event description. *Automatisierungstechnik*, **47**:358–365, 1999.
- [423] J. Lunze, B. Nixdorf, and H. Richter. Hybrid modelling of continuous-variable systems with application to supervisory control. In *Proc. 1997 European Control Conf.*, Brussels, Belgium, 1997.
- [424] J. Lunze, B. Nixdorf, and J. Schröder. Deterministic discrete-event representations of continuous-variable systems. *Automatica*, **35**:101–109, 1999.
- [425] J. Lygeros and O. Watkins. Stochastic reachability for discrete time systems: an application to aircraft collision avoidance. In *Proc. 42th IEEE Conf. Decision and Control*, vol. 5, pp. 5314–5319, Maui, USA, 2003.
- [426] J. Lygeros, D. N. Godbole, and S. Sastry. Verified hybrid controllers for automated vehicles. *IEEE Trans. Automat. Control*, **43**(4):522–539, 1998.
- [427] J. Lygeros, K. H. Johansson, S. N. Simić, J. Zhang, and S. Sastry. Dynamical properties of hybrid automata. *IEEE Trans. Automat. Control*, **48**(1):2–17, 2003.
- [428] N. Lynch, R. Segala, F. Vaandrager, and H. B. Weinberg. Hybrid I/O automata. In *Hybrid Systems III*, pp. 496–510. Springer-Verlag, Berlin, 1996.
- [429] N. Lynch, R. Segala, and F. Vandraager. Hybrid I/O automata revisited. In *Proc. 4th Int. Workshop on Hybrid Systems, Computation and Control*, pp. 403–417, 2001.
- [430] N. Lynch, R. Segala, and F. Vaandrager. Hybrid I/O automata. *Inf. Comput.*, **185**(1):105–157, 2003.
- [431] Y. Ma and R. Vidal. Identification of deterministic switched ARX systems via identification of algebraic varieties. In M. Morari and L. Thiele, eds., *Hybrid Systems: Computation and Control*, pp. 449–465. Springer-Verlag, Berlin, 2005.

- [432] J. M. Maciejowski. *Predictive Control with Constraints*. Prentice Hall, Englewood Cliffs, NJ, 2002.
- [433] M. Maesumi. An efficient lower Bound for the generalized spectral radius. *Linear Algebr. Appl.*, **240**:1–7, 1996.
- [434] A. Makhorin. *GLPK (GNU Linear Programming Kit) User's Guide*, 2004. Available at: <http://www.gnu.org/software/glpk/glpk.html>.
- [435] D. Maksimovic, R. Zane, and R. Erickson. Impact of digital control in power electronics. In *Proc. ISPSD*, pp. 13–22, Kitakyushu, Japan, 2004.
- [436] K. L. Man and R. R. H. Schiffelers. Formal specification and analysis of hybrid systems. Unpublished PhD thesis, Eindhoven University of Technology, Eindhoven, The Netherlands, 2006.
- [437] J. L. Mancilla-Aguilar and R. A. Garcia. A converse Lyapunov theorem for nonlinear switched systems. *Syst. Control Lett.*, **41**(1):67–71, 2000.
- [438] X. Mao and C. Yuan. *Stochastic Differential Equations with Markovian Switching*. Imperial College Press, London, 2006.
- [439] K. Marriot and P. J. Stuckey. *Programming with Constraints: An Introduction*. MIT Press, New York, 1998.
- [440] G. Martin. The future of high-level modelling and system level design: some possible methodology scenarios. In *Proc. 9th IEEE/DATC Electronic Design Processes Workshop*, Monterey, USA, 2002.
- [441] G. Martin. Guest editor's introduction: The reuse of complex architectures. *IEEE Des. Test. Comp.*, **19**(6):4–5, 2002.
- [442] P. Mason, M. Sigalotti, and J. Daafouz. On stability analysis of linear discrete-time switched systems using quadratic Lyapunov functions. In *Proc. 46th IEEE Conf. Decision and Control*, New Orleans, LA, 2007.
- [443] S. Mattson and G. Söderlind. Index reduction in differential-algebraic equations using dummy derivatives. *SIAM J. Sci. Statist. Comp.*, **14**(3):677–692, 1993.
- [444] A. Matveev and A. Savkin. *Qualitative Theory of Hybrid Dynamical Systems*. Birkhäuser, Boston, MA, 2000.
- [445] A. S. Matveev and A. V. Savkin. Multi-rate stabilization of linear multiple sensor systems via limited capacity communication channels. *SIAM J. Control Optim.*, **44**(2):584–617, 2005.
- [446] D. Q. Mayne and S. Raković. Optimal control of constrained piecewise affine discrete time systems using reverse transformation. In *Proc. 41st IEEE Conf. Decision and Control*, pp. 1546–1551, Las Vegas, NV, 2002.
- [447] D. Q. Mayne and S. Raković. Model predictive control of constrained PWA discrete-time systems. *Int. J. Robust Nonlinear Control*, **13**(3–4):261–279, 2003.
- [448] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert. Constrained model predictive control: stability and optimality. *Automatica*, **36**(6):789–814, 2000.
- [449] J. L. Menaldi. Stochastic hybrid optimal control Models. *Aportaciones Matematicas (Sociedad Matematica Mexicana)*, **16**:205–250, 2001.

- [450] S. P. Meyn and R. L. Tweedie. *Markov Chains and Stochastic Stability*. Online edition, available at: <http://probability.ca/MT/>, 2005.
- [451] R. D. Middlebrook and S. Cuk. A general unified approach to modeling switching-converter power stages. In *Proc. IEEE Power Electronics Specialists Conf.*, pp. 18–34, 1976.
- [452] D. Mignone, G. Ferrari-Trecate, and M. Morari. Stability and stabilization of piecewise affine and hybrid systems: an LMI approach. In *Proc. 39th IEEE Conf. Decision and Control*, pp. 504–509, Sydney, Australia, 2000.
- [453] M. Milanese and A. Vicino. Optimal estimation theory for dynamic systems with set membership uncertainty: an overview. *Automatica*, **27**(6):997–1009, 1991.
- [454] L. Mirkin and Z. J. Palmor. Control issues in systems with loop delays. In D. Hristu-Varsakelis and W. S. Levine, eds., *Handbook of Networked and Embedded Control Systems*, pp. 627–648. Birkhäuser, Boston, MA, 2005.
- [455] G. Mitra, C. Lucas, and S. Moody. Tool for reformulating logical forms into zero-one mixed integer programs. *Eur. J. Operational Res.*, **72**(2):262–276, 1994.
- [456] S. Mitra and D. Liberzon. Stability of hybrid automata with average dwell-time: an invariant approach. In *Proc. 43th IEEE Conf. Decision and Control*, pp. 1394–1399, Atlantis, Bahamas, 2004.
- [457] S. Mitra, D. Liberzon, and N. Lynch. Verifying average dwell time of hybrid systems. *ACM Trans. Embedded Comp. Syst.*, **8**(1): article no. 3, 2008.
- [458] MoBIES team. HSIF semantics. Technical report, University of Pennsylvania, Philadelphia, PA 2002.
- [459] R. Mobus, M. Baotic, and M. Morari. Multi-object adaptive cruise control. In O. Maler and A. Pnueli, eds., *Hybrid Systems: Computation and Control*, pp. 359–374. Springer-Verlag, Berlin, 2003.
- [460] B. E. Moision, A. Orlitsky, and P. H. Siegel. On codes that avoid specified differences. *IEEE Trans. Inform. Theor.*, **47**(1):433–442, 2001.
- [461] A. Molchanov and E. Pyatnitskiy. Criteria of asymptotic stability of differential and difference inclusions encountered in control theory. *Syst. Control Lett.*, **13**(1):59–64, 1989.
- [462] M. D. P. Monteiro Marques. *Differential Inclusions in Nonsmooth Mechanical Problems: Shocks and Dry Friction*. Birkhäuser, Basel, 1993.
- [463] T. Moor and J. Raisch. Think continuous, act discrete: DES techniques for continuous systems. In *Proc. 10th Mediterranean Conf. Control and Automation*, p. 67, Lisboa, Portugal, 2002.
- [464] T. Moor and J. Raisch. Supervisory control of hybrid systems within a behavioural framework. *Syst. Control Lett.*, **38**(3):157–166, 1999.
- [465] T. Moor, J. M. Davoren, and B. D. O. Anderson. Robust hybrid control from a behavioural perspective. Technical report, RSISE, Australian National University, 2002. Accepted for presentation at the 41st Int. Conf. on Decision and Control.

- [466] T. Moor, J. Raisch, and S. D. O'Young. Discrete supervisory control of hybrid systems based on l -complete approximations. *J. Discrete Event Dynam. Syst.*, **12**(1):83–107, 2002.
- [467] T. Moor, J. Raisch, and J. M. Davoren. Admissibility criteria for a hierarchical design of hybrid control systems. In *Proc. 1st IFAC Conf. Analysis and Design of Hybrid Systems*, pp. 389–394, Saint-Malo, France, 2003.
- [468] T. Moor, J. M. Davoren, and J. Raisch. Learning by doing: systematic abstraction refinement for hybrid control synthesis. *IEE Proc. Control Theor. Appl.*, **153**(5):591–599, 2006.
- [469] C. Moore. Unpredictability and undecidability in dynamical systems. *Phys. Rev. Lett.*, **64**:2354–2357, 1990.
- [470] M. Morari and J. Lee. Model predictive control: past, present and future. *Comp. Chem. Eng.*, **23**(4):667–682, 1999.
- [471] M. Morari, J. Buisson, B. de Schutter, and G. Papafotiou. Report on the assessment of hybrid control methods for electric energy management problems. Technical report, HYCON Deliverable, 2006.
- [472] M. Morari, J. Buisson, B. de Schutter, and G. Papafotiou. Final report on modeling tools, benchmarks and control methods. Technical report, HYCON Deliverable, 2006.
- [473] T. M. Y. Mori and Y. Kuroe. A solution to the common Lyapunov function problem for continuous-time systems. In *Proc. 36th IEEE Conf. Decision and Control*, vol. 4, pp. 3530–3531, San Diego, LA, 1997.
- [474] P. Morin and C. Samson. Robust stabilization of driftless systems with hybrid open-loop/feedback control. In *Proc. 2000 American Control Conf.*, pp. 3929–3933, Chicago, IL, 2000.
- [475] A. S. Morse. Supervisory control of families of linear set-point controllers—part 1: exact matching. *IEEE Trans. Automat. Control*, **41**(10):1413–1431, 1996.
- [476] P. J. Mosterman. HyBrSim—A modeling and simulation environment for hybrid bond graphs. *J. Syst. Control Eng.*, **216**(1):35–46, 2002.
- [477] P. J. Mosterman. An overview of hybrid simulation phenomena and their support by simulation packages. In *Hybrid Systems: Computation and Control*, pp. 165–177, Springer-Verlag, Berlin, 1999.
- [478] P. J. Mostermann, J. Sztipanovits, and S. Engell. Computer automated multi-paradigm modeling in control systems technology. *IEEE Trans. Control Syst. Technol.*, **12**(2):223–234, 2004.
- [479] E. Münz and V. Krebs. Continuous optimization approaches to the identification of piecewise affine systems. In *Proc. 16th IFAC World Congress*, Prague, Czech Republic, 2005.
- [480] G. N. Nair and R. J. Evans. Stabilizability of stochastic linear systems with finite feedback data rates. *SIAM J. Control Optim.*, **43**(2):413–436, 2004.
- [481] G. N. Nair, F. Fagnani, S. Zampieri, and R. J. Evans. Feedback control under data rate constraints: an overview. *Proc. IEEE*, **95**(1):108–137, 2007.

