RWTH Aachen

Department of Computer Science
Technical Report

Comparative Evaluation and Improvement of Computational Approaches to Reachability Analysis of Linear Hybrid Systems

Ibtissem Ben Makhlouf

ISSN 0935–3232 · Aachener Informatik-Berichte · AIB-2016-02

RWTH Aachen · Department of Computer Science · March 2016

The publications of the Department of Computer Science of *RWTH Aachen University* are in general accessible through the World Wide Web.

http://aib.informatik.rwth-aachen.de/

Comparative Evaluation and Improvement of Computational Approaches to Reachability Analysis of Linear Hybrid Systems

Von der Fakultät für Mathematik, Informatik und Naturwissenschaften der
RWTH Aachen University zur Erlangung des akademischen Grades
einer Doktorin der Ingenieurwissenschaften genehmigte Dissertation

vorgelegt von

Diplom-Informatikerin
Ibtissem Ben Makhlouf
aus
Djerba / Tunesien

Berichter: Universitätsprofessor Dr.-Ing. Stefan Kowalewski
Universitätsprofessor Dr. Goran Frehse

Tag der mündlichen Prüfung: 26. Januar 2016

Diese Dissertation ist auf den Internetseiten der Universitätsbibliothek online verfügbar.

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Zugl.: D 82 (Diss. RWTH Aachen University, 2016)

Ibtissem Ben Makhlouf
Lehrstuhl Informatik 11 - Embedded Software
makhlouf@embedded.rwth-aachen.de

Aachener Informatik Bericht AIB-2016-2

Herausgeber: Fachgruppe Informatik
RWTH Aachen University
Ahornstr. 55
52074 Aachen
GERMANY

ISSN 0935-3232

Copyright Shaker Verlag 2016
Alle Rechte, auch das des auszugsweisen Nachdruckes, der auszugsweisen oder vollständigen Wiedergabe, der Speicherung in Datenverarbeitungsanlagen und der Übersetzung, vorbehalten.

Printed in Germany.

ISBN 978-3-8440-4376-1

Shaker Verlag GmbH, Postfach 101818, 52018 Aachen
Telefon: 02407/9596-0, Telefax: 02407/9596-9
Internet: www.shaker.de, E-Mail:info@shaker.de

Acknowledgments

I would like to express my special appreciation and heartfelt gratitude to my advisor Professor Stefan Kowaleswki, you have been a great mentor for me. I would like to thank you for supporting my research and for allowing me to grow as a research scientist. Your advice on both research as well as on my career has been priceless. Your moral support and understanding of my particular situation has helped me a lot through the course of my thesis. I hope that my work lived up to some of your expectations.

I would also like to thank, Prof. Erika Abraham, Prof. Thomas Seidl, Dr. Goran Frehse for taking their time to serve on my thesis committee. I also want to thank you for letting my defense be an enjoyable moment, and for your brilliant comments and suggestions. Erika and Goran, it has been a great pleasure for me to discuss and exchange thoughts with you.

I would especially like to thank Prof. Lars Grüne, Prof. Matthias Althoff, Dr. André Platzer, Dr. Stefan Matthias Ratschan, Dr. Colas Le Guernic for the precious support. You have always been ready to response quickly to my questions.

This dissertation could not have been completed without the great support that I have received from my students Norman Hansen, Jonathan Gan and Malte Neuss.

Special thanks goes to Prof. Wolfgang Thomas, Helen Bolke-Hermanns and the AlgoSyn group for the financial support and for allowing me to be a member of your group.

Furthermore, I would like to thank my colleagues at the Informatik 11 - Embedded Systems Institute for creating a friendly environment, for the stimulating discussions, for the creative power and the quality of the team dynamics.

I would like to thank my former supervisor Prof. Meriem Jaidane. My work with you has helped to develop my fascination towards research.

I wish to offer my most heartfelt thanks to Dr. Eva M. Navarro-López. I have always been able to turn to you when I needed a listening ear and a dissenting opinion. You have become one of my best friends. We have supported and helped each other through both good times and bad times. Our online discussions in various research directions were a great source of inspiration.

A special thanks to my family. I have an amazing family, unique in many ways. Your support has been unconditional all these years. You have given up many things for me to finally attain this longstanding goal. Thank you Raid, Rim and Rashid for your never ending patience and support.

Words cannot express how grateful I am to my husband Dr. Adel Mhamdi for all the love and support throughout the past few years. Thank you for pushing me to gain more self-confidence and self-belief in dealing with such a difficult topic. You were, in some way, my second advisor. Your knowledge and constant guidance have helped me a long way towards completing this thesis.

Particular thanks to my mother for all of the sacrifices. Your prayer for me was what sustained me thus far. And to my dad. I wish you were here. But you know a great part of you is in me. You taught me that hard work pays off. I feel so very

grateful for all your love that I received.

Finally, would like to thank my sister, my brothers and all of my friends who supported me by striving towards my goal.

Ibtissem Ben Makhlouf
Aachen, March 2016

Zusammenfassung

Diese Dissertation befasst sich mit dem Problem der Erreichbarkeitsanalyse, fokussiert auf lineare hybride Systeme. Hybride Systeme sind eine Mischung von kontinuierlichen und diskreten Verhalten. Ein hybrider Automat, bestehend aus einem Graph, in dem die Knoten das kontinuierliche und die Kanten das diskrete Verhalten beschreiben, bietet sich als das passende formale Modell für solche Systeme an. Es besteht aus einen Formalismus, das Differentialgleichungen und logische Ausdrücke im gleichen Rahmen umfasst und damit neue Horizonte für die Forschung und Entwicklung, vor allem in Richtung neuer Methoden und neuer Algorithmen, eröffnet. Trotz des Fortschrittes, der in den letzten Jahren zu verzeichnen war, gibt es vor allem in Hinsicht auf die praktische Anwendung noch viele offene Fragen. Begonnen haben wir diese Arbeit mit der Bewertung einiger Verifikationstools. Speziell für diese Aufgabe wurde eine Reihe von Benchmarks erdacht und zusammengefasst. Die Benchmarks besitzen besondere Eigenschaften in Bezug auf die Prüfung der Effizienz, Anwendbarkeit, Skalierbarkeit und Leistungsfähigkeit dieser Tools. Wir geben einen ausführlichen Überblick über bestehende Methoden zum Berechnen einer Überapproximation der erreichbaren Mengen für lineare zeitinvariante hybride Systeme. Dieser erfasst unterschiedliche Ansätze für die Berechnung einer Überapproximation für die kontinuierliche Dynamik mit und ohne Invarianten sowie auch für die Berechnung der Schnittmenge bei Übergängen. Basierend auf diesen Ergebnissen wurden neue Approximationsmethoden zur Berechnung der erreichbaren Mengen für den kontinuierlichen Teil als auch für den diskreten Teil des hybriden Automaten sowie eine modulare skalierbare Implementierung verschiedene Ansätze vorgeschlagen. Für diese Implementierungen werden zuerst Stützfunktionen und danach Zonotopen verwendet. Für die jeweiligen geometrischen Darstellungen wurde eine Reihe von verschiedenen Ansätzen für den Umgang mit Invarianten, Sprungbedingungen und Transitionen vorgestellt. Zwei Tools sind daraus entstanden. Beide Tools integrieren die oben beschriebenen Methoden und erlauben möglicher Kombinationen. Sie verfügen über eine GUI und ermöglichen eine vom Benutzer konfigurierbare Erreichbarkeitsanalyse. Beide Tools wurden zur Durchführung eines Leistungsvergleiches verschiedener Methoden verwendet. Einen Zusammenhang zwischen diesen Leistungen und die Komplexität der Benchmarks wurde dabei festgestellt. Die Studie mit den vorgeschlagenen Benchmarks führte zu dem Ergebnis, dass der Unterschied zwischen Methoden in Bezug auf die Genauigkeit der Überapproximation und die Rechenzeit unbedeutend ist. Um die Vielfältigkeit der Anwendbarkeit der Erreichbarkeitsanalyse zu veranschaulichen, kam es zum Vorschlag einer vernetzten Fahrzeugkolonne. Zuerst wurde die Analyse verwendet, um sichere und kurze Abständen zwischen Fahrzeugen in einer LMI-geregelten Kolonne zu bestimmen. Nachfolgend wurde eine Erreichbarkeitsanalyse durchgeführt um bei der Entscheidung über den leistungsfähigsten H_2- or H_∞-Regler der gleiche Kolonne zu unterstützen. Sie wurde außerdem eingesetzt um zeitkritische Bedingungen für eine Kreuzung mit einer annähernden Kolonne zu bestimmen und damit den Verkehr innerhalb der Kreuzung sicher zu verwalten.

Abstract

This thesis addresses the problem of reachability analysis with the focus on linear hybrid systems. Hybrid systems are a mixture of continuous and discrete behaviors. The Hybrid automaton consisting of a graph, in which the locations describe the continuous and the transitions the discrete behavior, represents the best formal model for such kind of systems. It provides a formalism integrating differential equations and logic expressions in a same framework, thus opening new horizons in research and development of new methods and novel algorithms. Despite recent progress made in this field in the last years, actual verification methods and available tools have exhibited their shortcomings.

We started this work with the assessment of some verification tools using a suite of benchmarks conceived specially for this task. The benchmarks possess particular characteristics for testing of efficiency, applicability, scalability, capability and performances of these tools.

We offer a theoretical overview of existing methods for computing an overapproximation of reachable sets for linear time invariant hybrid systems. This covers approaches for overapproximating reachable sets of the continuous dynamics with and without invariants as well as methods for solving the problem of guard intersection at transitions. We furthermore propose new overapproximation techniques for treating the continuous part as well as the discrete part of the hybrid automaton. We suggest scalable, modular implementations of these diverse methods allowing thereby possible combinations between them first using support functions and then with zonotopes. The implementations include different approaches for handling invariants, guards and transitions for the above-mentioned set representations. Two toolboxes are the results of this implementation effort. Both tools integrate the methods described above. They offer a GUI and allow for a user-configurable reachability analysis. We use both tools to carry out a performance comparison of different methods. We note thereby that there is a correlation between these performances and the complexity of the tested example. However, we note during this survey using the proposed benchmark suite that the difference in the performance with regards to the tightness of the over-approximation and the computation time is not so crucial for low dimensional systems.

We propose a networked platoon of vehicles to demonstrate different context where reachability analysis can be useful. We first perform a reachability analysis to determine unsafe gaps between the vehicles which are controlled using LMI-formalism. Reachability analysis can be helpful for control design. The choice between controllers on the basis of reachability results has led to controller ensuring the best compromise between safe and small gaps when applying H_2 or H_∞ control design techniques. Reachability can also be used to determine time-critical conditions. As demonstration, we opt for a platoon approaching an intersection.

Contents

1 Introduction **1**
- 1.1 Motivation and Objectives . . . 3
- 1.2 Contributions . . . 4
- 1.3 Outline . . . 5
- 1.4 Bibliographic Notes . . . 6

2 Assessment and Performance Comparison of Tools **9**
- 2.1 Introduction . . . 9
- 2.2 Benchmarks . . . 10
- 2.3 Description of Tools . . . 11
  - 2.3.1 SpaceEx: The PHAVer and the LGG Scenarios . . . 11
  - 2.3.2 KeYmaera . . . 13
  - 2.3.3 HSolver . . . 14
  - 2.3.4 iSAT . . . 15
- 2.4 Results . . . 16
  - 2.4.1 SpaceEx . . . 18
  - 2.4.2 KeYmaera . . . 25
  - 2.4.3 HSolver . . . 27
  - 2.4.4 iSAT . . . 28
  - 2.4.5 Performance Evaluation . . . 28
- 2.5 Conclusion . . . 31

3 Overview of Methods for Computing Reachable Sets of Linear Hybrid Systems **33**
- 3.1 Introduction . . . 33
- 3.2 Hybrid Automaton . . . 34
- 3.3 Run Semantics . . . 35
- 3.4 Reachable State within a Discrete Mode . . . 36
- 3.5 Reachable Set of LTI-Systems . . . 38
- 3.6 Computing an Over-approximation of the Input Contribution . . . 39
  - 3.6.1 Norm-bounded Uncertain Input . . . 40
  - 3.6.2 Bounded Uncertain Input . . . 40
  - 3.6.3 Toward a Tighter Approximation of the Input Contribution . . . 40
  - 3.6.4 Input Constant within a Time Step . . . 41

3.7 Computing an Over-approximation Ω_0 of $\mathcal{R}_{[0,r]}(X_0)$ 42
3.7.1 Using a Bloating Factor α_r [47] 42
3.7.2 Alternative Approach for Computing a Bloating Factor α_r [73] 43
3.7.3 Toward a Tighter Approximation of the Initial Set [71] 43
3.7.4 Forward and Backward Approximations for Computing an Over-approximation of the Initial Set [44, 96] 44
3.8 Handling Invariants Inside Discrete Modes 44
3.8.1 Handling Invariants as Guards 46
3.8.2 Recursive Scheme with Invariants [71] 46
3.9 Handling Transitions 49
3.10 Conclusion 51

4 Support Function Technique for Computing Reachable Sets 53
4.1 Introduction 53
4.2 Definition and Properties of Support Functions/Support Vectors . . 54
4.3 Computation of the Reachable Set within a Continuous Mode 59
4.3.1 Approximation of the Input Contribution 61
4.3.2 Initial Set Over-approximation Scenarios 62
4.3.3 Impact of the Approximation Method of the Initial Set on the Tightness of the Reachable Set 64
4.4 Collision Detection Between Two Convex Sets 68
4.5 Intersection Computation of Two Convex Sets 69
4.6 Intersection with Hyperplanes or Halfspaces 70
4.6.1 From n to 2 Dimension and the Dichotomous Search 70
4.6.2 The Sandwich Algorithm 74
4.6.3 Optimization Techniques 82
4.7 Handling Invariants with Support Function Techniques 85
4.7.1 Classical Method for Handling Invariants 85
4.7.2 Recursive Scheme Fusing Reachable Sets with Domain/Invariant Conditions 86
4.8 Handling Guards 90
4.9 Simultaneously Handling of Invariants and Guards 90
4.10 Handling Spontaneous Transitions 91
4.11 Handling Time-Triggered Transitions 92
4.12 MATLAB/Simulink Implementation 92
4.13 Experimentation 96
4.14 Conclusion 104

5 Zonotopic Approximation for Computing Reachable Sets 107
5.1 Introduction 107
5.2 Zonotopes 108
5.3 Properties and Geometric Operations of Zonotopes 109
5.4 Zonotope/Hyperplane Intersection 111
5.4.1 Intersection Check Between a Zonotope and a Hyperplane . . 111

5.4.2 From Dimension n to Dimension 2 . . . 112
5.4.3 Zonotope/Strip Intersection . . . 122
5.4.4 Zonotope/Hyperplane Intersection using Singular Value Decomposition (SVD) . . . 123
5.4.5 Comparison of Zonotope/Hyperplane Intersection Methods . 124
5.5 Zonotope/Halfspace Intersection . . . 127
5.6 Zonotope/Polyhedron Intersection . . . 128
5.7 Zonotope/Zonotope Intersection . . . 128
5.7.1 Zonotope/Zonotope Collision Detection . . . 129
5.7.2 Intersection of Two Zonotopes using Optimization Techniques 134
5.7.3 Intersection of Two Zonotopes using SVD . . . 135
5.7.4 Comparison of Both Zonotope/Zonotope Intersection Methods 136
5.8 Computing Reachable Sets within Discrete Modes . . . 137
5.9 Handling Invariants . . . 139
5.10 Handling Transitions . . . 139
5.10.1 Taking the First Intersection . . . 140
5.10.2 The Over-approximative Convex Hull Method . . . 141
5.10.3 Pre- and Post-clustering . . . 141
5.10.4 Finding the Min/Max at a Guard Transition . . . 141
5.10.5 Fixpoint-/Time-Triggered Transition . . . 141
5.11 Implementation . . . 142
5.12 Experimental Results and Performance Evaluation . . . 143
5.12.1 Comparison of Intersection Methods . . . 143
5.12.2 Experimental Evaluation at Guard Intersection . . . 145
5.12.3 Handling Invariant . . . 151
5.13 Conclusion . . . 151

6 Reachability Analysis of a Networked Platoon of Trucks 153
6.1 Introduction . . . 153
6.2 Description of a Networked Cooperative Platoon . . . 153
6.2.1 Platoon Model . . . 153
6.3 Information Flow and Interconnection Topology . . . 155
6.4 Control Design . . . 156
6.5 LMI-based Control . . . 157
6.5.1 Hybrid Modeling . . . 158
6.5.2 Safety Verification of the LMI-based Controlled Platoon . . . 159
6.6 H_2/H_∞-based Control . . . 164
6.6.1 Reachability Analysis . . . 172
6.7 Managing Platoons at Intersections . . . 176
6.7.1 The Intersection Infrastructure . . . 177
6.7.2 The Intersection Model . . . 178
6.7.3 Reachability using Zonotopes . . . 178
6.8 Conclusion . . . 183

7 Conclusion **185**

A Appendix A **189**
A.1 Infinity test . . . 189
A.2 The Bouncing Ball Example . . . 189
A.3 The Colliding Masses Benchmark . . . 190
A.4 The Transient in Flower Benchmark . . . 190
A.5 The Two-Tank Benchmark . . . 191
A.6 The Navigation Benchmark . . . 193
A.7 The Heating Benchmark . . . 195
A.8 Cooperative Platoon of Trucks . . . 197
A.8.1 Platoon under full communication (single mode) . . . 198
A.8.2 Platoon under a dropout of communication (2 modes) . . . 198
A.9 A Five Dimensional Linear Switching System (5D LSS) . . . 199

B Appendix B **201**
B.1 Proof of Lemma 1: Norm-bounded uncertain input . . . 201
B.2 Proof of Lemma 3: Toward a tighter approximation of the input contribution . . . 201
B.3 Proof of Lemma 4 . . . 203
B.4 Proof of Lemma 3.7.3 . . . 204

1. Introduction

Over the last years, the classical definition of embedded systems as an integration of physical systems and computing entities to achieve an intended task has been extended to the networking amongst embedded systems. Networked embedded systems are systems that involve multiple autonomous entities exchanging information via a communication network to cooperatively achieve a predefined common objective. Networking devices are nowadays standard in the automotive and avionic industries, widely spread in manufacturing and chemical processes and effectively applied in the medical field. They are making rapid inroads in high performance applications. This trend benefits from the progress made towards efficient, fast and reliable communication and sensors technologies and new methods for networked control in recent years.

Owing to their distributed nature, their dependency on the communication topology and their safety-critical requirements, the design of such systems is certainly a challenging task. Realizing correctness and ensuring safety by construction based only on numerical simulations during the design process has been shown to be impossible. Formal verification and validation is the only method that can cover a wide range of test cases during design steps. Although formal verification and validation are not feasible for all designs, they can be applied successfully in many cases.

A particularly challenging task is to find a suitable model framework and appropriate tools or verification methods to analyze and validate this kind of systems. However, looking more closely at the definition of such systems, we can differentiate between two major interacting components. The physical process, generally described by differential equations, constitutes the continuous part of the system which can be influenced by discrete events coming from computation or from communication. This mixture of discrete and continuous dynamics can naturally be modeled using the framework of hybrid automaton. A hybrid automaton is briefly defined as graph of indexed locations describing different continuous dynamics and transitions between them. Commonly, these transitions possess jump conditions and eventually reset maps.

The rapid expansion of connected controlled systems has spurred the control and computer science communities to support research in the field of hybrid systems. However, many problems, like the checking whether unsafe states are reachable beginning from an initial set, for example, are in generally undecidable [6, 9, 56, 57, 66, 91]. Therefore, most of the existing methods have focused on decidable classes like timed or rectangular automata using, for example, abstractions on some undecidable classes like the linear time invariant (LTI) systems for which relaxation

and approximation techniques have been proved to be feasible [89]. Over the last years, the focus was on developing and improving approaches to handle linear, as well as, nonlinear hybrid systems. As results of long-term efforts, several verification tools have emerged in the course of the last years.

These tools can be categorized in two classes, the logic-theoretic tools and the set-theoretic tools. The logic-theoretic tools construct logic proofs for checking of correctness, as found in the toolbox KeYmaera [88] which uses theorem provers and deductive verification for proving correctness. The toolboxes iSAT [99, 100] and dReach [27, 62] integrate SMT-solving techniques with constraint propagation and interval arithmetic to check for correctness.

The set-theoretic tools, however, focus on computing geometric representations of reachable sets using thereby geometric operations and if necessary optimization. Approximation methods are used to over-approximate the result of geometric operations and result in the loss of the original set representation. In addition, the adopted basic recursive scheme is generally approximative. Tools like CheckMate [101] and d/dt [7] use polyhedra to compute flowpipe approximations. However, the complexity of operations on polyhedra are usually exponential in the number of dimensions and makes it hard for the reachable set computation to scale [25]. The toolbox PHAVer [41] circumvents this problem by allowing polyhedron complexity reduction nonetheless at the cost of the approximation tightness. Tools like HSolver [94] and HyCreate [13] combine boxes with interval arithmetic and constraint propagation techniques. They have scalability problems due to the adopted state space or reachable set partition technique when the number of variables goes up. Such an approach has been shown to be inefficient even for six dimensional systems [17, 19]. Furthermore approaches adopting polytopes [55], ellipsoids [24, 63] were proposed. Zonotopes [4, 47] and support functions [72], however, have led to more efficient algorithms. The COntinuous Reachability Analyzer CORA [2] and SpaceEX (State Space EXplorer) the scalable verification tool for affine hybrid systems have been developed on the basis of these algorithms [42]. SpaceEx is by far the most efficient and widely used tool for affine hybrid systems. Ariadne [14, 15, 23] and the Taylor Model-Based Analyzer for Hybrid Systems Flow* [30–32] also use over-approximations to compute the flowpipe of the continuous dynamics. The former makes use of integrations and the latter is based on Taylor model and interval arithmetic. The Toolbox of Level Set Methods [81], however, adopts an entirely different method. It approximates the solutions of a class of Hamilton-Jacobi (HJ) partial differential equations (PDEs) and make thereby use of level set methods.

The drawbacks of set-theoretic approach are the propagation and accumulation of over-approximation errors and the difficulty to handle the logical jump conditions with geometric approximations. Transitions are the main source of over-approximation errors due to the intersection operation which frequently yield large over-approximations. These errors are furthermore bloated because of the wrapping effects accompanying the computation of the flowpipe of the continuous dynamics. For the proof and logic based tools, the main problems lie in the difficulty of handling the continuous dynamics with logic.

Despite this variety of available tools and approaches, practical experiences show that these are not mature enough for the verification of real applications. The major problems are the time and space complexity, which critically increase with the dimension of the application, and the conservative approximation adopted within these tools to represent and compute reachable sets. In fact, it was shown [17, 19, 22, 29, 51] that these methods were pushed to their limits with a continuous state space dimension of more than six when applied to examples inspired from automotive or intelligent transportation fields and even simple academic benchmarks. Moreover, the number of locations, the number of transitions, the jump conditions, their nature and their logic expression complexity may indeed limit their applicability. Another hindrance is the restriction of tools and methods to specific classes of hybrid systems which are practically simple compared with real models.

1.1. Motivation and Objectives

At the beginning of this work, our first goal within the DFG-Priority Program 1305 *Control Theory of Digitally Networked Dynamical Systems* [102] was to carry out a safety analysis of a networked platoon of trucks using existing verifications tools. Platooning can significantly improve the efficiency of existing road systems and increase their capacity. However, communication networks are generally subject to delays, loss of packets and breakdowns. For this reason, a typical objective for this system is to maintain a constant relative distance between trucks. The control design of such cooperative vehicles is practically a challenging task, because the controller must not only guarantee stability, robustness and safety under faults and failures in the mechanics and in the hardware of the vehicles, but also under changes in the topology of the communication network. A critical challenge here is to find a controller which ensures for short and safe gaps between the vehicles in case of nominal communication as well as under disturbances or even under a total loss of communication. It is practically infeasible to tackle all these factors, along with string stability in control design, into account. The idea was to find a controller satisfying a part of these requirements and to check the satisfiability the rest with verification tools. Therefore, it was necessary to derive a hybrid model for the controlled networked platoon and to restrict the verification to some safety-critical scenarios.

Results of the safety analysis of a platoon of 2 and 3 vehicles obtained using some at that time available tools have encouraged us to begin the implementation using zonotopes. The implementation was gradually extended with new approaches and methods striving thereby for more efficiency and better results.

As previously stated, our main goal is the safety analysis of a cooperative platoon of vehicles using reachability analysis. In the achieving of this objective, three further goals are attained. First of all, a new assessment of available tools was a necessary prerequisite for understanding how each of them works. Their individual

strengths and weaknesses had to be investigated. A suite of linear benchmarks was collected for this purpose. Particular benchmarks, like the classical navigation oder the heater benchmarks [37], were extended in order to increase their complexity with regard to the number of locations, the number of transitions and the nature of their jump conditions. In addition, new benchmarks have been specially designed to test the capability of the tools to handle large systems with dimension of more than 20 state variables. An overview of existing theoretical methods and approaches was necessary for making a decisive step towards implementing a new tool.

The implementation first began under MATLAB with the zonotopic approach by adopting thereby the simplest approximations methods for the initial and the input sets and allowing only hyperplane or time jump conditions in transitions. We then worked on support function implementation with the goal of allowing for a more general representation of sets. This implementation was extended with a variety of methods for handling the initial set, the input set, transitions and diverse optimization algorithms for computing the support functions. We offer this implementation for the common use and allow users to create their own reachability analysis. A user-friendly interface was necessary to easily set the parameters and combine different choices. Furthermore, an intuitive textual and a graphical description of hybrid automata was also recommended to make simple and extended input more easier. The graphical description permits a straightforward description of rather small dimensional systems. The textual description, however, can be used for large systems. It was also in our interest to permit an automatic acquisition of control parameters directly in the textual input file. This is beneficial if control decisions are made on the base of verification analysis and practical in case of large systems like the platoon application.

1.2. Contributions

In this work, we only focus on linear time invariant hybrid systems. Our main contributions in this context include:

- A benchmark collection for the evaluation of verification tools. A complete description is given in Appendix A and some of them are available under the link [61],
- An overview of most of the methods and approaches proposed in the literature for computing reachable sets,
- A detailed description and overview of methods for handling guard transitions,
- Suggestion of new proposals for improvement and alternative approaches.
- A user-friendly implementation of a MATLAB toolbox based on support function allowing a choice between the above-mentioned methods,
- A comparative evaluation of guard intersection methods for support functions,

- An assessment and comparative evaluation of different methods for the over-approximation of the input contribution and of the initial set,
- A MATLAB prototypical implementation of the zonotope based reachability analysis,
- A user-friendly implementation of a C++ toolbox using zonotopes allowing the choice between different methods for handling guard condition,
- A comparative evaluation of guard intersection methods and clustering strategies for zonotopes,
- A control design of scalable cooperative platoon of vehicles based on a strategy using H_2 or H_∞ control and reachability,
- A particular fixpoint-transition hybrid model for the platoon of vehicles. This hybrid automaton is marked by transitions which are triggered once subsequent reachable sets remain unchangeable. This kind of hybrid model can generally be used to model networked systems for the goal of safety verification.

Both implemented tools possess intuitive graphic user interfaces to ease the setting of parameters and the option choice. They also allow the user to choose the target location if many transitions can be triggered from the current location. The main goal of these tools is to offer more flexibility towards a user-configurable analysis.

1.3. Outline

This thesis is structured in seven chapters including this introduction and two appendices.

- In Chapter 2, we give an intensive survey and provide a comparative study of the most important available tools to investigate their strengths and weaknesses and demonstrate the limits of their applicability in real applications.
- In Chapter 3, we focus more on the theoretical aspect of this work. We first introduce the basic terminology and definitions of a hybrid automaton. Therefore, we briefly explain key steps towards the computation of reachable sets for linear time invariant systems. We then give a detailed description of available approaches for the approximation of the input contribution and for the initial set. We complete this theoretical overview with a survey of methods for handling transitions.
- Chapter 4 focuses on the theoretical and the implementation aspects of the reachability analysis using support functions. We briefly introduce main definitions and properties. Thereafter, we describe how the methods presented

in Chapter 3 are transformed in terms of support functions. We furthermore provide a detailed description and overview of methods for handling guard transitions and propose new alternative methods. Next, we present a prototypical implementation in the form of a toolbox. We carry out some experimentations with the benchmark suite of Appendix A to demonstrate the applicability of this implementation and at the same time evaluate the performances of different methods in a comparative way.

- Chapter 5 is dedicated to the reachability analysis based on zonotopes. After presenting the approach we adopted for computing the reachable sets for the continuous part of the hybrid automaton, we overview available techniques, adapt approaches used in automatic control and suggest new methods to solve the problem of guard and invariant intersection. We propose several clustering approaches for handling transitions. Thereafter, we present the experimental results with a performance comparison on the various intersection methods.

- Chapter 6 focuses on demonstrating which context reachability can be useful. We used it first to check for the safety conditions guaranteeing no collision inside a networked platoon of three vehicles with a leader ahead. We then demonstrate how reachability can help in the control design of complex systems using the same application but instead of LMI, H_2 and H_∞ control were applied. We subsequently show that time safety-critical conditions can be determined using a platoon approaching an intersection as application.

- In Chapter 7, we conclude our work and address potential research directions.

- In Appendix A, we give a detailed description of the linear hybrid benchmarks collected with the goal of testing verification tools.

- In Appendix B, we present the proofs of Chapter 3.

1.4. Bibliographic Notes

Parts of this thesis were subject of refereed publications.

- Some reflections on the challenges in the verification of hybrid systems mentioned in Chapter 2 can be also retrieved in [33].

- Similar tool assessment results to those presented in Chapter 2 can be found in papers [17, 19].

- The main algorithm for the computation of reachable sets with support functions described in [18] and the adopted approximations for the input contribution and the initial set are reproduced in Chapter 5 Section 4.3.

- The algorithm for the computation of reachable sets using zonotopes presented in Chapter 5 Section 5.8, was used in [20, 21] to carry out a verification analysis of a platoon of 3 vehicles. In [20], the case of nominal communication was treated whereas the case of complete outage of communication was investigated in [21].
- The corresponding results and the hybrid model strategy adopted for the networked platoon appear again in Chapter 6 in Section 6.5.2 and Section 6.5.1 respectively .
- Part of the results presented in Chapter 6 Section 6.6 concerning the H_2-based platoon controller are given in [18].
- Reachability experimentation with a platoon approaching an intersection as presented in [22] are described in Chapter 6 Section 6.7.

2. Assessment and Performance Comparison of Tools

2.1. Introduction

Over the last ten years, many steps have been taken to develop Open Source Toolboxes for the verification of hybrid systems. This has resulted in a variety of academic tools. Some of them, like KeYmaera [87], are purely based on theorem proving techniques. Others, such as HySAT [39] and iSAT [40], are model checkers for generally bounded hybrid systems. Another class of tools uses interval arithmetic and constraint propagation techniques, like HSolver [95], to guide and therewith reduce the complexity of the verification. Some others, however, are purely reachability tools, like for example the tool SpaceEx (State Space Explorer). The latter offers an open development environment for the verification of hybrid systems. It encompasses the toolbox PHAVer [41] for affine systems using polyhedra, the LGG toolbox based on support function techniques [44] and the STC scenario an enhancement of the LGG algorithms. These toolboxes are based on the computation of reachable sets by means of approximations.
The problem of using different toolboxes resides in the fact that these tools differ not only in their numerical approach but also in the input semantic and the type of systems they can analyze. Furthermore, the techniques adopted by these tools are generally valid only for a specific class of systems. For instance, SpaceEx handles in its current version only LTI systems with the LGG and STC scenarios and affine systems with the PHAVer one whereas HSolver can deal with systems with restricted nonlinear dynamics including polynomial, *sin* and *cos* functions. KeYmaera and iSAT, however, impose no restrictions on the nature of the systems and functions they can handle. But the fact that they require a monolithic model, while systems are typically modeled from components, can be considered as restriction.
In recent years, comparative studies of the performances of available tools have clearly revealed that despite considerable effort, the tools have shown their limits particularly in practical applications. In [17] and [51], both tools HSolver and PHAVer have been evaluated using simple hybrid examples. A wide range of tools for modeling and analyzing hybrid systems have also been reviewed and compared in [29]. The simulation tools BHPC and Hybrid Chi as well as the applicability of verification tools like KeYmaera, HySAT and iSAT have been tested in [76] using some commonly known examples. In another work [19], the performances of KeYmaera, HSolver and PHAVer have been assessed with the goal of checking the safety of a cooperative platoon of vehicles. A similar practical example with some

other benchmarks has been used in [22] to compare reachability techniques based on zonotopes and support functions and their approach for computing the invariant of continuous systems.
In this chapter, we investigate the features of the tools SpaceEx, HSolver, KeYmaera and iSAT by using a specific suite of benchmarks [33]. We compare their performances with regard to different aspects, like semantics, dynamics, dimension of the benchmarks, efficiency of the computation, intended results and many others. This review begins with a preview of different benchmarks followed by a brief description of the tools. Numerical results with the corresponding tool settings are thereafter presented and commented. These results are subsequently used to assess the performances of the tools, to conclude about their features and to provide guidelines for improvement and future work.

2.2. Benchmarks

In this section, we give a short description of the benchmarks suite used to test the tools. A detailed description of the benchmarks is, however, given in Appendix A. We confine our analysis to linear hybrid systems as the SpaceEx toolbox is restricted to this class of systems. Our benchmark suite includes some classical benchmarks such as the bouncing ball example, the navigation and the heating benchmarks. It comprises the particular continuous infinity test example for which an exact reachable set can be computed [22]. The colliding masses [50], the two-tank benchmark [47] and the transient in flower benchmark [36] are also considered. The suite is reused with extensions and improvements of some available benchmarks to derive new versions that provide a worthy challenge for the tested tools. We suggest, for example, navigation benchmarks for a 4x4-grid and a 5x5-grid. We moreover propose beyond the known 3 room heating benchmark with 2 movable heaters [94] and the 6 room heating benchmark [5] with 2 anchored heater, a 3 room benchmark and three different layouts for a 4 room benchmark both as well with 2 anchored heaters. We include in the suite the cooperative platoon of trucks. The continuous model involves a functioning communication among the trucks while a hybrid model with spontaneous transitions involves the case where communication among all trucks is lost. The 5-Dimensional Linear Switching System (5DLSS), the result of a self-conceived hybrid automaton, lastly complements the benchmark suite. The modes are governed by a linear dynamics stabilized using the linear quadratic regulation (LQR) technique with transitions between them deduced heuristically. Table 2.1 summarizes essential characteristics of the benchmarks. The corresponding hybrid models vary in the dimensionality of their state vector, the number of continuous modes, the invariant conditions, the number of transitions as well as their corresponding guard and reset conditions. The aim of the variety of benchmarks is to tackle different aspects such as the expressiveness of the input semantic in practice, the efficiency of computation, the accuracy of the output and the limits of applicability of different hybrid system verification tools.

Benchmark	continuous dynamics	dimension	modes	invariants	transitions	guards	guard condition	reset
1. infinity test	Ax+Bu	2	1	yes	-	-	-	-
2. bouncing ball	Ax+Bu+b	4	1	yes	loop	multiple	1 eq.+ 1 ineq.	yes
3. colliding masses	Ax+Bu	4	1	yes	loop	1	1 eq.	yes
4. transient in flower	Ax	2	4	yes	2	1	1 eq.	no
5. two tanks	Ax+Bu+b	2	4	yes	7	1	1 eq.	no
6. nav3x3	Ax+Bu	4	7	yes	16	multiple	1 eq.+ 2 ineq.	no
7. nav4x4	Ax+Bu	4	14	yes	34	multiple	1 eq.+ 2 ineq.	no
8. nav5x5	Ax+Bu	4	23	yes	58	multiple	1 eq.+ 2 ineq.	no
9. 3 rooms+ 2 movable heaters layout Figure A.9(c)	Ax+b	3	7	yes	22	multiple	2 ineq.	no
10. 3 rooms+ 2 fixed heaters layout Figure A.9(a)	Ax+b	3	4	yes	12	multiple	2 ineq.	no
11. 4 rooms+ 2 fixed heaters layout Figure A.9(e)	Ax+b	4	4	yes	12	muliple	2 ineq.	no
12. 4 rooms+ 2 fixed heaters layout Figure A.9(f)	Ax+b	4	4	yes	12	multiple	2 ineq.	no
13. 4 rooms+ 2 fixed heaters layout Figure A.9(g)	Ax+b	4	4	yes	12	multiple	2 ineq.	no
14. 6 rooms+ 2 fixed heaters layout Figure A.9(b)	Ax+b	6	4	yes	12	multiple	2 ineq.	no
15. platoon single mode	Ax+Bu+b	9	1	no	-	-	-	no
16. platoon two modes	Ax+Bu+b	9	2	no	2	spontaneous	-	no
17. 5D LSS	Ax+Bu	5	5	no	5	1	1 eq.	no

Table 2.1.: Essential characteristics of the benchmarks used for the tool assessment.

2.3. Description of Tools

We provide a brief overview of the features offered by each tool involved in this study along with a short description of the different input formats. The usability and the applicability of each tool are also investigated.

2.3.1. SpaceEx: The PHAVer and the LGG Scenarios

SpaceEx is a framework for verification tools of hybrid affine systems. The first release comprised of two scenarios, the LGG support function package and the toolbox

PHAVer (Polyhedral Hybrid Automaton Verifier) [41]. The STC scenario was newly added to SpaceEx to reduce the complexity and tighten the over-approximation at discrete transitions. In our study, we test both the PHAVer and the LGG support function scenarios. The toolbox PHAVer handles piecewise linear bounded derivative systems and affine dynamics described by a conjunction of constraints of the form:

$$a_i^T \dot{x} + \hat{a}_i^T x \bowtie_i b_i, \tag{2.1}$$

with a_i^T, $\hat{a}_i^T \in \mathbb{Z}^n$, $b_i \in \mathbb{Z}$, $\bowtie_i \in \{<, \leqslant, =\}$ and $i \in \{1, \ldots, m\}$. It makes use of the Parma polyhedra library (PPL) [10] to compute a polyhedral approximation of the reachable sets. It additionally includes techniques to control the increasing complexity by limiting the number of bits and constraints of polyhedra used to over-approximate the reachable sets. The LGG scenario uses a method proposed in [72] to compute a template polyhedral approximation of the reachable set for the continuous linear dynamics and the method proposed in [43] to compute this approximation at jumps. The template polyhedral approximating set is the result of the computed support functions in a predefined template of directions. SpaceEx treats linear hybrid automaton with a continuous dynamics described by the following differential equation:

$$\dot{x} = Ax + Bu + b \tag{2.2}$$

with $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times m}$ are constant matrices, $b \in \mathbb{R}^n$ a constant vector and u an uncertain bounded input. The guard are polyhedral constraints and the reset conditions take this form:

$$\mathcal{R}e\,(X) = RX \oplus W, \tag{2.3}$$

with $R \in \mathbb{R}^{m \times n}$, $X \subseteq \mathbb{R}^n$ and $W \subseteq \mathbb{R}^m$. SpaceEx offers the possibility to specify the input with an XML-file. This input can be written by hand or produced using a graphical editor which allows for an intuitive and convenient representation of hybrid systems as hybrid automata. Both input possibilities are illustrated for the bouncing ball example in Figure 2.2 and Figure 2.1 respectively.

SpaceEx also supports compositional and assume-guarantee reasoning analysis. Besides the many possibility offered to visualize results, the graphical user interface also helps to set up parameters to initialize and guide the reachability analysis. We particularly focus on the meaning of some parameters we often used for the LGG scenario during our study. A part from the setting of the time horizon, sampling time and maximum number of iterations, SpaceEx also provides the possibility to control the accuracy and the complexity of the intersection with guards. The problem thereby is that an intersection may occur for many successive iterations. Handling each intersection set separately will lead to state explosion. This problem is addressed by setting the option *clustering* which computes a template hull of all intersections or a percentage of them by setting off or enabling an error tolerance. The option *aggregate sets* computes a convex hull of the already clustered sets and improves therewith the accuracy of the approximation.

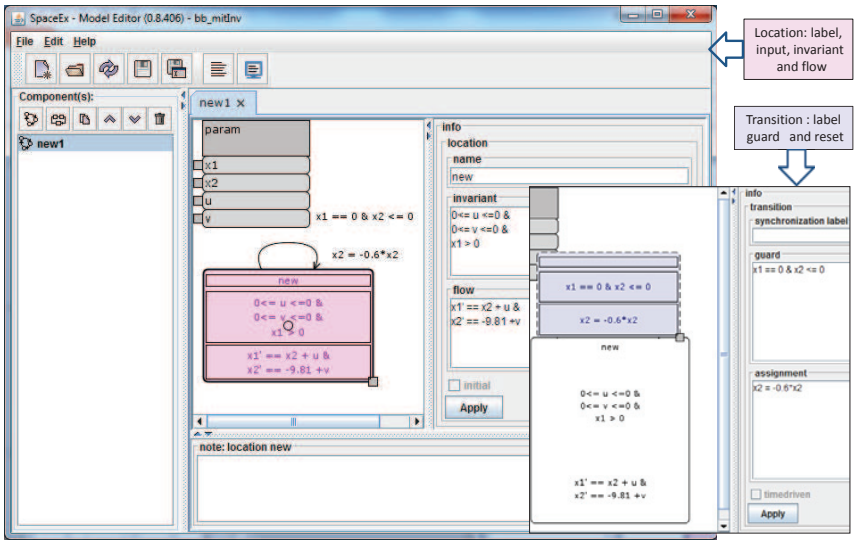
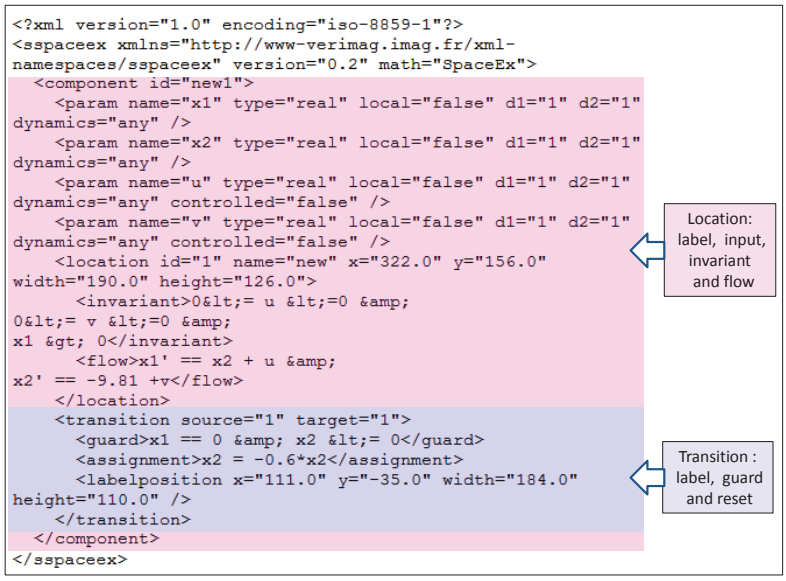


Figure 2.1.: The SpaceEx graphical editor: the bouncing ball model as example. A location is described by a label, an invariant including also the input and a flow which can be edited in the corresponding fields on the right side with a click on the location. The synchronization label, the guard and the reset conditions are shown after selecting the transition.

2.3.2. KeYmaera

KeYmaera is a verification toolbox for hybrid systems based on theorem proving techniques. In KeYmaera hybrid automata are entered as hybrid programs with respect to a specially proposed syntax detailed in [87], [86] and [85]. As example the hybrid program corresponding to the bouncing ball is given in Figure 2.3. The formalism of hybrid programs embeds logic, in particular the differential dynamic logic $d\mathcal{L}$, in a conventional programming framework. KeYmaera allows real-valued formula with linear as well as nonlinear functions. The differential dynamic logic $d\mathcal{L}$ is a straightforward transformation of the differential behavior of the hybrid system. Its related axioms and proof rules are purely syntactic leading consequently to the renouncement of their mathematical semantics to derive verification proofs. Combined with the induction rules of ordinary logics, the $d\mathcal{L}$ logic complements the induction engine of KeYmaera in a way that takes into account the interaction between the discrete aspect and the continuous dynamics of the hybrid system. Besides the verification of specific properties, KeYmaera can generate a counterexample using Counter Example Guided Abstraction Refinement CEGAR-techniques in case the verification fails. Furthermore, KeYmaera offers the possibility to deal with parallel compositions of hybrid systems.




```
<?xml version="1.0" encoding="iso-8859-1"?>
<sspaceex xmlns="http://www-verimag.imag.fr/xml-
namespaces/sspaceex" version="0.2" math="SpaceEx">
  <component id="new1">
    <param name="x1" type="real" local="false" d1="1" d2="1"
dynamics="any" />
    <param name="x2" type="real" local="false" d1="1" d2="1"
dynamics="any" />
    <param name="u" type="real" local="false" d1="1" d2="1"
dynamics="any" controlled="false" />
    <param name="v" type="real" local="false" d1="1" d2="1"
dynamics="any" controlled="false" />
    <location id="1" name="new" x="322.0" y="156.0"
width="190.0" height="126.0">
      <invariant>0&lt;= u &lt;=0 &amp;
0&lt;= v &lt;=0 &amp;
x1 &gt; 0</invariant>
      <flow>x1' == x2 + u &amp;
x2' == -9.81 +v</flow>
    </location>
    <transition source="1" target="1">
      <guard>x1 == 0 &amp; x2 &lt;= 0</guard>
      <assignment>x2 = -0.6*x2</assignment>
      <labelposition x="111.0" y="-35.0" width="184.0"
height="110.0" />
    </transition>
  </component>
</sspaceex>
```

Figure 2.2.: The SpaceEx textual XML-input file of the bouncing ball example.

2.3.3. HSolver

HSolver is a toolbox developed by Ratschan and She [93] to check the safety of a class of nonlinear hybrid systems. It allows the use of nonlinear functions like *sin*, *cos* and *square*, for instance. The implemented strategy is based on the partitioning of the state space into a grid of hyper-rectangles. For this reason, the *STATESPACE* entry is required in the input format of HSolver. This can be seen in Figure 2.4 illustrating the bouncing ball HSolver input file. HSolver avoids the computation of reachable sets and instead uses interval arithmetic and constraint propagation techniques to abstract the flowpipe of discrete states of the hybrid automaton. Furthermore, an interval splitting step is forced in case the resulting abstraction exceeds a permitted threshold, above which the over-approximation is considered to be rough. Splitting can be also avoided by steps where the constraint propagation is incomplete. In addition to this, the method is able to recognize unsatisfiable constraints at an early stage of the computation and thereby eliminates stepwise unreachable sets. Each of these pruning steps will consequently shrink the search state space. Jump conditions, initial states and unsafe states are given by constraints. HSolver outputs some computation steps in textual form and offers the possibility to plot reachable sets.

The verification result returned by HSolver may be one of the following statements:

a) *safe* if no intersection between the unsafe set and the resulting over-approximation of the reachable sets is found or

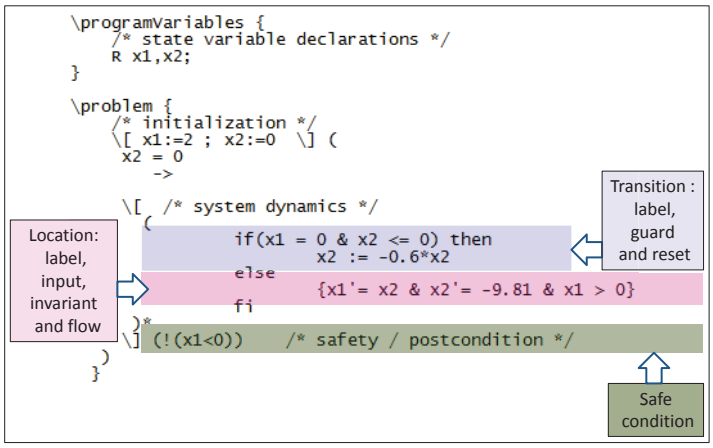


Figure 2.3.: The hybrid program of the bouncing ball example: input file in KeYmaera.

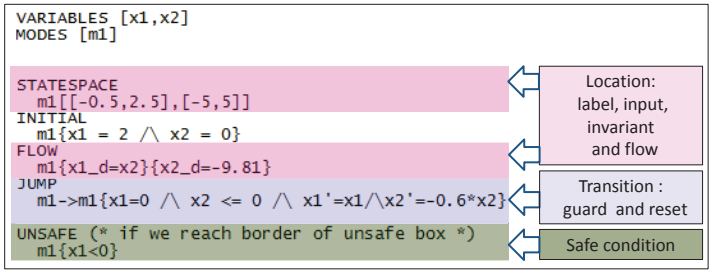


Figure 2.4.: The input file of HSolver for the bouncing ball example.

b) *unknown* otherwise.

2.3.4. iSAT

We introduce in this section a bounded model checker for hybrid systems based on satisfiability checking (SAT) combined with interval constraint propagation technique (ICP) for handling linear and nonlinear constraints of integers as well as real variables. The iSAT algorithm initialized with interval ranges for each type of variables performs its computation in two steps. The decision step consists of first making a variable choice among the set of all variables. Next, the corresponding interval is split in a lower and an upper range [99]. Beginning with the upper range, for example, the algorithm proceeds with the deduction step. It uses unit propagation technique and ICP to deduce new interval ranges for unit clause variables. A unit clause is in general a clause in which exactly one atom remains unassigned and other assigned atoms make the clause inconsistent. The deduction process continues until no more interval refinement is possible. That reveals that a fixpoint

is reached. The possibility of an empty deduced interval range cannot be ruled out and indicates a possible conflict. A conflict resolution analysis helps in a next step to backtrack the search and avoid visiting other already conflicted branches by adding a resolved clause to the original formula. If no further resolution is possible, the unsatisfiability of the problem is proven. The toolbox offers some user parameters to control the progression of the computation and also to enforce termination. The minimal splitting width parameter fixes the threshold at which intervals can be split into sub-intervals. The minimum progress parameter, however, is a measure criterion for the acceptance of the actual computed upper bound in comparison with the last one. The iSAT-ODE is an extension of the iSAT core to allow for the handling ordinary differential equations (ODEs) by unwinding the dynamical behavior of the hybrid automaton as a sequence of transitions involving continuous as well as discrete transitions. The continuous transitions often governed by the ODE of the discrete modes of the hybrid automaton occur between two predefined time instances, a pre-time t_k and a post-time t_{k+1} with $k \in \{0, \ldots, N\}$ for a user defined timestep r and a time horizon $T = Nr$. Discrete transitions, on the other hand, are instantaneous. The following formula:

$$\begin{aligned} \Phi_N = \quad & init(x_0) \wedge trans(x_0, x_1) \wedge \ldots \wedge \\ & trans(x_{N-2}, x_{N-1}) \wedge target(x_N) \end{aligned} \tag{2.4}$$

is, hence, embedded in the bounded model checking engine of the iSAT solver. The predicates $init(x_0)$ and $target(x_N)$ correspond respectively to the initial and the target conditions. The transition predicate $trans(x_i, x_{i+1})$ is an instantiation of the dynamical evolution of the hybrid system within a timestep. Consequently, an instantiation of variables in the iSAT core corresponds to a k-fold unwinding of the constraints in the transition system. An ODE enclosure engine is furthermore integrated to help with the deduction and the decision steps. Regarding the input, the hybrid automaton is given using a textual description, in which a timestep and a state space range must be defined at the beginning. At the end of the file, a definition of a target set is also required. The core of the input file constitutes of a specific syntactical description of the hybrid automaton. A description of this syntax is given in Figure 2.5 for the bouncing ball example.

2.4. Results

We tested the above verification tools with the previous suite of benchmarks. Although some tools like KeYmeara, HSolver and iSAT allow nonlinear dynamics, we restrict our actual study to linear hybrid systems. Our goal is to investigate features such as availability, usability, efficiency, correctness and quality of the results. We will particularly focus on the kind and classes of hybrid systems these tools are able to handle. We take a closer look into the type of constraints permitted by each tools to describe, for example, invariants inside discrete modes, guard as well as reset conditions for transitions. We furthermore examine their performances

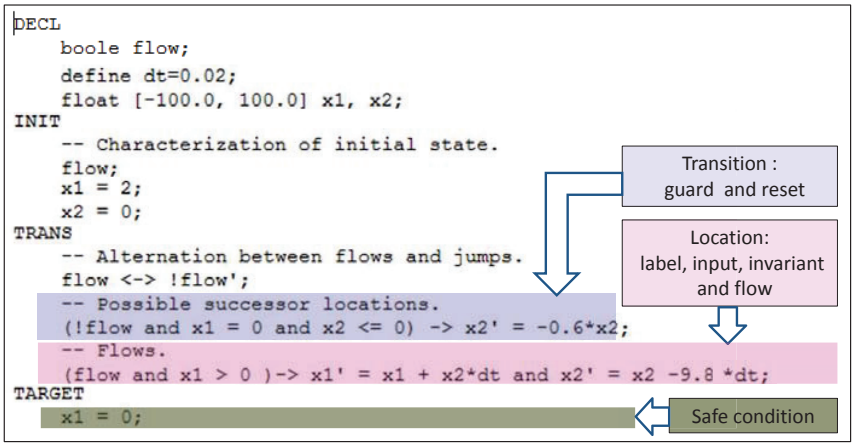


Figure 2.5.: The input file of iSAT for the bouncing ball example.

with regards to the time and space complexity. In addition, we test their ability to handle increasing state dimension. We aim to acquire a meticulously detailed knowledge of the applicability of each tool at the end of these tests.

We first begin with the comparative overview of Table 2.2. It reveals the nature

Tool	dynamics	input set	safety property	visual ouput	guards	invariants
SpaceEx	linear	yes	optional	yes	multiple	yes
PHAVer	linear	no[1]	optional	yes	multiple	yes
KeYmaera	nonlinear	yes	mandatory	no	multiple	yes
HSolver	nonlinear	yes	mandatory	no	multiple	yes
iSAT	nonlinear	no	mandatory	yes	multiple	yes

Table 2.2.: Comparison of some features supported by different tools.

of the hybrid model which can be treated by each tool. In particular, we are interested to know if invariants and multiple guards are considered and in which form they are allowed. It is also of particular interest to know how far it is possible to allow the disturbance or the control input variables to vary within a predefined set. It is moreover important to have an idea of the kind of outputs and information delivered by each tool. Secondly, we focus on the results of each tool separately in order to provide a detailed and concise evaluation and a founded comparison of the tested tools. All tests have been performed on a 2,9 GHz quad-core CPU with 4GB memory, but is not of much relevance for performance measurement because SpaceEx runs on a virtual machine on the specified computer. The virtual machine

[1] In PHAVer, the input u in $\dot{x} = Ax + u$ with $u \in U$ can be considered by using the flow predicate " $\dot{x} - Ax \in U$" in the invariant of the corresponding location. For this study, we consider inputs as part of the flow condition.

has not been modified for our test. The machine under test has 1 virtual CPU and 512 MB virtual memory[2].

2.4.1. SpaceEx

For these tests we used SpaceEx v0.9.7beta with the SpaceEx Web Interface v1.0-BETA1.4. We tested the LGG Support Function scenario and the PHAVer scenario, which corresponds to setup 3 in Table 2.3. In SpaceEx, many user parameters are made available, among others, to guarantee on one hand termination, by restricting for example the local time horizon T, fixing a sampling time r or by limiting the number of allowed transitions *max trans*. Other parameters, such as the relative and absolute error parameters, are used to control the arithmetic precision error during the computation. However, the most critical and impotant parameters are those which have a direct impact on the accuracy and tightness of the over-approximation of reachable sets such as the *flowpipe tolerance*, the *clustering percentage* and the *aggregation* parameters. The first parameter fixes an upper bound error for the over-approximation of reachable sets with their template polyhedron. The second and the third ones are applied specially to make a compromise between accuracy and complexity when handling transitions. In fact, owing to the iterative computation scheme adopted by SpaceEx, the flowpipe may contain many reachable sets that collide with the guard. Handling each intersecting set separately may substantially increase the computational complexity. Initially constructing the template hull of the bundle of intersecting sets may circumvent this problem but result in decreased tightness of the approximation. As a compromise, template hulls can be constructed in a piecewise manner for each subgroup of sets up to the error bound given by the parameter *clustering percentage*. With the *aggregation* parameter, the user can decide whether to compute the Convex Hull (CH) of the template sets resulting from the last step before proceeding to the next step. The user can furthermore choose between *box* directions, *oct* directions or *m uniformly* distributed directions. The number of directions directly effects the shape of the polyhedron used to cover the actual flowpipe. Results can be given as graphs illustrating the flowpipe of the hybrid evolution or as text in form of intervals, or forbidden states are found to be reachable or not. It is moreover possible to know if the algorithm was able to detect the existence of a fixpoint. Given its existence, the corresponding number of transition is displayed.
For the sake of completeness, we tested the impact of the choice of the available user parameters on the results for each benchmark separately. We choose for this purpose the setups shown in Table 2.3 to test SpaceEx. The obtained results corresponding to various examples with varying initializations are given in Tables 2.7, 2.8 and 2.9 for the navigation benchmarks. The results of the heating benchmarks are summarized in Tables 2.10 and 2.11 while verification results for the platoon benchmark are collected in Tables 2.12. Table 2.6 is reserved for results of the two-tank

[2]During our tests the memory and one CPU core have been physically made available for the virtual machine

benchmark with invariants (Figure A.5(b)) and without invariants (Figure A.5(b)). For the remaining benchmarks, the bouncing ball, the colliding masses, the transient in flower benchmark, the infinity test and the the 5-dimensional benchmark the results are collected respectively in Tables 2.4, 2.5 and 2.13.

	scenario	directions	clustering	aggregate sets	sampling time	flowpipe tolerence	time horizon	max. iterations
setup1	LGG	box	-	ch	0.5	-1	50	100
setup2	LGG	oct	-	ch	0.5	-1	50	100
setup3	PHAVer	-	-	-	-	-	-	100
setup4	LGG	oct	50	ch	0.5	-1	50	100
setup5	LGG	oct	50	NONE	0.5	-1	50	100
setup6	LGG	oct	-	ch	0.1	-1	50	100
setup7	LGG	oct	-	ch	0.05	-1	50	100
setup8	LGG	box	-	ch	0.05	-1	50	100

Table 2.3.: Description of the different options used to perform the reachability analysis of the benchmarks.

Certain benchmark tables do not contain information corresponding to the PHAVer setup (setup3 in Table 2.2). This is attributable to the fact that the PHAVer scenario cannot be used to analyze hybrid systems with free input set U as it is not supported. Such is the case in the colliding masses example. In the 3 rooms heating benchmark with exchangeable heaters, however, the input u has been considered to be a constant and the model is constructed accordingly.

	Infinity test example			Bouncing ball example		
	fixpoint found	time (s)	forbidden states reached	fixpoint found	time (s)	forbidden states reached
setup1	1	0.06	yes	no	30.678	yes
setup2	1	0.092	yes	no	2.157	yes
setup4	1	0.1	yes	no	2.059	yes
setup5	1	0.095	yes	6	0.096	yes
setup6	1	0.361	yes	no	3.352	yes
setup7	1	0.715	yes	no	4.42	yes
setup8	1	0.352	yes	no	282.37	yes

Table 2.4.: SpaceEx results for the infinity test and bouncing ball examples.

The infinity test benchmark

With regard to the results for the infinity test benchmark shown in Table 2.4, we note in spite of the system being conceived safe, no setup was able to prove this fact.

The bouncing ball example

The benchmark results for the bouncing ball example are summarized in Table 2.4. We note the remarkably long computation time for setups 1 and 8. The computation using setup 5 is faster owing to the early detected fixpoint. Although the system is theoretically safe, our different SpaceEx setups have failed to prove its safety.

	Colliding masses			Transient in flower		
	fixpoint found	time (s)	forbidden states reached	fixpoint found	time (s)	forbidden states reached
setup1	3	0.199	yes	no	37.680	yes
setup2	3	0.673	yes	no	36.248	yes
setup3	-	-	-	6	0.002	yes
setup4	3	0.742	yes	no	30.065	yes
setup5	3	0.668	yes	7	1.192	yes
setup6	4	2.829	yes	no	127.769	no
setup7	-	-	-	no	110.47	no
setup8	3	1.266	yes	no	114.366	no

Table 2.5.: SpaceEx results for the colliding masses and the transient in flower examples.

The colliding masses example

The results for the colliding masses benchmark are shown in Table 2.5. Similar to the bouncing ball example, the unattainability of the forbidden states, which we have intendedly chosen to be unreachable, has not been proven using the mentioned setups.

The transient in flower benchmark

The results for the transient in flower benchmark are shown in Table 2.5. The most remarkable ones are the computation times and the fixpoint results obtained with PHAVer (setup 3) and with setup 5. The PHAVer scenario is the fastest setup at $0.002s$ while setup 5 with a clustering percentage of 50 and a deactivated convex hull option at the aggregation step is the second fastest one. We further note that, contrary to the results obtained with the setups 1-5, the system is proven to be safe with the setups 6-8.

The two-tank benchmarks

For our tests, we consider the two-tank benchmark with and without invariants. The corresponding hybrid automata are illustrated in Figure A.5. For simplicity, we choose to check whether x_1 or x_2 could reach level -1 under an uncertain input verifying $u \in [-0.1, 0.1]$.

The results are given in Table 2.6 for the benchmark both with and without invariants. We note that unlike the two-tank benchmark without invariants, a fixpoint was found for the benchmark with invariants with all chosen setups. Comparing both *forbidden states reachable* columns in Tables 2.6, it is interesting to remark how the presence of invariants in modes could completely reverse the results of the verification. In fact, the benchmark with invariants is proven to be safe with all setups, but not so in the case without invariants. It is particularly notable the computation times are reduced when invariants are considered as compared to when they are not considered. Furthermore, the results of the benchmark with invariants show that the number of iterations necessary to reach a fixpoint is dependent on the chosen setup but does not necessarily increases with small timesteps.

	Without invariants			With invariants		
	fixpoint found	time (s)	forbidden states reached	fixpoint found	time (s)	forbidden states reached
setup1	no	6.455	yes	7	0.092	no
setup2	no	9.928	yes	7	0.114	no
setup4	no	9.702	yes	7	0.104	no
setup5	no	9.174	yes	8	0.128	no
setup6	no	39.495	yes	3	0.088	no
setup7	no	87.988	yes	3	0.136	no
setup8	no	43.105	yes	4	0.117	no

Table 2.6.: SpaceEx results for the two-tank benchmark without and with invariants.

The navigation benchmark

We carry out a verification analysis for the navigation benchmarks with the setups of Table 2.3 and the same initial and safe conditions as specified in Appendix A. A comparison of the results in Table 2.7 obtained with setups 2 and 6 for the 3x3-navigation example with the initial point given in Table A.1 shows that the choice of a smaller sampling time does not necessary lead to longer computation time. The same observation can be made for setup 6 and setup 7. Only setup 1 was not able to prove the system safety. We furthermore note that increasing the number of directions or reducing the timestep can help to prove the unattainability of the forbidden states. This comes, however, at the expense of the computational

efficiency. The fastest safety proof was performed with the PHAVer scenario (setup 3 in Table A.1).
Similar to the results with an initial point, the results with the initial set in Table 2.7 show that with setup 1, the forbidden states are proved to be reachable. The PHAVer scenario is again shown to be the most efficient reachability analysis method for this example. Contrary to the same example with an initial point, we remark in this case that decreasing the sampling time results in considerably slower computations. PHAVer was also the only setup capable of computing a fixpoint after 6 iterations for the initial point and after 19 iterations for the initial set.
For the 4x4-navigation benchmark with an initial point, we note in Table 2.8 from

	With initial point			With initial set		
	fixpoint found	time (s)	forbidden states reached	fixpoint found	time (s)	forbidden states reached
setup1	no	9.690	yes	no	9.328	yes
setup2	no	28.173	no	no	23.787	no
setup3	6	0.061	no	19	0.321	no
setup4	no	20.770	no	no	20.649	no
setup5	no	12.490	no	no	15.640	no
setup6	no	13.605	no	no	38.912	no
setup7	no	12.345	no	no	60.528	no
setup8	no	4.841	no	no	15.198	no

Table 2.7.: SpaceEx results of the 3x3-navigation benchmark with the initial point given in Table A.1 and the initial set in Table A.2 .

setup1 and setup 8 that combining a smaller sampling time with the *box directions* option can in some cases speed up the computation. In addition, we observe that the PHAVer scenario was not able to prove the safety with this example. We notice also that the number of directions does not seem to affect the reachability of forbidden states. Furthermore, the non-reachability of forbidden states was proven by reducing the timestep for the point initialization but failed for set initialization.
By comparing the results corresponding to the *time elapsed* in setup 1 and setup 8, it appears that the choice of a smaller timestep does not necessarily lead to longer computation time. It is noteworthy that the safety proof failed with setups 6,7 an 8 for the 4x4-navigation benchmark with an initial point while succeeding with an initial set.

Using point initialization, the PHAVer scenario was still the fastest but was unable to prove the safety of the system. For other setups, the results for the 5x5-navigation benchmark have been found to be largely similar to those of the 4x4-navigation benchmark. setups 2,4 and 5, however, show an increasing time complexity corresponding to the complexity of the benchmarks. Setup1 proved

	With initial point			With initial set		
	fixpoint found	time (s)	forbidden states reached	fixpoint found	time (s)	forbidden states reached
setup1	no	7.712	yes	no	10.817	yes
setup2	no	13.471	yes	no	28.399	yes
setup3	no	0.061	yes	no	1.372	yes
setup4	no	12.056	yes	no	22.854	yes
setup5	no	4.869	yes	no	17.790	yes
setup6	no	19.636	no	no	76.023	yes
setup7	no	34.943	no	no	137.568	yes
setup8	no	6.973	no	no	37.990	yes

Table 2.8.: SpaceEx results of the 4x4-navigation benchmark with the initial point given in Table A.1 and the initial set in Table A.2 .

to be an exception, taking about a minute less in computation time on the 5x5 benchmark than in the 4x4 benchmark.

The heating benchmark

The results for the heating example with 2 exchangeable heaters are presented in Table 2.10. Similar to the 5x5-navigation with initial set, only the PHAVer scenario proved the reachability of forbidden states.

The results for the heating benchmark with 6 rooms and 2 fixed heaters are shown in Table 2.11. Comparing the elapsed time using the setups 1 with 8 and 2 with 7, it is observed that the choice of a smaller timestep does not necessarily yield a longer computation time for this particular benchmark.

The results for the heating benchmark with 2 fixed heaters and 3 rooms presented in Table 2.10 are similar to those of the benchmark with 6 rooms despite the justifiable difference in the computation time. It is particularly notable that a combination of smaller timestep and more directions does not necessarily slow down the computation.

The results for the heating benchmarks with 4 rooms and 2 fixed heaters using room layouts in Figure A.9(d),(e) and (f) are given in Table 2.11. These results are similar to the results of the benchmark with fixed heaters.

The platoon benchmark

The results for the platoon benchmark are collected in Table 2.12 with the two-mode benchmark requiring a longer time for computation as compared to the single-mode

	With initial point			With initial set		
	fixpoint found	time (s)	forbidden states reached	fixpoint found	time (s)	forbidden states reached
setup1	no	6.544	yes	no	24.212	yes
setup2	no	17.708	no	no	36.569	yes
setup3	no	2.252	yes	no	2.591	yes
setup4	no	15.473	no	no	30.509	yes
setup5	no	14.149	no	no	20.477	yes
setup6	no	52.698	no	no	225.098	yes
setup7	no	39.126	no	no	204.610	yes
setup8	no	13.435	no	no	59.805	yes

Table 2.9.: SpaceEx results of the 5x5-navigation benchmark with the initial point given in Table A.1 and the initial set in Table A.2 .

platoon. It is important to note, at this juncture, that it is occasionally useful to simultaneously reduce the timestep and set the *oct* option for the direction choice to produce a different result, e.g. for example setup 7 from Table 2.12. In fact, with this setup, the system has been proved to be safe while all the other setups have failed to prove the safety. Moreover, a fixpoint was found in all setups after the first iteration for the single mode and after the seventh for the two modes example. For the platoon benchmark, we are interested in finding the minimum gaps assuring a collision-free platooning. These can be determined by taking the maximum boundaries of the reachable sets in the specific state variables, in this case the state variables x_1, x_4 and x_7 corresponding to the gaps e_1, e_2 and e_3. We chose 128 for the direction option, the time horizon $T = 20s$ and the timestep $r = 0.001$. The resulting reachable sets are shown in Figure 2.6. We note the maximal values in the negative direction. We therefore deduce that under full communication, a collision-free platooning is guaranteed if $e_1 > 26m$, $e_2 > 13m$ and $e_3 > 10m$. In case of an abrupt breakdown of the communication between all trucks, the safe gaps become larger: $e_1 > 29m$, $e_2 > 26m$ and $e_3 > 11m$.

The 5D linear switching system benchmark

Results for the 5 dimensional linear switching benchmark are shown in Table 2.13. In this example, setups 1, 4 and 5 caused an internal error marked here with E. We observed that with the setup 2, this error disappeared but the computation took longer than three hours. The SpaceEx development team has since resolved this problem. In addition, setups 6, 7 and 8 which are characterized by small steptimes ended with a collision detection between the flowpipe and the chosen forbidden states. Moreover, it is worthnoting the increase in computation time of around eight-fold after switching the direction choice from *box* (setup 8) to *oct* (setup 7).

	Movable heaters			Fixed heaters		
	fixpoint found	time (s)	forbidden states reached	fixpoint found	time (s)	forbidden states reached
setup1	no	34.523	yes	no	15.113	yes
setup2	no	65.207	yes	no	32.470	yes
setup3	43	0.17	no	-	-	-
setup4	no	59.584	yes	no	24.756	yes
setup5	no	91.488	yes	no	675.657	yes
setup6	no	625.658	yes	no	11.597	no
setup7	no	1108.20	yes	no	190.671	no
setup8	no	425.207	yes	no	8.912	no

Table 2.10.: Results of the 3 rooms heating benchmark with movable heaters (layout c Figure A.9).

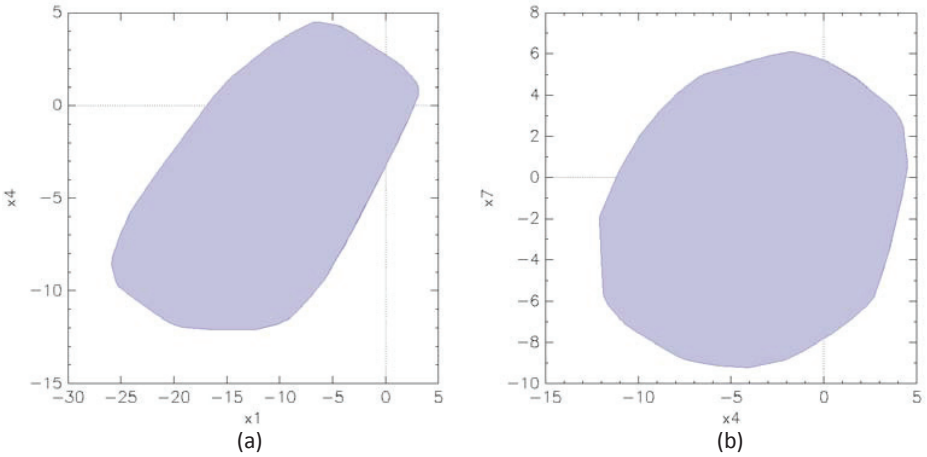


Figure 2.6.: SpaceEx results of the one mode platoon for a time horizon $T = 20s$, a timestep $r = 0.01s$, 128 as a direction choice. (a) x1x4 corresponding to the gaps e_1 and e_2. (b) x4x7 corresponding to the gaps e_2 and e_3.

Figure 2.8 shows the projection of the flowpipe on the plane x_1x_2 computed with the timestep $r = 0.01$ for a time horizon $T = 10s$, the *oct* option for the directions and 10 maximal allowed transitions.

2.4.2. KeYmaera

We tested KeYmaera on an opensuse 13.1 (milestone 4) 64bit operating system. For this study, we use KeYmaera 3.5 and Mathematica 9.0. We took the same configurations given by the first initializing of KeYmaera, except the maximal number

		4 rooms layout d	4 rooms layout e	4 rooms layout f	6 rooms layout b	
	fixpoint found	time (s)	time (s)	time (s)	time (s)	forbidden states reached
setup1	no	26.067	50.291	15.113	84.550	yes
setup2	no	101.178	155.723	97.938	894.191	yes
setup4	no	24.756	122.294	62.379	581.752	yes
setup5	no	48.132	103.284	47.882	486.511	yes
setup6	no	20.410	29.714	20.743	126.886	no
setup7	no	190.671	31.436	42.624	127.550	no
setup8	no	10.362	7.889	8.207	16.110	no

Table 2.11.: Results of the heating benchmark with 4 and 6 rooms and 2 fixed heaters (layout b,d,e,f in Figure A.9).