- [482] H. Nakada, K. Takaba, and T. Katayama. Identification of piecewise affine systems based on statistical clustering technique. *Automatica*, **41**(5):905–913, 2005.
- [483] K. S. Narendra and J. Balakrishnan. A common Lyapunov function for stable LTI systems with commuting A-matrices. *IEEE Trans. Automat. Control*, **39**(12):2469–2471, 1994.
- [484] I. Necoara, B. De Schutter, T. van den Boom, and H. Hellendoorn. Min-max model predictive control for uncertain max-min-plus-scaling systems. In *Proc. 8th Int. Workshop Discrete Event Systems*, pp. 439–444, Ann Arbor, MI, 2006.
- [485] J. Neidig, C. Falkenberg, J. Lunze, and M. Fritz. Qualitative diagnosis of an automotive air path. *ATP Int. Automation Technol.*, **3**(1):37–42, 2005.
- [486] X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. An approach to the description and analysis of hybrid systems. In R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, eds., *Hybrid Systems*, pp. 149–178. Springer-Verlag, Berlin, 1993.
- [487] J. Nilsson. Real-time control systems with delays. Unpublished PhD thesis, Lund Institute of Technology, Lund, Sweden, 1998.
- [488] J. Nilsson, B. Bernhardsson, and B. Wittenmark. Stochastic analysis and control of real-time systems with random time delays. *Automatica*, **34**(1):57–64, 1998.
- [489] M. Noguchi, Y. Tanaka, T. Tanaka, and Y. Takeuchi. A study on gasoline engine combustion by observation of intermediate reactive products during combustion. SAE Technical Papers 790840, 1979.
- [490] M. Oh and C. C. Pantelides. A modelling and simulation language for combined lumped and distributed parameter systems. *Comp. Chem. Eng.*, **20**(6–7): 611–633, 1996.
- [491] Y. Ohta, H. Imanishi, L. Gong, and H. Haneda. Computer generated Lyapunov functions for a class of nonlinear systems. *IEEE Trans. Circuits Syst. I*, **40**(5): 343–354, 1993.
- [492] J. O. Olsson, P. Tunestål, and B. Johansson. Closed-loop control of an HCCI engine. SAE Technical Papers 2001-01-1031, 2001.
- [493] S. Onishi, S. Hong Jo, K. Shoda, P. Do Jo, and S. Kato. Active thermo-atmosphere combustion (ATAC)—a new combustion process for internal combustion engines. SAE Technical Papers 790501, 1979.
- [494] P. Ortner and L. del Re. Predictive control of a diesel engine air path. *IEEE Trans. Control Syst. Technol.*, **15**(3):449–456, 2007.
- [495] M. Otter, M. Remelhe, S. Engell, and P. Mostermann. Hybrid models of physical systems and discrete controllers. *Automatisierungstechnik*, **48**:426–437, 2000.
- [496] C. M. Özveren and A. S. Willsky. Observability of discrete event dynamic systems. *IEEE Trans. Automat. Control*, **35**(7):797–806, 1990.
- [497] L. Pallottino, V. G. Scordio, E. Frazzoli, and A. Bicchi. Probabilistic verification of a decentralized policy for conflict resolution in multi-agent systems. In *IEEE Int. Conf. Robotics and Automation*, pp. 2448–2453, Orlando, FL, 2006.

- [498] L. Pallottino, V. G. Scordio, E. Frazzoli, and A. Bicchi. Decentralized and scalable conflict resolution strategy for multi-agents systems. In *Proc. 17th Int. Symp. Mathematical Theory of Networks and Systems*, Kyoto, Japan, 2006.
- [499] Y. J. Pan, H. J. Marquez, and T. Chen. Remote stabilization of networked control systems with unknown time varying delays by LMI techniques. In *Proc. 44th IEEE Conf. Decision and Control*, pp. 1589–1594, Seville, Spain, 2005.
- [500] A. Panousopoulou, G. Nikolakopoulos, and A. Tzes. Experimental controller tuning and QoS optimization of a wireless transmission scheme for real-time remote control applications. In *Int. Conf. Industrial Technology*, pp. 807–808, Hammamet, Tunisia, 2004.
- [501] C. C. Pantelides. The consistent initialization of differential-algebraic systems. *SIAM J. Sci. Stat. Comp.*, **9**(2):213–231, 1988.
- [502] C. C. Pantelides. SPEEDUP—recent advances in process simulation. *Comp. Chem. Eng.*, **12**(7):745–755, 1988.
- [503] S. Paoletti. Identification of piecewise affine models. Unpublished PhD thesis, Department of Information Engineering, University of Siena, Siena, Italy, 2004.
- [504] S. Paoletti and J. Roll. PWAID: piecewise affine system identification toolbox, 2007. Available at: <http://www.control.isy.liu.se/~{jroll}/PWAID>.
- [505] S. Paoletti, A.Lj. Juloski, G. Ferrari-Trecate, and R. Vidal. Identification of hybrid systems: a tutorial. *Eur. J. Control*, **513**(2–3):242–260, 2007.
- [506] C. H. Papadimitriou. *Computational complexity*. Addison-Wesley, Washington, DC, 1994.
- [507] G. Pappas. Hybrid system tools Wiki. University of Pennsylvania, Philadelphia, PA, 2008. Available at: <http://wiki.grasp.upenn.edu/graspdoc/wiki/hst>.
- [508] G. J. Pappas, G. Lafferriere, and S. Sastry. Hierarchically consistent control systems. *IEEE Trans. Automat. Control*, **45**(6):1144–1160, 2000.
- [509] P. A. Parrilo and A. Jadbabaie. Approximation of the joint spectral radius using sum of squares. *Linear Algebr. Appl.*, **428**(10):2385–2402, 2008.
- [510] A. Pavlov, N. van de Wouw, and H. Nijmeijer. Convergent piecewise affine systems: analysis and design. In *Proc. 44th IEEE Conf. Decision and Control*, pp. 5391–5396, Seville, Spain, 2005.
- [511] P. Peleties and R. A. DeCarlo. Asymptotic stability of m -switched systems using Lyapunov-like functions. In *Proc. 1991 American Control Conf.*, pp. 1679–1684, Boston, MA, 1991.
- [512] D. L. Pepyne and C. G. Cassandras. Modeling, analysis, and optimal control of a class of hybrid systems. In *Discrete Event Dynamic Systems: Theory and Applications*, pp. 175–201, 1998.
- [513] M. A. Pereira Remelhe and S. Engell. Modeling and simulation of large-scale hybrid systems. In *Encyclopedia of Life Support Systems*. EOLSS Publishers, Oxford, 2004.

- [514] M. A. Pereira Remelhe and S. Engell. Combining Modelica models with discrete event formalisms for simulation using the DES/M environment. *Int. J. Software Eng. Knowledge Eng.*, **2**(15):349–355, 2005.
- [515] C. De Persis. Nonlinear stabilizability via encoded feedback: the case of integral ISS systems. *Automatica*, **42**(10):1813–1816, 2006.
- [516] S. Pettersson. Switched state jump observers for switched systems. In *Proc. 16th IFAC World Congress*, p/n Tu-F12-TO/5, Prague, Czech Republic, 2005.
- [517] S. Pettersson and B. Lennartson. Stability analysis of hybrid systems—a gearbox application. In R. Johansson and A. Rantzer, eds., *Nonlinear and Hybrid Systems in Automotive Control*, pp. 373–389. Springer-Verlag, Heidelberg, 2003.
- [518] L. R. Petzold. A description of DASSL—a differential/algebraic system solver. In *Proc. 10th World Congress System Simulation and Scientific Computation*, pp. 430–432, Montreal, Canada, 1983.
- [519] B. Picasso. Stabilization of quantized linear systems: analysis, synthesis, performance and complexity. Unpublished PhD thesis, Scuola Normale Superiore di Pisa, Pisa, Italy, 2008.
- [520] B. Picasso and A. Bicchi. On the stabilization of linear systems under assigned I/O quantization. *IEEE Trans. Automat. Control*, **52**(10):1994–2000, 2007.
- [521] B. Picasso, S. Pancanti, A. Bemporad, and A. Bicchi. Receding horizon control of LTI systems with quantized inputs. In *Proc. 1st IFAC Conf. Analysis and Design of Hybrid Systems*, pp. 259–264, Saint-Malo, France, 2003.
- [522] B. Picasso, L. Palopoli, A. Bicchi, and K. H. Johansson. Control of distributed embedded systems in the presence of unknown-but-bounded noise. In *Proc. 43th IEEE Conf. Decision and Control*, pp. 1448–1453, Paradise Islands, Bahamas, 2004.
- [523] B. Piccoli. Hybrid systems and optimal control. In *Proc. 37th IEEE Conf. Decision and Control*, vol. 1, pp. 13–18, Tampa, FL, 1998.
- [524] B. Piccoli. Necessary conditions for hybrid optimization. In *Proc. 38th IEEE Conf. Decision and Control*, vol. 1, pp. 410–415, Phoenix, AZ, 1999.
- [525] M. J. H. Pijpers, H. Preisig, and M. Weiss. A discrete modelling procedure for continuous processes based on state discretization. In *Proc. 2nd MATHMOD Conf.*, pp. 189–194, Vienna, Italy, 1997.
- [526] A. Pinto, L. P. Carloni, R. Passerone, and A. L. Sangiovanni-Vincentelli. Interchange format for hybrid systems: abstract semantics. In J. P. Hespanha and A. Tiwari, eds., *Hybrid Systems: Computation and Control*, pp. 491–506. Springer-Verlag, Santa Barbara, CA, 2006.
- [527] A. Pnueli. The temporal logic of programs. In *Proc. 18th Annual Symp. Foundations of Computer Science*, pp. 46–57. IEEE Computer Science Press, New York, 1977.
- [528] A. Podelski and A. Rybalchenko. ARMC: the logical choice for software model checking with abstraction refinement. In *Proc. 9th Int. Symp. Practical Aspects of Declarative Languages*, pp. 245–259, Nice, France, 2007.