	With one mode			With two modes		
	fixpoint found	time (s)	forbidden states reached	fixpoint found	time (s)	forbidden states reached
setup1	1	0.218	yes	7	0.985	yes
setup2	1	1.766	yes	7	45.026	yes
setup4	1	1.786	yes	7	45.418	yes
setup5	1	1.802	yes	7	45.255	yes
setup6	1	8.784	no	7	199.17	yes
setup7	1	17.162	no	7	398.293	no
setup8	1	2.024	no	7	8.332	yes

Table 2.12.: SpaceEx results for the platoon benchmark with a single mode and with modes.

of rules, which was set to 5000. KeYmaera allows the user to change this configuration by choosing between different external plug-ins such as the tools QEPCAP, RedLog, Z3-SMT, Orbital and Mathematica for simplifying, solving differential equations and proving quantified formulas involving real arithmetic. The user can also add new rules to the rule base of KeYmaera. Setting these operations, however, requires a well-founded knowledge of the techniques behind KeYmaera. For this reasons, we confined our study to the initial KeYmaera configuration. Compared to SpaceEx, KeYmaera requires a property to be proven and has no visual output to display trajectories. We use the same forbidden/unsafe states for KeYmaera as before for SpaceEx.

Table 2.14 summarizes the results obtained with KeYmaera for the various supported benchmarks. During our experience with KeYmaera, we noted that the computation time changed considerably for some benchmarks after running the

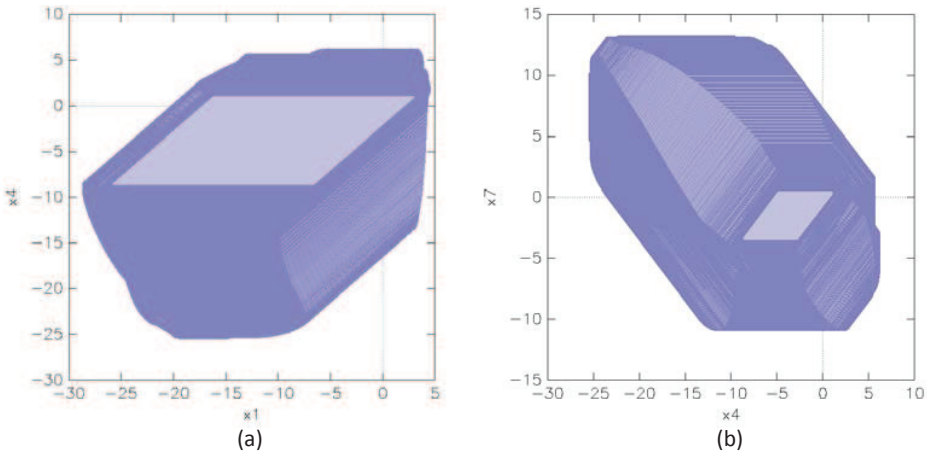


Figure 2.7.: SpaceEx results of the two mode platoon for a time horizon $T = 20s$, a timestep $r = 0.01s$, *oct* as a direction choice. (a) x1x4 corresponding to the gaps e_1 and e_2. (b) x4x7 corresponding to the gaps e_2 and e_3.

same test. This phenomena was particularly noticeable in the heating benchmark with exchangeable heaters and the platoon benchmark with 2 modes as shown in line 6 and line 13 of Table 2.14. A possible reason for this behavior could be the underlying Java virtual machine and the load of the operating system.
KeYmaera was only able to prove the safety of just a few of the examples, specifically the heating benchmark with exchangeable heaters and the bouncing ball benchmark.

2.4.3. HSolver

It was not possible to test all benchmarks because HSolver does not allow free input variables. In fact, only a constant input is supported. Consequently only the navigation benchmarks, the bouncing ball, the colliding masses, the transient in flower benchmark and the heating benchmark with exchangeable heaters have been tested. Specially the heating benchmark could be tested because it has a constant input u which does not change and does not need to be represented as variable in the model file.

We performed the HSolver tests on the same machine and operating system as KeYmaera and we used for this purpose the HSolver 3.1 release. Compared to the executions with KeYmaera, HSolver seems not to benefit from multicore CPUs. In fact, we observed that three of the four cores were nearly idle during the whole computation. However, KeYmaera has involved more than one CPU core for its computation.

HSolver requires a safety property to perform the verification. We chose the same as for KeYmaera and SpaceEx. The HSolver results are shown in Table 2.15. Cells with $> 3h$ mean that the computation took longer than 3 hours and was therefore

	fixpoint found	time (s)	forbidden states reached
setup1	no	E	-
setup2	no	>3h	-
setup4	no	E	-
setup5	no	E	-
setup6	no	216.809	yes
setup7	no	2843.04	yes
setup8	no	333.209	yes

Table 2.13.: SpaceEx results for the 5D linear switching system (5D LSS) example

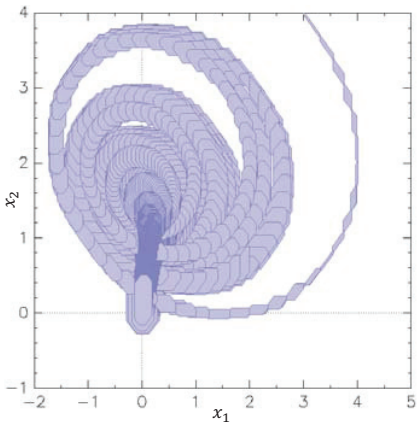


Figure 2.8.: The SpaceEx result of the 5D LSS: projection of the reachable sets on the plan x_1x_2.

aborted. The safety has been however proven only for the bouncing ball example.

2.4.4. iSAT

In this study, we used iSAT 1.0 release. Because of the restrictive iSAT input format, only the benchmarks in Table 2.16 have been tested. For the platoon and the two-tank benchmarks with invariants shown in Figure A.5(b), the computation was stopped after 3 hours. The result of the verification obtained for the colliding masses and the transient in flower benchmark was *unknown*. We have observed after testing both benchmarks that the output has shown intervals intersecting the chosen forbidden states.

2.4.5. Performance Evaluation

As previously mentioned, we explored the features of the tools with specific benchmarks varying from commonly known to practical ones. We proposed examples scalable in their state dimension, in the number of discrete modes and discrete transitions. The continuous dynamics inside discrete modes are given by linear ordinary differential equations. We allow the input to vary within a given range for some examples, like in the platoon benchmark. The transitions are described by guards given as a conjunction of equalities or inequalities and affine reset conditions.
The first problem we faced during this study is to put our suite of benchmarks in the appropriate semantic and input format of each tool. While this task was straightforward and intuitive for SpaceEx and HSolver, this was not the case for iSAT and KeYmaera. The unintuitive iSAT modeling of transitions was particularly difficult

Benchmark	time (s)	open goal	property proved
infinity test	0.4	1	no
bouncing ball	2,7	0	yes
colliding masses	6.6	1	no
transient in flower	50.0	1	no
two tank example with invariants	30.7	1	no
Nav3x3	105.7	1	no
Nav4x4	123.7	1	no
Nav5x5	248.9	1	no
3 rooms 2 movable heaters (layout Figure A.9(c))	6.5 to 11.3	0	yes
3 rooms 2 fixed heaters (layout Figure A.9(a))	17.9	1	no
4 rooms 2 fixed heaters (layout Figure A.9(e))	32.9	1	no
4 rooms 2 fixed heaters (layout Figure A.9(f))	30.0	1	no
4 rooms 2 fixed heaters (layout Figure A.9(g))	28.2	1	no
6 rooms 2 fixed heaters (layout Figure A.9(b))	113.5	1	no
platoon with single mode	79.2	1	no
platoon with two modes	132.7 to 166.1	1	no
5D LSS	523.7	1	no

Table 2.14.: KeYmaera benchmark results.

because the systems are encoded as formulas decoupled from their hybrid model. A similar problem arose with KeYmaera especially for large examples like the 5×5-navigation benchmark. Furthermore unlike SpaceEx, no possibility is offered by HSolver and iSAT 1.0 to test benchmarks with uncertain inputs varying within a given range. In contrary to SpaceEx, HSolver, iSAT and also KeYmaera request a safety property to proceed with the verification process. For HSolver and iSAT, we have to additionally define the state space which corresponds to the search domain for the state variables (see lines 3 and 4 in Figure 2.4 and line 4 in Figure 2.5). SpaceEx is the only tool among them offering a graphical model builder and thereby making the introduction of complex examples more easier.

Depending on the possibilities offered by the different tools, the user has more or less complete control over the whole computation process. In fact, while HSolver and iSAT only allow the user to set few parameters like the timestep, SpaceEx offers a set of parameters, which can be optimally tuned to meet intended requirements. Besides the standard parameters like the time horizon and the sampling time, SpaceEx allows the user to decide on the precision and efficiency of the computation of the reachable sets when encountering a jump. This could be done by setting the *clustering* and *aggregation* parameters. However, if we try to make a general observation of the contribution of the clustering and aggregation engine of SpaceEx to time reduction, we note that a switch from setup 2 to setup 4 or setup

Benchmark	time (s)	reachability
bouncing ball	<1s	safe
colliding masses	>3h	?
transient in flower	>3h	?
Nav3x3	>3h	?
Nav4x4	>3h	?
Nav5x5	>3h	?
3 rooms 2 movable heaters (layout Figure A.9(c)	>3h	?

Table 2.15.: HSolver benchmark results.

Benchmark	time (s)	iterations	reachability
colliding masses	0.12	20	unknown
transient in flower	1.18	13	unknown
two-tank benchmark with invariants	>4h	82	aborded
platoon with single mode	>4h	149	aborded
platoon with two modes	>3h	468	aborded

Table 2.16.: iSAT benchmark results.

5 could result in a significant decrease in computation time for benchmarks with many transitions, as shown in the heating and the navigation benchmarks. The expected improvement by enabling the convex hull in the aggregation step could not generally be confirmed by our results obtained with setup 5.

Regarding KeYmaera, the fact that hybrid automata are represented as hybrid programs with the usual programming language constructs offers users many possibilities to describe their systems decoupled from the formal hybrid model. This can yield different hybrid programs for the same hybrid system. Our experience with the platoon benchmark has shown that different hybrid programs can lead to different computation times. Furthermore, KeYmaera allows the choice between different SMT-solves, ODE-solvers, counterexample generators, first order strategies and many others for besides the usual parameters. The problem with the configuration of the parameters lies in the need for in-depth knowledge on theorem proving techniques and on the way KeYmaera works.

We note generally that although SpaceEx runs in a virtual machine, it achieves the best results. Furthermore, the geometric approach seems to be more feasible than the other approaches for linear hybrid systems.

2.5. Conclusion

We tested toolboxes for the verification of hybrid systems with a particularly selected range of benchmarks. We focused on linear benchmarks which differ in their state dimension, the number of modes as well as the number of transitions of their corresponding hybrid automata. Therefore, while the differential equations describing the flows inside the modes of some systems include an input varying in predefined range, the others only have a constant or no input. In addition, the benchmarks can differ depending on whether the guard transitions are described by equalities or inequalities.
The tools are based on different techniques. While the LGG scenario of SpaceEx uses reachability technique based solely on support functions, HSolver applies interval constraint propagation technique in combination with pruning and refinement to check for intersection of reachable sets with unsafe regions. KeYmaera, however, is a first-order theorem prover able to provide a proof for the validity of predefined properties or a counterexample otherwise. The toolbox iSAT proposes a merging of interval constraint propagation technique into a SAT solver which is in further release enhanced with a nonlinear enclosure technique based on Taylor expansion and Taylor model arithmetic. With the exception of SpaceEx, all the other tools, with some restrictions for HSolver, allow for nonlinear hybrid systems.
It was possible to process all proposed benchmarks with the verification and the computation of the reachable sets using SpaceEx. It was moreover possible to acquire better results by making an appropriate choice of user parameters like the *timestep*, *time horizon*, *clustering* and *aggregation*. However, the set of valid benchmarks shrinks considerably if tested with KeYmaera, HSolver and iSAT. The obtained results have shown as well to take a turn for the worse when KeYmaera, HSolver and iSAT are subsequently used.
Effective decisions on choosing the right tool to use for the purpose of verification require not only the gathering of information about the tool itself and the method behind it, but, in many of the cases, require also a comparative evaluation of the obtained results with results issued from a variety of other tools. Although the most available tools are based on profound theoretical concepts, proven methods and continually improving techniques, experiences like ours have shown that their success rate still remains below the expectations regarding the respective needs and objectives of the verification of practical hybrid systems. Furthermore, the recent tendency to fuse different techniques in the same framework, like for example the tools iSAT3 [100] and dReach [27], has shown its limits [33]. It will be interesting for future work to find an explanation of the problems related to each promising technique. This knowledge may allow a combination of these methods in an optimal way. Alternatively, it is possible to offer these techniques on the same framework with a Hybrid Systems Interchange Format like HSIF [84], [103] and to pave in future work the way for possible combinations. This is certainly bound to an enormous effort and consolidated knowledge. However, the first step in this direction has been already taken with the platform SpaceEx. We are working on an open-

source platform which will provide a complete library of set representations (like, polytopes, zonotopes, support functions, Taylor models,...) with the corresponding set operations. Additionally, we are building many verification algorithms and implementing techniques using this library. This work is progressing within the framework project HyPro [61]. For future work, this platform can be potentially extended by introducing and allowing combinations of these techniques with SMT-solvers.

3. Overview of Methods for Computing Reachable Sets of Linear Hybrid Systems

3.1. Introduction

For a continuous dynamical system, the computation of reachable states must guarantee the enclosure of all possible trajectories given by the dynamics governing the behavior of the system and that beginning from all points included in an a-priori defined initial set. Generally, to assure computer termination this task is done over a bounded time.

Using simulation methods to check for safety is obviously infeasible if we have to account for uncountable initial set or for systems submitted to disturbances or to uncertain inputs. On the other hand even reachability has been shown to be generally undecidable. Decidability has been proven under stringent conditions on the continuous dynamics for particular classes of hybrid systems such as timed, multirate and rectangular automata [56, 91], piecewise constant derivative systems, systems with linear vector fields and subclasses of linear hybrid systems with either a nilpotent or diagonalizable with rational eigenvalues system matrix or also matrices with purely imaginary eigenvalues and 2x2 block diagonal Jordan form [65, 66, 89]. But works like [6, 8, 9] have revealed that even inside these classes the decidability problem remains for particular low-dimensional systems unresolved. Furthermore, the proposed algorithmic solutions have shown their limits in the practice, even for low dimensional systems [17, 19].

Two different directions have been fundamentally adopted during the last years to tackle this problem. The first direction uses logical formalizations involving for example temporal logic, abstractions and bisimulations to choose under some conditions a finite number of trajectories sufficiently covering the dynamical behavior of the whole system and prove the correctness of a-priori predefined condition thereafter. The second direction, however, over-approximates the set of states reached by all possible trajectories using well-known geometric shapes such as polytopes [60, 64, 80], ellipsoids [24, 63], boxes [25, 92, 104], polyhedra [41, 44, 52] and zonotopes [48, 71] and checks afterwards the intersection with an unsafe set.
In this chapter, we are concerned with linear hybrid automata in which each discrete mode is described by an LTI-system. The guards are defined by equality/inequality conditions. Resets are simple linear maps.

The reachability analysis of hybrid systems consists of computing an approximation of the reachable set in each discrete mode beginning from an initial set. This is, generally, the result of the intersection of the reachable set of the mode before the transition and the guard triggering this mode.

We provide an overview of methods and techniques developed in the last few years to compute reachable sets based on over-approximation. We handle different aspects related to the computation of reachable sets in continuous modes beginning with the deduction of a recursive schema less exposed to problems such as the wrapping effect. This requires a close approximation of the system input contribution and in a next step an accurate approximation of the initial set. Many approaches have been proposed to solve these two approximation problems. On the other hand, we describe methods for handling transitions with a particular focus on approaches for computing the intersection with guards if many successive reachable sets are involved in this intersection.

This chapter gives a survey of already available methods enriched with our suggestions for improvement and approaches like the approximation of the input set contribution under piecewise assumption. A practical performance comparison of these methods using support functions is given in Chapter 4

3.2. Hybrid Automaton

A hybrid automaton is a framework to formally describe a system characterized by an interaction between continuous dynamics and discrete events. It is a graph $G = (Q, Trans)$ with vertices in Q describing different continuous behaviors the system can take under specific conditions in Inv and edges in $Trans$ specifying the jump and eventually the reset conditions between them. The vertices are called locations or discrete modes of the hybrid automaton while the edges are referred to as transitions. A formal definition is given as follows:

Definition 1. A hybrid automaton is generally given as tuple

$$H = (Q, Var, Inv, Flow, Trans, Init)$$

where:

- $Q = \{q_1, \ldots, q_p\}$ is the set of different locations. A *state* is a pair $\langle q, v \rangle$ of a location q and $v : Var \to \mathbb{R}$ a valuation over Var.
- Var is the set of state vectors associated to each location in Q.
- Inv is the set of domains corresponding to each location in Q.
- $Flow$ is a set of continuous maps describing the continuous behavior of each location.

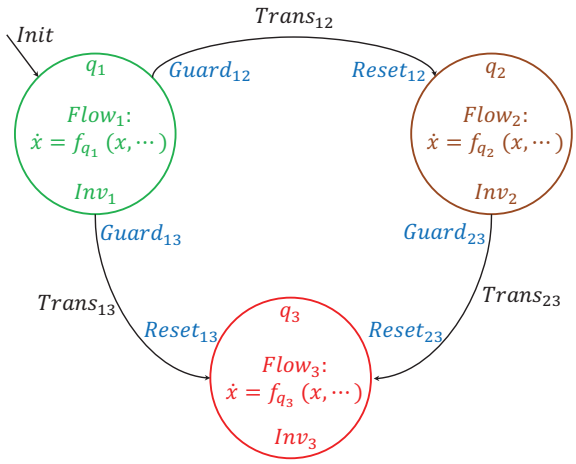


Figure 3.1.: Formal definition of a hybrid automaton.

- $Trans$ is a set of edges. Each edge $e = (q_i, q_j)$ of them is characterized by a *source* location q_i and a *target* destination q_j as well as by a $Guard_{ij}$ condition and a $Reset_{ij}$ map. The guard condition is a condition on the continuous variables that enables the discrete transition once this is satisfied. The reset map is a relation that describes how the continuous state vector changes when entering the target location.
- $Init$ consists of an initial condition also called initial set X_0 as well as an initial location in Q (see Figure 1).

3.3. Run Semantics

The semantics of a hybrid automaton is specified by two different types of transitions:

- The continuous transition describes the evolution over the time of state variables according to the flow inside the same location.
  $\langle q_i, r\rangle \xrightarrow{t}_C \langle q_i, s\rangle \iff$ $\exists$ a continuous trajectory ξ such that $r = \xi(0)$, $s = \xi(t)$ and for each $t' \in [0, t[$ Inv_i and $Flow_i$ hold.
- The discrete transition which corresponds to an instantaneous transition between two different locations.
  $\langle q_i, r\rangle \xrightarrow{e}_D \langle q_j, s\rangle \iff$ $e \in Trans$ and $Inv_i, Guard_{ij}, Reset_{ij}, Inv_j$ hold.

A sequence of alternating continuous and discrete transitions defines a *run* of a hybrid automaton.
A state s is reached by the hybrid automaton H from an initial set $Init$ if $\exists s_0 \in Int$ and a run of H from s_0 to s.

3.4. Reachable State within a Discrete Mode

In general linear systems are defined as follows.

Definition 2. Linear System [12] The dynamics of linear systems is described by the following differential equation for a $t \in \Gamma = [t_0, t_f]$

$$\begin{array}{ll} \dot{x}(t) & = A(t)x(t) + B(t)u(t) \\ x(t_0) & = x_0 \in X_0, \\ u(t) & \in U \end{array} \tag{3.1}$$

where:

- $A, B, u : \Gamma \rightarrow \mathbb{R}^{n\times n}, \mathbb{R}^{n\times m}, \mathbb{R}^m$ Lebesgue-integrable and
- $X_0, U \subset \mathbb{R}^n$ respectively the nonempty initial and input sets,

The state $x(t)$ reached from $x(t_0)$ at time t is given by

$$x(t) = \Phi(t, t_0)x(t_0) + \int_{t_0}^{t} \Phi(t, s)B(t)u(s)ds \tag{3.2}$$

where $\Phi(t, t_0) \in \mathbb{R}^{n\times n}$ is the transition matrix or the fundamental solution of the homogeneous counterpart of (3.1).

Remark 1. For any t, t_0, τ and σ, the transition matrix satisfies the following properties:

- $\Phi(t, t) = I_n$ where I_n is the identity matrix
- $\Phi(t, \tau) = \Phi(t, \sigma)\Phi(\sigma, t)$ the semi-group property of the fundamental solution
- $\frac{\partial}{\partial t}\Phi(t, \tau) = A(t)\Phi(t, \tau)$
- $\Phi(t, t_0)^{-1} = \Phi(t_0, t)$

Definition 3. Reachable Set [12] The reachable set at time t of the system described by Definition 2 is given by:

$$\mathcal{R}(t, t_0, x_0) := \{y \in \mathbb{R}^n | \, \exists u(.) \in U \wedge \exists x(.) \text{ solution of (3.1) with y=x(t) } \} \tag{3.3}$$

$$\mathcal{R}(t, t_0, X_0) := \cup_{x0\in X_0}\mathcal{R}(t, t_0, x_0) \tag{3.4}$$

These sets are, according to [12] [page 161], also the reachable sets of systems described by the following differential inclusion.

Definition 4. Linear Differential Inclusion [12] Let Γ, $A(.)$, $B(.)$, X_0 and U be as in Definition 2. Then $x(.) : \Gamma \rightarrow \mathbb{R}^n$ defined in equation 3.2 is also the solution of the linear differential inclusion

$$\dot{x}(t) \in A(t)x(t) + B(t)U \tag{3.5}$$

$$x(t_0) = x_0 \in X_0. \tag{3.6}$$

and $x(.)$ is absolutely continuous for almost all $t \in \Gamma$.

Proposition 1. The reachable set at time t_f of systems described in Definitions 2 or 4 is then given by

$$\mathcal{R}(t_f, t_0, X_0, U) = \Phi(t_f, t_0)X_0 \oplus \int_{t_0}^{t_f} \Phi(t, s)B(t)Uds \tag{3.7}$$

where $\oplus$ is the Minkowski set addition.

Remark 2. The reachable set $\mathcal{R}(t_f, t_0, X_0, U)$ is nonempty, convex and compact in $\mathbb{R}^n$ if

- X_0 is nonempty, convex and compact in $\mathbb{R}^n$,
- U is nonempty and compact in $\mathbb{R}^m$,
- $A(.)$,$B(.)$ are Lebesgue-integrable.

If the values of the fundamental solution are known, an iterative set-valued approximation of the reachable set at time t_f can be deduced by using its semi-group property.

Remark 3. Semi-group Property of Reachable Sets We choose for $\Gamma = [t_0, t_f]$ the partition t_i, $i \in \{i = 0, \ldots, N\}$ with $N \in \mathbb{R}^+$ and a constant time-step $r = \frac{t_f}{N}$. Let t_{i+1}, t_i such that $t_0 \leqslant t_i \leqslant t_{i+1} \leqslant t_f$

$$\mathcal{R}(t_{i+1}, t_0, X_0, U) = \mathcal{R}(t_{i+1}, t_i, \mathcal{R}(t_{i+1}, t_i, X_0, U), U) \tag{3.8}$$

We adopt the abbreviation $\mathcal{R}_i = \mathcal{R}(t_i, t_0, X_0, U)$ for the reachable set obtained at time t_i. Furthermore, we use the semi-group property and (3.7) to derive the iterative scheme for the computation of reachable sets for a linear system defined by 2 or 4. This leads to the basic reachability algorithm given by algorithm 3.1.

Algorithm 3.1 Iterative scheme for computing reachable sets of linear systems

$\mathcal{R}_0 = X_0$
for $i = 0 : N$ **do**
  $\mathcal{R}_{i+1} = \Phi(t_{i+1}, t_i)\mathcal{R}_i \oplus \int_{t_i}^{t_{i+1}} \Phi(t_{i+1}, s)B(t)Uds$
end for
$\mathcal{R}_N = \mathcal{R}(t_f, t_0, X_0, U)$

Remark 4. Fixpoint The algorithm 3.1 reaches a fixpoint if from a certain iteration i_F successive computed sets verify $\mathcal{R}_{i+1} = \mathcal{R}_i$ for all $i \geqslant i_F$.

The iterative scheme described by algorithm 3.1 is practically not feasible. In fact, the implementation of this algorithm requires first the computation of the fundamental solution in each iteration. However, that is only feasible for limited class of systems, such as LTI-systems. Otherwise, special methods for the approximation of the fundamental solution as well as numerical approximation techniques for the integral counterpart in (3.8), like the set-valued quadrature technique [11, 12], could be applied. Second, the reachable sets have to take a geometric form to allow their numeric manipulation.

3.5. Reachable Set of LTI-Systems

In this section, we consider Linear Time-Invariant LTI-systems which are described by the following differential equation

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{3.9}$$

where $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$ are constant matrices. These systems are effectively closed-loop controlled systems. They have consequently to fulfill the following necessary conditions:

1. The matrix A is stable, i.e., all eigenvalues of A possess negative real parts.
2. The system represented by (3.9) is completely controllable for any initial time t_0.
3. The input $u(t)$ is measurable and for all $t \in \mathbb{R}_+$, $u(t) \in U \subset \mathbb{R}^m$.
4. The set U is compact and convex.

The input $u(t)$ is consequently considered here as an uncertain input. For LTI-systems, the fundamental solution takes the following particular form.

$$\Phi(t, t_0) = e^{(t-t_0)A} \text{ for } t \geqslant t_0. \tag{3.10}$$

For the initial time, we can assume $t_0 = 0$ without loss of generality. This results in the following equation:

$$x(t) = e^{At}x_0 + \int_0^t e^{(t-s)A}Bu(s)ds. \tag{3.11}$$

The above equation gives rise to the following explicit form of the reachable set function.

$$\begin{array}{rl} R(.,.) : \mathbb{R}_+ \times \mathbb{R}^n & \to \mathbb{R}^n \\ (t, X) & \mapsto e^{tA}X \oplus \int_0^t e^{(t-s)A}BUds. \end{array} \tag{3.12}$$

In this equation the matrix exponential and the matrix exponential integral are set-valued functions.
We adopt, from now on, the same notations as in [71] for the reachable sets, i.e.

$$\begin{array}{ll} \mathcal{R}_t(X) & = R(t, X) \\ & = e^{tA}X \oplus \int_0^t e^{(t-s)A}BUds \\ & = e^{tA}X \oplus \mathcal{R}_t(\{0_{\mathbb{R}^n}\}), \\ \mathcal{R}_{[t_1,t_2]}(X) & = \cup_{t\in[t_1,t_2]}\mathcal{R}_t(X). \end{array} \tag{3.13}$$

We appeal, beside the semi-group property, the following property of the reachable set:

$$\mathcal{R}_{[t_0,t_1]}\left(\mathcal{R}_{\left[t_0',t_1'\right]}(X)\right) = \mathcal{R}_{\left[t_0+t_0',t_1+t_1'\right]}(X) \tag{3.14}$$

for $t_0, t_1, t_0^{'}, t_1^{'} \in [0, t_f]$, to derive the following recursion:

$$\begin{aligned} \mathcal{R}_{[t_k,t_{k+1}]}(X) &= \mathcal{R}_r\left(\mathcal{R}_{[t_{k-1},t_k]}(X)\right) \\ &= e^{rA}\mathcal{R}_{[t_{k-1},t_k]}(X) \oplus \mathcal{R}_r(\{0_{\mathbb{R}^n}\}) \end{aligned} \tag{3.15}$$

A concrete implementation of this recurrence imposes the choice of a geometric representation to compute iteratively successive reachable sets. An exact computation is quite impossible. This is first due to the dynamics of the system which continuous evolution from an initial set and particularly in the presence of uncertainty or disturbances can be enclosed only with approximation techniques. Second, the computational complexity imposes the use of finite numerical representation of the geometric maps. Furthermore, these sets are in general not enclosed under some operations involved in the computation. As consequence, over-approximations are for this purpose required to retrieve the original form.

A wide variety of geometric maps have been intensively investigated in the context of reachability analysis of hybrid systems. Each geometric form such as hyperrectangles [25, 92, 104], polytopes [60, 64], polyhedra [41, 44, 52] and ellipsoids [24, 63] have been shown to have strengths and weaknesses. The main issues are that, on one hand, the complexity of computation increases with the complexity of the representation. On the other hand, however, the more complexer the geometry is, the more tighter the approximation of the reachable set becomes. The challenge is then to find a trade-off between the complexity of computation and the accuracy of the approximation. Zonotopes [4, 47] and support functions [12, 72] are later proposed in the literature as solutions for this challenging problem.

We note Ω_k and $\mathcal{V}$ the computed approximate of respectively the exact reachable set $\mathcal{R}_{[t_k,t_{k+1}]}(X_0)$ and the left hand side of (3.15) after the choice of a geometric set.

$$\begin{aligned} \mathcal{R}_{[t_k,t_{k+1}]}(X_0) &\subseteq \Omega_k, \\ \mathcal{R}_r(\{0_{\mathbb{R}^n}\}) &\subseteq \mathcal{V}_r \end{aligned} \tag{3.16}$$

That leads consequently to the following implementable recursion [71]:

$$\Omega_k = e^{rA}\Omega_{k-1} \oplus \mathcal{V}_r \tag{3.17}$$

If we choose geometric shapes closed under Minkowski sum, it is evident that the approximation Ω_k takes directly, owing to (3.17), the same chosen shape.

To implement recursion (3.17), we have first to determine the stepwise evolution of the approximation of the input contribution and second to compute an approximation of the initial set Ω_0.

3.6. Computing an Over-approximation of the Input Contribution

In this section, we describe different methods for over-approximating the reachable set due the input contribution. This over-approximation is highly dependent on

the nature of the input signals and, consequently, the specific assumptions made about them. The input are initially supposed to be uncertain but confined in a convex and compact set U. This guarantees the compactness and convexity of the over-approximation.

3.6.1. Norm-bounded Uncertain Input

In many applications, the uncertain input is assumed to be bounded according to a predefined norm. Practically, this limitation can be justified, because submitting systems to unbounded uncertainties in the input can certainly destabilize them. One of the simplest ways to do this is to determine a constant μ for a given norm $\|.\|$ such that $\mu = sup_{u \in U} \|Bu\|$. By adopting this assumption for the uncertain input, [47] proposed the following over-approximation for the input contribution.

Lemma 1. [47] Let $\|.\|$ be a chosen norm and $\mu = sup_{u \in U} \|Bu\|$. The input contribution verifies then

$$\mathcal{R}_r(\{0_{\mathbb{R}^n}\}) \subseteq \mathcal{B}(\beta_r) \tag{3.18}$$

where $\beta_r = \mu \frac{e^{r\|A\|}-1}{\|A\|}$ and $\mathcal{B}(\beta_r)$ is defined as a ball of radius β_r according to the norm $\|.\|$. Therefore, we get

$$\mathcal{V}_r = \mathcal{B}(\beta_r). \tag{3.19}$$

The proof can be found in Appendix B.

3.6.2. Bounded Uncertain Input

It is quite evident that the afore-stated assumption leads to rapid growth of the over-approximation of the input contribution. This consequently results in a conservative approximation of the reachable set [20]. In fact, under this assumption, the reachable set has to be bloated in each time step with the ball of radius equal to the largest input norm, although generally individual inputs have their norm very far below the maximum value. In [73], an over-approximation based on the maximum radius $R_V = max_{v \in V} \|v\|$ of the set $V = BU$ was proposed.

Lemma 2. The input contribution can be over-approximated by

$$\mathcal{V}_r = rBU \oplus \beta_r \mathcal{B} \tag{3.20}$$

where $\beta_r = \left(e^{r\|A\|} - 1 - r\|A\|\right) \frac{R_V}{\|A\|}$.

3.6.3. Toward a Tighter Approximation of the Input Contribution

In [71], a new approach for a tighter approximation of the input contribution was suggested. This approach avoids the use of maximum norm but instead involves the input set U in the approximation. Two particular set definitions are at this stage required.

Definition 5. The *Interval Hull* of a set $S \subset \mathbb{R}^n$ is the set

$$\square(S) = [\underline{x}_1, \overline{x}_1] \times \ldots \times [\underline{x}_n, \overline{x}_n] \ \forall i, \quad \begin{array}{l} \underline{x}_i = min\{x_i : x \in S\} \\ \overline{x}_i = max\{x_i : x \in S\} \end{array} \tag{3.21}$$

Definition 6. The *Symmetric Interval Hull* of a set $S \subset \mathbb{R}^n$ is the set

$$\boxdot(S) = \left[-\overline{|x_1|}, \overline{|x_1|}\right] \times \ldots \times \left[-\overline{|x_1|}, \overline{|x_1|}\right] \forall i, \ \overline{|x_1|} = max\{|x_i| : x \in S\} \tag{3.22}$$

Lemma 3. Let $\mathcal{U}(r, U)$ be the set defined as follows

$$\mathcal{U}(r, U) = rBU \oplus \mathcal{E}_U \tag{3.23}$$

with

$$\mathcal{E}_U = \square\left(|A|^{-2}\left(e^{r|A|} - I_n - r|A|\right)\right) \boxdot (ABU). \tag{3.24}$$

The over-approximation of the input contribution therefore verifies the following equality

$$\mathcal{V}_r = \mathcal{U}(r, U). \tag{3.25}$$

where I_n is the identity matrix in $\mathbb{R}^n$ and $|A|$ is the component-wise absolute value operation of the matrix $A \in \mathbb{R}^{n \times n}$.

A more elaborated proof inspired from the one proposed in [71] is given in Appendix B.

3.6.4. Input Constant within a Time Step

In [3] a method based on Riemann integration and Taylor series is suggested to compute an approximation of the integral in equation (3.11). However, this method is valid only if $u(t)$ is interval-wise constant. Independently from this particular property of u, an approximation of the input contribution can be derived from the Riemann sum approximation of the integral

$$\begin{array}{c} \int_0^r e^{A(r-s)}Bu(s)ds = \lim\limits_{n\to\infty} \sum_{i=1}^n e^{A(r-s_i^*)}Bu(s_i^*)\Delta s \\ s_i^* = (\Delta s)i \qquad and \qquad (\Delta s) = \frac{r}{n} \end{array} \tag{3.26}$$

by first dropping the limit in (3.26) then choosing a large n and finally computing a supremum of $u(t)$ in each (Δs). Otherwise, it is always possible to choose a small discretization step r within which the piecewise assumption is practically hold. Furthermore, this assumption is practically justified because the input u to a continuous-time system is generally discrete and consequently constant between samples. Under this assumption, the input contribution is over-approximated with the following set:

$$\mathcal{V}_r = \mathbb{B}U \tag{3.27}$$

where $\mathbb{B} = \int_0^r e^{A(r-s)}Bds$.

3.7. Computing an Over-approximation Ω_0 of $\mathcal{R}_{[0,r]}(X_0)$

We are now concerned with the computation of an over-approximation Ω_0 of the set

$$\begin{aligned}\mathcal{R}_{[0,r]}(X_0) &= \cup_{t\in[0,r]}\left(e^{tA}X_0 \oplus \mathcal{V}_t\right)\\ &= \cup_{t\in[0,r]}\left(e^{tA}X_0\right) \oplus \cup_{t\in[0,r]}\left(\mathcal{V}_t\right)\\ &= \mathcal{R}^*_{[0,r]}(X_0) \oplus \mathcal{V} \subseteq \Omega_0\end{aligned} \tag{3.28}$$

with $\mathcal{R}_t(\{0_{\mathbb{R}^n}\}) \subseteq \mathcal{V}_t$, $\cup_{t\in[0,r]}(\mathcal{V}_t) \subseteq \mathcal{V}$ and $\mathcal{R}^*_{[0,r]}(X_0) = \cup_{t\in[0,r]}\left(e^{tA}X_0\right)$ the reachable set of the autonomous system $\dot{x} = Ax$. Many approaches have been proposed during the last few years to cope with this problem. The most recent methods aim at further tightening this approximation. In fact, the quality of the initial approximation can have a crucial impact on the approximation of the reachable sets obtained at the end of the computation (see [20]). In the following, we review some of the most practically applied approaches.

3.7.1. Using a Bloating Factor α_r [47]

We begin with the approach proposed in [47, 70]. Under convexity assumptions of considered sets, the set $\mathcal{R}^*_{[0,r]}(X_0)$ is there first approximated with the convex hull $C = CH(X_0 \cup e^{rA}X_0)$ and then bloated with a factor α_r guaranteeing the enclosure of all the states reachable from X_0 within the time interval $[0, r]$ (see Figure 3.2). The methods adopted in [34, 47, 70] to find such an α_r are quite

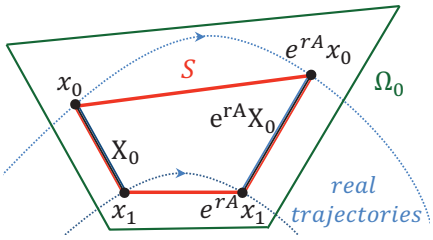


Figure 3.2.: Approximation of the initial set.

similar. To encounter the non trivial computation of C, the set S of segments joining each point $x \in X_0$ and its corresponding image $e^{rA}x$ in $e^{rA}X_0$ is defined.

$$S = \left\{y = x + \tfrac{t}{r}(e^{rA}x - x) \mid x \in X_0 \ \wedge \ t \in [0, r]\right\} \tag{3.29}$$

It is evident that $S \subseteq C$. The minor difference between methods in [34] and in [47, 70] resides, however, in the resulting final form of α_r. Its value is, in general, assumed to be equal to the upper bound on the distance between the set of real trajectories (blue in Figure 3.2) and their approximating set S (red in Figure 3.2). In [47], the Hausdorff distance d_H is used to evaluate α_r.

Lemma 4. The initial set $\mathcal{R}_{[0,r]}(X_0)$ is over-approximated by Ω_0 verifying

$$\Omega_0 = CH(X_0 \cup e^{rA}X_0) \oplus \mathcal{B}(\alpha_r) \oplus \mathcal{V} \tag{3.30}$$

where $\mathcal{B}(\alpha_r)$ is the ball of radius α_r and

$$\alpha_r = \left(e^{r\|A\|} - 1 - r\,\|A\|\right) sup_{x\in X_0}\,\|x\|\,. \tag{3.31}$$

A proof for this lemma is given in Appendix B.

3.7.2. Alternative Approach for Computing a Bloating Factor α_r [73]

An over-approximation based on maximum radii $R_V = max_{u\in V}\,\|u\|$, where $V = BU$ and $R_{X_0} = max_{x\in X_0}\,\|x\|$ of respectively the input set U and the initial set X_0 was proposed in [73]. The proposed approach has resulted in a slightly different bloating factor as that proposed in Section 3.7.1.

$$\alpha_r = \left(e^{r\|A\|} - 1 - \|A\|\,r\right)\left(R_{X_0} + \frac{R_V}{\|A\|}\right) \tag{3.32}$$

The authors defined the unit ball $\mathcal{B}$ associated to the chosen norm $\|.\|$ and proposed the following set Ω_o as an over-approximation for the first element of the sequence of successive reachable sets

$$\Omega_0 = CH(X_0, e^{rA}X_0 \oplus rBU \oplus \alpha_r\mathcal{B}). \tag{3.33}$$

3.7.3. Toward a Tighter Approximation of the Initial Set [71]

The tightness of the approximation of the initial set is crucial for the quality of successive approximations. For this reason, a method has been proposed in [71] aiming at reducing the boating effect of the set $CH(X_0, e^{rA}X_0)$ in each direction and aims as far as possible to restrict it in the vector field directions of the system dynamics.

Lemma 5. [71] For each $\lambda \in [0,1]$ a convex set $\Omega_{0,\lambda}$ is defined as follows

$$\Omega_{0,\lambda} = (1-\lambda)\,X_0 \oplus \lambda e^{rA}X_0 \oplus \lambda\,(1-\lambda)\,\mathcal{E}_{X_0} \oplus \lambda rBU \oplus \lambda^2\mathcal{E}_U \tag{3.34}$$

where

$$\begin{aligned}\mathcal{E}_{X_0} &= \square\left(|A|^{-1}\left(e^{r|A|} - I_n\right) \boxdot \left(A\left(I_n - e^{rA}\right)X_0\right)\right)\\ &\quad \oplus \square\left(|A|^{-2}\left(e^{r|A|} - I_n - r\,|A|\right) \boxdot \left(A^2e^{rA}X_0\right)\right)\\ \mathcal{E}_U &= \square\left(|A|^{-2}\left(e^{r|A|} - I_n - r\,|A|\right) \boxdot (ABU)\right)\end{aligned} \tag{3.35}$$

The initial set can then be approximated by

$$\Omega_0 = CH(\bigcup_{\lambda\in[0,1]} \Omega_{0,\lambda}) \tag{3.36}$$

A complete proof of this lemma can be found in Appendix B.

3.7.4. Forward and Backward Approximations for Computing an Over-approximation of the Initial Set [44, 96]

In [44, 96] a revised approximation of the initial set is proposed. The main difference between the last approximation, and this one resides in only one term. In fact, the approach involves forward and backward interpolations leading to error terms proportional to λ and $(1-\lambda)$ respectively. It results in the replacement of the term $\lambda(1-\lambda)\mathcal{E}_{X_0}$ in equation (3.34) with the intersection of both approximations and is made up as follows.

Lemma 6. [44, 96] For each $\lambda \in [0,1]$ a convex set $\Omega_{0,\lambda}$ is defined

$$\Omega_{0,\lambda} = (1-\lambda) X_0 \oplus \lambda e^{rA} X_0 \oplus \left(\lambda \mathcal{E}^+_{X_0} \cap (1-\lambda)\mathcal{E}^-_{X_0}\right) \oplus \lambda r B U \oplus \lambda^2 \mathcal{E}_U, \tag{3.37}$$

where

$$\begin{aligned} \mathcal{E}^+_{X_0} &= \boxdot\left(|A|^{-2}\left(e^{r|A|} - I_n - r|A|\right) \boxdot \left(A^2 X_0\right)\right) \\ \mathcal{E}^-_{X_0} &= \boxdot\left(|A|^{-2}\left(e^{r|A|} - I_n - r|A|\right) \boxdot \left(A^2 e^{rA} X_0\right)\right) \\ \mathcal{E}_U &= \boxdot\left(|A|^{-2}\left(e^{r|A|} - I_n - r|A|\right) \boxdot (ABU)\right). \end{aligned} \tag{3.38}$$

The initial set is then approximated with

$$\Omega_0 = CH\Big(\bigcup_{\lambda\in[0,1]} \Omega_{0,\lambda}\Big). \tag{3.39}$$

For the proof of this lemma, we refer to the corresponding literature.

A summarizing overview of the above-presented over-approximation methods for the input contribution as well as for the initial set is given by Figure 3.3 It should be noted in regard to this figure that it is possible to combine the $Method_{\Omega_0}(1)$ and $\text{Method}_{\Omega_0}(2)$ for the initial set approximation with all presented methods for the approximation of the input contribution in the left side of the figure. In contrast to the previous two methods $\text{Method}_{\Omega_0}(3)$ and $\text{Method}_{\Omega_0}(4)$ which are from the outset based on the approximation $\text{Method}_{\mathcal{V}_r}(3)$.

Claims have been made that the quality of the initial set and the input approximations has a crucial impact of the tightness of further computed reachable sets. A practical assessment of these different approximations, however, seems to be lacking in the corresponding literature. In Chapter 4, we present a performance comparison of these methods using support functions and polyhedra for approximating the reachable sets. We use some known benchmarks for which the reachable sets resulting from each method are compared in regard to their computation time and the tightness of the approximation.

3.8. Handling Invariants Inside Discrete Modes

According to the definition of hybrid automaton (Section 3.2), each location q_i is characterized by a flow condition $\dot{x} = f_{q_i}(x, ...)$ describing the continuous dynamics

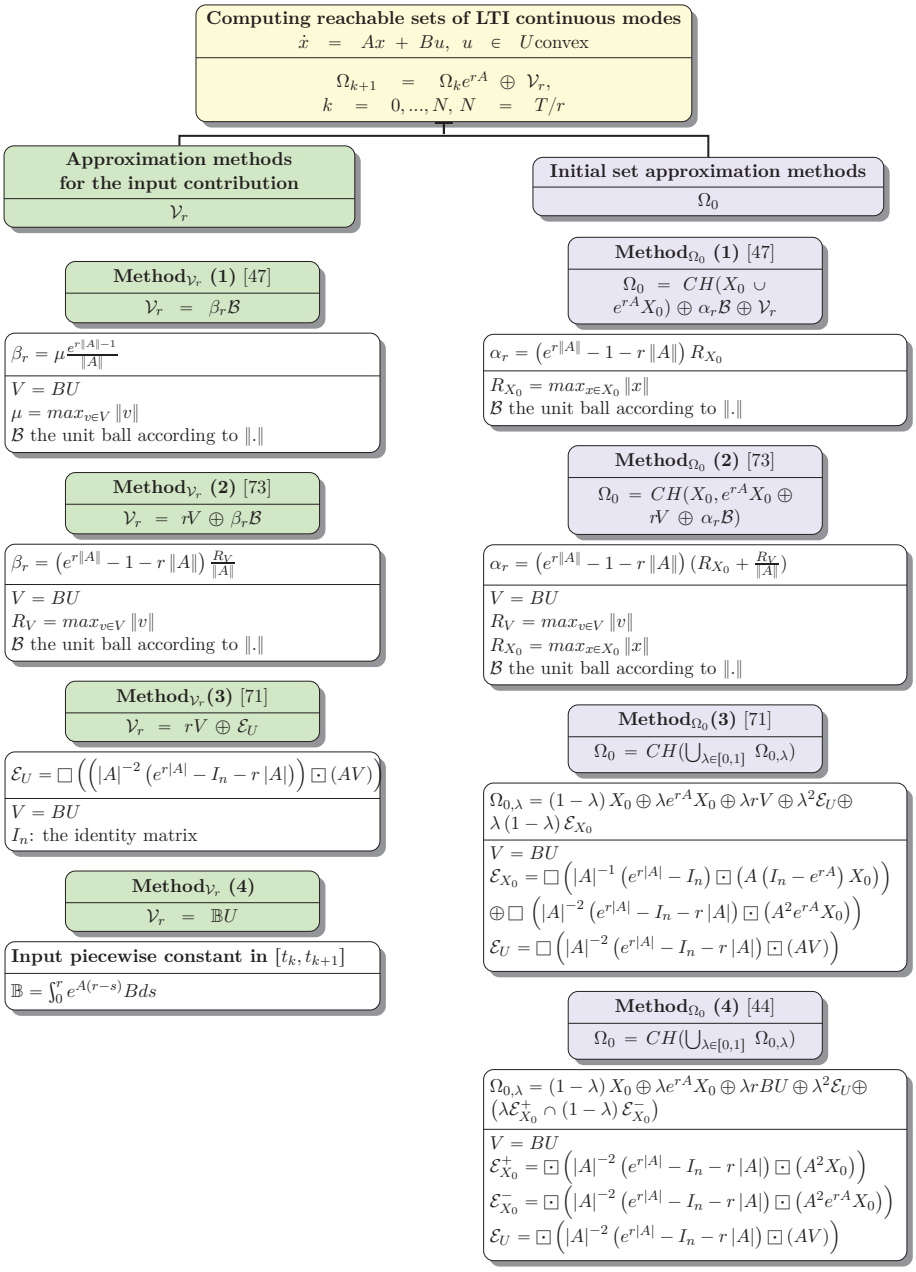


Figure 3.3.: Approximation methods of the initial set and the input contribution.

generally valid in a predefined domain Inv_i subset of the state space of the system. For this, we have to test in each iteration k in the considered location, like by guards, for an intersection between the computed reachable set Ω_k and the domain also called invariant which we abbreviate here for simplicity with $\mathcal{I}$. In case of a non emptiness, two actions may be taken. The first one consists in computing the intersection set $\tilde{\Omega}_k = \Omega_k \cap \mathcal{I}$. The resulting set $\tilde{\Omega}_k$ is then considered in the following computation steps as the new reachable set. The second possibility is to spring to the next location if the guard condition is met. Otherwise the computation will be broken off.
In this section, we describe two different approaches to cope with the problem of invariants within continuous locations. These two different concepts have been already proposed in [71, 96] for handling this problem using support function techniques.

3.8.1. Handling Invariants as Guards

The first alternative for treating invariant is to use the same concept for handling guards. It consists, as shown in Algorithm 3.2 and detailed in the next section, first in checking a collision between the invariant and the reachable set. In a second step the intersection is computed. This intersection is hence taken as the reachable set for further iteration steps.

3.8.2. Recursive Scheme with Invariants [71]

It is obvious that invariant constraints can be treated in the same manner and consequently with the same approaches as guard conditions. In [71], however, a recursive scheme has been suggested to handle them while computing reachable sets. In each iteration step an emptiness check is carried out. If the intersection between the reachable set and the invariant is revealed to be not empty, the latter is then computed. The resulting intersection is therefore the reachable set of the current iteration. The proposed scheme is derived by taking under consideration the following proposition

Proposition 2. Let X, Y, $Z \subseteq \mathcal{R}^n$. Then the following holds:

$$(X \cap Y) \oplus Z \subseteq (X \oplus Z) \cap (Y \oplus Z) \tag{3.40}$$

while computing successive reachable sets with the following recursion

$$\tilde{\Omega}_k = \left(e^{rA}\tilde{\Omega}_{k-1} \oplus \mathcal{V}_r\right) \cap \mathcal{I}. \tag{3.41}$$

This leads to the following scheme for the computation of reachable set in the presence of invariants within continuous modes

$$\tilde{\Omega}_k \subseteq \Omega_k \cap \bigcap_{j=0}^{k-1} \mathcal{I}_j \tag{3.42}$$

Algorithm 3.2 Reachability Algorithm for a Hybrid Automaton in the Presence of Invariant and Guards

Input: r, N, $\Phi = e^{rA}$, Ω_0, $\mathcal{V}_r$, I, r, $\}_1, \ldots, \}_m$;
Output: $\Omega_0, \ldots, \Omega_N$;

```
1:  X = Ω := Ω_0;
2:  V := V_r;
3:  S := {0};
4:  for k = 1 to N do
5:    if Ω ∩ I = ∅ then
6:      break;
7:    end if
8:    for j = 1 to m do
9:      if Ω ∩ I ∩ g_j ≠ ∅ then
10:       collect the indexes of the intersecting sets with the index of the corresponding intersected guard condition;
11:     end if
12:   end for
13:   X = ΦX;
14:   S = S ⊕ V;
15:   V = ΦV;
16:   Ω = X ⊕ V;
17:   Ω_k := Ω;
18: end for
19: return {Ω_0, ..., Ω_N} and indexes of intersecting sets with the corresponding guard index;
```

with

$$\begin{aligned} \Omega_k &= e^{rA}\Omega_{k-1} \oplus \mathcal{V}_r \\ \mathcal{I}_k &= e^{rA}\mathcal{I}_{k-1} \oplus \mathcal{V}_r, \end{aligned} \tag{3.43}$$

where $\tilde{\Omega}_0 = \Omega_0$ and $\mathcal{I}_0 = \mathcal{I}$. This scheme can be easily, as already done in [71], proved using induction. Although it seems to bloat once more the over-

Algorithm 3.3 Reachability Algorithm for Handling Invariant Intersection while Computing Reachable Sets

Input: r, N, $\Phi = e^{rA}$, Ω_0, $\mathcal{V}_r$, I, r, $\}_1, \ldots, \}_m$;
Output: $\Omega_0, \ldots, \Omega_N$;
1: $X = \Omega := \Omega_0$;
2: $Y := I$;
3: $V := \mathcal{V}_r$;
4: $S := \{0\}$;
5: $K := \mathbb{R}^n$;
6: $\tilde{\Omega} = \Omega_0 \cap I$;
7: **for** $k = 1$ **to** N **do**
8: **if** $\tilde{\Omega} = \varnothing$ **then**
9: break;
10: **end if**
11: **for** $j = 1$ **to** m **do**
12: **if** $\tilde{\Omega} \cap g_j \neq \varnothing$ **then**
13: collect the indexes of the intersecting sets with the index of the corresponding intersected guard condition;
14: **end if**
15: **end for**
16: $X = \Phi X$;
17: $Y = \Phi Y$;
18: $S = S \oplus V$;
19: $V = \Phi V$;
20: $\Omega = X \oplus S$;
21: $K = K \cap I$;
22: $I = Y \oplus S$;
23: $\tilde{\Omega} = \Omega \cap K$;
24: $\tilde{\Omega}_k := \tilde{\Omega}$;
25: **end for**
26: **return** $\left\{\tilde{\Omega}_0, \ldots, \tilde{\Omega}_N\right\}$ and indexes of intersecting sets with the corresponding guard index;

approximation of the reachable set because of the inclusion in (3.42) this scheme has the advantage of first involving Ω_k instead of $\tilde{\Omega}_k$. In this way, if an intersection takes place because of the over-approximation of the reachable set, this error will thereby not be propagated along the computation. It second allows us

to continue taking advantage of the superposition principle to compute the exact reachable set. The third benefit of this scheme resides in shifting the intersection operation towards the end of the recursion so that it essentially affects the invariant set evolution through the computation. The corresponding algorithm, however, calls for two set-intersection operations as shown in lines 21 and 23 of Algorithm 3.3. The intersection operation efficiency and accuracy of sets depends largely on the choice of the reachable set representation. For this reason, this algorithm may suffer from inefficiency and large bloating effect in comparison to Algorithm 3.2 if the representation set is not closed under intersection and the over-approximation of the intersection is rough and computationally complex. We will hence adopt this algorithm for support functions but not for zonotopes in the next sections.

3.9. Handling Transitions

Hybrid automaton are particularly characterized by the strong interaction between continuous dynamics and discrete events. As mentioned, the latter are modeled as transitions between modes describing the continuous behavior. After handling the computation of reachable sets for continuous dynamics, this section is dedicated to available methods for treating transitions. In general, a transition is described by a guard condition $\mathcal{G}$ and optionally a reset map $\mathcal{R}e$. A transition is triggered as soon as the guard is fulfilled. The reset map is thereafter applied before entering the target mode. The most treated types of guards are hyperplanes, halfspaces or recently also polyhedra [96].

To handle guards, a collision detection test of the reachable set with the guard

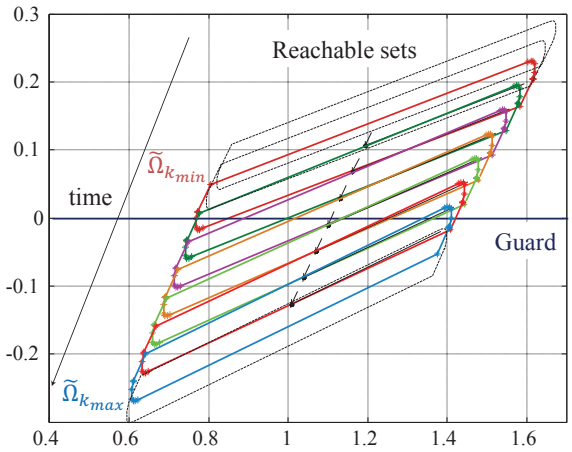


Figure 3.4.: Detecting successive reachable sets $\tilde{\Omega}_{k_{min}}, \ldots, \tilde{\Omega}_{k_{max}}$ intersecting the guard $y = 0$ for the two tanks benchmark.

is first introduced to each iteration in the computation schema (3.42). The test

may be positive for several successive iteration steps $k \in [k_{min}, k_{max}]$. Figure 3.4 shows, for example, seven successive reachable sets intersecting the guard given as a hyperplane $x = 0$ for the two tanks benchmark. Those are marked with continuous different colored lines to differentiate them from the dashed sets irrelevant for the intersection. Once the bundle of intersecting sets is known, their intersections with the guard $\mathcal{Y}$ is in a next step computed. Two different concepts may be considered for computing the entire intersection:

- The pre-clustering method consists of first computing the convex hull of the intersecting bundle and then the intersection of this last with the guard according to the following formulation:

$$\mathcal{Y} = CH\left(\left(\bigcup_{k=k_{min}}^{k_{max}} \tilde{\Omega}_k\right) \cap \mathcal{G}\right) \tag{3.44}$$

- or the post-clustering method which computes as follows each intersection separately and then build their convex hull:

$$\mathcal{Y} = CH\left(\bigcup_{k=k_{min}}^{k_{max}} \left(\tilde{\Omega}_k \cap \mathcal{G}\right)\right). \tag{3.45}$$

We note that the set intersection operation is required for pre-clustering, as well as, for post-clustering. This operation is, in general, a hard operation for the most set presentations. Moreover, the closure property of sets may be lost under this particular operation. In this case, an over-approximation with the chosen set presentation is required. This may lead consequently to the propagation of the over-approximation error through the computation each time this operation is called. From this point of view the pre-clustering scheme is clearly advantageous over the post-clustering because it uses this operation only once. However, experimental results have revealed that for some particular cases, the set resulting from post-clustering (3.45) may be tighter. The complexity of computation and the choice of the clustering method is set dependent. In the next sections, this problem will be investigated using support functions (Section 4) and zonotopes (Section 5) as set representations.

Once the intersection is computed, the initial set for the transition target mode will be the image of $\mathcal{Y}$ under the reset map $\mathcal{R}e$:

$$X_0^T = \mathcal{R}e\left(\mathcal{Y}\right). \tag{3.46}$$

An example of guard handling with reset within a transition for the bouncing ball example is illustrated in Figure 3.5. The sets in yellow represent the intersections of the flowpipe with the guard. The intersections in the first and second transition involve only one intersecting set. We note the effect of the reset map which shifts the resulting intersection in this case.

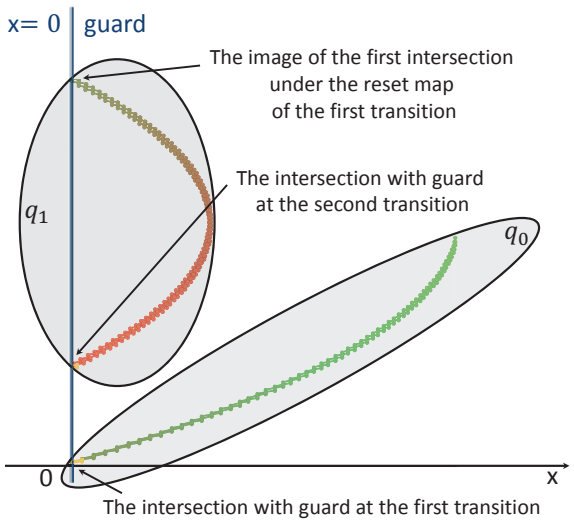


Figure 3.5.: Illustration of the intersection with guard and its image under the reset map for the bouncing ball example.

3.10. Conclusion

This chapter investigates methods for computing reachable set for hybrid linear systems. Different techniques for approximating the input contribution as well as the initial sets and approaches for handling invariants are explored. We also surveyed approaches for handling guards and resets. In the next chapters, we apply the above-mentioned methods practically by using two different set representations. We furthermore carry out a practical performance comparison of these methods. Based on this, a first concept and a prototypical implementation of a reachability analysis toolbox for linear hybrid systems based on support functions is described in the next chapter. The main features of this toolbox is to provide these different methods on the same platform. The user may choose amongst them accordingly. Additionally, an implementation using zonotopes is discussed in the following chapter. We discover both common and different implementation techniques for these particular approximation sets. Consequently, we draw some conclusions about the efficiency and accuracy of implementations when using support functions or zonotopes based on experimental results.

4. Support Function Technique for Computing Reachable Sets

4.1. Introduction

Approximation methods of reachable sets are generally based on a preliminary choice of the geometric representation. Once this choice is made, it is difficult, in the vast majority of cases, to change it while computing reachable sets. For some arithmetic set operations, like the intersection and containment operations, however, the closure property may be lost for some geometric shapes. The complexity of those operations also depends heavily on the class of the chosen geometry. To cope with these both problems, methods based on transformation between different geometric shapes are proposed (see [5]). The transformation from a geometric shape into another may in its turn be a source of error propagation because of the over-approximation usually necessary to turn back to the original chosen shape and vice versa.

Support functions provide the best alternative to represent convex sets without making an initial binding choice of geometric shape. They also possess specific properties making them attractive for implementation. Arithmetic set operations are simply transformed in algebraic function operations. But in addition to their advantages they also have some important drawbacks. In fact, some of those operations as well as the evaluation of the support function in different directions for visualization scope require the use of different optimization and linear programming algorithms. Our experience has revealed that choosing the right linear programming algorithm or optimization is practically difficult. The reason is that the performances of these algorithms are tightly dependent on the form of the reachable sets and the dynamical behavior of the system.

To our knowledge, [11, 110] were one of the first works which concretely introduced support functions for approximating reachable sets. Support functions, however, took on a much more significant role in reachability analysis with the appearance of the works [49, 73]. Algorithms for the computation of the support functions of successive reachable sets were proposed for time-discrete as well as for time-continuous systems. The idea behind these algorithms is the deduction of a recursion scheme for the computation of the support functions of successive reachable sets based on results of previous computation steps.

Further works extended the proposed algorithms to consider the invariants of continuous modes and guards in transitions. They suggested for example methods to compute some specific set intersections, like convex sets/hyperplanes [72] or

convex sets/halfspaces intersections [43, 96], using support functions combined with optimization techniques. SpaceEx is one of most recent open source platform for the verification of hybrid system based on support functions.
In this chapter, we overview previously existing methods and present further details of our solutions for their practical implementations. We suggest alternatives to these methods and compare their performances. We begin first with the definition and the geometric interpretation of support functions. We then present some of their properties which we judged to be relevant for the derivation of practically efficient algorithms for the computation of reachable sets. Furthermore, we review the most recent techniques in the approximation of the input contribution as well as those in the approximation of the initial set for an LTI-system with an uncertain input confined in a bounded compact set. We propose a variety of methods to deal with invariants and guards. We therefore present a method for treating hybrid systems with spontaneous transitions. We conclude with the description, guiding instructions and some testing results of our MATLAB implementation.

4.2. Definition and Properties of Support Functions/Support Vectors

Support function can be considered as an alternative representation to geometric sets. This representation has the advantage to transform some geometric manipulations of n-dimensional sets, which are hard to imagine and visualize, in ordinary algebraic transformations.

Definition 7. : Support Functions [45]
Let $S \subset \mathbb{R}^n$ and $l \in \mathbb{R}^n$. The function

$$\begin{aligned} \rho_S : \quad & \mathbb{R}^n \longrightarrow \mathbb{R} \cup \{\pm\infty\} \\ & l \longmapsto \rho_S(l) := sup_{x \in S} \langle l, x \rangle \end{aligned} \tag{4.1}$$

is called the support function of the set S.

$\langle l, x \rangle$ is the dot product of vectors ordinarily calculated as $l^T.x$ or $x^T.l$. If the support function of an non empty set $S \subset \mathbb{R}^n$ is known, we can compute the convex hull of S, which we abbreviate here for the convenience of notation to ch(S), as follows:

$$ch(S) = \{x \in \mathbb{R}^n | \, \forall l \in \mathbb{R}^n : \langle l, x \rangle \leqslant \rho_S(l)\} \tag{4.2}$$

Consequently, we deduce that $\rho_S = \rho_{ch(S)}$.
Support functions also possess some interesting properties which simplify their manipulation.

Proposition 3. : Properties of Support Functions
Let S, S_1, $S_2 \subset \mathbb{R}^n$ be nonempty sets, l, l_1, $l_2 \in \mathbb{R}^n$, $A \in \mathbb{R}^{n \times m}$ and $\lambda \geqslant 0$, the following properties of the support function hold [45].

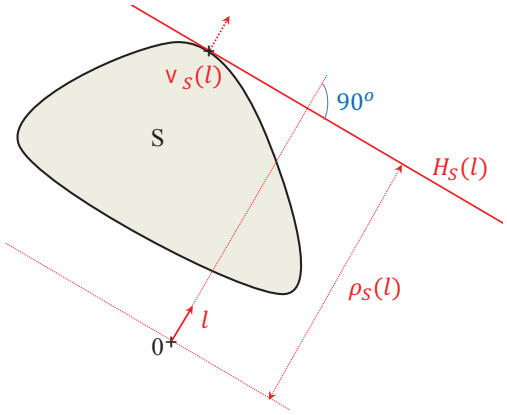


Figure 4.1.: Illustration of the notions of support function, support vectors and supporting hyperplanes of a convex set S.

1. $\rho_{\lambda S}(.) = \lambda \rho_S(.)$
2. $\rho_S(\lambda l) = \lambda \rho_S(l)$
3. $\rho_S(l_1 + l_2) = \rho_S(l_1) + \rho_S(l_2)$
4. $\rho_{AS}(l) = \rho_S\left(A^T l\right)$
5. $\rho_{S_1 \oplus S_2}(.) = \rho_{S_1}(.) + \rho_{S_2}(.)$
6. $\rho_{CH(S_1 \cup S_2)}(.) = max\left(\rho_{S_1}(.), \rho_{S_2}(.)\right)$
7. $\rho_{S_1 \cap S_2}(l) \leqslant min\left(\rho_{S_1}(l), \rho_{S_2}(l)\right)$

Beside these properties, it is also important to know the support function of some particular sets. Those are useful for the computation of the support functions of the initial and the input sets. Some of them are summarized in Table 4.1.

The support functions of hyper-rectangles and symmetric hyper-rectangles are considered separately. These types of convex sets are commonly used for approximating the initial set and the input contribution. They appear there as ordinary or symmetric interval hull of previously known convex sets.

Proposition 4. : Support function of hyper-rectangles
The support function of a hyper-rectangle $I = [a_1, b_1] \times \ldots \times [a_n, b_n]$, $a_i, b_i \in \mathbb{R}$, in a direction $l = (l_1, \ldots, l_n)^T$ is given by

$$\begin{aligned} \rho_I(l) &= \sum_{i=1}^{n} \left[a_i l_i (l_i < 0) + b_i l_i (l_i \geqslant 0)\right] \\ &= \sum_{i=1}^{n} \left|(\tfrac{a_i - b_i}{2}) l_i\right| + \sum_{i=1}^{n} (\tfrac{a_i + b_i}{2}) l_i. \end{aligned} \tag{4.3}$$

Convex set S	$\rho_S(l)$
$\{u\}$ with $u \in \mathbb{R}^n$	$\langle l, u \rangle$
$[a, b]$,$a \leqslant b$, $a, b, l \in \mathbb{R}$	$\begin{cases} a.l & \text{for } l < 0 \\ b.l & \text{for } l \geqslant 0 \end{cases}$
$\{x \in \mathbb{R}^n : \|x\|_1 \leqslant 1\}$	$\|l\|_\infty = max_{i=1,\ldots,n} \|l_i\|$
$\{x \in \mathbb{R}^n : \|x\|_2 \leqslant 1\}$	$\|l\|_2$
$\{x \in \mathbb{R}^n : \|x\|_\infty \leqslant 1\}$	$\|l\|_i = \sum_{i=1}^n \|l_i\|$
Ball: $B_r(m)$ with $m \in \mathbb{R}^n$,$r \geqslant 0$	$r \|l\|_2 + \langle l, m \rangle$
Ellipsoid: $\left\{x \in \mathbb{R}^n : x^T Q^{-1} x \leqslant 1\right\}$, Q positive definite symmetric matrix	$\sqrt{l^T Q l}$
Zonotope: $\{\sum_{i=1}^r \alpha_i g_i : \alpha_i \in [-1, 1]\}$, $g_i \in \mathbb{R}^n$	$\sum_{i=1}^r \|g_i l_i\|$
Polytope: $\{x \in \mathbb{R}^n : Cx \leqslant d\}$, C, d are matrix and vector of compatible dimension	$\max\ l^T x$ $Cx \leqslant d$

Table 4.1.: Support function of some particular sets.

The box operator of a convex set S, $\square S$, is the interval hull over-approximation of the set S. The evaluation of the support function of the set S in the directions of the canonical basis $(e_1, \ldots, e_n)^T$ together with their opposites leads to the determination of the upper and lower limits of the hyper-rectangle $\square S$.

Proposition 5. : The box operator of sets
Let $\square S = [a_1, b_1] \times \ldots \times [a_n, b_n]$, then

$$\begin{aligned} a_i &= -\rho_S(-e_i) \\ b_i &= \rho_S(e_i), \end{aligned} \tag{4.4}$$

where each vector $e_i = (0, \ldots, 0, 1, 0, \ldots 0)^T$ has 1 at the i^{th} position, otherwise the coordinates are equal to 0.

Proposition 6. : Support function of symmetric hyper-rectangles
The support function of a symmetric hyper-rectangle $I = [-h_1, h_1] \times \ldots \times [-h_n, h_n]$, $h_i \in \mathbb{R}^+$, in a direction $l = (l_1, \ldots, l_n)^T$ is calculated as follows

$$\rho_I(l) = \sum_{i=1}^{n} |h_i l_i| = \sum_{i=1}^{n} h_i \, |l_i| \, . \tag{4.5}$$

The symmetric box operator of a convex set S, $\boxdot S$, is the symmetric interval hull over-approximation of the set S. It is computed by means of support functions with the following proposition.

Proposition 7. : The symmetric box operator of sets
Let $\boxdot S = [-h_1, h_1] \times \ldots \times [-h_n, h_n]$, then h_i is obtained as follows

$$h_i = max(\rho_S(e_i), \rho_S(-e_i)). \tag{4.6}$$

It is important to retrieve the convex set representation from its support function representation. However an infinite number of well chosen directions are necessary for an exact representation of a convex set. Usually, a set of directions uniformly arranged on the boundary of a unit ball is chosen.

Proposition 8. : From support function to convex set
Let $B_1(0_{\mathbb{R}^n})$ be the euclidean unit ball and $\partial B_1(0_{\mathbb{R}^n})$ its boundary. If S is a convex, closed nonempty set then

$$\begin{aligned} S &= \{x \in \mathbb{R}^n |\, \forall l \in \partial B_1(0_{\mathbb{R}^n}) : \langle l, x\rangle \leqslant \rho_S(l)\} \\ &= \bigcap_{l \in \partial B_1(0_{\mathbb{R}^n})} \{x \in \mathbb{R}^n |\, \langle l, x\rangle \leqslant \rho_S(l)\} \end{aligned} \tag{4.7}$$

Definition 8. : Supporting hyperplane
The hyperplane given by

$$H(l, \alpha) = \{z \in \mathbb{R}^n |\, \langle l, z\rangle = \alpha\} \tag{4.8}$$

contains the set S in its lower halfspace when $\langle l, x\rangle \leqslant \alpha,\ \forall x \in S$. In this case, $H(l, \alpha)$ is called the supporting hyperplane of S in the direction l which we note for convenience $H_S(l)$. The value of α is the maximum signed distance of the set S from the origin in the direction l which is exactly the definition of the support function (see Figure 4.1).

Proposition 9. : From support function to boundary of a convex set
We can also in the same way compute the boundary ∂S of S:

$$\begin{aligned} \partial S = \{v \in \mathbb{R}^n |\quad & \forall l \in \partial B_1(0_{\mathbb{R}^n}) : \langle l, v\rangle \leqslant \rho_S(l), \\ & \exists l_0 \in \partial B_1(0_{\mathbb{R}^n}) : \langle l_0, v\rangle = \rho_S(l_0)\}. \end{aligned} \tag{4.9}$$

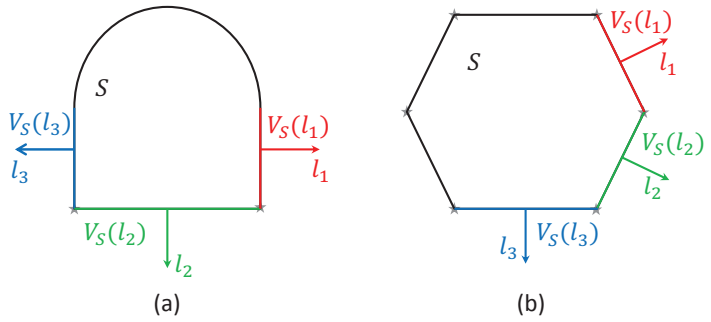


Figure 4.2.: Convex sets for which the support vectors in direction l_1, l_2 and l_3 are respectively the segments of line $V_S(l_1)$, $V_S(l_2)$ and $V_S(l_3)$.

The point v is called the **support vector** of S in direction l_0. Consequently the boundary of S can also be defined as the set of support vectors in all directions $l_i \in \partial B_1(0_{\mathbb{R}^n})$

$$\partial S = \bigcup_{l_i \in \partial B_1(0_{\mathbb{R}^n})} V_S(l_i) \tag{4.10}$$

with $i \in \{1, 2, \ldots\}$ which is generally infinite.
It should be noted that the intersection of a supporting hyperplane in a direction l_i with the convex set S is not necessary a unique point. In general, the result of this intersection is the so called *exposed face* (the colored lines $V_S(l_i)$ in Figure 4.2), which is a set of all support vectors (also named *exposed points*) in a predefined direction. Similar to support functions, support vectors exhibit some interesting properties, which simplify their manipulation during computation.