- [529] A. Podelski and S. Wagner. Model checking of hybrid systems: from reachability towards stability. In J. P. Hespanha and A. Tiwari, eds., *Hybrid Systems: Computation and Control*, pp. 507–521. Springer-Verlag, Berlin, 2006.
- [530] A. Podelski and S. Wagner. A sound and complete proof rule for region stability of hybrid systems. In A. Bemporad, A. Bicchi, and G. Butazzo, eds., *Hybrid Systems: Computation and Control*, pp. 750–753. Springer-Verlag, Berlin, 2007.
- [531] A. Podelski and S. Wagner. Region stability proofs for hybrid systems. In J. F. Raskin and P. S. Thiagarajan, eds., *Proc. FORMATS 2007*, pp. 320–335. Springer-Verlag, Berlin, 2007.
- [532] A. Pogromsky, W. P. M. H. Heemels, and H. Nijmeijer. On solution concepts and well-posedness of linear relay systems. *Automatica*, **39**:2139–2147, 2003.
- [533] G. Pola, M. L. Bujorianu, J. Lygeros, and M. D. Di Benedetto. Stochastic hybrid models: an overview with applications to air traffic management. In *Proc. 1st IFAC Conf. Analysis and Design of Hybrid Systems*, pp. 45–50, Saint-Malo, France, 2003.
- [534] L. S. Pontryagin, V. G. Boltyanskiĭ, R. V. Gamkrelidze, and E. F. Mishchenko. *Matematicheskaya Teoriya Optimalnykh Protsessov*, 4th edn. Nauka, Moscow, 1983.
- [535] C. de Prada, S. Cristea, and J. J. Rosano. Optimal start-up of an evaporation station. In *Proc. 8th Int. Symp. Dynamics and Control of Process Systems*, pp. 115–120, Cancun, Mexico, 2007.
- [536] S. Prajna, A. Jadbabaie, and G. J. Pappas. A framework for worst-case and stochastic safety verification using barrier certificates. *IEEE Trans. Automat. Control*, **52**(8):1415–1428, 2007.
- [537] M. Prandini and J. Hu. A stochastic approximation method for reachability computations. In H. A. P. Blom and J. Lygeros, eds., *Stochastic Hybrid Systems: Theory and Safety Applications*, pp. 107–139. Springer-Verlag, Berlin, 2006.
- [538] M. Prandini and J. Hu. Stochastic reachability: theory and numerical approximation. In C. G. Cassandras and J. Lygeros, eds., *Stochastic Hybrid Systems*, pp. 107–138. Taylor & Francis Group/CRC Press, Boca Raton, FL, 2006.
- [539] M. Prandini, J. Hu, J. Lygeros, and S. Sastry. A probabilistic approach to aircraft conflict detection. *IEEE Trans. Intelligent Transportation Syst.*, **1**(4):199–220, 2000.
- [540] C. Prieur. Asymptotic controllability and robust asymptotic stabilizability. *SIAM J. Control Optim.*, **43**:1888–1912, 2005.
- [541] C. Prieur and A. Astolfi. Robust stabilization of chained systems via hybrid control. *IEEE Trans. Automat. Control*, **48**(10):1768–1772, 2003.
- [542] C. Prieur and E. Trélat. Quasi-optimal robust stabilization of control systems. *SIAM J. Control Optim.*, **45**(5):1875–1897, 2006.
- [543] C. Prieur, R. Goebel, and A. R. Teel. Hybrid feedback control and robust stabilization of nonlinear systems. *IEEE Trans. Automat. Control*, **52**(11):2103–2117, 2007.

- [544] V. Y. Protasov. The generalized spectral radius: a geometric approach. *IzvestiyaMathematika*, **61**(5):995–1030, 1997.
- [545] A. Puri, P. Varaiya, and V. Borkar. ϵ -approximation of differential inclusions. In *Proc. 34th IEEE Conf. Decision and Control*, pp. 2892–2897, New Orleans, LA, 1995.
- [546] D. E. Quevedo, G. C. Goodwin, and J. A. De Doná. Finite constraint set receding horizon control. *Int. J. Robust Nonlinear Control*, **14**(4):355–377, 2004.
- [547] A. Raghunathan and L. Biegler. Mathematical programs with equilibrium constraints (MPEC) in process engineering. *Comp. Chem. Eng.*, **27**(10): 1381–1392, 2003.
- [548] J. Raisch. Nondeterministic automata as approximations for continuous systems—an approach with an adjustable degree of accuracy. In *IMACS Symp. Mathematical Modelling*, pp. 195–202, Vienna, Austria, 1997.
- [549] J. Raisch and T. Moor. Hierarchical hybrid control of a multiproduct batch Plant. In *Control and Observer Design for Nonlinear Finite and Infinite Dimensional Systems*, pp. 199–216. Springer-Verlag, Berlin, 2005.
- [550] J. Raisch and S. D. O’Young. Discrete approximation and supervisory control of continuous systems. *IEEE Trans. Automat. Control*, **43**:569–573, 1998.
- [551] P. J. Ramadge and W. M. Wonham. Supervisory control of a class of discrete event systems. *SIAM J. Control Optim.*, **25**:206–230, 1987.
- [552] P. J. Ramadge and W. M. Wonham. The control of discrete event systems. *Proc. IEEE*, **77**:81–98, 1989.
- [553] R. Raman and I. E. Grossmann. Relation between MILP modeling and logical inference for chemical process synthesis. *Comp. Chem. Eng.*, **15**(2):73–84, 1991.
- [554] A. Rantzer and M. Johansson. Piecewise linear quadratic optimal control. *IEEE Trans. Automat. Control*, **45**(4):629–637, 2000.
- [555] J. B. Rawlings. Tutorial overview of model predictive control. *IEEE Control Syst. Magazine*, **20**(3):38–52, 2000.
- [556] J. Richalet, A. Rault, J. L. Testud, and J. Papon. Model predictive heuristic control: applications to industrial processes. *Automatica*, **14**(5):413–428, 1978.
- [557] P. Riedinger, F. Kratz, C. Iung, and C. Zanne. Linear quadratic optimization for hybrid systems. In *Proc. 38th IEEE Conf. Decision and Control*, vol. 3, pp. 3059–3064, Phoenix, AZ, 1999.
- [558] Y. Rogozhin. Small universal Turing machines. *Theor. Comp. Sci.*, **168**:215–240, 1996.
- [559] J. Roll, A. Bemporad, and L. Ljung. Identification of piecewise affine systems via mixed-integer programming. *Automatica*, **40**(1):37–50, 2004.
- [560] M. Rubensson. Stability properties of switched dynamical systems—a linear matrix inequality approach. Unpublished PhD thesis, Chalmers University of Technology, Göteborg, Sweden, 2003.
- [561] J. Rufino. An overview of the controller area network. In *Proc. CiA Forum—CAN for Newcomers*, Braga, Portugal, 1997.

- [562] R. W. Saaty. The analytic hierarchy process—what it is and how it is used. *Mathematical Modelling*, **9**(3–5):161–176, 1987.
- [563] S. Sager, H. G. Bock, M. Diehl, G. Reinelt, and J. P. Schlöder. Numerical methods for optimal control with binary control functions applied to a Lotka-Volterra type fishing problem. In A. Seeger, ed., *Recent Advances in Optimization, Proc. 12th French-German-Spanish Conf. Optimization*, pp. 269–289. Springer-Verlag, Berlin, 2006.
- [564] T. Saito. A chaos generator based on a quasi-harmonic oscillator. *IEEE Trans. Circuits Syst.*, **32**(4):320–331, 1985.
- [565] A. L. Sangiovanni-Vincentelli. Defining platform-based design. *EEdesign*, online, 2002.
- [566] A. L. Sangiovanni-Vincentelli, L. Carloni, F. De Bernardinis, and M. Sgroi. Benefits and challenges for platform-based design. In *Proc. Design Automation Conf.*, San Diego, CA, 2004.
- [567] E. De Santis, M. D. Di Benedetto, and G. Pola. On observability and detectability of continuous-time linear switching systems. In *Proc. 42th IEEE Conf. Decision and Control*, pp. 5777–5782, Maui, USA, 2003.
- [568] E. De Santis, M. D. Di Benedetto, S. Di Gennaro, A. D’Innocenzo, and G. Pola. Critical observability of a class of hybrid systems and application to air traffic management. In H. A. P. Blom and J. Lygeros, eds., *Stochastic Hybrid Systems: Theory and Safety Applications*, pp. 141–170. Springer-Verlag, Berlin, 2005.
- [569] E. De Santis, M. D. Di Benedetto, and G. Pola. Digital idle speed control of automotive engine: a safety problem for hybrid systems. *Nonlinear Anal.*, **65**:1705–1724, 2006.
- [570] D. Sarabia, F. Capraro, L. F. S. Larsen, and C. de Prada. Hybrid control of a supermarket refrigeration system. In *Proc. 2007 American Control Conf.*, pp. 4178–4185, New York, 2007.
- [571] A. J. van der Schaft and H. Schumacher. *An Introduction to Hybrid Dynamical Systems*. Springer-Verlag, London, 2002.
- [572] A. J. van der Schaft and J. M. Schumacher. Complementarity modelling of hybrid systems. *IEEE Trans. Automat. Control*, **43**(4):483–490, 1998.
- [573] A. J. van der Schaft and J. M. Schumacher. The complementary-slackness class of hybrid systems. *Math. Control Signals Syst.*, **9**(3):266–301, 1996.
- [574] A. Schild and J. Lunze. Stabilization of limit cycles of discretely controlled continuous systems by controlling switching surfaces. In *Hybrid Systems: Computation and Control*, pp. 515–528. Springer-Verlag, Berlin, 2007.
- [575] A. Schild and J. Lunze. Switching surface design for periodically operated discretely controlled continuous systems. In *Hybrid Systems: Computation and Control*, pp. 471–485. Springer-Verlag, Berlin, 2008.
- [576] J. Schröder. *Modelling, State Observation, and Diagnosis of Quantized Systems*. Springer-Verlag, Berlin, 2003.
- [577] B. De Schutter. Optimal control of a class of linear hybrid systems with saturation. *SIAM J. Control Optim.*, **39**(3):835–851, 2000.