Proposition 10. : Properties of Support Vectors
Let S, S_1, $S_2 \subset \mathbb{R}^n$ be nonempty sets, $l \in \mathbb{R}^n$, $A \in \mathbb{R}^{n \times m}$ and $\lambda \geqslant 0$, then the following properties of the support function are fulfilled

1. $V_{\lambda S}(.) = \lambda V_S(.)$,
2. $V_{S_1 \oplus S_2}(.) = V_{S_1}(.) + V_{S_2}(.)$,
3. $V_{AS}(l) = A V_S\left(A^T l\right)$.

Particular interesting for our analysis are singleton intersections, which are faces of dimension 0 where $V_S(l_i) = \{v_i\}$. In these cases v_i is called an *extreme point* of S in the direction l_i. The convex hull of these extreme points build up the tightest over-approximation of the convex set S.

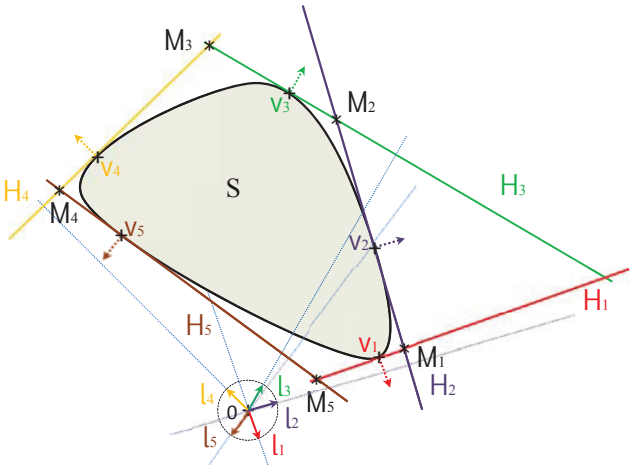


Figure 4.3.: Representation of a convex set via support functions and supporting hyperplanes.

Practically, a finite set or a template $D = \{l_1, \ldots, l_m\}$ of directions uniformly distributed on the unit ball $\partial B_1(0_{\mathbb{R}^n})$ is chosen. For simplicity, the notations of the support hyperplane and support vector in case of a singleton or a set in a direction l_j, $j \in \{1, \ldots, m\}$ are respectively abbreviated in H_j and v_j or $V_S(l_j)$ (Figure 4.3). Generally the directions are arranged in ascending or descending order according to

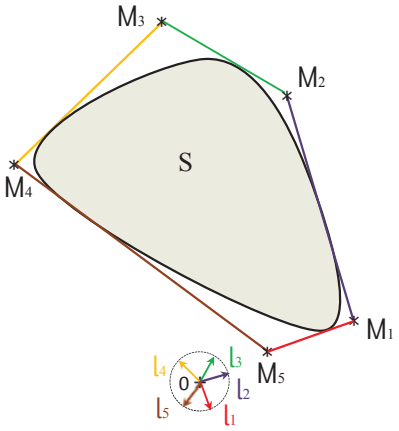


Figure 4.4.: The over-approximating polyhedron of the convex set S for a template of directions $D = \{l_1, l_2, l_3, l_4, l_5\}$ (see also Figure 4.3) .

their angular deviations from a predefined reference. In this way, a set of points can be computed. Each of these points is the result of the intersection of two adjacent supporting hyperplanes (see Figure 4.3). The obtained set defines in its turn a polyhedron which constitutes an over-approximation of the original convex set (see Figure 4.4).

4.3. Computation of the Reachable Set within a Continuous Mode

For the computation of the reachable set in continuous modes described with the differential equation $\dot{x} = Ax + Bu$ where A and B are constant matrices, we recall the recursion scheme given by (3.17) in Section 3.5. In fact, if the approximations for the initial set Ω_0 and the input contribution $\mathcal{V}_r$ are available and parameters like the time step r and the time horizon T are already chosen, the reachable set Ω_k at time $t = kr$ can be theoretically obtained by calling the set equation $\Omega_k = \Phi\Omega_{k-1} \oplus \mathcal{V}_r$ where $\Phi = e^{rA}$. However, it is evident that a practical implementation requires a preliminary choice of a geometric map for representing reachable sets. We aim to consistently take advantage of the support function presentation of convex sets to derive a practical computation scheme of reachable sets. For this purpose, we use the support function ρ in a direction l and its properties to derive the following algebraic equations

$$\begin{aligned} \rho_{\Omega_k}(l) &= \rho_{\Phi\Omega_{k-1}}(l) + \rho_{\mathcal{V}_r}(l) \\ &= \rho_{\Omega_{k-1}}(\Phi^T l) + \rho_{\mathcal{V}_r}(l). \end{aligned} \tag{4.11}$$

It therefore follows

$$\begin{aligned}
\rho_{\Omega_k}(l) &= \rho_{\Omega_{k-1}}(\Phi^T l) + \rho_{\mathcal{V}_r}(l) \\
&= \left[\rho_{\Omega_{k-2}}(\Phi^T\,\Phi^T l) + \rho_{\mathcal{V}_r}\left(\Phi^T l\right)\right] + \rho_{\mathcal{V}_r}(l) \\
&= \rho_{\Omega_{k-2}}(\Phi^{T^2} l) + \rho_{\mathcal{V}_r}(\Phi^T l) + \rho_{\mathcal{V}_r}(l) \\
&= \left[\rho_{\Omega_{k-3}}(\Phi^T\,\Phi^{T^2} l) + \rho_{\mathcal{V}_r}\left(\Phi^{T^2} l\right)\right] + \rho_{\mathcal{V}_r}(\Phi^T l) + \rho_{\mathcal{V}_r}(l) \\
&= \rho_{\Omega_{k-3}}(\Phi^{T^3} l) + \rho_{\mathcal{V}_r}\left(\Phi^{T^2} l\right) + \rho_{\mathcal{V}_r}(\Phi^T l) + \rho_{\mathcal{V}_r}(l) \\
&\;\;\vdots \\
&= \rho_{\Omega_0}(\Phi^{T^k} l) + \textstyle\sum_{j=0}^{k-1} \rho_{\mathcal{V}_r}\left(\Phi^{T^j} l\right).
\end{aligned} \tag{4.12}$$

If we adopt the notations $d_k = \Phi^{T^k} l$ and $s_k = \sum_{j=0}^{k-1} \rho_{\mathcal{V}_r}\left(\Phi^{T^j} l\right)$, the above equations lead to the algorithm (4.1) proposed in [71] for the evaluation of the support functions $\rho_{\Omega_1}(l), \ldots, \rho_{\Omega_N}(l)$ of successive reachable sets $\Omega_1, \ldots, \Omega_N$ in one of the direction $l \in \partial B_1(0_{\mathbb{R}^n})$.

To get the polyhedron approximating the reachable sets, the obtained support func-

Algorithm 4.1 Computation of $\rho_{\Omega_0}(l), \ldots, \rho_{\Omega_N}(l)$

Input: Φ, ρ_{Ω_0}, $\rho_{\mathcal{V}_r}$, l, N
Output: $\rho_0 = \rho_{\Omega_0}(l), \ldots, \rho_N = \rho_{\Omega_N}(l)$
1: $d_0 := l$
2: $s_0 := 0$
3: $\rho_0 = \rho_{\Omega_0}$
4: **for** $k = 1$ **to** N **do**
5: $\quad d_k := \Phi^T d_{k-1}$
6: $\quad s_k := s_{k-1} + \rho_{\mathcal{V}_r}(d_{k-1})$
7: $\quad \rho_k := \rho_0(d_k) + s_k$
8: **end for**
9: **return** $\{\rho_0, \ldots, \rho_N\}$

tions must be evaluated for each direction element of the template $D = \{l_1, \ldots, l_m\}$. The choice of this template has a crucial impact on the tightness and precision of the over-approximation and also on the efficiency of the computation. The problem thereby is to find practical criteria helping to take effective compromise between precision and complexity. For these reasons, many approaches have been suggested to cope with this problem. In [71] (pages 74-75) some improvements of the algorithm 4.1 has been proposed by making a particular choice of the template of directions D. For a given direction l_0 and some indices $p_1 < .. < p_m$, the directions have been chosen in the following way

$$l_j = (\Phi^T)^{p_j} l_0,\ j \in \{1, \ldots, m\}. \tag{4.13}$$

The problem indeed resides in the fact that the above defined directions would progressively tend towards the same direction with increasing indices. Particular care

must consequently be taken for the choice of these indices to ensure the enclosure and coverage of the reachable set. Our experience has shown that this particular choice has led to ill-defined polyhedra in many cases.
Beside the choice of the template, the choice of the approximation method for the initial set and the input contribution can have also a crucial impact of the quality of the approximation of the reachable sets. Therefore, we propose the implementation of some algorithms involving these different approximation methods and combinations between them in this work. A thorough comparison study is also correspondingly carried out.
We begin with the approaches adopted in our implementation for computing an over-approximation of the input contribution $\mathcal{V}_r$.

4.3.1. Approximation of the Input Contribution

In this work, we opt for the implementation of the next three methods to compute an over-approximating set $\mathcal{V}_r$ of the input contribution. The approximation is made not only in each new recursion for the computation of the reachable set of the Algorithm 4.1, but also in the computation of an over-approximation of the initial set Ω_0. These methods offer, as presented in the next sections, increasing tightness in the approximation nonetheless with additional computation effort. Our goal is, on one hand, to offer a multiple method choice and, on the other hand, to allow the user to accordingly configure the analysis.

Approach 1: Using a bloating factor [73]

The details of this approach are given in Section 3.6.2 ($Method_{\mathcal{V}_r(2)}$). The support function of the proposed over-approximation in a direction $l \in \mathbb{R}^n$ is then computed as follows

$$\begin{aligned} \rho_{\mathcal{V}_r}(l) &= r\rho_{BU}(l) + \beta_r \rho_{\mathcal{B}}(l) \\ &= r\rho_U(B^T l) + \beta_r \rho_{\mathcal{B}}(l) \end{aligned} \tag{4.14}$$

where $\beta_r = \left(e^{r\|A\|} - I_n - \|A\| r\right) \frac{R_V}{\|A\|}$, I_n is the identity matrix and $R_V = max_{v \in V} \|v\|$ with $V = BU$. The support function $\rho_{\mathcal{B}}$ of the unit ball according to the chosen norm $\|.\|$ can be found in Table 4.1.

Approach 2: Towards a best over-approximation of the input contribution [71]

This second approach, previously described in Section 3.6.3 ($Method_{\mathcal{V}_r(3)}$), provides a tighter approximation of the input contribution and reduces thereby the cumulative bloating effect in the recursive computation of Algorithm 4.1. The right hand side of equation (4.14) is then replaced by the support function of the set $\mathcal{E}_U = \square\left(|A|^{-2}\left(e^{r|A|} - I_n - r|A|\right)\right) \boxdot (ABU)$, which is computed using Propositions 6 and 5.

$$\rho_{\mathcal{V}_r}(l) = r\rho_U(B^T l) + \rho_{\mathcal{E}_U}(l). \tag{4.15}$$

Approach 3: Piecewise constant input

We consider the same approach adopted for the zonotope implementation in Chapter 5. The input signal is assumed to be constant within a time step r. Therefore the support function of the input contribution in a direction $l \in \mathbb{R}^n$ is computed as follows

$$\rho_{\mathcal{V}_r}(l) = \rho_{\mathbb{B}U}(l) = \rho_U(\mathbb{B}^T l) \tag{4.16}$$

with $\mathbb{B} = \int_0^r e^{(r-s)A} B ds$.

As previously seen in Section 3.7, an input contribution is also implied in the initial set over-approximation.
Next, we explore multiple support function based approaches for computing the initial set over-approximation.

4.3.2. Initial Set Over-approximation Scenarios

Regarding the initial set over-approximation, we choose the following scenarios for our actual implementation.

Scenario 1 (*NoScale*):

We begin with considering one of the first approximation, proposed in [72], for computing the initial set using support functions. This approach was elaborated in Section 3.7.2 ($Method_{\Omega_0(2)}$). The direct application of properties 4, 5, 6 of Proposition 3 in (3.33) allows the deduction of the following support function

$$\begin{aligned} \rho_{\Omega_0}(l) &= max(\rho_{X_0}, (\rho_{e^{rA}X_0}(l) + r\rho_{BU}(l) + \alpha_r \rho_{\mathcal{B}}(l))) \\ &= max(\rho_{X_0}, (\rho_{X_0}((e^{rA})^T l) + r\rho_U(B^T l) + \alpha_r \rho_{\mathcal{B}}(l))) \end{aligned} \tag{4.17}$$

where $l \in \mathbb{R}^n$, $\alpha_r = \left(e^{r\|A\|} - 1 - \|A\| r\right)\left(R_{X_0} + \frac{R_V}{\|A\|}\right)$ (3.32), $R_{X_0} = max_{x \in X_0} \|x\|$ and $R_V = max_{v \in V} \|v\|$ with $V = BU$.

Scenario 2 (*PreciseOmega0*):

The method proposed in [71] aims to further tighten the approximation of the initial set in comparison with previous methods. This approach was previously elaborated in Section 3.7.3 ($Method_{\Omega_0(3)}$). We aim, in this section, to express the recommended over-approximation in term of support functions and then suggest the way to implement it. In fact, according to the equations (3.34), (3.35) and (3.36), the support function of this over-approximation in a direction $l \in \mathbb{R}^n$ is then equal to

$$\rho_{\Omega_0}(l) = max_{\lambda \in [0,1]} \rho_{\Omega_{0,\lambda}}(l), \tag{4.18}$$

where

$$\begin{aligned}\rho_{\Omega_{0,\lambda}}(l) &= (1-\lambda)\rho_{X_0}(l) + \lambda\rho_{(e^{rA}X_0)}(l) + \lambda r\rho_{(BU)}(l) + \lambda(1-\lambda)\rho_{\mathcal{E}_{X_0}}(l) \\ &+\lambda^2\rho_{\mathcal{E}_U}(l) \\ &= (1-\lambda)\rho_{X_0}(l) + \lambda\rho_{X_0}\left((e^{rA})^T l\right) + \lambda r\rho_U\left(B^T l\right) + \lambda(1-\lambda)\rho_{\mathcal{E}_{X_0}}(l) \\ &+\lambda^2\rho_{\mathcal{E}_U}(l).\end{aligned} \tag{4.19}$$

The support functions of the sets

$$\begin{aligned}\mathcal{E}_{X_0} &= \square\left(|A|^{-1}\left(e^{r|A|} - I_n\right) \boxdot \left(A\left(I_n - e^{rA}\right)X_0\right)\right) \\ &\oplus \square\left(|A|^{-2}\left(e^{r|A|} - I_n - r|A|\right) \boxdot \left(A^2 e^{rA} X_0\right)\right) \\ \mathcal{E}_U &= \square\left(|A|^{-2}\left(e^{r|A|} - I_n - r|A|\right) \boxdot (ABU)\right)\end{aligned} \tag{4.20}$$

are computed using Propositions 7 and 5. Finally $\rho_{\Omega_0}(l)$ is calculated by using optimization techniques and available optimization packages like the MATLAB optimization toolbox, the CVX or the MPT toolboxes.

Scenario 3 (*SpaceEx*):

The computation scheme proposed in [44, 96] is fundamentally the same scheme on which the algorithm 4.1 is based. The authors, however, formulated it differently to allow the choice of variable time steps r_j, $j \in \{0, \ldots, N-1\}$.

$$\begin{aligned}\Psi_{k+1} &= \Psi_k \oplus e^{At_k}\mathcal{V}_{r_k} \\ \Omega_k &= e^{At_k}\Omega_{[0,r_k]}(X_0, U) \oplus \Psi_k\end{aligned} \tag{4.21}$$

where $t_{k+1} - t_k = r_k$.
We consider, in this work, a constant time step r. Therefore, the algorithm implementing the recursion (4.21) is practically algorithm 4.1. The difference in other algorithms lies in the approximation methods adopted for the initial set and the input contribution. In [44, 96], the authors suggested a new method for the approximation of the initial set , which they deemed to provide a tighter approximation ($Method_{\Omega_0(4)}$ Section 3.7.4). In term of support functions and using their dual properties with convex sets operations, the proposed approximation is transformed as follows

$$\begin{aligned}\rho_{\Omega_{0,\lambda}}(l) &= (1-\lambda)\rho_{X_0}(l) + \lambda\rho_{e^{rA}X_0}(l) + \lambda r\rho_{BU}(l) + \lambda^2\rho_{\mathcal{E}_U}(l) \\ &+\rho_{(\lambda\mathcal{E}^+_{X_0} \cap (1-\lambda)\mathcal{E}^-_{X_0})}(l) \\ &= (1-\lambda)\rho_{X_0}(l) + \lambda\rho_{X_0}\left((e^{rA})^T l\right) + \lambda r\rho_U\left(B^T l\right) + \lambda^2\rho_{\mathcal{E}_U}(l) \\ &+\rho_{(\lambda\mathcal{E}^+_{X_0} \cap (1-\lambda)\mathcal{E}^-_{X_0})}(l).\end{aligned} \tag{4.22}$$

where

$$\begin{aligned}\mathcal{E}^+_{X_0} &= \boxdot\left(|A|^{-2}\left(e^{r|A|} - I_n - r|A|\right) \boxdot \left(A^2 X_0\right)\right) \\ \mathcal{E}^-_{X_0} &= \boxdot\left(|A|^{-2}\left(e^{r|A|} - I_n - r|A|\right) \boxdot \left(A^2 e^{rA} X_0\right)\right) \\ \mathcal{E}_U &= \boxdot\left(|A|^{-2}\left(e^{r|A|} - I_n - r|A|\right) \boxdot (ABU)\right).\end{aligned} \tag{4.23}$$

If we now consider property 7 of the support function (given in Proposition 3), then for a given direction $l \in \mathbb{R}^n$, the last term is approximated as follows

$$\begin{aligned} \rho_{(\lambda\mathcal{E}^+_{X_0} \cap (1-\lambda)\mathcal{E}^-_{X_0})}(l) \leqslant \quad & \underbrace{min(\rho_{\lambda\mathcal{E}^+_{X_0}}(l), \rho_{(1-\lambda)\mathcal{E}^-_{X_0}}(l))}_{=}. \\ & min(\lambda\rho_{\mathcal{E}^+_{X_0}}(l), (1-\lambda)\rho_{\mathcal{E}^-_{X_0}}(l)) \end{aligned} \tag{4.24}$$

Property 6 of support functions, hence, allows the computation of the support function of the initial set

$$\rho_{\Omega_0}(l) = max_{\lambda\in[0,1]}\rho_{\Omega_{0,\lambda}}(l). \tag{4.25}$$

Focusing now on the set $\lambda\mathcal{E}^+_{X_0} \cap (1-\lambda)\mathcal{E}^-_{X_0}$ (noted for convenience $\mathcal{E}_0$), we see that the computation of its support function involves the evaluation of the support functions of the two symmetric interval hulls $\mathcal{E}^+_{X_0}$ and $\mathcal{E}^-_{X_0}$ given in (4.23).

Proposition 7 facilitates now the computation of the support function of the set $\mathcal{E}_0 = \lambda\mathcal{E}^+_{X_0} \cap (1-\lambda)\mathcal{E}^-_{X_0}$. First, two vectors $h^+ = (h^+_1, \ldots, h^+_n)$ and $h^- = (h^-_1, \ldots, h^-_n)$ for a direction $l \in \mathbb{R}^n$ respectively :

$$\begin{aligned} \rho_{\mathcal{E}^+_{X_0}}(l) = \quad & h^+ |l|, \\ \rho_{\mathcal{E}^-_{X_0}}(l) = \quad & h^- |l| \end{aligned} \tag{4.26}$$

are evaluated. Second, the following expression is derived

$$\rho_{\mathcal{E}_0}(l) = \sum_{i=1}^{n} min\left(\lambda h^+_i, (1-\lambda)h^-_i\right)|l_i|, \; i \in \{1, \ldots, n\}, \tag{4.27}$$

which in turn can be written as a piecewise linear function of λ. This expression can alternatively be directly embedded in equation (4.25). After adding $\rho_{\mathcal{E}_U}$, the support function of the initial set given in (4.25) is subsequently computed by using already existing optimization toolboxes and algorithms.

4.3.3. Impact of the Approximation Method of the Initial Set on the Tightness of the Reachable Set

It would be interesting at this stage to assess the implication of the initial set approximation method on the tightness of the reachable sets as well as on the computation time. During tests and evaluations of our implementation, computations of reachable sets were carried out for the same examples under different choices of the initialization methods. In addition to initial approximation choice, a nonlinear optimization algorithm is required for the second and the third initial set approximation methods. Choices can also be made between different algorithms and optimization toolboxes.

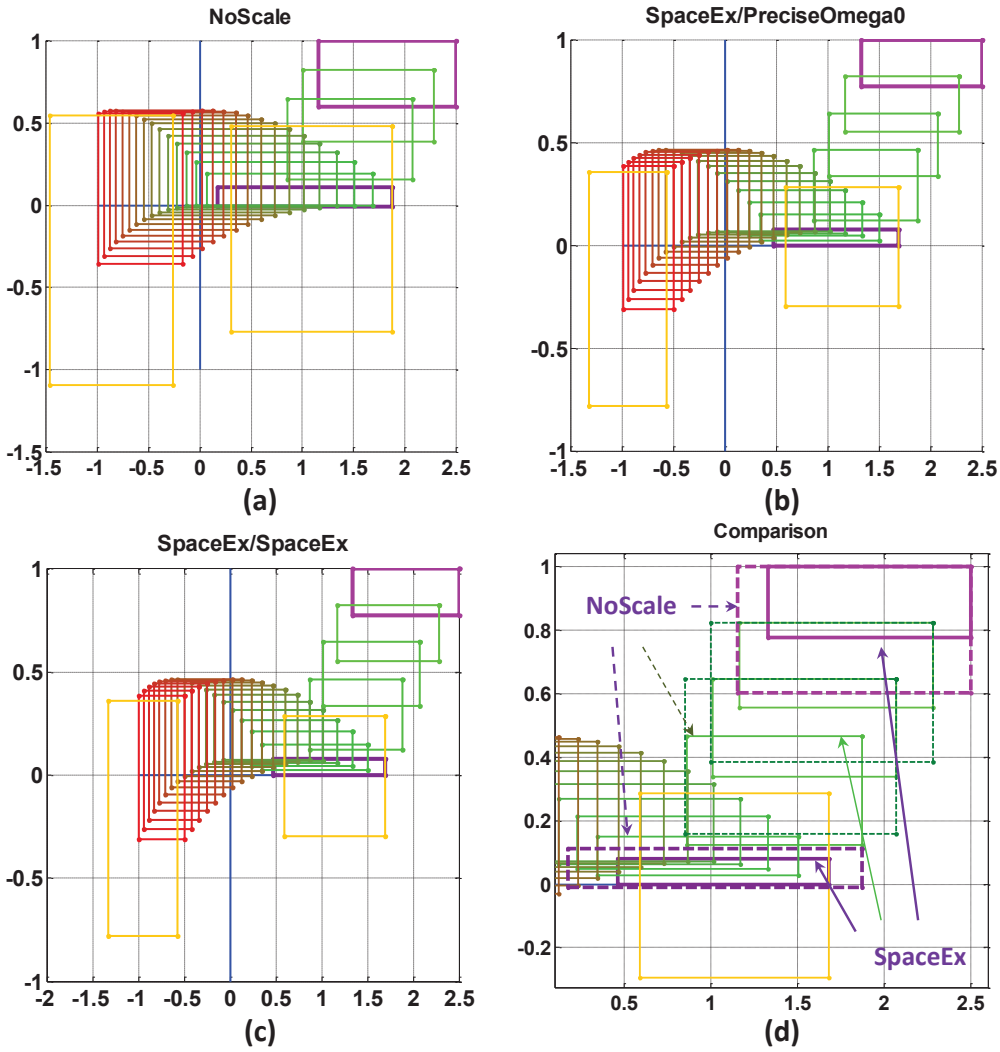


Figure 4.5.: The resulting approximation of the initial set for the two-tank benchmark depending on the method chosen for the initial set computation with a time step $r = 0.05$, a time horizon $T = 3s$, the initial conditions $1.5 \leqslant x1 \leqslant 2.5 \wedge x2 = 1$, an input $u = 0$ and the *box* directions option. (a) The first method (*NoScale*). (b) The second method (*SpaceEx/PreciseOmega0*). (c) The third method (*SpaceEx/SpaceEx*). (d) Comparison of the computed initial reachable sets with different methods.

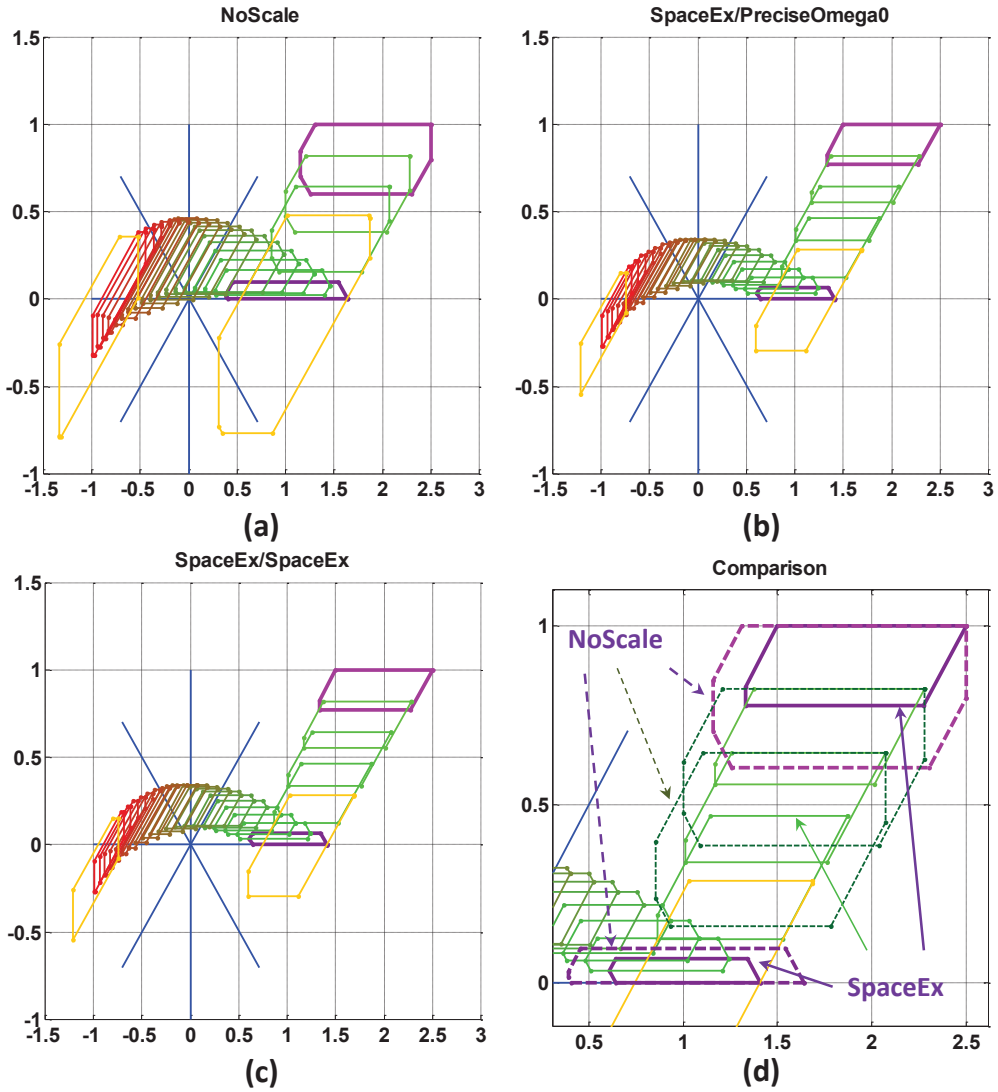


Figure 4.6.: The resulting approximation of the initial set for the two-tank benchmark depending on the method chosen for the initial set computation with a time step $r = 0.05$, a time horizon $T = 3s$, the initial conditions $1.5 \leqslant x1 \leqslant 2.5 \wedge x2 = 1$, an input $u = 0$ and the *octagonal* (*oct*) directions option. (a) The first method (*NoScale*). (b) The second method (*SpaceEx/PreciseOmega0*). (c) The third method (*SpaceEx/SpaceEx*). (d) Comparison of the computed initial reachable sets with different methods.

With regards to the benchmarks and the hybrid systems we tested, we note the increase in tightness of the approximation if the second or the third method are

chosen instead of the first.

For purpose of illustration, the results obtained for the two-tank system of Figure A.5(a) for an input $u = 0$ and using the algorithm *CDD Criss-Cross* of the *MPT*-toolbox are shown in Figure 4.5 for *box* directions and in Figure 4.6 for *octagonal* directions. We note that with the *box* direction choice, the first computed reachable set (purple in Figure 4.5) and the flowpipe resulting at the end of the time horizon $T = 3s$ practically do not show any difference in respect with the tightness (see Figure 4.6). In contrast, the results obtained with the *octagonal* directions, reveal a little improvement between the initial set computed with the first method (NoScale in Figure 4.6(a)) and that obtained with the second method (PreciseOmega0 in Figure 4.6(b)). This can be seen in Figure 4.6(d). We observe, however, that the approximations issued from the second method and the third method (the SpaceEx adopted method in Figure 4.6(c)) coincide in the Figure 4.6(d). The box approximation of the last computed reachable set is equal to $[1.514, 1.528] \times [-0.3101, 0.7338]$ for the second and the third methods and to $[1.514, 1.530] \times [-0.3139, 0.7351]$ for the first one. We note that the resulting difference is not significant enough to conclude about its origin. It may also be due to finite precision errors.

The time required for the computation increases also with the complexity of the approximation for example from $t = 0.111109s$ for the first method, to $t = 0.172788s$ for the second method and $t = 0.226099s$ for the third method.

The approximation of the input contribution is a part of the initial set approximation. Its quality consequently has an impact on the performance of the initial set computation. We consider for this reason an input $u \in [-0.1, 0.1]$ and compute for the same two-tank benchmark and the same setting parameters as above, the flowpipe for different approximation methods and scenarios. We observe in Figure 4.7 the same trend for the *SpaceEx* scenarios with both allowed initialization methods, namely, but note that the *SpaceEx* initial set approximation is faster. The *NoScale* scenario is fast but computes the most bloated over-approximation. The *ConstU* scenario is shown, however, to be fast and to provide a better approximation as the *NoScale* scenario.

Based on our experience with the suite of benchmarks described in Appendix A, general observations can be derived. On a one hand the *SpaceEx* and the *PreciseOmega0* methods guarantee best tightness of the initial set approximation. On the other hand, the *NoScale* and the *ConstU* are the most time efficient methods. It is evident that a trade-off must be made between tightness and efficiency while carrying out a reachability analysis. We decide to let the user decide which method suits the intended system requirements.

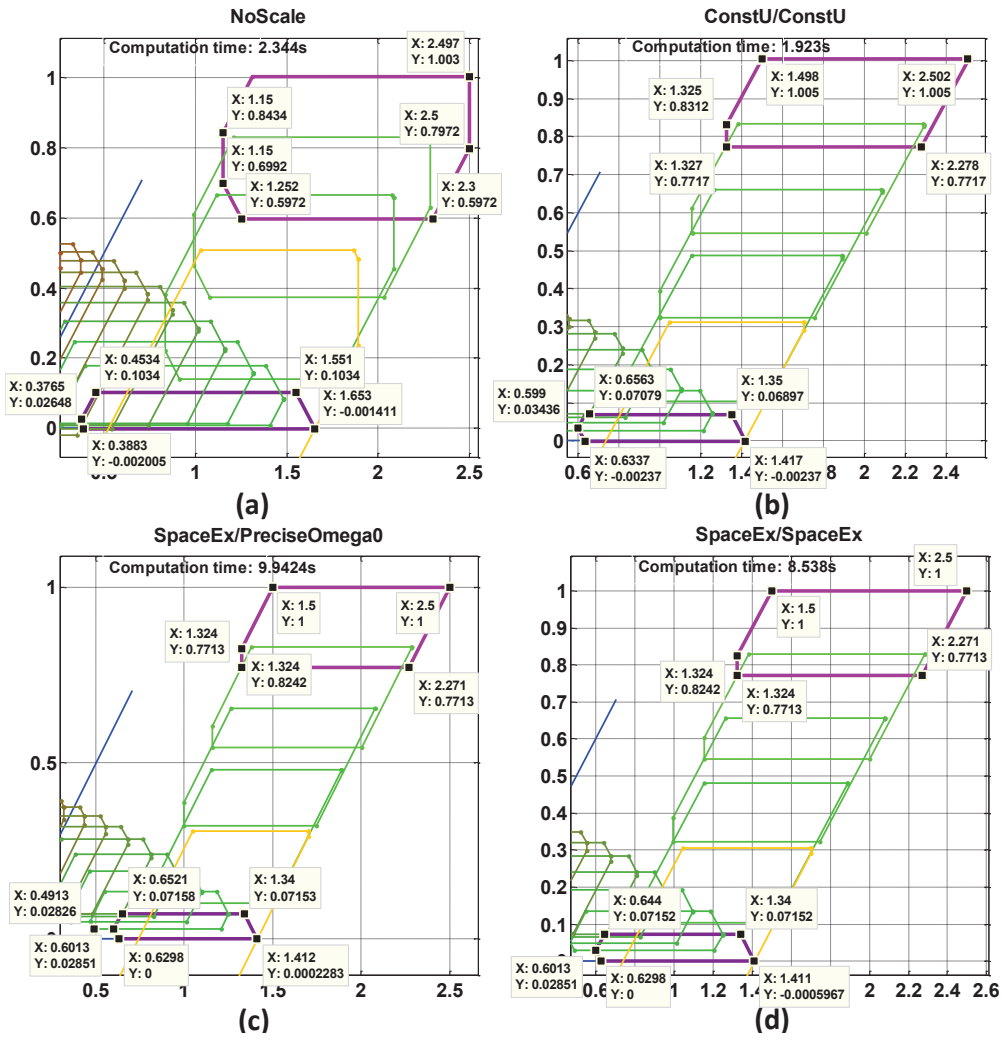


Figure 4.7.: The resulting approximation of the initial set for the two-tank benchmark depending on the method chosen for the initial set computation with a time step $r = 0.05$, a time horizon $T = 3s$, the initial conditions $1.5 \leqslant x1 \leqslant 2.5 \wedge x2 = 1$, an input $-0.1 \leqslant u \leqslant 0.1$ and the *octagonal* directions option. (a) The first method (*NoScale*). (b) The second method (*ConstU/ConstU*). (c) The third method (SpaceEx/-PreciseOmega0). (d) The third method (*SpaceEx/SpaceEx*).

4.4. Collision Detection Between Two Convex Sets

While computing reachable sets of hybrid automata, we look to collision detection between convex sets in two areas: first to check for intersection with invari-

ants within the locations, and second to check for intersection with guards by transitions. Invariant and guard conditions can take the form of a hyperplane $H_p = \left\{x \in \mathbb{R}^n : d^T x = e\right\}$, a halfspace $H_s = \left\{x \in \mathbb{R}^n : d^T x \leqslant e\right\}$ or a polyhedron $P = \{x \in \mathbb{R}^n : D.x \leqslant E\}$ with $D \in \mathbb{R}^{m \times n}$ and $E \in \mathbb{R}^m$. Most of the available collision detection algorithms are based on separation techniques and use support functions in some way to look for the existence of separating hyperplanes between sets (see Section 5.4.1 and Section 5.7.1). For our implementation, we make use of the following lemmas.

Lemma 7. Let $\Omega \subseteq \mathbb{R}^n$ be a convex set and $H_p = \left\{x \in \mathbb{R}^n : d^T x = e\right\}$ be a hyperplane then the following holds

$$(\Omega \cap H_p = \varnothing) \Longleftrightarrow (-\rho_\Omega(-d) \leqslant e \leqslant \rho_\Omega(d)). \tag{4.28}$$

Lemma 8. Let $\Omega \subseteq \mathbb{R}^n$ be a convex set and $H_s = \left\{x \in \mathbb{R}^n : d^T x \leqslant e\right\}$ be a halfspace then the following holds

$$(\Omega \cap H_p = \varnothing) \Longleftrightarrow (-\rho_\Omega(-d) > e). \tag{4.29}$$

The proofs of both lemmas are given in [96].
The extension of the detection collision to polytopes is straight forward. A polytope is basically an intersection of hyperplanes. We treat each of them separately by applying the appropriate lemma.

4.5. Intersection Computation of Two Convex Sets

In this section, we are interested in methods based on support functions for computing an approximating set for the intersection of two convex sets. This intersection is used first in continuous modes to force the system to remain inside the invariants I and then used in the computation of the transition successors by considering guard conditions. In general, there are three possible formulations for computing a convex over-approximation $S \subseteq \mathbb{R}^n$ of the intersection of two convex sets $S_1, S_2 \subseteq \mathbb{R}^n$ characterized by their support functions ρ_{S_1} and ρ_{S_2}. For the directions $l, l_1, l_2 \in \mathbb{R}^n$, the support function ρ_S of the set S can be computed in different ways. The first formulation given in (4.30) is based on the definition of so called *epi-sum* of convex functions (see [97, 98] page 23). This is also called the *infimal convolution* of closed proper convex functions on $\mathbb{R}^n$ [16, 58] and is therewith the origin of the second formulation (4.31).

$$\rho_{S_1 \cap S_2}(l) = \inf_{l=l_1+l_2}\left(\rho_{S_1}(l_1) + \rho_{S_2}(l_2)\right), \tag{4.30}$$

$$\rho_{S_1 \cap S_2}(l) = \inf_{w \in \mathbb{R}^n}\left(\rho_{S_1}(w) + \rho_{S_2}(l-w)\right), \tag{4.31}$$

The operation $''+''$ in (4.30) and (4.30) is obviously extended to the so-called *inf-addition* by adopting the following conventions

$$\begin{array}{l} \alpha + \infty = \infty + \alpha = \infty \text{ for } -\infty < \alpha < \infty \\ \alpha - \infty = \infty + \alpha = -\infty \text{ for } -\infty < \alpha < \infty \end{array} \tag{4.32}$$

The third formulation comes from the seventh property of the support function given in Property 3.

$$\rho_{S_1 \cap S_2}(l) \leqslant \min(\rho_{S_1}(l), \rho_{S_2}(l)) \tag{4.33}$$

We note that (4.33) leads to an over-approximation of the intersection because of the inequality. The tightness of this over-approximation depends crucially on the choice of the template of directions particularly in the case of polyhedral approximation. This point will be treated in details in the part dealing the intersection of the flowpipe with hyperplanes or halfspaces. We also note that the first two formulations require optimization techniques of in general piecewise linear functions. The last one, however, is just a comparison between two values.

4.6. Intersection with Hyperplanes or Halfspaces

In this section, we will deal with the intersection of a convex set S with a hyperplane $H_p = \left\{x \in \mathbb{R}^n : d^T x = e\right\}$ or a hyperspace $H_s = \left\{x \in \mathbb{R}^n : d^T x \leqslant e\right\}$. If we furthermore take into consideration the particularity of their corresponding support functions,

$$\rho_{Hp}(l) = \begin{cases} \lambda e & \text{if } l = \lambda d \text{ and } \lambda \in \mathbb{R} \\ +\infty & \text{otherwise} \end{cases} \tag{4.34}$$

$$\rho_{Hs}(l) = \begin{cases} \lambda e & \text{if } l = \lambda d \text{ and } \lambda \in \mathbb{R}^+ \\ +\infty & \text{otherwise,} \end{cases} \tag{4.35}$$

the support functions of the considered intersections according to formula (4.31) can be reduced to the minimization of the following univariate linear piecewise functions as given in [71, 96]

$$\rho_{S \cap H_p}(l) = \inf_{\lambda \in \mathbb{R}} \left(\rho_S(l - \lambda d) + \lambda e\right), \tag{4.36}$$

$$\rho_{S \cap H_s}(l) = \inf_{\lambda \in \mathbb{R}^+} \left(\rho_S(l - \lambda d) + \lambda e\right). \tag{4.37}$$

To solve this minimization problem, many methods have been already suggested. In [71], the author adapted the dichotomous search technique already proposed for zonotopes (5.4.2) to support functions. The method is, however, restricted only for the intersection of a convex set S with a hyperplane H_p. The classical descending search method was also suggested as an alternative to solve this problem. Furthermore, a performance comparison between both methods was given. The method proposed in [96], however, is an adaptation of the sandwich optimization technique to this particular minimization problem.

4.6.1. From n to 2 Dimension and the Dichotomous Search

This method is suggested for computing an over-approximation of a convex set S defined by its support functions in directions given by the template $D = \{l_1, \ldots, l_m\}$ and a hyperplane $H_p = \left\{x \in \mathbb{R}^n : d^T x = e\right\}$. The idea behind the dichotomous

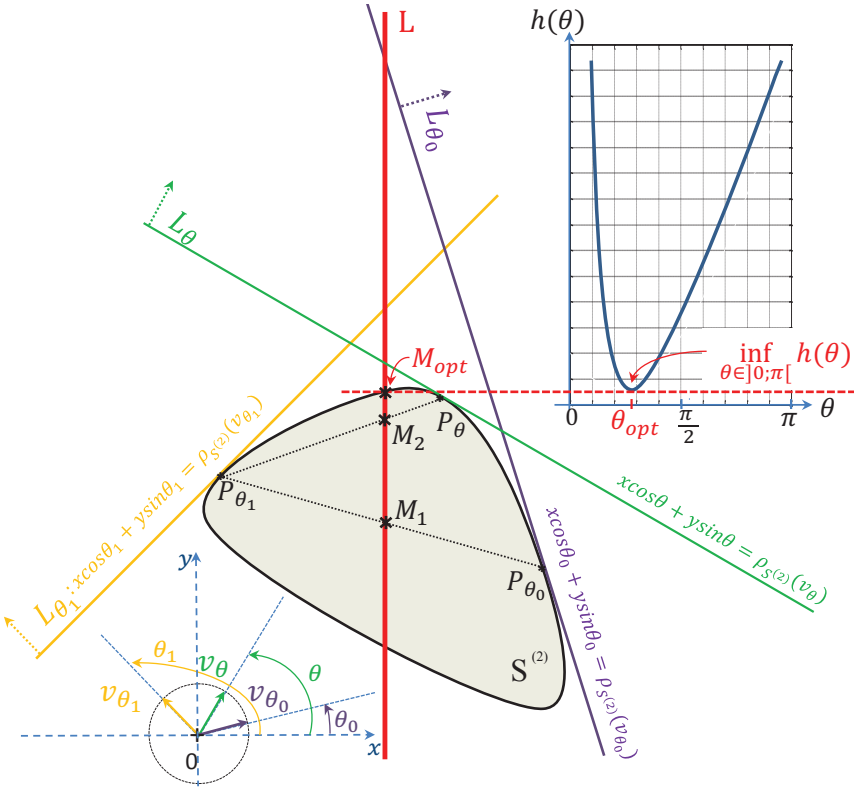


Figure 4.8.: Illustration of the dichotomous search method in relation with the computation of the infinimum of the function $h(\theta)$.

method proposed in [71] is, as already explained in Section 5.4.2, transforming the n dimensional minimization problem in at most $|D|$ two dimensional identical problems ($\frac{|D|}{2}+1$ if L contains both l_j and $-l_j$). Here $|D|$ is the size of the template of directions D. This transformation is a projection of the convex set $S \subseteq \mathbb{R}^n$ into the plane spanned by the vectors d and l_j, $j \in \{1,\ldots,m\}$, which is obtained by a simple matrix multiplication $S_j^{(2} = \mathbb{M}_{d,l_j} S$. The hyperplane is therefore transformed into a line $L = \big\{(x,y) \in \mathbb{R}^2 : x = e\big\}$. In the second step, the coordinates of at maximum two intersection points between the line L and $S_j^{(2)}$ are computed. These correspond, as given in [71], to the minima of the following function:

$$\begin{array}{rcl} h :]0;\pi[& \rightarrow & \mathbb{R} \\ \theta & \mapsto & \frac{\rho_{\mathbb{M}_{d,l_j} S}(v_\theta) - e\, cos\theta}{sin\theta} = \frac{\rho_S(\mathbb{M}^T_{d,l_j} v_\theta) - e\, cos\theta}{sin\theta} \end{array} \tag{4.38}$$

where $v_\theta = (cos\theta, sin\theta)^T$ is the direction defined by $\theta \in]0;\pi[$. In fact, $\rho_S(\mathbb{M}^T_{d,l_j} v_\theta) = \rho_S(cos\theta\, n + sin\theta\, l_j)$. This support function defines in its turn the line $L_\theta : x\, cos\theta + y\, sin\theta = \rho_{\mathbb{M}_{d,l_j} S}(v_\theta)$. The intersection of this line with L leads to points in the form $P = (e, y)$, where $y = \frac{\rho_{\mathbb{M}_{d,l_j} S}(v_\theta) - e\, cos\theta}{sin\theta}$.
Furthermore, the Theorem 8.1 in [71], states that the function defined by (4.38) is monotonic and unimodal and that its infinimum in $]0,\pi[$ corresponds to the support function of the intersection $S^{(2)} \cap L$ in direction $(0,1)^T$:

$$\inf_{\theta \in]0,\pi[} h(\theta) = \rho_{S^{(2)} \cap L}((0,1)^T). \tag{4.39}$$

The algorithm proposed in [71] (page 116) computes, however, the coordinate of the support vector which is also an extreme point of the intersection line $S^{(2)} \cap L$ in direction $(0,1)^T$. The coordinate of the second extreme point is obtained when considering the nD to $2D$ projection into the plane spanned by d and $-l_j$. This is automatically achieved during the computation since our choice of the template of directions D includes different directions with their opposites.

The idea behind this algorithm is to use an approximation approach to estimate the infinimum of the function $h(\theta)$ by using just support function techniques. As illustrated in Figure 4.8, the approach consists of first choosing a search interval $[\theta_0;\theta_1]$ for which the support vectors P_{θ_0} and P_{θ_1} in the directions defined by θ_0 and θ_1 are evaluated respectively. Second, the intersection point $M = (e, y_{min})$ between the line L and the line $\overline{P_{\theta_0} P_{\theta_1}}$ is computed. On the third step, we choose a new angle $\theta \in [\theta_0;\theta_1]$. We opt in our implementation for the simple choice $\theta = \frac{\theta_0+\theta_1}{2}$ with which we compute $P_\theta = (x_\theta, y_\theta)$ and $h(\theta)$. Other choices can also be made. For example, the normal direction to the line $\overline{P_{\theta_0} P_{\theta_1}}$ can be taken as a pivot. This choice was proposed as alternative in Algorithm 5.3. The choice of the new search interval $[\theta_0;\theta]$ or $[\theta;\theta_1]$ for the next iteration depends however on the position of P_θ to the line L. This can be determined with a simple comparison of x_θ with e. The above three steps are repeated iteratively until $y_{max} - y_{min} \leqslant \epsilon$, where ϵ is a chosen precision. The Algorithm 4.2 returns finally the value $y_{max} = min(h(\theta), y_{max})$. To

Algorithm 4.2 dichotomous search using support function

Input: ϵ, h, e, $\mathbb{M}_{d,l_j}$, $\rho_{\mathbb{M}_{d,l_j}S}$
Output: y_{max}

$\theta_0 := 0$
$\theta_1 := \pi$
$P_{\theta_0} := \mathbb{M}_{d,l_j} \,.\rho_S(\mathbb{M}_{d,l_j}^T \, v_{\theta_0})$ ▸ compute the support vector
$P_{\theta_1} := \mathbb{M}_{d,l_j} \,.\rho_S(\mathbb{M}_{d,l_j}^T \, v_{\theta_1})$
$y_{min} := -\infty$
$y_{max} := \infty$
while $(y_{max} - y_{min}) > \epsilon$ **do**
  $\theta := \frac{\theta_0+\theta_1}{2}$
  $P_\theta := \mathbb{M}_{d,l_j} \,.\rho_S(\mathbb{M}_{d,l_j}^T \, v_\theta)$
  if $x_\theta < e$ **then**
    $P_{\theta_1} := P$
    $\theta_1 := \theta$
  else
    $P_{\theta_0} := P$
    $\theta_0 := \theta$
  end if
  $M = (e, y_{min}) := \overline{P_{\theta_0} P_{\theta_1}} \cap L$
  $y_{max} := min(h(\theta), y_{max})$
end while
return y_{max}

compute the intersection in nD, the Algorithm 4.2 is called $|D|$ times, and that for each direction of the set $D = \{l_1, \ldots, l_m\}$.

As an alternative to the dichotomous search, the 2D minimization problem (4.39) can also be solved by using pre-existing optimization methods and algorithms. The MATLAB Optimization Toolbox, for example, offers the function *fminbnd* which uses the golden section search method or the Parabolic Interpolation approach to find a minimum of a unimodal function over a fixed interval. The golden search method is principally similar to the dichotomous search. The difference prevails, however, through using a constant factor -the golden factor- to reduce the size of the search interval in each iteration. The decision on the choice between the left and the right sides of the search interval as a new search interval depends on the values of the function on its boundaries ([90] Section 10.2). The golden search method approximates the function with a piecewise linear function while the parabolic extrapolation approach fits the function with a parabola within the search interval. For more details about optimization methods, we refer the reader to the corresponding literature.

Based on the bracketing principle of the dichotomous and golden searches, many proposals were suggested to improve the convergence rate. An interesting key suggestion involves first considering a fixed number of points in the search interval for which the function values are evaluated. A new point is then computed by using the golden factor for example. A comparison of the function values on these different points allows the determination of the new bracketing interval. The method stops when a predefined precision is attained. The next section introduces an algorithm based on this method for solving the problems (4.36) and (4.37) directly in nD without recourse to 2D projections.

4.6.2. The Sandwich Algorithm

The minimization approach proposed in [96] combines the bracketing search method with its dichotomous and golden ration alternatives for the choice of new bracket points. A comparison of the function values on these points also assists in the region elimination technique. Furthermore, for this last criteria, a lower bound for the function is determined without any recourse to derivative calculation. This lower bound helps in choosing the new bracketing interval. In fact, based on the convexity property of the function f, the chord relating two points $(x_i, f(x_i))$ and $(x_j, f(x_j))$, $x_i < x_j$ given by:

$$L_{ij} : y = \frac{f(x_j) - f(x_i)}{x_j - x_i} (x - x_i) + f(x_i) \tag{4.40}$$

allows, as shown in Figure 4.9, a simultaneous determination of a lower bound of the minimum of the function f outside the interval $[x_i, x_j]$. Concretely, this means that $\forall x \notin [x_i, x_j]$, $y < f(x)$.

This property will be later used to formulate an appropriate criteria for finding a lower bound of the minimum of the function. This helps make the right choice of

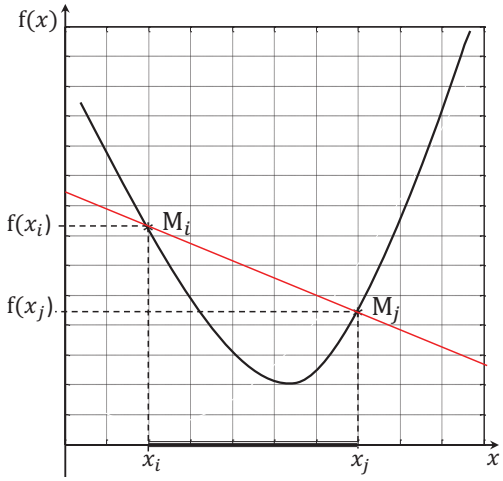


Figure 4.9.: The chord relating two points $(x_i, f(x_i))$ and $(x_j, f(x_j))$, $x_i < x_j$ is a lower bound of $f(x)$ outside $[x_i, x_j]$.

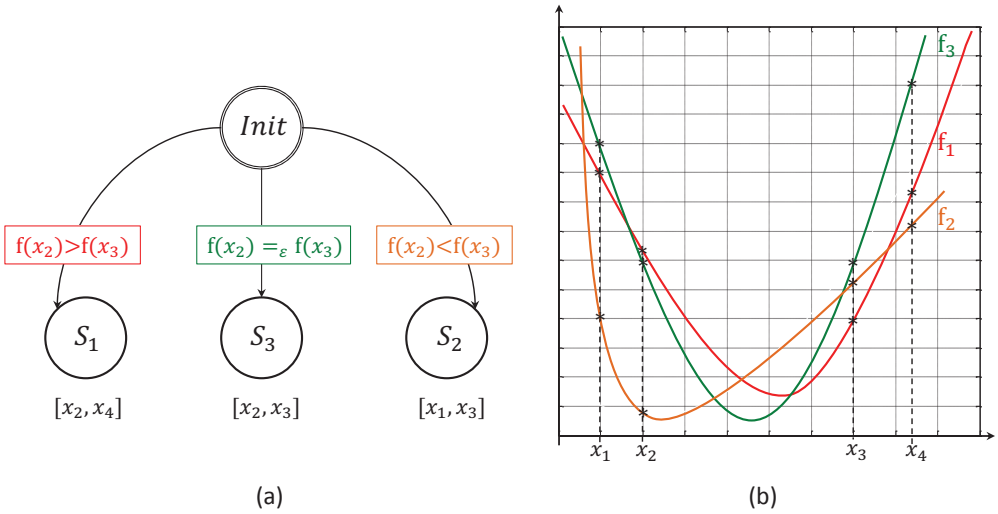


Figure 4.10.: Decision about the choice of the bracketing interval depending on the values of the function in x_2 and x_3. (a) Illustration as state machine. (b) Illustration as functions.

the next bracketing interval.
The solution proposed in [96] to the problems (4.36) and (4.37) begins with an initial interval I_0 in which four pivot points x_1, x_2, x_3 and x_4 satisfying the conditions $f(x_1) > f(x_2)$ and $f(x_3) < f(x_4)$ are selected. The search interval is thereby decomposed into subintervals. To decide on the new search interval, three different possibilities or states S_1, S_2 and S_3 depending on the values of the function on x_2 and x_3 are distinguished:

1. if $f(x_2) > f(x_3) \Rightarrow \ x_{min} \in [x_2, x_4] \Rightarrow S_1$ (state S_1 in Figure 4.10(a) and function f_1 in Figure 4.10(b))
2. $f(x_2) < f(x_3) \Rightarrow \ x_{min} \in [x_1, x_3] \Rightarrow S_2$ (state S_2 in Figure 4.10(a) and function f_2 in Figure 4.10(b))
3. $f(x_2) = f(x_3) \Rightarrow \ x_{min} \in [x_2, x_3] \Rightarrow S_3$ (state S_3 in Figure 4.10(a) and function f_3 in Figure 4.10(b))

where x_{min} is the intended minimum.

In the new bracketing interval $I_1 = [x_i, x_j]$ a new pivot x is then computed

- based on the bisection rule ($x = \frac{x_i + x_j}{2}$), or
- by applying the golden ratio rule ($\frac{x_j - x_i}{x - x_i} = \frac{x - x_i}{x_j - x} = \frac{1+\sqrt{5}}{2}$) or
- as the abscissa of a particularly predefined lower bound of the function f inside the interval $[x_i, x_j]$. This will be detailed below.

The above mentioned lower bound property is later used to determine the bracketing interval by states S_1 and S_4.

Steps by state S_1

The search interval for the state S_1 corresponds to $[x_2, x_4]$. This can be split into two subintervals $[x_2, x_3]$ and $[x_3, x_4]$. To decide about the next bracketing interval, two particular points are computed.

- The intersection point (α_1, min_1) between the chord L_{12} defined by the points $(x_1, f(x_1))$ and $(x_2, f(x_2))$ and the chord L_{34} joining both points $(x_3, f(x_3))$ and $(x_4, f(x_4))$ is first given (see Figure 4.12 and Figure 4.13).

$$(\alpha_1, min_1) = L_{12} \cap L_{34} \tag{4.41}$$

- Second the intersection point (α_2, min_2) of the chord L_{23} defined by the points $(x_2, f(x_2))$ and $(x_3, f(x_3))$ and the vertical through the point $(x_4, f(x_4))$, which we note here for convenience L_4, is determined (see Figure 4.12 and Figure 4.13).

$$(\alpha_2, min_2) = L_{23} \cap L_4. \tag{4.42}$$

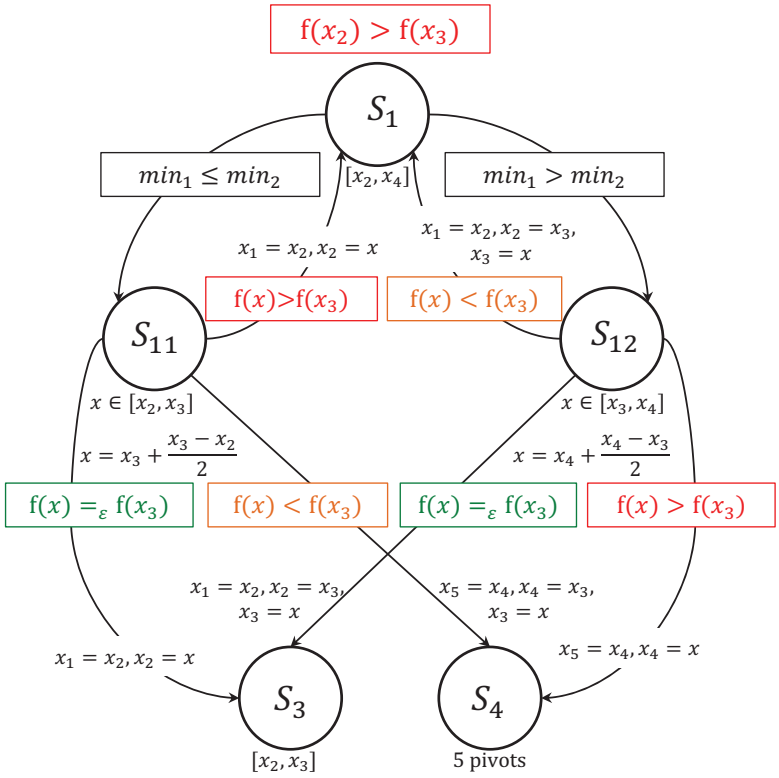


Figure 4.11.: State machine describing further possible steps for S_1 as initial state.

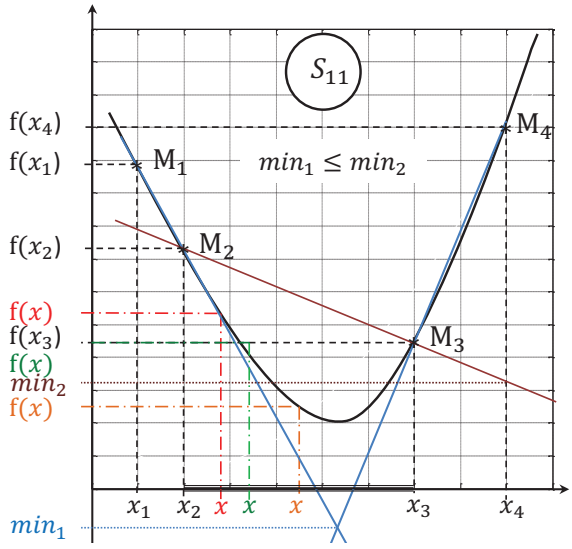


Figure 4.12.: Illustration of the state S_{11}.

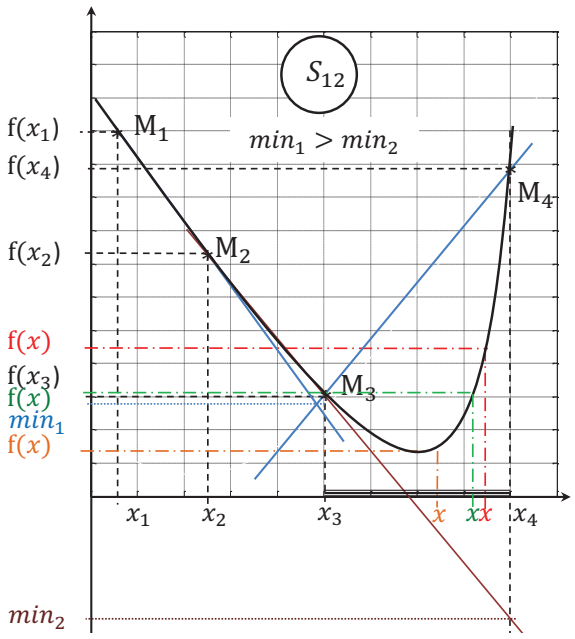


Figure 4.13.: Illustration of the state S_{12}.

The next bracketing interval I_1 corresponds to the subinterval in which $min = min(min_1, min_2)$ is located. This means

- if $min = min_1$ then $I_1 = [x_2, x_3]$. That corresponds to the state S_{11} in Figure 4.11. The pivot x is then computed by using the golden ratio rule or the bisection rule. We opt for the last one because [96] exhibits no difference in performance between both choices. We get hereby $x = \frac{x_2+x_3}{2}$. Depending on the value of $f(x)$ in comparison with $f(x_3)$, three different states are possible. Details of these steps are given by the state machine of Figure 4.11 and illustrated in Figure 4.12.

- if $min = min_2$ then $I_1 = [x_3, x_4]$. This however corresponds, however, to the state S_{12} in Figure 4.11. The next pivot is $x = \frac{x_3+x_4}{2}$. Details of further steps are given in Figure 4.11 and illustrated in Figure 4.13.

Depending on the comparison result between $f(x)$ and $f(x_3)$, the state S_1 or S_3 or the five pivots state S_4 is enabled according to the state machine of Figure 4.11.

Steps by state S_2

The search interval for the state S_2 corresponds to $[x_1, x_3]$. This can be split into two subintervals $[x_1, x_2]$ and $[x_2, x_3]$. For the choice of the next bracketing interval, We proceed as by the state S_1 and compute two particular points.

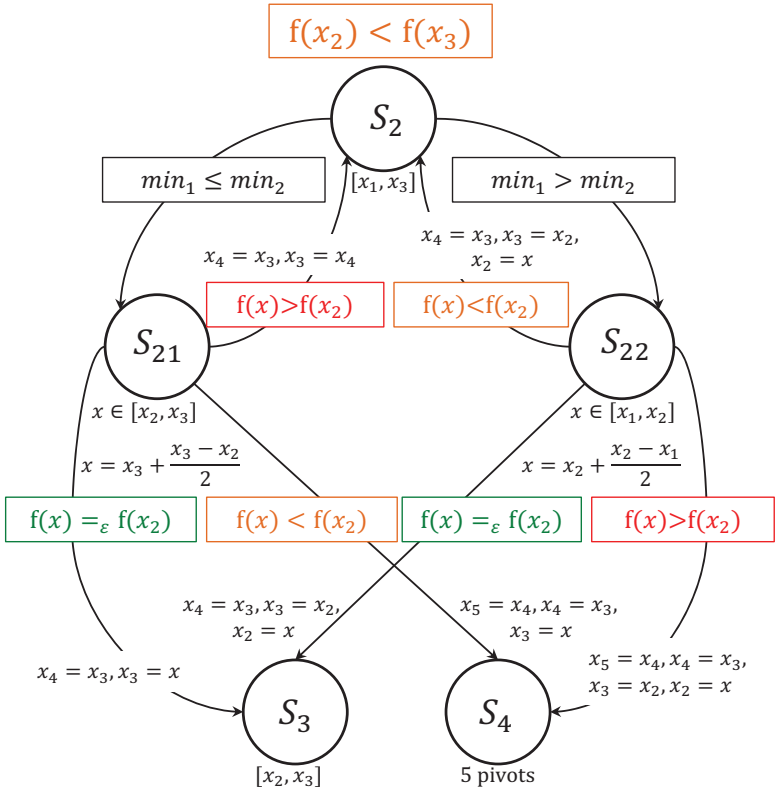


Figure 4.14.: State machine describing further possible steps for S_2 as initial state.

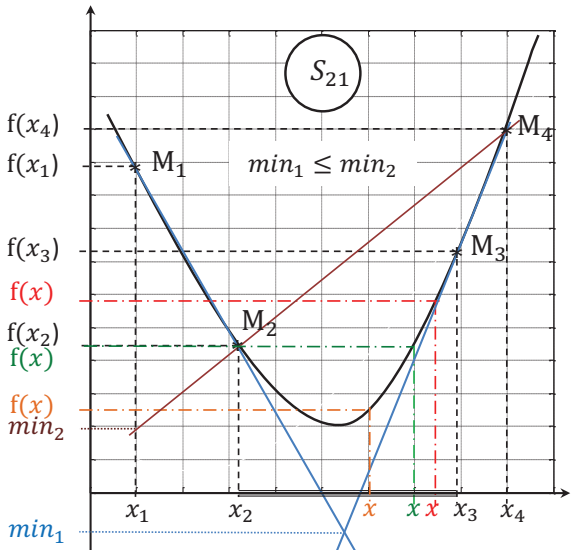


Figure 4.15.: Illustration of the state S_{21}.

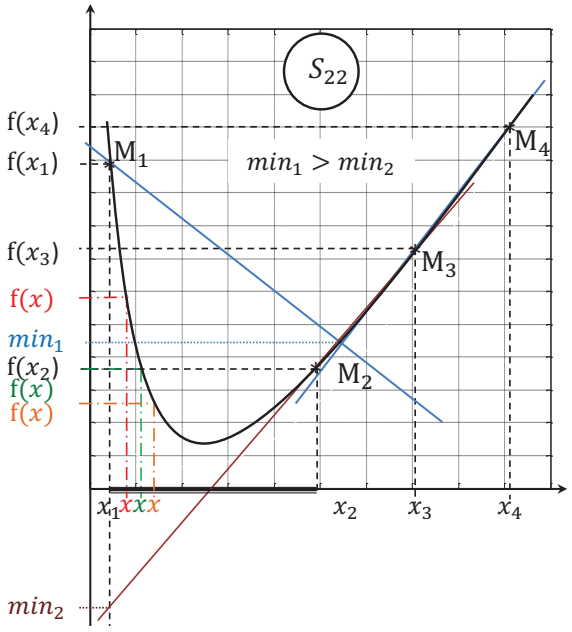


Figure 4.16.: Illustration of the state S_{22}.

- We begin with the computation of the intersection point (α_1, min_1) between the chord L_{12} joining the points $(x_1, f(x_1))$ and $(x_2, f(x_2))$ and the chord L_{34} joining the points $(x_3, f(x_3))$ and $(x_4, f(x_4))$ (see Figure 4.15 and Figure 4.16).

$$(\alpha_1, min_1) = L_{12} \cap L_{34} \tag{4.43}$$

- We then compute the intersection point (α_2, min_2) between the chord L_{23} joining the points $(x_2, f(x_2))$ and $(x_3, f(x_3))$ and the vertical through the point $(x_1, f(x_1))$, which we denote here as L_1 for convenience (see Figure 4.15 and Figure 4.16).

$$(\alpha_2, min_2) = L_{23} \cap L_1 \tag{4.44}$$

Depending on the location of $min = min(min_1, min_2)$, the next bracketing interval I_1 is therefore

- $[x_2, x_3]$ if $min = min_1$. This corresponds to the state S_{21} in Figure 4.14 which is also illustrated in Figure 4.15. The new pivot is then given by $x = \frac{x_2+x_3}{2}$. The enable condition to the next state is governed by the result of the comparison between $f(x)$ and $f(x_2)$. Details of these steps and new pivot-naming are given in the state machine of Figure 4.14.

- $[x_1, x_2]$ if $min = min_2$. This corresponds to the state S_{22} in Figure 4.14. The next pivot is $x = \frac{x_1+x_2}{2}$. Details of further steps are given in Figure 4.11 and illustrated in Figure 4.16.

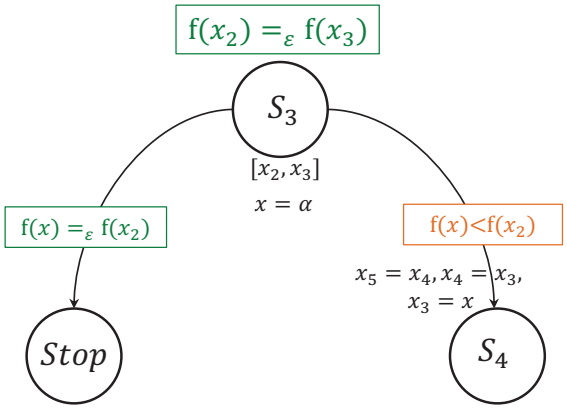


Figure 4.17.: State machine describing further possible steps for S_3 as initial state.

The result of the comparison between $f(x)$ and $f(x_2)$ will trigger the decision about the next state as illustrated in the state machine of Figure 4.14.

Steps by state S_3

In this case, it is obvious from the condition $f(x_2) = f(x_3)$ that the bracketing interval will be $[x_2, x_3]$. For the choice of the new pivot x, the maximum error rule is in [96] adopted. This consists of first computing the intersection point

$$(\alpha, min) = L_{12} \cap L_{34} \tag{4.45}$$

and second taking $x = \alpha$, as illustrated in Figure 4.18.
The algorithm terminates if $f(x) = f(x_2)$. Wherever this is not the case, the five pivots state is selected (see Figure 4.17).

The five pivots state S_4

To accelerate the search, the new pivot x is added to the already chosen four pivots so that the search proceeds as below but with five pivots. A minor difference is in the determination of the values min_1 and min_2. The search interval is, in this case, $[x_2, x_4]$ which can be split into $[x_2, x_3]$ and $[x_3, x_4]$. The lower bound on the minimum of the function f in $[x_2, x_3]$ is given by the intersection point

$$(\alpha_1, min_1) = L_{12} \cap L_{34}, \tag{4.46}$$

whereas in $[x_3, x_4]$, this is given by

$$(\alpha_2, min_2) = L_{23} \cap L_{45}. \tag{4.47}$$

The interval where $min = min(min_1, min_2)$ is located corresponds to the new bracketing interval. Further computation steps are described by the state machine

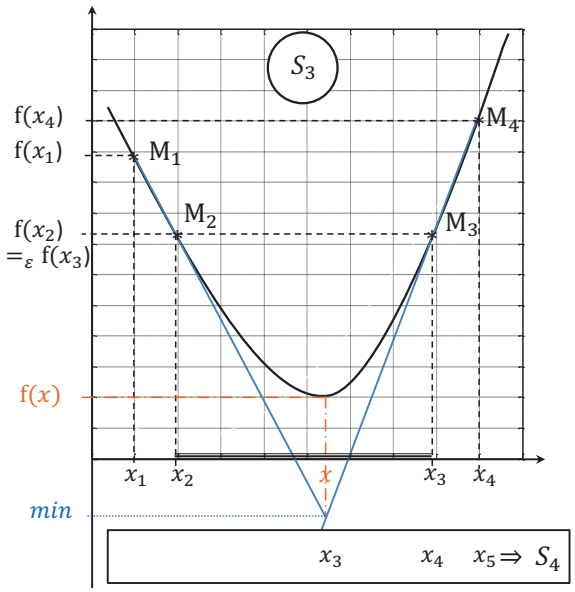


Figure 4.18.: Illustration of the state S_3.

of Figure 4.19 and illustrated in Figure 4.20 and Figure 4.21. For the new pivot the maximum rule is applied so that

$$x = \begin{cases} \alpha_1 & \text{if } min = min_1 \\ \alpha_2 & \text{if } min = min_2. \end{cases} \tag{4.48}$$

The algorithms stops if $f(x) = f(x_3)$.

4.6.3. Optimization Techniques

As an alternative to the previous proposed methods, the minimization problems (4.36) and (4.37) may also be solved using existing solvers. The MATLAB optimization toolbox, for example, offers a wide range of derivatives and non-derivatives based optimizers. It provides, for one, the derivative-free method *fminsearch* function and also the derivative based *fminunc* function to solve the unconstrained optimization problem (4.36). The *fminsearch* function is built on the Nelder-Mead simplicial direct search method. It compares the objective function at a finite number of points and decides whether to reflect, expand, contract or shrink the actual simplex to replace the simplex vertex with the worst function value with a best one. MATLAB offer two different algorithms for the derivative-based function *fminunc*. The quasi-Newton method uses the information about the objective function and its gradient to approximate the Hessian matrix and therefore build a quadratic approximation of the objective function. The standard trust-region method approximates in a predefined region the objective function with the two first terms of the Taylor

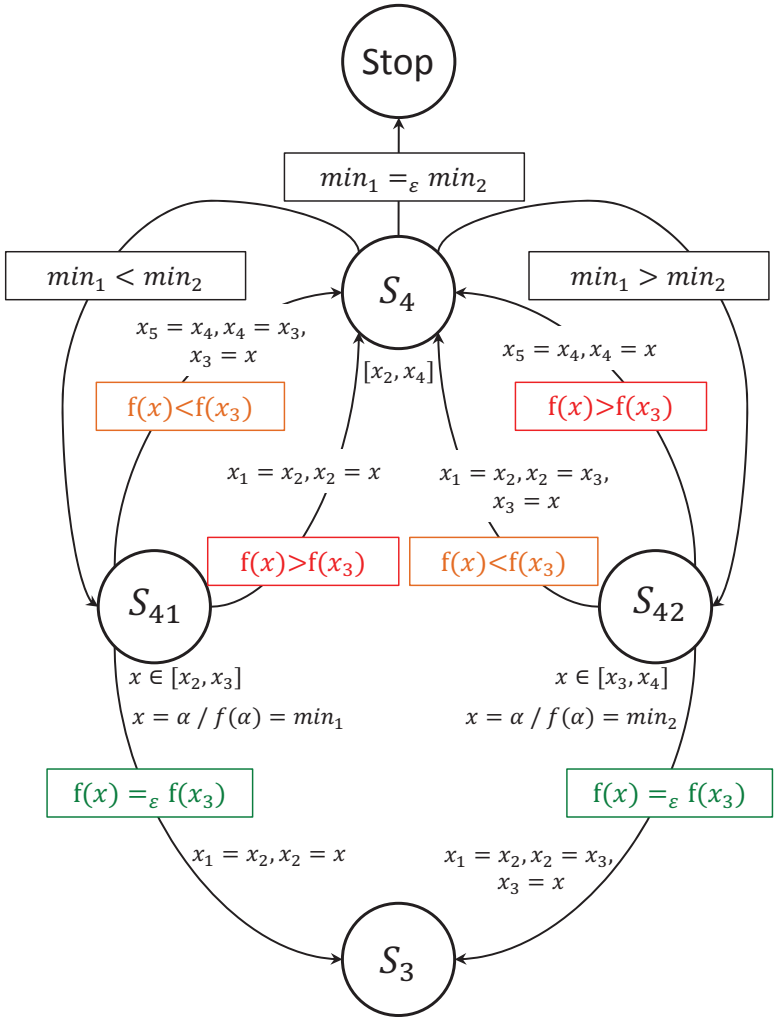


Figure 4.19.: State machine describing further possible steps for S_4 as initial state.