- [578] B. De Schutter and T. J. J. van den Boom. Model predictive control for max-min-plus-scaling systems. In *Proc. 2001 American Control Conf.*, pp. 319–324, Arlington, VA, 2001.
- [579] B. De Schutter and T. J. J. van den Boom. MPC for continuous piecewise-affine systems. *Syst. Control Lett.*, **52**(3–4):179–192, 2004.
- [580] M. Sgroi, A. Wolisz, A. Sangiovanni-Vincentelli, and J. Rabaey. A service-based universal application interface for ad-hoc wireless sensor networks. Technical report, Berkeley Wireless Research Center, Berkeley, CA, 2003.
- [581] G. M. Shaver, M. Roelle, and J. C. Gerdes. Decoupled control of combustion timing and work output residual-affected HCCI engines. In *Proc. 2005 American Control Conf.*, pp. 3871–3876, Portland, OR, 2005.
- [582] G. M. Shaver, M. J. Roelle, and J. C. Gerdes. Modeling cycle-to-cycle dynamics and mode transition in HCCI engines with variable valve actuation. *Control Eng. Practice*, **14**(3):213–222, 2006.
- [583] J. Shen and J. S. Pang. Linear complementarity systems: Zeno states. *SIAM J. Control Optim.*, **44**(3):1040–1066, 2005.
- [584] J. Shen and J. S. Pang. Semicopositive linear complementarity systems. *Int. J. Robust Nonlinear Control*, **17**(15):1367–1386, 2007.
- [585] R. Shorten and K. Narendra. Necessary and sufficient conditions for the existence of a CQLF for a finite number of stable LTI systems. *J. Adaptive Control Signal Processing*, **16**(10):709–728, 2002.
- [586] R. Shorten, F. Wirth, O. Mason, K. Wulff, and C. King. Stability criteria for switched and hybrid systems. *SIAM Review*, **49**(4): 545–592, 2007.
- [587] H. Siegelmann and E. Sontag. On the computational power of neural nets. *J. Comp. Syst. Sci.*, **50**(1):132–150, 1992.
- [588] H. T. Siegelmann and E. D. Sontag. Turing computability with neural nets. *Appl. Math. Lett.*, **4**(6):77–80, 1991.
- [589] B. I. Silva, K. Richeson, B. Krogh, and A. Chutinan. Modeling and verifying hybrid dynamic systems using CheckMate. In *Proc. 4th Int. Conf. Automation of Mixed Processes*, pp. 323–328, Dortmund, Germany, 2000.
- [590] S. N. Simić, K. H. Johansson, J. Lygeros, and S. Sastry. Structural stability of hybrid systems. In *Proc. 2001 European Control Conf.*, pp. 3858–3863, Porto, Portugal, 2001.
- [591] E. Skafidas, R. J. Evans, A. V. Savkin, and I. R. Petersen. Stability results for switched controller systems. *Automatica*, **35**(4):553–564, 1999.
- [592] F. Song and S. M. Smith. Cell-state-space-based search. *IEEE Control Syst. Magazine*, **22**(4):42–56, 2002.
- [593] C. Sonntag and O. Stursberg. Optimally controlled start-up of a multi-stage evaporation system. Technical report, Technische Universität Dortmund, Dortmund, 2005.
- [594] C. Sonntag, A. Devanathan, S. Engell, and O. Stursberg. Hybrid nonlinear model-predictive control of a supermarket refrigeration system. In *Proc. IEEE Multi-Conf. Syst. Control*, pp. 1432–1437, Suntec City, Singapore, 2007.

- [595] C. Sonntag, A. Devanathan, and S. Engell. Hybrid NMPC of a supermarket refrigeration system using sequential optimization. In *Proc. 17th IFAC World Congress*, vol. 17, part 1, Seoul, Korea, 2008.
- [596] C. Sonntag, W. Su, O. Stursberg, and S. Engell. Optimized start-up control of an industrial-scale evaporation system with hybrid dynamics. *Control Eng. Practice*, **16**(8):976–990, 2008.
- [597] E. D. Sontag. On the observability of polynomial systems, I: finite-time problems. *SIAM J. Control Optim.*, **17**(1):139–151, 1979.
- [598] E. D. Sontag. Nonlinear regulation: the piecewise linear approach. *IEEE Trans. Automat. Control*, **26**(2):346–357, 1981.
- [599] E. D. Sontag. Interconnected automata and linear systems: a theoretical framework in discrete-time. In R. Alur, T. A. Henzinger, and E. D. Sontag, eds., *Hybrid Systems III*, pp. 436–448. Springer-Verlag, Berlin, 1996.
- [600] E. D. Sontag. *Mathematical Control Theory: Deterministic Finite Dimensional Systems*. Springer-Verlag, New York, 1998.
- [601] A. Speranzon, C. Fischione, and K. H. Johansson. Distributed and collaborative estimation over wireless sensor networks. In *Proc. 45th IEEE Conf. Decision and Control*, pp. 1025–1030, San Diego, CA, 2006.
- [602] J. Sreedhar and P. Van Dooren. Pole placement via the periodic Schur decomposition. In *Proc. 1993 American Control Conf.*, pp. 1563–1567, San Fransisco, CA, 1993.
- [603] J. Sreedhar and P. Van Dooren. An orthogonal method for the controllable subspace of a periodic system. In *Proc. Conf. Information Sciences & Systems*, pp. 174–177, Baltimore, MD, 1993.
- [604] O. Stein, J. Oldenburg, and W. Marquardt. Continuous reformulations of discrete-continuous optimization problems. *Comp. Chem. Eng.*, **28**(10): 1951–1966, 2004.
- [605] C. Stirling. Bisimulation and language equivalence. In *Logic for Concurrency and synchronisation*, pp. 269–284. Kluwer, Norwell, MA, 2003.
- [606] J. A. Stiver, P. J. Antsaklis, and M. D. Lemmon. Interface and controller design for hybrid systems. In P. J. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, eds., *Hybrid Systems II*, pp. 462–492. Springer-Verlag, Berlin, 1995.
- [607] G. Stremersch and R. K. Boel. On the influencing net and forbidden state control of timed Petri nets with forced transitions. In *Proc. 37th IEEE Conf. Decision and Control*, pp. 3287–3292, Tampa, FL, 1998.
- [608] O. Stursberg. A graph search algorithm for optimal control of hybrid systems. In *Proc. 43th IEEE Conf. Decision and Control*, pp. 1412–1417, Paradise Islands, Bahamas, 2004.
- [609] O. Stursberg, S. Kowalewski, and S. Engell. Generating timed discrete models of continuous systems. In *Proc. 2nd MATHMOD Conf.*, pp. 203–209, Vienna, Italy, 1997.
- [610] R. Su, S. Abdelwahed, and S. Neema. Computing finitely reachable containable region for switching systems. *IEE Proc. Control Theory Appl.*, **152**(4): 477–486, 2005.
- [611] Z. Sun and S. S. Ge. *Switched Linear Systems*. Springer-Verlag, Berlin, 2005.

- [612] Z. Sun, S. Ge, and T. Lee. Controllability and reachability criteria for switched linear control systems. *Automatica*, **38**(5):775–786, 2002.
- [613] H. J. Sussmann. A nonsmooth hybrid maximum principle. In *Stability and Stabilization of Nonlinear Systems*, pp. 325–354. Springer-Verlag, London, 1999.
- [614] H. Suzuki, N. Koike, H. Ishii, and M. Odaka. Exhaust purification of diesel engines by homogeneous charge with compression ignition. SAE Technical Papers 970315, 1997.
- [615] S. C. Tatikonda and S. Mitter. Control under communication constraints. *IEEE Trans. Automat. Control*, **49**(7):1056–1068, 2004.
- [616] S. C. Tatikonda, A. Sahai, and S. Mitter. Stochastic linear control over a communication channel. *IEEE Trans. Automat. Control*, **49**(9):1549–1561, 2004.
- [617] L. Tavernini. Differential automata and their discrete simulators. *Nonlinear Anal. Theor. Meth. Appl.*, **11**(6):665–683, 1987.
- [618] M. Tawarmalani and N. V. Sahinidis. A polyhedral branch-and-cut approach to global optimization. *Math. Prog.*, **103**(2):225–249, 2005.
- [619] A. R. Teel and R. Goebel. Lyapunov characterization of Zeno behavior in hybrid systems. In *Proc. 47th IEEE Conf. Decision and Control*, pp. 2752–2757, Cancun, Mexico, 2008.
- [620] R. Tempo, G. Calafiore, and F. Dabbene. Randomized algorithms for analysis and control of uncertain systems. In *Hybrid Systems IV: Computation and Control*. Springer-Verlag, Berlin, 2003.
- [621] The MathWorks, Inc. Using Simulink, version 6, 2005. Available at: <http://www.mathworks.com>.
- [622] J. Till, S. Engell, S. Panek, and O. Stursberg. Applied hybrid system optimization: an empirical investigation of complexity. *Control Eng. Practice*, **12**(10): 1291–1303, 2004.
- [623] M. Tiller. *Introduction to Physical Modeling with Modelica*. Springer-Verlag, New York, 2001.
- [624] Y. Tipsuwan and M. Y. Chow. Control methodologies in networked control systems. *Control Eng. Practice*, **11**(10):1099–1111, 2003.
- [625] O. Tokar. On the algorithmic unsolvability of some stability problems for discrete event systems. In *Proc. 13th IFAC World Congress*, pp. 353–358, 1996.
- [626] J. Tolsma and P. I. Barton. DAEPACK: an open modeling environment for legacy models. *Industrial Eng. Chem. Res.*, **39**(6):1826–1839, 2000.
- [627] J. E. Tolsma, J. A. Clabaugh, and P. I. Barton. Symbolic incorporation of external procedures into process modeling environments. *Industrial Eng. Chem. Res.*, **41**(16):3867–3876, 2002.
- [628] T. Tometzki, O. Stursberg, C. Sonntag, and S. Engell. Optimizing hybrid dynamic processes by embedding genetic algorithms into MPC. In *Proc. IFAC Int. Symp. ADCHEM*, pp. 977–982, 2006.