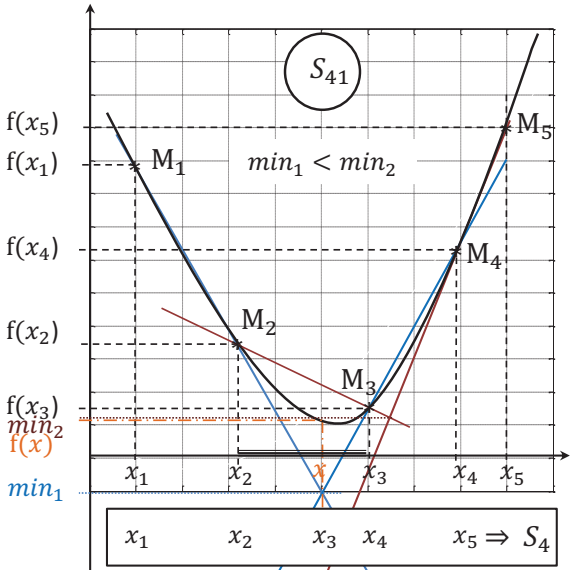


Figure 4.20.: Illustration of the state S_{41}.

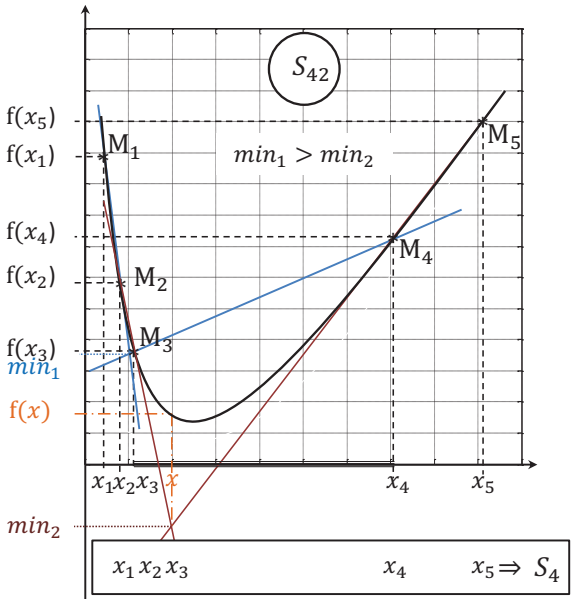


Figure 4.21.: Illustration of the state S_{42}.

expansion. Both functions *fminsearch* and *fminunc* are proposed as alternative options to solve the unconstrained optimization problem (4.36). On the other hand, the handling of inequality guards requires the constrained optimization problem (4.37) to be considered. For this kind of optimization problems, other techniques are though required. The *fmindnd* MATLAB function uses the golden Search approach to solve such constrained problems for cases where the solution is far away from the actual state and parabolic interpolation otherwise. The *fmincon* function, however, offers the possibility to choose between different algorithms, the interior-point, the trust-region reflective, the sequential quadratic programming (SQP) and the active-set, to find the minimum for the problem (4.37). This section gives just a brief list with a short description of the MATLAB optimization toolbox solvers we proposed as alternative to the algorithms of sections 4.6.2 and 4.6.1. This work does not intend to go deeper into convex optimization techniques. Relevant details can be found in the corresponding literature, in particular the references [26] and [79].

4.7. Handling Invariants with Support Function Techniques

In this section, we describe how the two different approaches for handling invariants presented in Section 3.8 of 3 are formulated and implemented with support functions. We suggest several modifications on the original algorithms proposed in [71, 96] to circumvent some potential problems encountered during the implementation. Furthermore, we make some suggestions to enhance its efficiency and tighten the over-approximation of the computed reachable set.

4.7.1. Classical Method for Handling Invariants

Invariants can be handled using the same methods as with guards. Principally, this begins in each iteration with a collision detection check of the computed reachable set with the invariant set and then the computation of the intersection in the case of partial collision. It is however important to differentiate between the two types of constraints because the behavior of the hybrid system after the collision detection depends largely on the nature of the constraints. In fact, in the case of intersection with guards, the corresponding transition of the hybrid automaton is triggered. However, in the case that the invariants are fulfilled (albeit partially), the hybrid automaton remains in the same location. It hence makes sense to use the same collision detection function as well as the same intersection computation method for guards and invariant during the analysis process as proposed in Algorithm 3.2. Alternatively, the property (4.33) of set intersection operation with support functions facilitates the implementation of Algorithm 3.3 where invariants and guards are treated separately.

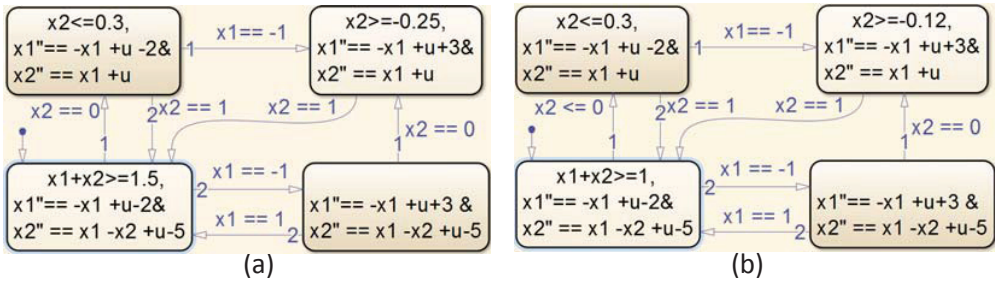


Figure 4.22.: The hybrid automata of the two-tank benchmark with different invariant conditions within discrete modes.

4.7.2. Recursive Scheme Fusing Reachable Sets with Domain/Invariant Conditions

This approach was first proposed in [71]. The resulting recursive scheme has already been introduced in its general form in Section 3.8 with the equations (3.42) and (3.43). If we now express these equations in terms of support functions and adopt the property in equation (4.33) for the intersection, we obtain the following formula for the computation of the polyhedral approximation P_k of the set $\tilde{\Omega}_k$

$$P_k = \bigcap_{i=0}^{m} \left\{ x : {l_i}^T.x \leqslant min\left(\rho_{\Omega_k}(l_i), min_{j=0}^{k-1}\rho_{\mathcal{I}_j}(l_i)\right)\right\} \tag{4.49}$$

where m is the number of directions in a predefined direction template $D = \{l_1, \ldots, l_m\}$.
The algorithm proposed in [71] computes the support function $\rho_{\tilde{\Omega}}(l)$ of the reachable set $\tilde{\Omega}$ in a given direction l. It first checks for an intersection of the computed reachable set with the invariant $\mathcal{I}$ by testing, in each iteration, if the condition $\rho_{\tilde{\Omega}}(l) \geqslant -\rho_{\mathcal{I}}(-l)$ holds. This last condition is valid for reachable sets entirely or partially contained within the invariant. Figures 4.23 and 4.24 show the reachable sets computed for the initial location of the two-tank benchmark of Figure 4.22(a). In these figures, the whole flowpipe of the considered location without considering the invariant is plotted. The reachable sets intersecting the invariant are marked in red. The invariant set $\mathcal{I}$ described by the inequality $x_1 + x_2 \geqslant 1.5$ in Figure 4.22(a) is outlined in blue. We note, for example, in Figure 4.23 that the set Ω_0 in dark blue is entirely inside the invariant. However, the set Ω_1, outlined in dark blue in Figure 4.24, partially intersects the invariant.

By applying the property (4.33), the intersection, if it ever exists, is then computed. Otherwise, the algorithm terminates. We note that this algorithm requires an evaluation of $\rho_{\mathcal{I}}$ in both direction l and $-l$. For this reason, we decided to extend this algorithm to allow a simultaneous computation of $\rho_{\tilde{\Omega}}(l)$ and $\rho_{\tilde{\Omega}}(-l)$ while executing the iteration. Since the direction template contains simultaneously both directions and their opposites, this enhancement will drastically reduce double

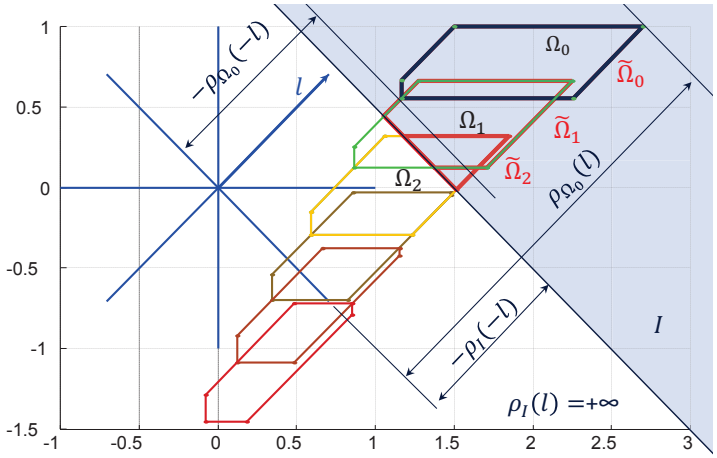


Figure 4.23.: The reachable set is entirely included in the invariant.

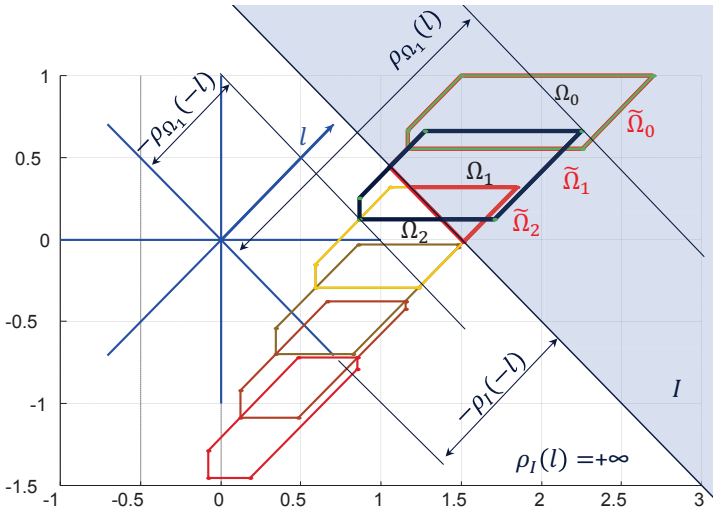


Figure 4.24.: The reachable set is partly included in the invariant.

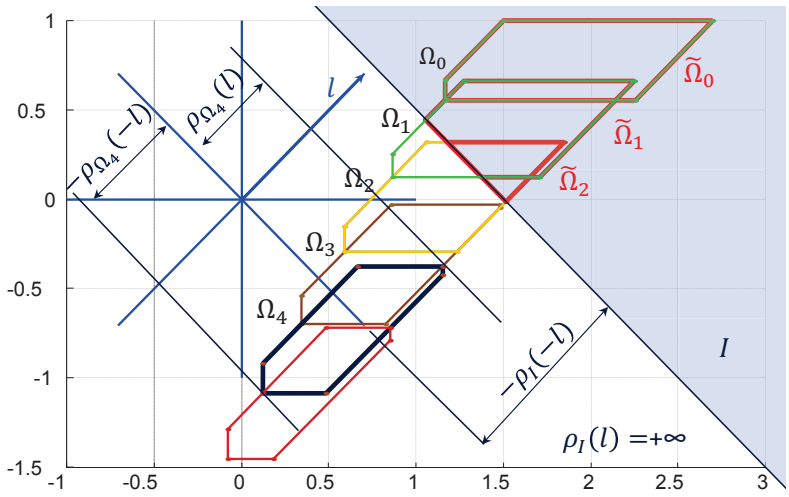


Figure 4.25.: The reachable set is not included in the invariant.

evaluations of support functions. To count for the direction $-l$, we added in our algorithm 4.3 the condition $\rho_{\tilde{\Omega}}(-l) \geqslant -\rho_{\mathcal{I}}(l)$ to the while loop condition at line 13. Furthermore, it may be observed, that the more relevant direction for the check and computation of the intended intersection is the invariant direction $l_{\mathcal{I}}$. If for an iteration k the intersection check in this direction fails, we can be sure that the support functions computed for all the other directions in this iteration and the next ones are irrelevant because they describe sets outside the invariant. In this case, we only have to compute $\rho_{\tilde{\Omega}_1}, \ldots, \rho_{\tilde{\Omega}_{k-1}}$ for all directions. Otherwise, we have to continue the computation until the condition $k < N$ is no longer fulfilled. This case means that during the chosen time horizon $T = Nr$, the reachable sets remain inside the invariant. To account for both cases, two alternatives are possible. The first one consists in storing, for each direction l_i, the number of iteration k_i after which no intersection is detected, taking their minimum $\boldsymbol{min} = min_m^{i=1} k_i$ and computing with that the support functions $\rho_{\tilde{\Omega}_j}(l_i)$ for $j = 1, \ldots, \boldsymbol{min}$ and $i = 1, \ldots, m$ representing the intended reachable sets. Alternatively, if $l_{\mathcal{I}}$ is added in a prior step at the top of the direction template D, the computation will therefore first begin with this direction $l_{\mathcal{I}}$. We note thereby the number of sets intersecting the invariant and regard it as the maximum number of iterations in the while loop for further computations.

On the other hand, we added an if-condition -the negation of the while condition without its right hand side- inside the while loop in line 31 of Algorithm 4.3. The goal of this is to detect the first reachable set of the flowpipe leaving the invariant completely and note its index.

Algorithm 4.3 AlgoInv: Computing the intersection with the invariant while computing reachable set inside a location.

Input: N, r, A, l, ρ_I, ρ_{Ω_0}, $\rho_{\mathcal{V}_r}$
Output: $\rho_{\tilde{\Omega}_0}(l), \ldots, \rho_{\tilde{\Omega}_{k-1}}(l), \rho_{\tilde{\Omega}_0}(-l), \ldots, \rho_{\tilde{\Omega}_{k-1}}(-l)$

1: $k = 1$
2: $r = r_{old} = l$ ▸ Initialization step
3: $s_{old} = s_{old}^{-} = 0$
4: $h_{old} = h_{old}^{-} = h = h^{-} = +\infty$
5: $p_\Omega = \rho_{\Omega_0}(l_i)$
6: $p_\Omega^{-} = \rho_{\Omega_0}(-l_i)$
7: $p_I = \rho_I(l)$
8: $p_I^{-} = \rho_I(-l)$
9: $I^{-} = -p_I^{-}$
10: $I^{+} = -p_I$ ▸ End of the initialization step
11: $results(1,1) = min(p_\Omega, p_I)$
12: $results(2,1) = min(p_\Omega^{-}, p_I^{-})$
13: **while** $(results(1,k) >= I^{-}) \wedge (results(2,k) >= I^{+}) \wedge (k < N)$ **do**
14: $k = k + 1$
15: $r = \left(e^{rA}\right)^T * r_{old}$
16: $s = s_{old} + \rho_{\mathcal{V}_r}(r_{old})$
17: $s^{-} = s_{old} + \rho_{\mathcal{V}_r}(-r_{old})$
18: $p_\Omega = \rho_{\Omega_0}(r) + s$
19: $p_\Omega^{-} = \rho_{\Omega_0}(-r) + s^{-}$
20: $h = min(h_{old}, p_I)$
21: $h^{-} = min(h_{old}^{-}, p_I^{-})$
22: $p_I = \rho_I(r) + s$
23: $p_I^{-} = \rho_I(-r) + s^{-}$
24: $results(1,k) = min(p_\Omega, h)$
25: $results(2,k) = min(p_\Omega^{-}, h^{-})$
26: $r_{old} = r$
27: $s_{old} = s$
28: $s_{old}^{-} = s^{-}$
29: $h_{old} = h$
30: $h_{old}^{-} = h^{-}$
31: **if** $results(1,k) < I^{-} \vee results(2,k) < I^{+}$ **then**
32: $k = k - 1$
33: **end if**
34: **end while**
35: **return** $results(1,1), \ldots, results(1,k), results(2,1), \ldots, results(2,k)$

For the purposes of illustration, we note in Figure 4.25 that the sets Ω_3, Ω_4 and Ω_5 verify the if-condition and are hence outside the invariant. The intended index is equal to 3 here. We decided furthermore to treat the computation of the intersection

of the initial reachable set Ω_0 with the invariant in the initialization step to avoid further computations in case of no intersection.

4.8. Handling Guards

As mentioned in Section 3.9, two tests must be done at the end of each iteration k to compute the reachable set $\tilde{\Omega}_k$ at a time point t_k. The first test checks for intersection with the invariant of the location. This was already dealt with in the former section. The second test aims at checking for collision between the computed reachable set $\tilde{\Omega}_k$ and the guard condition $\mathcal{G}$ and to note, at the same time, the corresponding iteration number. Since the intersection can take place for a number of successive iterations, a k_{min} and a k_{max} marking the first and the last intersection detections are noted. To compute the intersection in an upcoming step, two strategies may be pursued. The support function of the intersection can be computed directly using one of the methods presented in Section 4.5 after collision detection and that within the same iteration resulting in the support function $\rho_{\tilde{\Omega}_k^{int}} = \rho_{\tilde{\Omega}_k \cap \mathcal{G}}$ describing the set $\tilde{\Omega}_k^{int} = \tilde{\Omega}_k \cap \mathcal{G}$. All the reachable states which join the guard are enclosed in set represented by the following support function $max_{k=k_{min}}^{k=k_{max}} \rho_{\tilde{\Omega}_k^{int}}$. This corresponds to the post-clustering alternative introduced previously in Section 3.9 of Chapter 3. The pre-clustering alternative computes first the support function $max_{k=k_{min}}^{k=k_{max}} \rho_{\tilde{\Omega}_k}$ of the set $\bigcup_{k=k_{min}}^{k=k_{max}} \tilde{\Omega}_k$ and in a second step the support function of the intersection with the guard $\min\left(max_{k=k_{min}}^{k=k_{max}} \rho_{\tilde{\Omega}_k}, \rho_{\mathcal{G}}\right)$ is computed. Once the intersection with the guard is computed, the reachable set after the transition Ω_{trans} is obtained by applying the following reset map $\mathcal{R}e$ to the resulting intersection.

$$\begin{aligned} \mathcal{R}e: \quad & \mathbb{R}^n \longrightarrow \mathbb{R}^n \\ & X \longmapsto \mathcal{R}e(X) := RX \oplus W \end{aligned} \tag{4.50}$$

where $X \subseteq \mathbb{R}^n$, $R \in \mathbb{R}^{n \times n}$ and $W \subseteq \mathbb{R}^n$ a convex set. The corresponding support function is then

$$\rho_{\Omega_{trans}}(l) = \rho_{\mathcal{Y}}\left(R^T l\right) + \rho_W(l) \tag{4.51}$$

where l is an arbitrary direction. A precise approximation Ω_{trans} of the intersection demands the addition of the normal directions of the guards and the invariants to the template of the support directions.
The resulting approximation will be consequently considered as the initial set of the upcoming continuous mode.

4.9. Simultaneously Handling of Invariants and Guards

The idea proposed in Section 4.7.2 to count for the invariant condition while computing the reachable sets can be also extended to guard conditions in the way

suggested in [72]. The recursion for the computation of the polyhedral approximation P_k of the intersection of the reachable set inside a mode with the invariant and the guard conditions at iteration k thereby takes the following form:

$$P_k = \bigcap_{i=0}^{m} \left\{ x : {l_i}^T.x \leqslant min\left(\rho_{\Omega_k}(l_i), \rho_{\mathcal{G}}(l_i), min_{j=0}^{k-1}\rho_{\mathcal{I}_j}(l_i)\right)\right\} \tag{4.52}$$

where m is the number of directions in a template $D = \{l_1, \ldots, l_m\}$ which should include in particular the normal directions describing the guard and invariant conditions.
In [96], a slightly different strategy was suggested with the goal of higher degree of precision of the approximation. It involves, besides the invariant I^- of the source mode q^-, the invariant I^+ of the transition target mode q_+. In fact, it is not possible to proceed with the computation in the mode q_+ if its invariant can be never fulfilled. For this reason, it makes sense to compute the following approximation corresponding to the initial set of mode q^+

$$\Omega_{trans} = \left[R\left(\Omega \cap \mathcal{G} \cap I^-\right) \oplus W\right] \cap I^+ \tag{4.53}$$

where Ω here denotes any reachable set. If we define the pre-image $I^* = \mathcal{R}e^{-1}\left(I^+\right)$ and consider the following lemma

Lemma 9. $(R\,\Omega \oplus W) \cap I^+ \subseteq R\left(\Omega \cap I^*\right) \oplus W$,

the following new approximation can accordingly be derived

$$\hat{\Omega}_{trans} = R\left(\Omega \cap \mathcal{G} \cap I^- \cap I^*\right) \oplus W. \tag{4.54}$$

It remains now to know how to practically compute the set I^* for a target invariant taking the following form $I^+ = \left\{x \in \mathbb{R}^n | \bigcap_{i=1}^{q} {n_i}^T x \leqslant e_i\right\}$. Simple computation steps lead to the following result $I^* = \left\{x \in \mathbb{R}^n | \bigcap_{i=1}^{q} \left(R^T n_i\right)^T \; x \leqslant e_i + \rho_W\left(-n_i\right)\right\}$. The template directions D once again should be extended with the directions $R^T n_i$ for $i = 1, \ldots, q$.

4.10. Handling Spontaneous Transitions

Spontaneous transitions are in this context defined as transitions which can take place arbitrarily. They are not triggered by a state-variable guard condition. They are referred to as *True* transitions. To guarantee the enclosure of all reached states in case of occurrence of such transitions, a jump to the target location is allowed only if successive reachable sets computed in the source location are ϵ-equal, where ϵ is the chosen tolerance. In this case, the reachability algorithm converges to a fixpoint. We consequently treat these *True* transitions as fixpoint-triggered transitions. We are now left with finding a way to check for fixpoints with support function techniques.

Lemma 10. Let S_1, $S_2 \subset \mathbb{R}^n$ be nonempty sets, and $\epsilon \in \mathbb{R}$. The sets S_1, S_2 are ϵ-equal if for all $j \in \{1, \ldots, m\}$

$$|\rho_{S_1}(l_j) - \rho_{S_2}(l_j)| \leqslant \epsilon. \tag{4.55}$$

We use this lemma to check for a fixpoint. If this is reached after a certain user specified number of iterations, it will be taken as the initial set for the target transition.

4.11. Handling Time-Triggered Transitions

Time triggered transitios can be regarded as state-variable guard transitions, if the time t is included as an ordinary state variable by adding a simple clock $\dot{t} = 1$ to the differential equations describing the continuous behavior of hybrid system and by involving it in the guard conditions too. Another alternative is to allow, depending on the objective, different time horizon choices for different locations. By doing so, a transition will take place directly after computation termination in the corresponding location.

4.12. MATLAB/Simulink Implementation

This section presents our first exploration prototyping MATLAB/Simulink implementation *HyReach* of the support function technique for the reachability analysis of linear hybrid systems. Figure 4.26 shows the breakout of its main components. It consists basically of an interactive graphical user interface (GUI) and a computation core (CC) exchanging information in the way illustrated in Figure 4.28. All suggested scenarios for the flowpipe computation and above-detailed methods for handling transitions have been modularly implemented within the CC of *HyReach* as shown in Figure 4.28. This modular structure allows for flexible combination. Through the GUI, the layout shown in Figure 4.27, the user can easily choose between different scenarios and methods, decide about their combination and configure the reachability analysis. For the construction of the flowpipe in continuous modes following options are available.

- The *NoScale* scenario is an implementation of the first support function based reachability algorithm proposed in [72]. This involves the Approach 1 of Section 4.3.1 for over-approximating the input contribution and Scenario 1 of Section 4.3.2 for the initial set.

- The *ConstU* scenario is based on the assumption that the input remains practically constant within small time steps. The basic algorithm given in [21] uses the Approach 3 of Section 4.3.1 for the input contribution and Scenarios 1, 2, and 3 of Section 4.3.2 for the initial set.

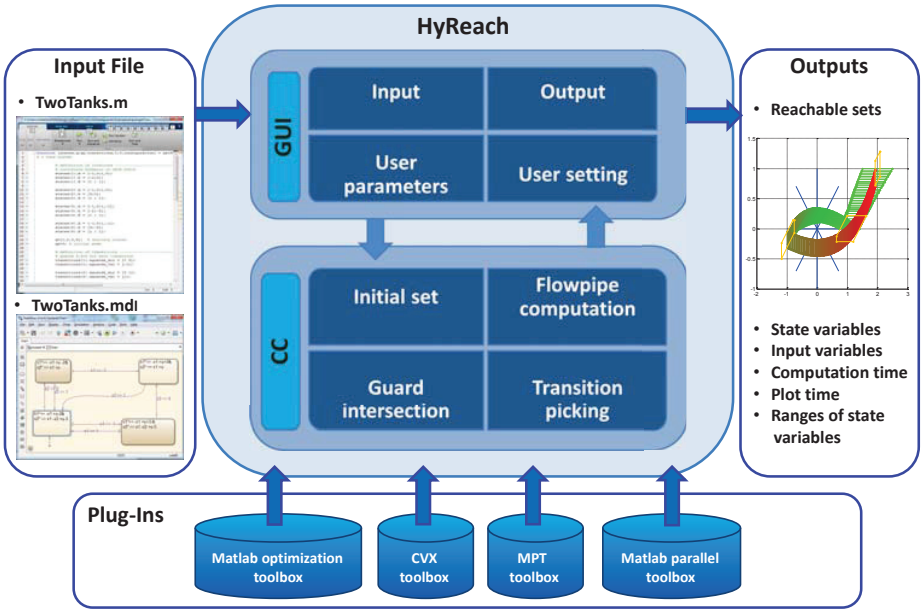


Figure 4.26.: HyReach architecture.

- The *SpaceEx* scenario is the implementation of the Algorithm 4.1. The user can choose between two initialization methods: the *PreciseOmega0* initial over-approximation and the *SpaceEx* over-approximation which correspond respectively to the second and the third scenarios of Section 4.3.2. For the input contribution Approach 2 of Section 4.3.1 is adopted.

- The *AlgoInv* scenario is the implementation of Algorithm 4.3 embedded in Algorithm 3.3. It adopts Approach 2 of Section 4.3.1 for input contribution and both approaches *PreciseOmega0* and *SpaceEx* for the initial set over-approximation.

- The *AlgoInv2* scenario is the implementation of Algorithm 3.2 with the same over-approximation choices as above.

In addition to these different scenarios for the flowpipe computation, it is also possible to choose between different optimization algorithms required for the computation of the support function. Besides the MATLAB optimization toolbox, the CVX and the MPT-toolboxes are added as Plug-ins to extend the choice of optimization algorithms.
The user also has the possibility to choose between the *Box*, *Oct* or *User* set-angle options to generate a template of directions for the evaluation of the support functions.
Choices are furthermore available for handling transitions with the different meth-

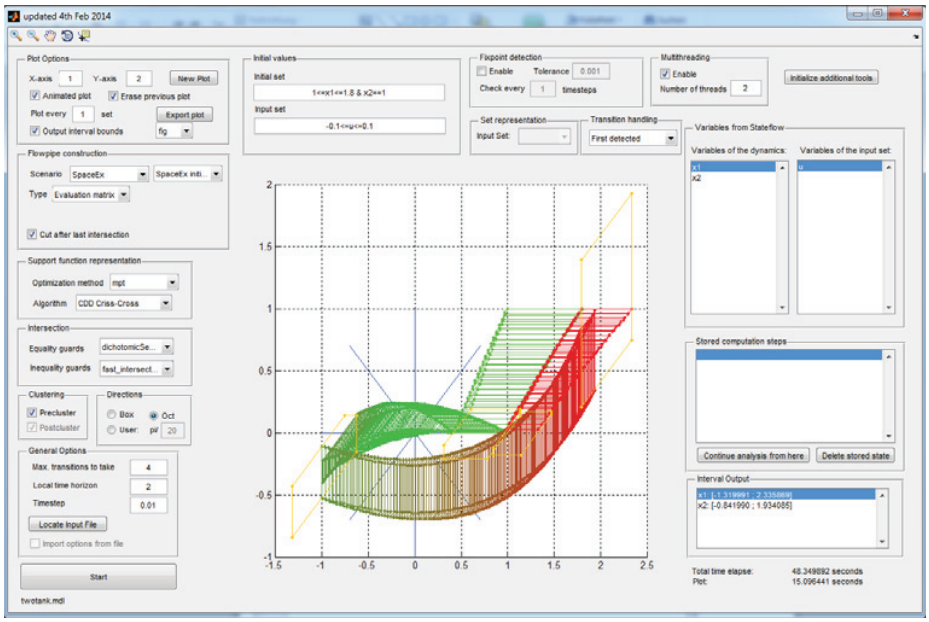


Figure 4.27.: Screenshot of the GUI.

ods presented in this section combined with different clustering strategies. We distinguish here between guard conditions given as equalities/hyperplanes and those described by inequalities/halfspaces.
For the intersection of reachable sets with a hyperplane following choices can be made.

- The *fminsearch* option involves a direct Nelder-Mead sequential simplex algorithm, part of the MATLAB optimization toolbox to solve the intersection problem.
- *fminunc* provides an alternative iterative method for solving unconstrained nonlinear optimization problems based on the Broyden Fletcher Goldfarb Shanno (BFGS) approach.
- *dichotomicSearch* is the implementation of Algorithm 4.2 for the dichotomous search method proposed in [71].
- *RayAlgo* is an enhancement of the sandwich approach suggested in [43] and detailed in Section 4.6.2.
- *fast intersection* is an application of property 7 of support functions.

For the intersection with halfspaces, following options are made available.

- *fminbnd* is a combination of the golden section search and the parabolic interpolation methods and is a part of the MATLAB optimization toolbox.

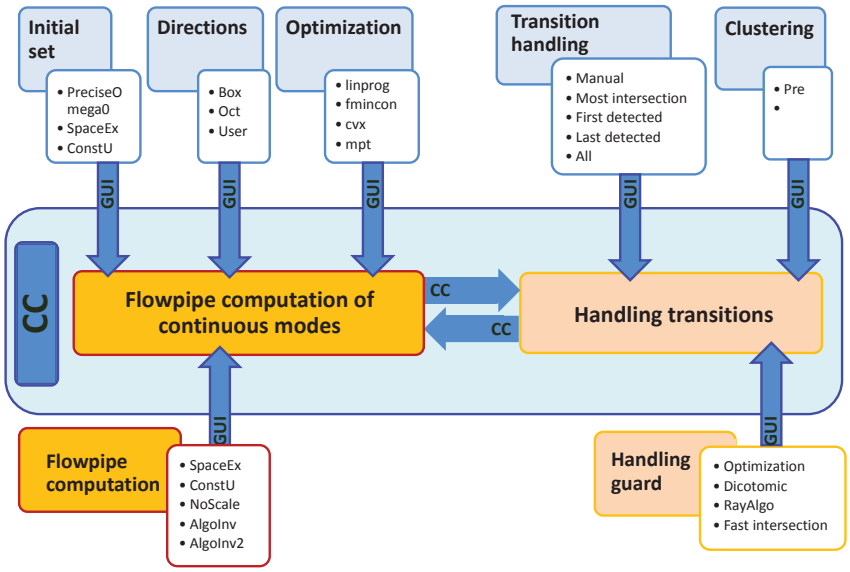


Figure 4.28.: User setting possibilities for the configuration of the reachability analysis.

- *fmicon* is a Sequential Quadratic Programming-based (SQP) algorithm provided by the MATLAB optimization toolbox.
- *RayAlgo* as above.
- *fast intersection* as above.

Moreover, different strategies for picking the target location, in the case of many resulting transitions, are placed at the disposal of the user. A choice between *Manual*, *First detected*, *Last detected*, *Most intersections* and *All* options can be made. With the *All* option, all possible runs through the hybrid automaton are considered. The *Manual* option, however, allows the choice of the next target location progressively during the computation. Flowpipes computed in different locations as well as the intersection sets corresponding to different transitions are saved in a tree structure. Each run through the hybrid automaton ends consequently with a leaf. Backtracking is afterwards applied to retrieve previous states if the *Manual* option or *All* option is set.

The HyReach GUI allows as shown in Figure 4.27 besides the setting of parameters and the configuration of the reachability analysis, the loading and the editing of parameters embedded in a graphical (*.mdl) or in a textual input model (*.m). The graphical hybrid model is build in the MATLAB/Stateflow environment using a slightly amended semantic. The MATLAB input file, however, describes the hybrid automaton textually with our own specially conceived semantic. Examples of both formats are given in Figure 4.29 and Figure 4.30 for the bouncing ball

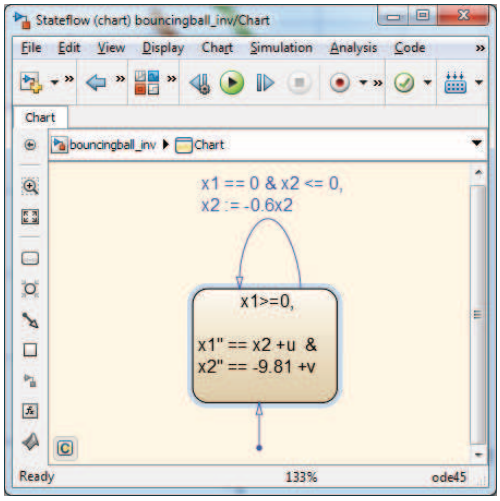


Figure 4.29.: Graphical input file of the bouncing ball example.

example. The textual input has shown its practice for large systems and for direct capture of hybrid automaton parameters from the control file. This is especially convenient if several controllers of the same system have to be tested.

After launching the analysis via the GUI, 2D-projections of the reachable sets can be plotted and saved at the end of the computation. The time progress of the computation is visualized by a color gradient, beginning with green sets and ending with red ones. The enclosure of intersecting sets with guards are plotted in yellow. In addition, information like the computation time and interval ranges can also be surveyed via the GUI.

4.13. Experimentation

We evaluate our implementation with the benchmark suite of [33] available under [61]. We select just some of them for the sake of brevity and present their corresponding results to highlight important features of HyReach.

The benchmarks differ from each other not only with regards to their dynamics but also to their state dimension and the number of modes and transitions. Furthermore, the guard conditions can make an important difference between the benchmarks as the number of equalities or inequalities can have a crucial impact of the complexity. In addition, we include benchmarks with invariants in locations as well as benchmarks with reset conditions in transitions. We first investigate features such as the scalability with respect to the number of variables, locations and transitions. We use for this investigation Intel Core i5-2520M @ 2.50GHz laptop with an 8GB RAM and MATLAB R2014a. We also select the options the *MPT-CDD-Criss-*

```
function [states,q,qa,transitions,I,U,configration] = getBouncingBallInput_Inv
% Bouncing Ball
    % definition of locations ---------------------------------------------
    % continuous dynamics in each state
    states{1}.A = [0,1;0,0];
    states{1}.b = [0;-9.81];
    states{1}.B = [1 0;0 1];
    % invariant in each state
    states{1}.C = [-1 0];
    states{1}.d = [0];
    q=[1]; % discrete states
    qa=1; % initial mode
    % definition of transitions -------------------------------------------
    % guards d.x=e for each transition
    transitions{1}.eguards_dir = [1 0];
    transitions{1}.eguards_val = [0];
    % Inequalities guards d.x<=e for each transition
    transitions{1}.iguards_dir = [0 1];
    transitions{1}.iguards_val = [0];
    % Reset
    transitions{1}.ResetMatrix = [1,0; 0,-0.6];
    transitions{1}.W = [0;0];
    % start and end location of  transition nbr 1
    transitions{1}.from = 1;
    transitions{1}.to = 1;
    % initial values ------------------------------------------------------
    % initial set (as polyheder)
    I.polyheder.C = vectorgenerator(2) ;
    I.polyheder.d = [2;-2;0;0];
    % input set (as zonotope)
    U.zonotope.c=[0;0];
    U.zonotope.g=0.001*eye(2,2);
    % input set (as box)
    U.box = [0;0];
    configuration.timehorizon = 0.8; % time horizon
    configuration.timestep = 0.01; % timestep
```

Figure 4.30.: Textual input file of the bouncing ball example.