- [629] C. Tomlin, G. Pappas, J. Lygeros, D. Godbole, and S. Sastry. Hybrid models of next generation of air traffic management. In *Hybrid Systems: Computation and Control*, pp. 378–404. Springer-Verlag, Berlin, 1997.
- [630] P. Tøndel, T. A. Johansen, and A. Bemporad. Evaluation of piecewise affine control via binary search tree. *Automatica*, **39**(5):945–950, 2003.
- [631] P. Tøndel, T. A. Johansen, and A. Bemporad. An algorithm for multi-parametric quadratic programming and explicit MPC solutions. *Automatica*, **39**(3):489–497, 2003.
- [632] F. D. Torrisi and A. Bemporad. HYSDEL—a tool for generating computational hybrid models. *IEEE Trans. Control Syst. Technol.*, **12**(2):235–249, 2004.
- [633] F. D. Torrisi, A. Bemporad, G. Bertini, P. Hertach, D. Jost, and D. Mignone. HYSDEL 2.0.5—user manual. Technical Report AUT02-28, Automatic Control Laboratory, ETH Zurich, Switzerland, 2002.
- [634] J. N. Tsitsiklis. The stability of the products of a finite set of matrices. In T. M. Cover and B. Gopinath, eds., *Open Problems in Communication and Computation*. Springer-Verlag, New York, 1987.
- [635] J. N. Tsitsiklis. On the stability of asynchronous iterative processes. *Math. Syst. Theor.*, **20**:137–153, 1987.
- [636] T. Tsubone and T. Saito. Manifold piecewise constant systems and chaos. *IEICE Trans. Fundamentals*, **E82-A**:1619–1626, 1989.
- [637] P. Tunestål. The use of cylinder pressure for estimation of the in-cylinder air/fuel ratio of an internal combustion engine. Unpublished PhD thesis, University of California, Department of Mechanical Engineering, Berkeley, CA, 2000.
- [638] Uppsala University and Aalborg University. UPPAAL version 4.0 online help, 2006. Available at: <http://www.it.uu.se/research/group/darts/uppaal/index.shtml>.
- [639] V. I. Utkin. Variable structure systems with sliding modes. *IEEE Trans. Automat. Control*, **22**(1):31–45, 1977.
- [640] V. I. Utkin. *Sliding Regimes in Optimization and Control Problems*. Nauka, Moscow, 1981.
- [641] L. Vandenberghe, B. L. De Moor, and J. Vandewalle. The generalized linear complementarity problem applied to the complete analysis of resistive piecewise-linear circuits. *IEEE Trans. Circuits Syst.*, **36**(11):1382–1391, 1989.
- [642] V. Vapnik. *Statistical Learning Theory*. John Wiley, New York, 1998.
- [643] A. Varga. Robust minimum norm Pole assignment with periodic state feedback. *IEEE Trans. Automat. Control*, **45**(5):1017–1022, 2000.
- [644] M. Vašak, M. Baotić, M. Morari, I. Petrović, and N. Perić. Constrained optimal control of an electronic throttle. *Int. J. Control*, **79**(5):465–478, 2006.
- [645] V. S. Vassiliadis, R. W. H. Sargent, and C. C. Pantelides. Solution of a class of multistage dynamic optimization problems—1. Problems without path constraints. *Industrial Eng. Chem. Res.*, **33**(9):2111–2122, 1994.
- [646] V. S. Vassiliadis, R. W. H. Sargent, and C. C. Pantelides. Solution of a class of multistage dynamic optimization problems—2. Problems with path constraints. *Industrial Eng. Chem. Res.*, **33**(9):2123–2133, 1994.

- [647] S. M. Veres. Geometric bounding toolbox (GBT) for MATLAB, 2003. Available at: <http://www.sysbrain.com>.
- [648] R. Vidal. Identification of PWARX hybrid models with unknown and possibly different orders. In *Proc. 2004 American Control Conf.*, pp. 547–552, Boston, MA, 2004.
- [649] R. Vidal, A. Chiuso, S. Soatto, and S. Sastry. Observability of linear hybrid systems. In A. Pnueli and O. Maler, eds., *Hybrid Systems: Computation and Control*, pp. 526–539. Springer-Verlag, Berlin, 2003.
- [650] R. Vidal, S. Soatto, Y. Ma, and S. Sastry. An algebraic geometric approach to the identification of a class of linear hybrid systems. In *Proc. 42th IEEE Conf. Decision and Control*, pp. 167–172, Maui, USA, 2003.
- [651] R. Vidal, S. Soatto, and S. Sastry. An algebraic geometric approach for identification of linear hybrid systems. In *Proc. 42th IEEE Conf. Decision and Control*, Maui, USA, 2003.
- [652] R. Vidal, A. Chiuso, and S. Soatto. Applications of hybrid system identification in computer vision. In *Proc. 2007 European Control Conf.*, Kos, Greece, 2007.
- [653] A. A. Vladimirov, L. Elsner, and W. J. Beyn. Stability and paracontractivity of discrete linear inclusions. *Linear Algebr. Appl.*, **312**:125–134, 2000.
- [654] G. Walsh, O. Beldiman, and L. Bushnell. Asymptotic behavior of nonlinear networked control systems. *IEEE Trans. Automat. Control*, **46**(7):1093–1097, 2001.
- [655] A. Wächter and L. T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Math. Programming*, **106**(1):25–58, 2006.
- [656] C. Weise and D. Lenzkes. Efficient scaling-invariant checking of timed bisimulation. In *STACS*, pp. 177–188. Springer-Verlag, Berlin, 1997.
- [657] A. Wernrud. Computation of approximate value functions for constrained control problems. In *Proc. 17th Int. Symp. Mathematical Theory of Networks and Systems*, Kyoto, Japan, 2006.
- [658] A. Wernrud. Approximate dynamic programming with applications. Unpublished PhD thesis, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden, 2008.
- [659] M. Wicks and R. DeCarlo. Solution of coupled Lyapunov equations for the stabilization of multimodal linear systems. In *Proc. 1997 American Control Conf.*, pp. 1709–1713, Albuquerque, New Mexico, 1997.
- [660] M. Wicks, P. Peleties, and R. DeCarlo. Switched controller synthesis for the quadratic stabilization of a pair of unstable linear systems. *Eur. J. Control*, **7**: 140–147, 1998.
- [661] J. C. Willems. Models for dynamics. *Dynamics Reported*, **2**:172–269, 1989.
- [662] J. C. Willems. Paradigms and puzzles in the theory of dynamic systems. *IEEE Trans. Automat. Control*, **36**:258–294, 1991.
- [663] J. C. Willems. Dissipative dynamical systems. *Arch. Rational Mech. Anal.*, **45**:321–393, 1972.

- [664] H. P. Williams. *Model Building in Mathematical Programming* 3rd edn. John Wiley & Sons, Chichester 1993.
- [665] H. S. Witsenhausen. A class of hybrid-state continuous-time dynamic systems. *IEEE Trans. Automat. Control*, **11**(2):161–167, 1966.
- [666] K. Wöllhaf, M. Fritz, C. Schulz, and S. Engell. BaSiP–batch process simulation with dynamically reconfigured process dynamics. *Comp. Chem. Eng.*, **20**:1281–1286, 1996.
- [667] K. C. Wong and W. M. Wonham. Hierarchical control of discrete-event systems. *Discrete Event Dynam. Syst.*, **6**:241–306, 1996.
- [668] W. Wong and R. Brockett. Systems with finite communication bandwidth constraints–part II: stabilization with limited information feedback. *IEEE Trans. Automat. Control*, **44**(5):1049–1053, 1999.
- [669] H. Wong-Toi. The synthesis of controllers for linear hybrid automata. In *Proc. 36th IEEE Conf. Decision and Control*, pp. 4607–4612, San Diego, CA, 1997.
- [670] K. Wulff, R. Shorten, and P. Curran. On the 45 degree region and the uniform asymptotic stability of classes of second order parameter varying and switched systems. *Int. J. Control*, **75**:812–823, 2002.
- [671] L. Xiao, A. Hassibi, and J. How. Control with random communication delays via a discrete-time jump system approach. In *Proc. 2000 American Control Conf.*, pp. 2199–2204, Chicago, IL, 2000.
- [672] X. Xu and P. J. Antsaklis. Optimal control of switched systems based on parameterization of the switching instants. *IEEE Trans. Automat. Control*, **49**(1):2–16, 2004.
- [673] X. Xu and P. J. Antsaklis. Stabilization of second-order LTI switched systems. *Int. J. Control*, **73**(14):1261–1279, 2000.
- [674] X. Xu and P. J. Antsaklis. A dynamic programming approach for optimal control of switched systems. In *Proc. 39th IEEE Conf. Decision and Control*, vol. 2, pp. 1822–1827, Sydney, Australia, 2000.
- [675] X. Xu, G. Zhai, and S. He. Stabilizability and practical stabilizability of continuous-time switched systems: a unified view. In *Proc. 2007 American Control Conf.*, New York, 2007.
- [676] H. Ye, A. N. Michel, and L. Hou. Stability theory for hybrid dynamical systems. *IEEE Trans. Automat. Control*, **43**(4):461–474, 1998.
- [677] D. Zambrano, C. Bordons, W. García-Gabín, and E. F. Camacho. A solar cooling plant: a benchmark for hybrid systems control. In C. G. Cassandras, A. Giua, C. Seatzu, and J. Zaytoon, eds., *Proc. 2nd IFAC Conf. Analysis and Design of Hybrid Systems*, pp. 199–204, Alghero, Italy, 2006.
- [678] D. Zambrano, C. Bordons, W. García-Gabín, and E. F. Camacho. Model development and validation of a solar cooling plant. *Int. J. Refrigeration*, **31**(2):315–327, 2007.
- [679] G. Zhai, B. Hu, K. Yasuda, and A. N. Michel. Qualitative analysis of discrete-time switched systems. In *Proc. 2002 American Control Conf.*, vol. 3, pp. 1880–1885, Anchorage, AK, 2002.
- [680] J. Zhang, K. H. Johansson, J. Lygeros, and S. Sastry. Zeno hybrid systems. *Int. J. Robust Nonlinear Control*, **11**:435–451, 2001.



- [681] W. Zhang. Stability analysis of networked control systems. Unpublished PhD thesis, CWRU, 2001.
- [682] W. Zhang, M. Branicky, and S. Phillips. Stability of networked control systems. *IEEE Control Syst. Magazine*, pp. 622–629, 2001.
- [683] J. Zheng. General lemmas for stability analysis of linear continuous-time systems with slowly time-varying parameters. *Int. J. Control*, **58**(6):1437–1444, 1993.

controlengineers.ir

Index

- χ , 55, 347
 2-counter machine, 134
20-sim, 55, 329, 346, 358
- ABACUSS**, 329, 346, 353
ABS, *see* anti-lock braking system
 abstraction, 62, 71, 72, 193, 228, 424, 443
 - a. refinement, 286
 - discrete a., 43, 230
 abstraction-based control, 224, 413
 abstraction-based modeling, 218
 accumulation point, 176, 179, 202
 activation duration, 27, 235
 active damping, 449
 active index set, 152
 active suspension, 453
 activity, 15
 activity function, 59
 adaptive cruise control, 453
 adjoint pair, 80
 affine dynamics, 36
 agent, 473
 - autonomous a., 474
 aggregation, 414
 air traffic control, 276
 air traffic management, 250
 airpath control, 454
 algebraic loop, 363
 algebraic optimization, 411
 algebraic procedure, 104
AMPL, 299, 354
 analysis, 130
 - controllability a., 52, 166
 - observability a., 52
 - reachability a., 64, 249, 264, 299, 308, 312
 - stability a., 52, 72, 121, 299
 anti-lock braking system, 441
 anti-windup, 391, 406
 application
 - air traffic control, 276
 - boost converter, 384
 - DC-DC converter, 379
 - Diesel engine, 221
 - electricity distribution, 393
 - Lake Mead, 287, 295
 - mining ventilation, 495
 - multiproduct batch plant, 419
 - power network, 393
 - solar air conditioning system, 502
 approximation, 458
 - discrete-time a., 399
 - robust affine a., 386
 architecture-specific format, 321, 324
ASCET, 444
 asynchronous communication, 442
 automatic gearbox, 22
 automatic voltage regulator, 396
 automaton
 - differential a., 55
 - cyclic linear a., 236
 - deterministic a., 178
 - discrete hybrid a., 300
 - finite a., 228
 - gain a., 72
 - hybrid a., 59

- interchange a., 365
- stochastic a., 213, 264
- stopwatch a., 64
- timed a., 40, 368
- AUTOSTAR initiative, 441
- auxiliary variables, 300
- average model, 440
- average spectral radius, 133
- AVR, *see* automatic voltage regulator

- BARON, 436
- barrier certificate, 274, 275
- BaSiP, 55, 329, 346, 358
- batch plant, 419
- Bayesian procedure, 102, 104
- BDC, *see* bottom dead center
- behavior, 225
 - behavioral framework, 224
 - chaotic b., 203
 - complete b., 226
 - Zeno b., 45
- behavioral relation, 219
- bisimulation, 62, 368
 - timed b., 64
- block diagram, 329, 444
- Bohl function, 162
- Bohl-type input, 162
- Boolean optimization, 304
- Boolean variable, 300
- bottle filling system, 370
- bottom dead center, 461
- bottom-up design, 417
- bounded-error procedure, 103, 105
- branch-and-bound search, 433
- branch-and-bound technique, 429
- Brockett's model, 55
- buffer, 159
- bumpless transfer, 25

- CAPE-OPEN initiative, 327
- Carathéodory solution, 51, 161, 181
- cardinality, 207
 - minimal c., 213
- causal, 416
- causal modeling, 331
- CCM, *see* continuous conduction mode
- CDD, 304
- certainly equivalence principle, 53

- CFTOC, *see* constrained finite time-optimal controller
- chaotic behavior, 203, 208
- characteristic multiplier, 240
- charge stratification, 463
- Charon, 346
- chattering, 332
- CheckMate, 347
- Chi, *see* χ
- Church-Turing thesis, 130
- CI, *see* compression ignition
- CIF, *see* compositional interchange format
- circuit theory, 160
- CITOC, *see* constrained infinite time-optimal control
- classification, 96, 97
- clock, 63
- clock-driven, 235
- closed orbit, 238
- CLP, 304
- clustering, 101
- clustering-based procedure, 101, 104, 316
- clutch, 447
- CMTOC, *see* constrained minimum time-optimal control
- co-state variable, 388
- collision avoidance, 477
- collocation-based technique, 352
- communication, 472
- communication constraint, 197, 204
- communication network, 5, 442
- complementarity condition, 151
- complementarity system, 151, 152, 177
 - linear c. s., 151
 - switched cone c. s., 154
- complete behavior, 226
- complete model, 218, 220
- complexity, 4, 50, 130, 209, 231, 433
- complexity reduction, 307
- component, 363
- compositional interchange format, 347, 362
- compression ignition, 461
- compression ratio, 461
- compression stroke, 461
- computation tree logic, 62
- computer science, 5
- computer-automated multi-paradigm modeling, 327
- conditional equation, 332

- cone, 153
- configuration
 - nonblocking c., 479
 - safe c., 479
- conflict management, 474
- consensus problem, 473
- consistency-based diagnosis, 220
- constrained finite optimal control, 383
- constrained finite time-optimal control, 303, 306
- constrained finite time-optimal controller, 487
- constrained infinite time-optimal control, 304, 306
- constrained minimum time-optimal control, 306
- constraint, 67, 287, 434
- constraint satisfaction, 304
- continuous conduction mode, 27, 244, 379
- continuous dynamics, 7, 59
- continuous evolution, 60, 252
- continuous input, 18
- continuous state variable, 19
- continuous-to-discrete interface, 34
- continuous-valued signal, 8
- control
 - abstraction-based c., 413
 - cooperative c., 472
 - decentralized c., 472
 - distributed c., 472
 - embedded c., 4
 - finite optimal c., 383
 - finite time-optimal c., 303, 306
 - hierarchical c., 414
 - high-level c., 423
 - hybrid c., 223, 506
 - infinite time-optimal c., 304, 306
 - low-level c., 414, 423
 - model-predictive c., 52, 381, 397, 452
 - multirate c., 442
 - nonlinear model-predictive c., 427
 - optimal c., 211, 303, 418, 434
 - optimal state-feedback c., 383
 - periodic c., 243
 - PI c., 23, 427
 - receding-horizon c., 305
 - rule-based c., 450
 - spatially decentralized c., 475
 - supervisory c., 227
 - control code, 452
 - control engineering, 5
 - control mode, 59
 - control parametrization, 351
 - control software, 406
 - control switch, 59
 - control system, 442
 - Control System Toolbox*, 305
 - control validation, 443
 - controllability, 52, 166
 - controllability analysis, 52
 - controlled Markov chain, 263
 - controlled switching, 18
 - controller
 - deadbeat c., 210
 - discrete-event c., 231
 - predictive logic c., 507
 - qdb-c., 210
 - sequential c., 406
 - controller synthesis, 67
 - convex hull operator, 71
 - convex optimization, 386
 - cooperative control, 472
 - coordination, 473
 - core dynamics, 151
 - cost function, 400, 418, 423
 - cost matrix, 158
 - counter, 76
 - Cplex*, 309
 - crank angle, 461
 - crankshaft sensor management, 449
 - cruisable graph, 474
 - cruise control, 449, 453
 - current-mode control, 245
 - cut-off control, 441, 449
 - cyclic process, 27
 - cylinder pressure, 464
- DAE, *see* differential-algebraic equation
- DASSL*, 331, 398
- data interchange format, 320
- DC-DC converter, 26, 379
- DCM, *see* discontinuous conduction mode
- dead zone, 389
- deadbeat controller, 210
- decentralized control, 472
- decentralized hysteresis controller, 426
- decidability, 62, 63
- decision problem, 130

- delay, 472
- density, 207
- dependency graph, 363
- derivative design, 444
- DES, *see* discrete-event system
- design flow, 442
- detectability, 52
- determinism, 179
- DHA, *see* discrete hybrid automaton
- diagnosis, 213
 - consistency-based d., 220
- diagnostic algorithm, 214, 449
- diagnostic tool, 448
- difference equation, 300
- differential equation with discontinuous
 - right-hand side, 156
- differential inclusion, 41, 63, 64, 174, 176, 185
 - impulse d. i., 188
 - linear d. i., 70, 115
- differential index, 331
- differential-algebraic equation, 327, 350, 398
- diffusion process, 257
- diffusion term, 271
- direct injection stratified charge engine, 455
- discontinuous conduction mode, 28
- discontinuous differential equation, 91
- discontinuous dynamical system, 35
- discontinuous model, 93
- discrete approximation, 413
- discrete dynamics, 7, 59
- discrete evolution, 252
- discrete hybrid automaton, 300
- discrete input, 18
- discrete sensor, 18, 197
- discrete state, 19
- discrete successor, 65
- discrete-event quantized system, 215
- discrete-event simulation, 55
- discrete-event system, 223, 332
- discrete-time hybrid system, 299
- discrete-time piecewise affine system, 41, 322
- discrete-time quantized system, 215
- discrete-time stochastic hybrid system, 261, 275
- discrete-time system, 130
- discrete-to-continuous interface, 34
- discrete-valued signal, 8
- discretely controlled continuous system, 39, 90, 231
- discretization, 352
- distinguishability, 109
- distributed control, 472
- disturbance, 234
- disturbance attenuation, 241
- drift term, 271
- drive-by-wire, 454
- driveline, 445, 447
- dtPWA, *see* discrete-time piecewise affine system
- DTSHA, *see* discrete-time stochastic hybrid automaton
- dual cone, 153
- duration calculus, 55
- duty cycle, 27, 381
- dwell time, 76, 107, 126, 235
- Dymola*, 354
- dynamical programming, 275, 386
- dynamical time scale, 416
- dynamics
 - continuous d., 7
 - discrete d., 7, 59
- EcosimPro*, 347, 365
- ECU, *see* electronic control unit
- EECI, *see* European Embedded Control Institute
- ego*, 410
- EGR, *see* exhaust gas recirculation
- ELC, *see* extended linear complementarity system
- electronic control unit, 441
- electronic throttle, 454
- embedded control, 4
- embedded controller, 7, 204, 440
- embedded logic, 427
- embedded map, 235
 - controlled e. m., 241
- embedded simulation, 410
- emission control, 449
- energy management, 378
- engaged gear identification, 447
- engine control, 440
- engine start-up, 448
- entry point, 73
- equilibrium, 72, 74, 114, 199, 233, 388

- equivalence ratio, 462, 463
 ESPRESSO, 304
 ETAS, 444
 European Embedded Control Institute, xv
 evaporation system, 407, 425, 430
 event, 34
 threshold e., 144
 event function, 234
 event generator, 34, 232, 300
 event-driven sampling, 224
 event-driven switching, 235
 example
 automatic gearbox, 22, 84
 boost converter, 26, 153, 233, 244
 bottle filling system, 370
 bouncing ball, 45
 flying ball, 173
 Leontiev economy, 157
 moving agent, 204
 quantized tank system, 215
 reset oscillator, 12
 tank system, 10
 thermostat, 7, 254
 two-tank system, 17, 61, 81, 154, 354
 user-ressource model, 158
 execution, 107, 175, 235, 252, 258, 262
 finite e., 179
 infinite e., 179
 maximal e., 179
 periodic e., 238
 sampled e., 238
 stable e., 239
 synchronous e., 363
 exhaust gas recirculation, 221, 449, 454, 462
 exhaust stroke, 461
 exhaust valve control, 448
 expansion stroke, 461
 expressiveness, 327
 extended linear complementarity system, 168
 extensible markup language, 321, 364
 extensible markup language document, 323
 extremal point, 73

 fault, 397, 401, 449
 fault detection
 f.d. of quantized system, 221
 fault diagnosis, *see* diagnosis
 fault identification
 f.i. of quantized system, 221
 fault probability, 220
 fault tolerance, 449, 472
 feasible subsystem, 95
 feature vector, 101
 feedback linearization, 458
 Filippov solution, 51, 189
 finite automaton, 228
 finite-state automaton, 299, 335
 finite-state machine, 106, 445, 505
 finiteness property, 132
 first-principle modeling, 92
 fixpoint, 240
 flow function, 63
 flow set, 40
 flowsheet, 407
 forbidden region, 67
 forced transition, 257, 262, 275
 forward solution, 161, 189
 fuel injection, 448
 functional development, 442
 functional integration, 443

 gain, 72
 gain automaton, 72
 GAMS, 354
 GBT, 304
 gear-box control, 449
 Geometric Bounding Toolbox, 304
 glcDirect, 410
 GLPK, 304
 gOPT, 353
 gPROMS, 328, 338, 353
 guard, 34, 60, 489
 guard set, 234

 HA, *see* hybrid automaton
 Hamiltonian, 79, 81
 port control H., 388
 port-H., 154
 HCCI, *see* homogeneous charge compression
 ignition
 HCCI engine, 461
 heat release, 464
 HHARX model, *see* hinging-hyperplane
 ARX model
 hierarchical connection, 369
 hierarchical control, 414
 hierarchical process, 508, 509

- high-level control, 423
- high-level supervisor, 414
- H_∞ -synthesis, 384
- hinging-hyperplane ARX model, 318
- HMP, *see* hybrid maximum principle
- homogeneous charge compression ignition, 461
- HW/SW components, 442
- HW/SW testing, 443
- hybrid automaton, 15, 17, 35, 59, 63, 145, 251, 407
 - network of h.a., 362
 - rectangular h.a., 63
 - regular h.a., 74
- hybrid control, 223
- hybrid decoupling constraint, 100
- hybrid decoupling polynomial, 100
- hybrid dynamical phenomenon, 9
- hybrid flow, 74
- hybrid identification, 316
- Hybrid Identification Toolbox*, 316, 320
- hybrid inclusion, 177
- hybrid jump, 275
- hybrid maximum principle, 80
- hybrid model parameter, 100
- hybrid model-predictive control, 452
- hybrid state, 59, 106
- hybrid state observer, 106
- hybrid state space, 59
- hybrid switch point, 235
- hybrid system, 4, 31
 - autonomous h. s., 8
 - continuous-time h. s., 151
 - phenomena of h. s., 9
 - structure of h. s., 33
- Hybrid System Interchange Format*, 362
- hybrid system theory, 4
- hybrid time basis, 107
- hybrid time set, 252
- hybrid time trajectory, 175
- Hybrid Toolbox*, 308, 320, 322, 459
- HyBrSim*, 348
- HYCON, xiv
- hyperplane, 72
- HYSDEL*, 299, 458
- HyTech*, 55, 286, 368
- HyVisual*, 367

- I/-behavior, 226

- I/S-machine, 225
- identification, 54, 92
 - actual engaged gear i., 451
 - set-membership i., 103
- if-then rule, 144, 299
- implicit description, 173
- impulse differential inclusion, 188
- inclusion, 131
- inequality constraint, 153, 400
- initial state probability, 219
- injector, 34
- input
 - Bohl-type i., 162
 - continuous i., 8, 18
 - discrete i., 8, 18
 - quantized i., 214
- input quantizer, 199
- input/output model, 93
- input/output pair, 95
 - spurious i/o p., 218
- input/output system, 152
- intake stroke, 461
- intake throttle valve control, 441
- integer optimization problem, 507
- integral-partial-differential-algebraic equation, 327
- integration and testing flow, 442
- intensity, 269
- interchange architecture, 321
- interchange format, 41, 362
 - compositional i.f., 362
- interface, 34
- interval region, 73
- invariant, 60, 287
 - i. hypercubes, 210
 - i. property, 76
- IPDAE, *see* integral-partial-differential-algebraic equation

- Jacobian, 399
- joint spectral radius, 122, 131
- jump, 10, 287
- jump function, 61
- jump rate, 269
- jump set, 12, 40

- KRONOS*, 55

- l -complete approximation, 228

- l -completeness, 228
- labeling function, 60
- Lagrangian, 77
- language-equivalence relation, 62
- LaSalle's invariance principle, 74
- LCP, *see* linear complementarity problem
- least squares, 104
- least squares fitting, 382
- LHA, *see* linear hybrid automaton
- lifting representation, 385
- limit cycle, 203, 238
- linear classifier, 317
- linear complementarity model, 37
- linear complementarity problem, 151, 159
- linear complementarity system, 152, 168
- linear constraint, 300
- linear dynamical system, 299
- linear hybrid automaton, 62, 286, 322
- linear hybrid system, 63
- linear matrix inequality, 52, 117
- linear passive system, 160
- linear programming, 288, 304
- linear quadratic regulator, 142
- linear saturation system, 183
- linear switched system, 106, 107
- linear temporal logic, 62
- Lipschitz continuity, 257
- live-lock, 178
- liveness, 107
- LMI, *see* linear matrix inequality
- load shedding, 397
- local regression, 101
- local stability, 206
- location, 15, 59, 287
- location observability, 110
- locational optimization, 208
- logic constraint, 301
- logic controller, 406
- low complexity setup, 306
- low-level control, 414, 423
- low-level plant model, 423
- LQR, *see* linear quadratic regulator
- LTL, *see* linear temporal logic
- Lyapunov function, 114, 205, 307
 - common L.f., 116, 388
 - control L.f., 207
 - multiple L.f., 119
 - piecewise linear L.f., 120
 - poly-quadratic L.f., 123
 - quasi-quadratic L.f., 120
 - weak L.f., 121
- Lyapunov stability theory, 114
- Lyapunov's indirect method, 75
- Lyapunov-like function, 120
- magnetic actuators, 455
- Markov chain, 264, 487
- Markov parameter, 189
- Markov property, 218
 - M.p. of quantized system, 218
- mathematical programming, 152
- max-min-plus-scaling system, 169
- mean-value model, 449
- measurement aggregation, 416
- MILP, *see* mixed-integer linear programming
- minimax optimization, 304
- MINLP, *see* mixed-integer nonlinear programming
- MIQP, *see* mixed-integer quadratic programming
- mixed discrete-continuous dynamical program, 423
- mixed logical dynamical model, 37, 506
- mixed logical dynamical system, 37, 145, 168, 299, 305, 458
- mixed switching, 235
- mixed-integer linear programming, 148, 304
- mixed-integer linear sequence, 312
- mixed-integer nonlinear optimization problem, 436
- mixed-integer nonlinear programming, 351, 427
- mixed-integer programming, 96, 303, 308, 398
- mixed-integer quadratic programming, 148
- MLD, *see* mixed logical dynamical system
- MMPS, *see* max-min-plus-scaling system
- mode
 - active m., 145
- mode of operation, 152, 445
- mode selector, 145
- mode sequence, 232
- model
 - average m., 43
 - complete m., 128
 - decisive power, 41
 - discrete-event m., 196, 218
 - input/output form, 93

- modeling power, 41
- nondeterministic m., 217
- sampled-data m., 384
- model checker, 275, 286
- model checking, 54, 274
- model integration, 327
- Model Predictive Control Toolbox*, 308
- model-based design, 444
- model-predictive control, 52, 211, 353, 397, 452, 454
 - dual mode m.p.c., 211
 - explicit m.p.c., 379
- model-predictive controller, 308, 406, 507, 509
- Modelica*, 55, 283, 329, 348, 353, 357, 365
- modeling framework, 31
- modularity, 326
- modulation scheme, 493
- MPC, *see* model-predictive control
- mpQP, *see* multi-parametric quadratic programming
- MPT, *see* *Multi-Parametric Toolbox*
- MRLP, *see* multicategory robust linear programming
- MS, *see* mode selector
- multi-agent system, 474
- multi-controller scheme, 113
- multi-disciplinary design, 4
- multi-parametric programming, 303, 383
- multi-parametric quadratic programming, 308
- Multi-Parametric Toolbox*, 303, 316, 320, 322
- multi-rate automaton, 54
- multi-regime system, 157
- multicategory robust linear programming, 317
- multiMin*, 410
- multiple shooting, 351, 436
- multiple state feedback, 389
- multiple time scales, 416
- multiple-vehicle system, 478
- multiproduct plant, 419
- multirate control, 442
- MUSCOD*, 354

- natural gas, 463
- natural projection, 226
- NCS, *see* networked control system

- nearest-neighbor quantizer, 207
- needle variation, 78
- networked control system, 5, 113, 203, 472, 487
- networked system, 441
- NMPC, *see* nonlinear model-predictive control
- nominal operation, 448
- nonanticipating system, 226
- nonblocking, 178, 179
- noncausal modeling, 329
- nonconflicting behavior, 227
- nondeterminism, 216, 250
- nondeterministic hybrid system, 250
- nonlinear differential equation, 20
- nonlinear dynamics, 19, 406
- nonlinear model-predictive control, 427
- nonlinear programming, 211, 428
- nonsmooth mechanics, 6
- nonsmooth system, 42
- nonswitching time, 164
- nontriviality condition, 81
- NP, 130
- NP-complete, 131
- NP-hard, 95, 99, 130, 131
- NPSOL*, 430
- numerical simulation tool, 329

- object-oriented implementation, 324
- observability, 52, 108, 182
 - incremental o., 108, 137
- observer, 53
- observer design, 451
- obstacle, 477
- ODE, *see* ordinary differential equation
- Omola*, 329, 348
- on-board diagnosis, 449
- on/off switch, 300
- operating point, 399
- operation mode, 15, 18, 232
- optimal control, 49, 211, 303, 418, 434
- optimal state-feedback control, 383
- Optimica*, 354
- optimization, 350, 410
 - smooth constrained o., 96
- Optimization Toolbox*, 309
- optimization-based analysis using simulation, 411
- optimization-based controller, 307

- orbital stability, 239
- ordinary differential equation, 327
- oscillation, 401
- output
 - continuous o., 8
 - discrete o., 8
 - quantized o., 214
- output quantizer, 197
- over-approximation, 67, 437

- P, 130
- P-matrix, 160
- p -periodic linear system, 243
- p -periodic state feedback, 243
- packet loss, 206, 472
- parallel composition, 362
- parallel connection, 369
- particle filtering, 103
- passivity, 151, 161, 163
- passivity constraint, 454
- PCH, *see* port control Hamiltonian
- pdf, *see* probability density function
- performance, 209, 389
- periodic control, 243
- periodic solution, 384
- periodic stationary execution, 238
- periodic stationary operation, 233
- perturbation, 74
- Petri net, 55
- phase-portrait approximation, 294
- physical process, 4
- PI control, 23, 427, 433
- piecewise affine ARX model, 318
- piecewise affine autoregressive exogenous model, 93
- piecewise affine dynamics, 299
- piecewise affine feedback, 303, 452
- piecewise affine model, 453, 454
- piecewise affine system, 36, 90, 134, 137, 167, 322
 - continuous-time p.a.s., 379
 - discrete-time p.a.s., 121
- Piecewise Affine System Identification Toolbox*, 317
- piecewise linear Lyapunov function, 120
- piecewise linear system, 151, 181
- piecewise smooth system, 35
- platform, 443
- platform-based design, 443
- platform-dependent language, 321
- platform-independent format, 324
- platform-specific format, 321
- PMP, *see* Pontryagin's maximum principle
- Poisson process, 269
- polyhedral partition, 93, 308, 383
- polyhedron, 69, 303
- polynomial embedding, 99
- polynomial-time algorithm, 130
- polytope, 304
- polytope manipulation, 304
- Pontryagin's maximum principle, 78, 85
- port control Hamiltonian, 388
- port-Hamiltonian complementarity system, 154
- power conversion, 378
- power converter, *see* DC-DC converter, boost converter
- power generation, 378
- power level, 493
- power outage, 393
- power system stabilizer, 396
- power transmission, 378
- powertrain control, 454
- practical stability, 200
- practically stabilizing controller, 209
- predicate
 - convex p., 62
 - deadline p., 367
- prediction model, 398
- predictive logic controller, 507
- primary controller, 401
- priority, 367
- probabilistic safety analysis, 265
- probability density function, 102
- probability map, 274
- programming logic controller, 49
- propositional logic, 299
- PSS, *see* power system stabilizer
- Ptolemy II*, 349
- pulse width modulator, 379
- PWA, *see* piecewise affine
- PWAID, *see* *Piecewise Affine System Identification Toolbox*
- PWARX, *see* piecewise affine autoregressive exogenous model
- PWARX model, *see* piecewise affine ARX model
- PWM, *see* pulse width modulator

- qdb-controller, 210
- QoS, *see* quality of service
- QP, *see* quadratic programming
- QP solver, 309
- quadratic optimization problem, 303
- quadratic programming, 142, 304
- quality of service, 473
- quantization, 193, 197, 198
 - dense q., 203
 - input q., 208
 - output q., 198, 208
 - q. density, 206
 - q. error, 200, 208
 - q. map, 196
 - q. noise, 472, 490
 - q. threshold, 491
 - q. width, 493
 - state q., 198
- quantized behavior, 196
 - nondeterminism of the q. b., 216
- quantized input, 214
- quantized model
 - completeness of q. m., 218, 219
- quantized output, 214
- quantized state, 214
- quantized system, 196
 - discrete-event q. s., 215
 - discrete-time q. s., 215
 - q. linear s., 196
- quantized value, 214
- quantizer, 214
 - logarithmic q., 201, 207
 - nearest neighbor q., 199
 - output q., 197
 - state q., 196
- quarter car model, 454
- quasi-quadratic Lyapunov function, 120
- queue, 159

- rapid prototyping, 309
- raw data classification, 103
- rbfSolve*, 410
- re-usability, 326
- reachability, 63, 64, 230, 306
- reachability analysis, 64, 249
- reachable space, 69
- reachable state, 179
- Real Time Workshop*, 307, 309
- real-time system, 349
- real-valued variable, 300
- receding-horizon control, 304, 305
- recovery, 449
- refinement, 103, 229, 443
- refrigeration system, 426
- region, 64, 73
- region estimation, 97, 98
- region graph, 64
- region stability, 73
- regression vector, 93
- regularity, 55, 273
- relaxation, 174
- relay characteristic, 156
- relay system, 156
- reliable operation, 393
- reset, 332
- reset map, 16, 60
- residual gas, 462
- residual signal, 451
- resolution, 210
- resource allocation, 212
- resource utilization, 473
- return map, 75, 239
 - controlled r. m., 241
- RHC, *see* receding-horizon control, 305
- right-accumulation point, 46
- RLP, *see* robust linear programming
- robust controller, 304
- robust linear programming, 98
- robustness, 54, 245
- roundabout policy, 475
- rule-based control, 450
- run, 175, 288

- safety, 437, 449
 - probabilistic s., 265
- safety analysis, 67, 265, 410
- sampling
 - event-driven s., 224
 - time-driven s., 230
- sampling period, 206
- saturated linear system, 135
- saturation, 37
- scheduler, 363
- Scilab*, 349
- scope, 368
- SeDuMi*, 304
- semantics, 175
- semi-active suspension, 453

- semi-analytical approach, 410
 semi-definite programming, 304
 semicontinuous, 186
 semicontinuous function, 186
 sensor network, 473
 separation technique, 97
 sequential controller, 406
 set-membership identification, 103
 set-membership test, 303
SHIFT, 329, 349
 shut-down, 406
 SI, *see* spark ignition
Siconos, 349
 simulation, 46, 55, 444
 block-diagram-based s., 329
 simulation tool, 326
Simulink, 307, 309, 444
 single-linkage clustering, 316
 single-shooting technique, 351
 slack variable, 400
 sliding behavior, 47
 sliding mode, 47, 181
 small-gain condition, 202
 small-gain theorem, 211
 smooth constrained optimization, 96
 snapshot, 73
 software engineering, 5
 solar air conditioning plant, 502
 solar energy, 393
 solution, 36, 185
 Carathéodory s., 51
 classical s., 50
 Filippov s., 51
 spark ignition, 448, 461
 specification, 227, 443
 spectral radius
 generalized s. r., 132
SPEEDUP, 338
 spontaneous transition, 257, 262, 275
 spurious input/output pair, 218
 stability, 43, 72, 113, 131, 165
 asymptotic s., 114
 exponential s., 114
 global asymptotic s., 114
 input-to-state s., 137
 local s., 206
 orbital s., 239
 practical s., 200, 201
 region s., 73
 stability analysis, 72, 121, 307
 stabilizability, 52, 166
 stabilization, 124, 199
 stabilizing switching law, 124
 stable system, 114
 start-up, 406, 433
 start-up performance, 389
 start-up time, 434
 state, 366
 continuous s., 8
 discrete s., 8
 quantized s., 214
 state feedback, 303
 state jump, 10, 22
 autonomous s.j., 12
 controlled s.j., 14
 state machine, *see* automaton
 state observer, 206
 state quantizer, 196
 state space
 partitioned s. s., 214
 state transition function, 14
 state-dependent switching, 157
 state-event detection, 363
 stationary operation, 233
 steady-state operation, 397, 407
 steepest descent, 389
 stochastic automaton, 219
 stochastic hybrid automaton, 256
 stochastic hybrid model, 55
 stochastic hybrid system, 249
 stopping condition, 367
 stopping time, 257
 stopwatch automaton, 64
 strictly passive, 165
 stroke detection, 448
 structural stability, 74
 subharmonic oscillation, 381
 supervisor, 225
 admissible s., 227
 generically implementable s., 227
 least restrictive s., 227
 supervisory control, 225, 227
 supervisory control theory, 226
 support vector machine, 317
 SVM, *see* support vector machine
 switched affine autoregressive exogenous
 model, 93
 switched circuits, 6

- switched cone complementarity system, 154
- switched linear system, 118, 425
- switched model-predictive controller, 454
- switched system, 35, 89, 137
 - autonomous s.s., 90, 112
 - controlled s.s., 89, 112
 - discrete-time s.s., 112
 - linear s.s., 106
- switching, 10, 47, 80, 131, 400
 - autonomous s., 10, 17
 - clock-driven s., 235
 - controlled s., 14, 18
 - event-driven s., 10, 235, 241
 - fast s., 124
 - mixed s., 235
 - stabilizing s., 124
 - time-driven s., 10
- switching behavior, 363
- switching condition, 80, 234
- switching diffusion system, 270
- switching frequency, 233
- switching function, 154
- switching logic, 232
- switching period, 27
- switching scheme, 27
- switching set, 10, 77, 80
- switching structure, 45
- switching surface, 47, 90
- switching system, 90
- switching time, 77, 80, 164
- synchronization, 60, 365, 427
- synthesis, 406
- synthesis flow, 442
- system
 - max-min-plus scaling s., 55
 - chaotic s., 203
 - continuous s., 43
 - discontinuous dynamical s., 35
 - discrete-event s., 43
 - discrete-time stochastic hybrid s., 275
 - hybrid s., 4
 - mixed logical dynamical s., 37
 - passive s., 165
 - piecewise smooth s., 35
 - switching diffusion s., 270
 - switching s., 90
 - timed s., 363
- System Identification Toolbox*, 305
- system integration, 441
- system specification, 442
- system testing, 443
- systems theory, 5
- target vector, 79
- target region, 434
- target set, 77
- TDC, *see* top dead center
- technological system, 4
- temporal logic, 62
- TG, *see* turbine governor
- theorem proving, 411
- thermostat, 7, 254
- throttle valve control, 448
- tight over-approximation, 294
- time scale, 378
- time-varying linear system, 131
- time-varying parameter, 128
- timed automaton, 40, 54, 63, 350, 409
- timed computation tree logic, 64
- TOMLAB*, 353, 410
- tool-specific layer, 321
- top dead center, 461
- traction control, 453
- transition function, 234
- transition intensity, 275
- transition kernel, 262
- transition probability, 264
- transition relation, 59
- transition system, 59
- transmission network, 393
- transversality condition, 81
- turbine governor, 395
- Turing machine, 134, 135
- two-layer architecture, 433
- two-level control architecture, 414
- two-level hierarchical solution, 418
- two-tank system, 17
- uncertainty, 304
- uncertainty set, 131
- undecidability, 63, 69, 130, 134
- uniform time scale, 415
- unilaterally constrained mechanical system, 182
- uniqueness, 187
- unsafe set, 264
- UPPAAL*, 55, 283, 329, 347, 349, 368
- urgency, 363, 366

- urgency predicate, 367
 urgent guard semantics, 367
 usability, 327
- valve assembly, 407
- variable, 141
 - connected v., 369
- variable geometry turbine, 221, 454
- VDC, *see* vehicle dynamical control
- vector field, 59, 60
- vehicle dynamical control, 441
- vehicle dynamics, 453
- verification, 62, 264, 303, 406
- VGT, 449, 454
- voltage instability, 393, 397
- voltage regulation, 381, 393, 396, 397
- VTG, *see* variable geometry turbine
- Wardrop principle, 158
- weak Lyapunov function, 121
- well-defined solution, 173
- well-formed expression, 175
- well-posedness, 36, 46, 173, 178
 - global w., 179
 - initial w., 178
 - local w., 179
- Wiener process, 256
- wind energy, 393
- wireless communication, 473
- worst-case analysis, 250
- XML, *see* extensible markup language
- XML Toolbox, 321
- YALMIP, 302
- Zeno
 - left-Zeno free, 164
 - right-Zeno free, 164
 - Zeno free, 164
- ZENO, 45
- Zeno behavior, 45, 180, 189, 332
- Zeno phenomenon, 191
- zero crossing, 46