Cross algorithm for the support function computation and the *fast-intersection* for both equality/inequality-guard intersections for all benchmarks. Other parameter settings with the computation time and the obtained interval enclosures of the reachable sets using HyReach are recorded in Table 4.2 for each benchmark. The resulting reachable sets of the cruise control, the transient in flower, the two-tank benchmarks and all navigation benchmarks described in [61] are shown for illustration in Figure 4.31. The scalable H_2-based platoon proposed in [75] for 5, 10, 15 and 20 vehicles corresponding respectively to systems of dimension 15, 30, 45 and 60 have been used as a testing benchmark. We note that we were able to get the intended reachable sets for all tested benchmarks. Furthermore, that was also possible with different scenario and option combinations.

In a further study, we carried out a comparative investigation of the impact of different parameter setting choices and scenarios combinations on the efficiency of the computation and the tightness of the reachable sets. With the bouncing ball of Figure 4.29 as example, we underline the importance of the direction template choice. We set the *SpaceEx* scenario with the *SpaceEx* initialization option ignoring thereby the presence of the invariant. The resulting reachable sets are shown in Figure 4.32.

We opt for the *Box*, the *Oct* and the *User pi/6* and *pi/10* options to carry out this comparison analysis. We note, as expected, an accuracy improvement of the approximation with an increasing number of support directions. This demands nevertheless more computation effort. In fact, we remark an increase in the computation time around 1.89s for each new direction. However, it is hard to distinguish a noticeable difference in the tightness of the flowpipes between the Figure 4.32.(c) and Figure 4.32(d), in contrast to the large computation time difference between these both choices. Consequently, a trade-off between tightness and efficiency must be found for the direction choice. Next, we tested different proposed flowpipe scenarios on the two-tank example. We begin with the example of Figure 4.22(b) without invariants. For comparison purposes, both scenarios for handling invariants in modes, *AlgoInv* and *AlgoInv2*, are involved in this survey because they can also treat such systems. We are interested in the computation time and the accuracy of the over-approximation of the flowpipe. We address first the *SpaceEx* Scenario and compare the results of both initial set options: the *SpaceEx* and the *PreciceOmega0*. Figure 4.33 shows practically no difference in the tightness of the resulting flowpipes. However, we note that the *SpaceEx* option is faster than the *PreciseOmega0* option for this example and in general for the suite of linear benchmarks in [61]. In a second step, we compared the scenarios *Noscale*, *ConstU* with both initial set options *ConstU* and *SpaceEx*. Figure 4.34 shows a small difference in the tightness of the over-approximation of the obtained reachable sets between the *NoScale* scenario and those computed with both remaining scenarios. The flowpipes issued from the last two scenarios are practically similar.

The computation time for the two-tank benchmark obtained with all possible flowpipe scenarios and available initial set options are summarized in Table 4.3. The *ConstU* and the *NoScale* scenarios are shown to be faster. However, the cor-

Benchmark	T(s)	r (s)	Max. trans.	Flowp. constr.	Init. approx.	Dir.	Init./Inp. set	Trans. picker	Time (s)	Intervals
1. BB Bouncing ball	1	0.01	4	AlgoInv	SpaceEx	Oct	x1=2,x2=0	First	33.499	x1:[0;2] x2:[-6.289;3.762]
2. CM: Colliding masses	1	0.01	4	AlgoInv	SpaceEx	Oct	a=0,b=3, c=2,d=-1	First	64.417	a:[0;2], b:[2;3] c:[-2;2], d:[-1;1]
3. CC: Cruise control	20	0.1	10	AlgoInv	SpaceEx	Oct	v=30,x=0, t=0	First	116.419	v:[2.587;30] x:[0;127.988] t:[0;2.5]
4. Flower: transient in flower	20	0.05	10	AlgoInv2	SpaceEx	Oct	-2.5<=x1 <=-1.5, x2=0	Most	19.606	x1:[-2.694;2.422] x2:[-2.224;2.637]
5. TT: 2-tanks	2	0.01	4	SpaceEx	SpaceEx	Oct	1.5<=x1 <=2.5, x2=1/ -0.1<=u <=0.1	All	119.214	x1:[-1.366;2.5] x2:[-5.229;2.036]
6. Nav3x3	20	0.05	4	AlgoInv2	SpaceEx	Oct	xx=0.5, yy=1.5 -0.01<=vx, vy<=0.01	Last	179.783	xx:[0.5;2.031] yy:[0.942;1.5] vx:[-0.010;0.749] vy:[-0.527;0.010]
7. Nav4x4	20	0.05	8	AlgoInv	SpaceEx	Oct	xx=0.5, yy=1.5 -0.01<=vx, vy<=0.01	Last	629.664	xx:[0.5;3.771] yy:[0.318;2.048] vx:[-0.010;0.919] vy:[-0.526;0.948]
8. Nav5x5	20	0.05	10	AlgoInv	SpaceEx	Oct	3.3<=xx, yy<=3.4 -0.01<=vx, vy<=0.01	Last	557.346	xx:[0.945;3.4] yy:[1.0;4.446] vx:[-0.882;0.299] vy:[-0.969;0.569]
9. 3R-2MH: 3 rooms+ 2 movable heaters	2	0.01	8	AlgoInv2	SpaceEx	Oct	x1=x2= x3=20	First	446.036	x1:[12.275;20] x2:[13.997;20.536] x3:[12.664;21.270]
10. 5D-LSS	2	0.01	10	ConstU	ConstU	Oct	a=3,b=4, c=d=e=0 / -0.01<=u <=0.01	First	87.160	a:[-4.185;4.538] b:[-1.163;6.824] c:[-4.018;1.129] d:[-0.681;8.308] e:[-0.791;5.421]
11. 3V-P: 3-vehicle-platoon	22	0.1	3	ConstU	ConstU	Oct	a=b=c= d=e=f= g=h=i=0	First (Fixpoint Tol.: 0.001 check every r)	378.133	a:[-28.540;4.479] b:[-8.171;8.184] c:[-15.803;9.163] d:[-25.634;6.274] e:[-10.740;10.919] f:[-23.285;12.267] g:[-11.061;13.316] h:[-9.252;8.729] i:[-18.304;10.517]
12. 5V-P: 5-vehicle-platoon	15	0.1	1	ConstU	ConstU	Oct	d1=d2= d3=d4= d5=d6= d7=d8= d9=d10= d11=d12= d13=d14= d15=0	First (irrelevant)	52.911	d1:[-31.438;3.958] d2:[-7.369;8.219] d3:[-10.450;2.830] d4:[-15.228;2.109] d5:[-3.516;3.602] d6:[-10.994;3.262] d7:[-9.690;1.411] d8:[-2.141;2.206] d9:[-11.288;3.533] d10:[-5.927;0.890] d11:[-1.275;1.321] d12:[-11.459;3.700] d13:[-2.836;0.433] d14:[-0.600;0.625] d15:[-11.539;3.779]

Table 4.2.: Setting and results of the testing benchmark list.

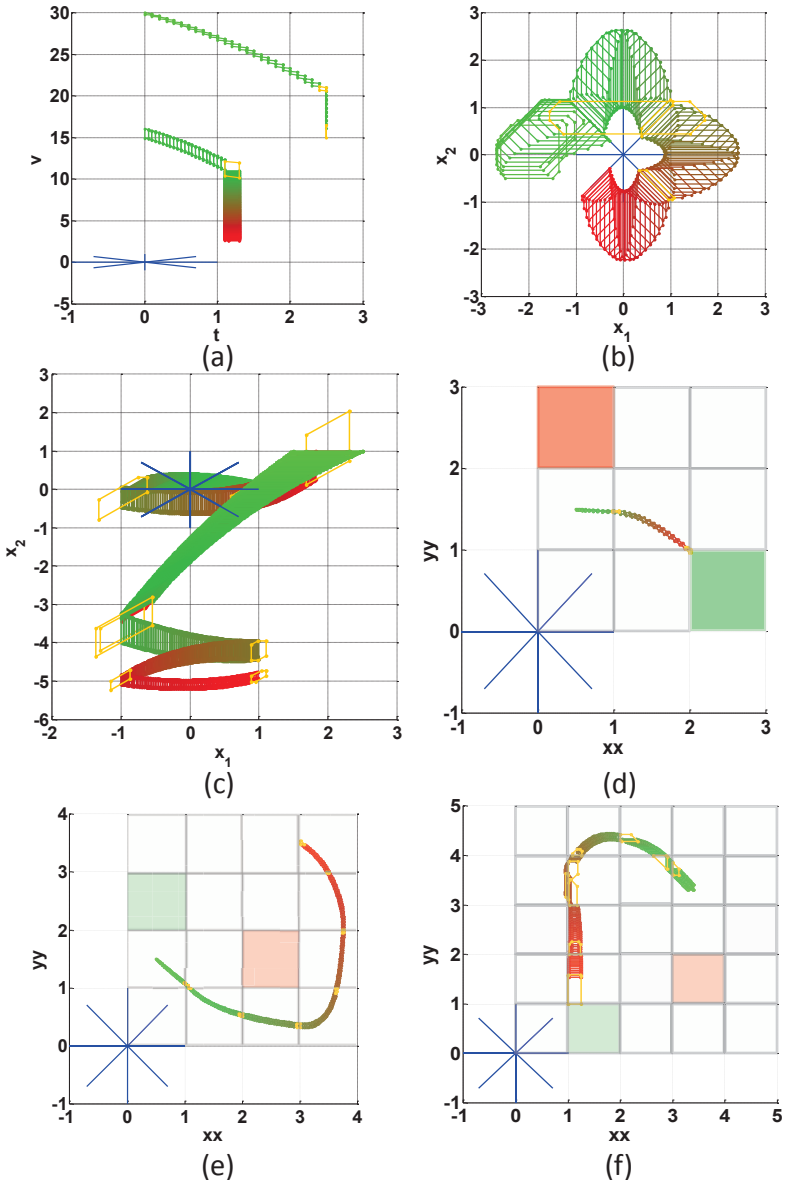


Figure 4.31.: The resulting flowpipes of some selected benchmarks. (a) The flowpipe of the cruise control as projected on the plane (t,v). (b) The flowpipe of the trajectory in flower benchmark. (c) The flowpipe of the two-tank benchmark. (d) The flowpipe of the navigation 3x3 benchmark as projected on the plane (xx,yy). (e) The flowpipe of the navigation 4x4 benchmark as projected on the plane (xx,yy). (d) The flowpipe of the navigation 5x5 benchmark as projected on the plane (xx,yy).

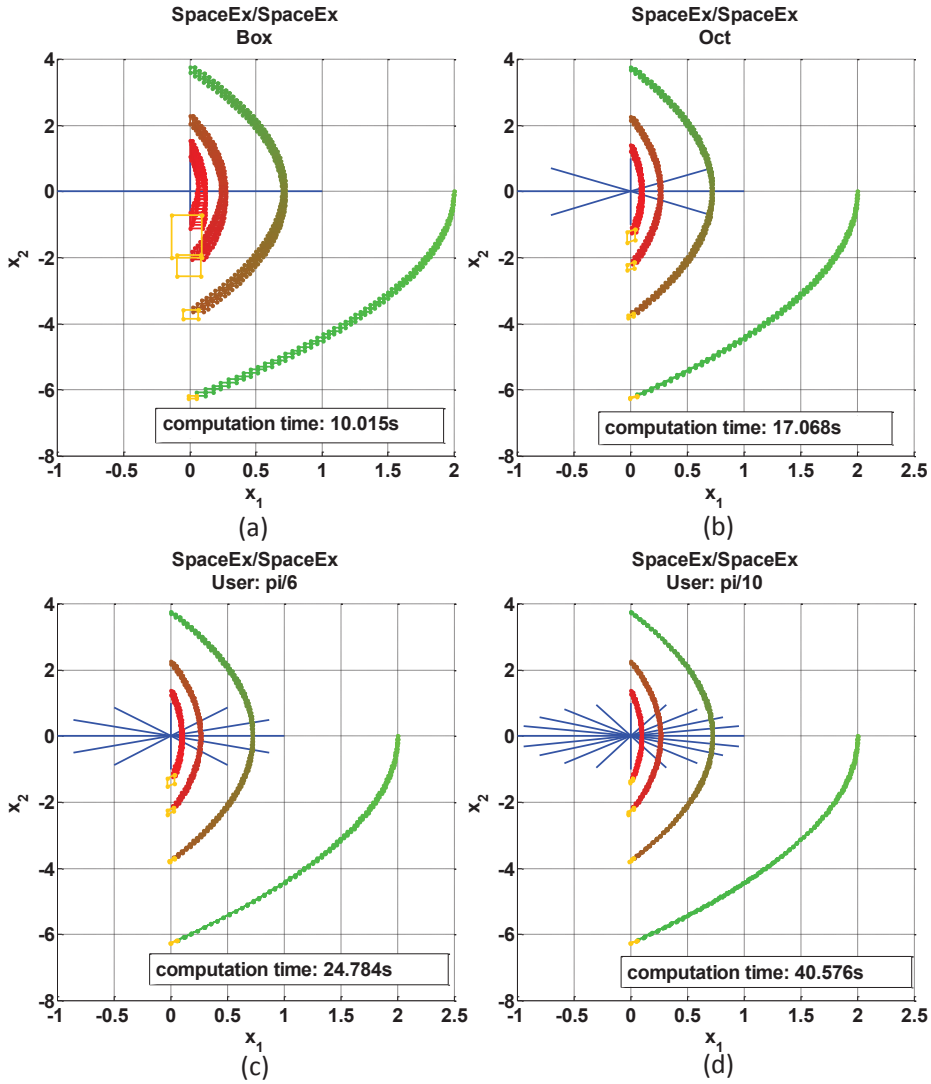


Figure 4.32.: Illustration of the impact of the choice of the direction template on the performance of the analysis using the bouncing ball example.

Scenario	SpaceEx		ConstU			NoScale	AlgoInv		AlgoInv2	
Init. set Approx.	SpaceEx	Precise Omega0	SpaceEx	Precise Omega0	ConstU	NoScale	SpaceEx	Precise Omega0	SpaceEx	Precise Omega0
Time (s)	36.957	40.141	36.3664	38.364	6.297	6.354	74.876	76.944	82.852	105.831

Table 4.3.: Computation times for different scenarios and different allowed initial set over-approximations for the two-tank example with an initial set $1 \leqslant x_1 \leqslant 1.8$, $x_2 = 1$, an input $u = 0$, a time horizon $T = 2s$, a time step $r = 0.01$, max. transitions equal to 3 and the *oct* as direction choice.

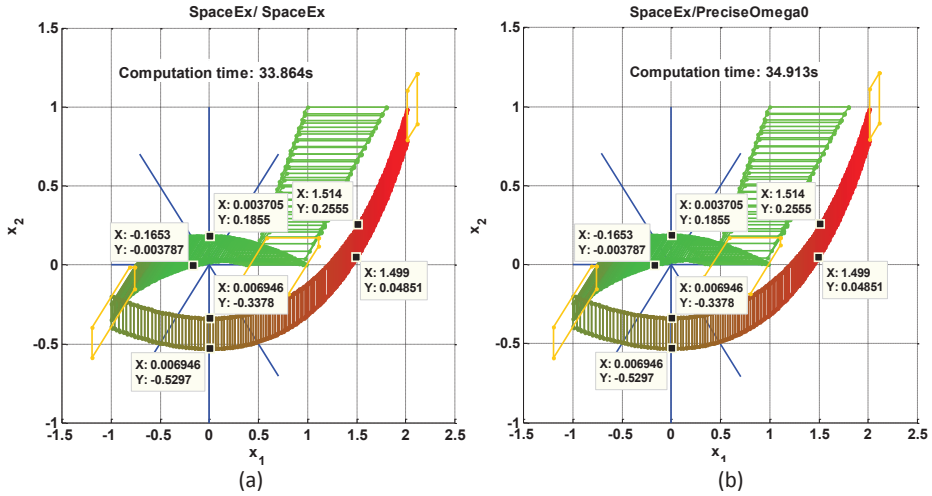


Figure 4.33.: Reachable sets of the two-tank benchmark obtained with an initial set $1 \leqslant x_1 \leqslant 1.8$, $x_2 = 1$, an input $u = 0$, a time horizon $T = 2s$, a time step $r = 0.01$, max. transitions equal to 3 and *oct* as direction choice. (a) Flowpipe construction with the *SpaceEx* scenario and the *SpaceEx* initial set over-approximation. (b) Flowpipe construction with the *SpaceEx* scenario and the *PreciseOmega0* initial set over-approximation.

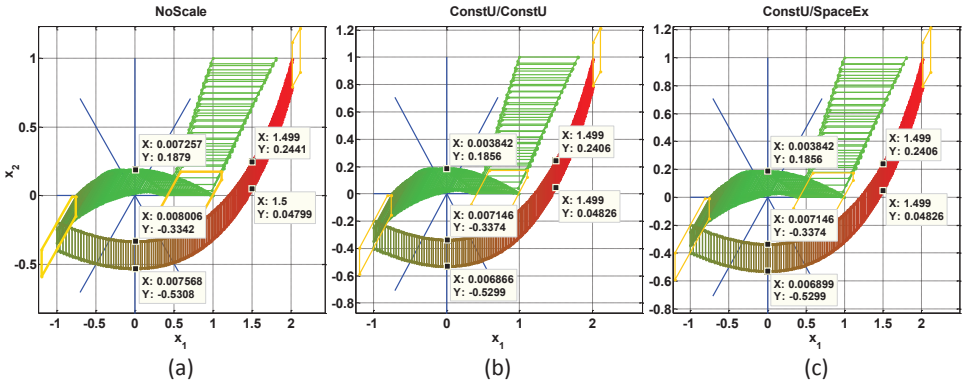


Figure 4.34.: Reachable sets of the two-tank benchmark obtained with an initial set $1 \leqslant x_1 \leqslant 1.8$, $x_2 = 1$, an input $u = 0$, a time horizon $T = 2s$, a time step $r = 0.01$, max. transitions equal to 3 and *oct* as direction choice. (a) Flowpipe construction with the *NoScale* scenario. (b) Flowpipe construction with the *ConstU* scenario and the *ConstU* initial set over-approximation. (c) Flowpipe construction with the *ConstU* scenario and the *SpaceEX* initial set over-approximation.

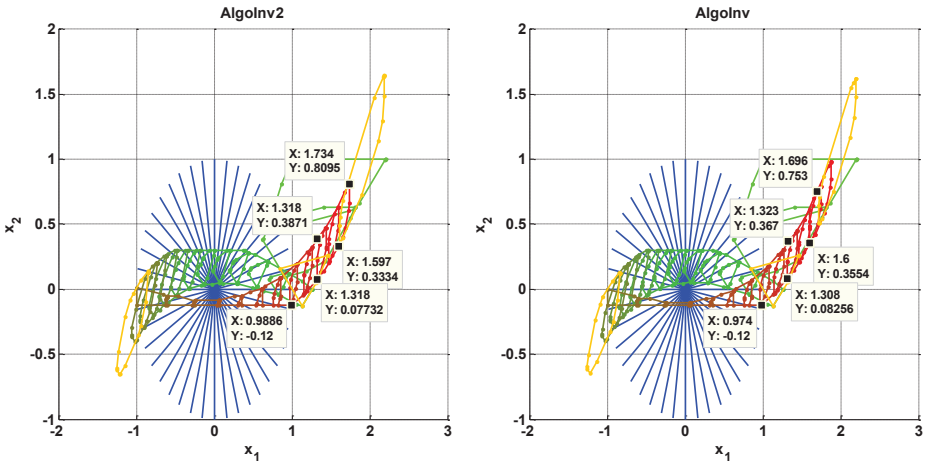


Figure 4.35.: Reachable sets of the two-tank benchmark obtained with an initial set $1 \leqslant x_1 \leqslant 2.2$, $x_2 = 1$, an input $-0.1 \leqslant u \leqslant 0.1$, a time horizon $T = 2s$, a time step $r = 0.1$, max. transitions equal to 6, *pi/20* as direction choice. (a) Flowpipe construction with the *AlgoInv* scenario and the *SpaceEx* initial set over-approximation. (b) Flowpipe construction with the *AlgoInv2* scenario and the *SpaceEx* initial set over-approximation.

responding flowpipes over-approximations are wider than those obtained with the other scenarios, although the difference is not considerable. It is recommended to avoid the choice of *AlgoInv* or the *AlgoInv2* scenarios for systems without invariants.

We compared both scenarios dedicated for hybrid automaton with invariants within continuous modes using the two-tank example of Figure 4.22(b) with an initial set $1 \leqslant x_1 \leqslant 2.2$, $x_2 = 1$, an input in $-0.1 \leqslant u \leqslant 0.1$ and two different choices of the direction template: *oct* and *pi/20*. We note for the first direction choice that the *AlgoInv* scenario is faster than the *AlgoInv2* scenario. For this reason, it was expected that for an increasing number of directions, this tendency would be significantly more noticeable. However, because of our parallel implementation of the AlgoInv2 algorithm, which was not possible for the AlgoInv algorithm, we observed the contrary during our tests. The *AlgoInv2* is shown in Table 4.4 to be faster for large templates of directions.

Concerning the tightness of the flowpipe, Figure 4.35 shows small differences between both scenarios.

Next, we tested different proposed intersection methods for equality guards first using the two-tank benchmark without invariants beginning with an initial of to $1 \leqslant x_1 \leqslant 1.8$, $x_2 = 1$ for an input $-0.1 \leqslant u \leqslant 0.1$, and with max. transitions of 2 and 4 subsequently. We note no difference in the tightness of the resulting reachable sets as well as practically similar computation times. We repeated, the same test in a second step with the transient in flower benchmark presented in [36]. We

Flowpipe	AlgoInv	AlgoInv2	AlgoInv	AlgoInv2
Directions	*oct*	*oct*	*pi/20*	*pi/20*
Time (s)	23.712	26.014	57.079	35.143
x_1	[-1.262802;2.200000]	[-1.255453;2.200000]	[-1.262802;2.200000]	[-1.255453;2.200000]
x_2	[-0.648818;1.632380]	[-0.655425;1.651646]	[-0.645434;1.621805]	[-0.652982;1.644013]

Table 4.4.: Time complexity and interval hull of the two-tank benchmark obtained with an initial set $1 \leqslant x_1 \leqslant 2.2$, $x_2 = 1$, an input $-0.1 \leqslant u \leqslant 0.1$, a time horizon $T = 2s$, a time step $r = 0.1$, max. transitions equal to 6, different direction choices and available algorithms for handling invariants with the *SpaceEx* option for the initial set.

chose the following initial conditions $-2.5 \leqslant x_1 \leqslant 1.5 \leqslant \wedge x_2 = 0$, a time horizon $T = 2s$ and a time step $r = 0.01s$. We set the *SpaceEx* scenario with the *SpaceEx* initial set option and used the *CDD Criss-Cross* algorithm of the MPT-toolbox for computing the support function. We also selected the option *Most intersections* for handling transitions and the *Oct* option for the directions. Furthermore, we set the inequality guards to fast intersection, although irrelevant here. We performed the same test with the different available options for handling equality guards with a maximum of one transition 1 and subsequently increased to four. We remark that all proposed equality guards intersection methods return practically the same results for the reachable sets and almost the same computation times. We note a duration of the computation of about 10s for one transition and around 48s for four transitions and practically the same resulting flowpipes. In a next step, we set a maximum number of eight transitions and chose the direction option *User: pi/11*. We notice in Table 4.5 a remarkable difference in the computation time between different options in favor of the *fast intersection* option. With regards to the the tightness of the reachable sets, no difference could be distinguished between the different methods.

Eq. guards methods	fast-int	dic. search	RayAlgo	fminunc	fminsearch
Time (s)	295.058	333.610	321.655	344.344	350.772

Table 4.5.: Computation time and of the transition in flower benchmark with the initial condition $-2.5 \leqslant x_1 \leqslant 1.5 \leqslant \wedge x_2 = 0$, a time horizon $T = 2s$ and a time step $r = 0,01s$, max. transitions equal to 8, $pi/11$ for the direction choice, the *SpaceEx* scenario with the *SpaceEx* initial set option, the *CDD Criss-Cross* algorithm and different proposed algorithms for handling equality guards.

4.14. Conclusion

This chapter treated the reachability problem of linear time invariant hybrid systems with uncertain inputs using support functions for presenting reachable sets.

We gave an overview of the definition and imported properties of support functions which were used to derive different algorithms for the computation of the flowpipe inside discrete modes with and without invariants. We also discussed the different options for the computation of an over-approximation of the initial set and of the input contribution. In addition, a variety of methods for checking and computing the intersection of the flowpipe with guards described as hyperplanes or halfspaces are described in detail. These scenarios and intersection methods were implemented within our prototyping MATLAB toolbox HyReach to allow for a user-configurable reachability analysis. To pave the way towards this goal, HyReach offers a GUI to facilitate the parameter setting and the combination of different options. Different optimization algorithms from the MATLAB optimization toolbox, the MPT and the CVX toolboxes are furthermore made available for the computation of the support function. In addition, the user can choose between different transition picking strategies if multiple transitions are triggered within the chosen time horizon. Results are provided in form of 2D-plots and interval ranges.

We present some experiences with HyReach and a comparative evaluations of the available methods and algorithms using a suite of benchmarks. We note, in general, that the performances of these different methods and algorithms are similar for small dimensional benchmarks with small number of transitions and small number of directions. Otherwise, these methods and algorithms were shown to perform differently depending on the dimension and the dynamics of the tested benchmark.

5. Zonotopic Approximation for Computing Reachable Sets

5.1. Introduction

Looking back over the last ten years, the review of different reachability approaches in relation with the geometric set choice has revealed that zonotopes, apart from support functions, provide the best trade-off between efficiency and tightness. It was furthermore found that approaches based on those sets allow up to 100-dimensional systems to be treated within an acceptable time [47, 71]. We therefore dedicate this chapter, to review, improve and combine various zonotopic techniques and approaches originating from different domains, like reachability analysis and control design. We aim to implement these different methods in the same framework for a user-configurable reachability analysis.

Zonotopes are closed under linear and Minkowski sum, properties that make them computationally attractive. But operations like intersection, union, often required in handling of invariants oder guards, lead in general to the loss of the original zonotopic form. This hence demands for appropriate techniques to find a tight zonotopic approximation for the resulting set. Apart from the complexity of reachable set computation inside continuous modes, the handling of intersections of reachable sets with guard conditions is also a challenge. While the use of zonotopes to represent, on one hand, reachable sets has been proven to be efficient [48], an intersection with a guard leads to a loss of the zonotopic form and, on the other hand, requires particular techniques to find the intersection.

The problem of computing an intersection of a hyperplane guard with a zonotope has been solved in [48] by transforming the multidimensional intersection problem into a series of two-dimensional problems applying projections. A zonotope/polytope transformation based solution was furthermore suggested in [5] to handle polytopic guards. An intersection with a tight parallelotopic over-approximation of the zonotope was computed using the MPT-toolbox, which was thereafter over-approximated using parallelotopes.

In the field of control theory, zonotopes have been proposed as a solution for the computational complexity problem of set-membership estimation methods. Algorithms for zonotope/strip intersection based on the minimization of the segments or the volume of the resulting zonotope were suggested in [1]. This approach was then elaborated in [69, 106] to handle zonotope/polytope intersection. In addition to these methods, Singular Value Decomposition SVD-based zonotope/hyperplane and zonotope/zonotope intersection techniques were also proposed in [67].

This chapter describes a variety of methods and approaches for computing zonotope/hyperplane, zonotope/halfspace and zonotope/polyhedron intersections besides the classical operations with zonotopes. We also propose and investigate various strategies for handling the bundle of zonotopes intersecting the guard. A performance comparison study of those methods is furthermore conducted in the context of guard intersections and handling transitions separately and in combination.

5.2. Zonotopes

A zonotope is generally defined as the image of a unit hypercube under affine (but not necessary invertible) transformation. A zonotope is a centrally symmetric convex polytope. Each facet of a zonotope corresponds to a congruent facet on the other side. Equivalently, a zonotope can also be defined as the Minkowski sum of a finite set of segments. The directions of these segments are given by a list of vectors called the generators of the zonotope.

Definition 9. A **zonotope** Z of order $r = p/n$ is defined by its center $c \in \mathbb{R}^n$ and its generators $g_1, \ldots, g_p \in \mathbb{R}^n$

$$\begin{aligned} Z &= (c, \langle g_1, \ldots, g_p \rangle) \\ &= \{x \in \mathbb{R}^n \mid x = c + \textstyle\sum_{i=1}^{p} s_i g_i,\ -1 \leqslant s_i \leqslant 1\} \end{aligned} \tag{5.1}$$

According to this definition, the maximum number of vertices V corresponds to the extreme points of an n-dimensional zonotope given by p generators ($p \geqslant n$) with $V \leqslant 2^p$. However, the exact upper bounds for the number of vertices V and the number of facets F of such a zonotope are shown [38] to satisfy

$$V \leqslant 2 \sum_{i=0}^{n-1} \binom{p-1}{i} \tag{5.2}$$

$$F \leqslant 2\binom{p}{n-1} \tag{5.3}$$

The upper bounds for any fixed n is for F of order $O(p^{n-1})$ and for V of order $O((p-1)^{n-1})$. Consequently the number of extreme points and the number of facets of a zonotope are polynomially bounded. The vertices describe the V-representation whereas the facets define the H-representation of a zonotope. Therefore the computation challenge is to find efficient polynomial algorithms and to choose efficiently between the V- and the H-representation of zonotopes during the implementation.

Alternatively, an equivalent definition of zonotopes involving matrix is frequently used.

Definition 10. Let $G \in \mathbb{R}^{n \times p}$ be the matrix given by the generators of the zonotope $g_1, \ldots, g_p \in \mathbb{R}^n$ as columns.

$$G = \begin{pmatrix} g_{1,1} & \cdots & g_{1,n} \\ \vdots & & \vdots \\ g_{n,1} & \cdots & g_{n,n} \end{pmatrix} \tag{5.4}$$

A **zonotope** Z of order p/n is then defined by

$$\begin{aligned} Z &= c \oplus GB^p \\ &= \{c + Gs : s \in B^p \mid B = [-1, 1]\} \end{aligned} \tag{5.5}$$

For implementation issues, it is practical to represent a zonotope as a matrix $\mathbf{Z}$ where c, g_i are their columns.

$$\mathbf{Z} = \begin{pmatrix} c_1 & \\ \vdots & G \\ c_n & \end{pmatrix} \tag{5.6}$$

This representation simplifies the manipulation of zonotopes and allows for an efficient implementation particularly under MATLAB.

5.3. Properties and Geometric Operations of Zonotopes

Zonotopes possess several properties that make them particularly attractive for use as approximating set in reachability analysis. They are closed under linear transformation, Minkowski sum and difference. This means in practice, that the testing for intersection between zonotopes can otherwise be described as testing for point inclusion in their Minkowski difference [54] as seen in the next sections. Furthermore, their implicit representation as a list (or a matrix) of vectors facilitates the implementation of such operations.

Linear image

Let $\mathbf{L}$ be a linear transformation and $Z = (c, \langle g_1, \ldots, g_p \rangle)$ a zonotope. The linear image of a zonotope is given as follows

$$\mathbf{L}Z = (\mathbf{L}c, < \mathbf{L}g_1, \ldots, \mathbf{L}g_p >) .$$

Minkowski sum and convex hull

Let $Z_1 = (c_1, \langle g_1, \ldots, g_p \rangle)$ and $Z_2 = (c_2, \langle h_1, \ldots, h_p \rangle)$ be two zonotopes. The Minkowski sum of two zonotopes is given by

$$Z_1 \oplus Z_2 = (c_1 + c_2, \langle g_1, \ldots, g_p, h_1, \ldots, h_p \rangle) .$$

The convex hull $CH(Z_1 \cup Z_2)$ can be over-approximated with the following zonotope Z_{CH}

$$Z_{CH} = \frac{1}{2} \begin{cases} (c_1 + c_2, \langle g_1 + h_1, \ldots, g_p + h_p, c_1 - c_2, \\ \qquad g_1 - h_1, \ldots, g_p - h_p \rangle) & \text{if } p = q \\ (c_1 + c_2, \langle g_1 + h_1, \ldots, g_q + h_q, c_1 - c_2, \\ \quad g_1 - h_1, \ldots, g_q - h_q, 2g_{q+1}, \ldots, 2g_p \rangle) & \text{if } p > q \\ (c_1 + c_2, \langle g_1 + h_1, \ldots, g_p + h_p, c_2 - c_1, \\ \quad h_1 - g_1, \ldots, h_p - g_p, 2h_{p+1}, \ldots, 2h_q \rangle) & \text{if } p < q. \end{cases} \tag{5.7}$$

Convex hull of Z and $e^{rA}Z$

In general, the convex hull of two zonotopes is not a zonotope. In [47] an approximation of convex hull of a zonotope Z and its image with the linear transformation e^{rA} is proposed.

$$CH(Z, e^{rA}Z) \subseteq P \tag{5.8}$$

where

$$P = (\tfrac{c+e^{rA}c}{2}, < \tfrac{g_1+e^{rA}g_1}{2}, \ldots, \tfrac{g_p+e^{rA}g_p}{2}, \tfrac{c-e^{rA}c}{2}, \tfrac{g_1-e^{rA}g_1}{2}, \ldots, \tfrac{g_p-e^{rA}g_p}{2} >) \tag{5.9}$$

These operations involve a concatenation of the generators of both zonotopes, which lead to a significant increase in the number of generators of the over-approximating zonotope. To have some control on the complexity, a global maximal order for zonotopes is often fixed in advance in combination with an order reduction operation.

Order reduction operation

Many approaches have been proposed to reduce the order of zonotopes. In general, a zonotope $O = (c, \langle g_1, \ldots, g_{rn} \rangle)$ of order r can be reduced to a zonotope of order q by replacing the $n(r - q + 1)$ less significant generators with their interval hull. To decide about the choice of generators, different sorting methods can be used. We make use of the method proposed in [47] which reduces the order by one by applying the following hard sorting criterion.
Let $O = \left(c, \left\langle g_1, \ldots, g_{(q+1)n} \right\rangle\right)$ be a zonotope satisfying

$$\|g_1\|_1 - \|g_1\|_\infty \leqslant \ldots \leqslant \left\|g_{(q+1)n}\right\|_1 - \left\|g_{(q+1)n}\right\|_\infty .$$

We construct the zonotope $\widehat{O} = \left(c, \left\langle Q, g_{2n+1}, \ldots, g_{(q+1)n} \right\rangle\right)$ where $Q \in \mathbb{R}^{n \times n}$ is the diagonal matrix that satisfies

$$Q_{ii} = \textstyle\sum_{j=1}^{2n} \left|g_j^i\right|, \; i = 1, \ldots, n$$

g_j^i is the i^th component of g_j. The obtained zonotope has hence the order q and satisfies $O \subseteq \widehat{O}$.

The next sections describe various approaches for intersection operations of zonotopes with different geometric shapes. We devote Section 5.4 to outline techniques for detecting and computing the intersection of a zonotope with a hyperplane. Section 5.5 describes zonotope/halfspace detection and intersection methods whereas Section 5.6 presents an extension of these methods to polyhedron. For the zonotope/zonotope intersection, various detection and over-approximating approaches are discussed in Section 5.7.

5.4. Zonotope/Hyperplane Intersection

The intersection of a zonotope with a hyperplane is in general not a zonotope. The computation of the intersection with a zonotope is found to be a difficult problem. Different methods have been proposed to solve this problem. A trivial method considers a zonotope as an ordinary polytope and uses existing methods to obtain a polytope as a resulting intersection and subsequently compute a zonotopic approximation of the result. The difficulty thereby lies in the complexity of the transformation of a zonotope into a polytope and vice-versa. In [5], different methods for the transformation of a zonotope from its generator representation to the halfspace representation as well as over-approximation techniques of polytopes by zonotopes are proposed and numerically evaluated. In [49], however, a dichotomous based algorithm for the computation of an intersection of a zonotope in its generator representation with a hyperplane was suggested. The idea behind this method is to transform the intersection problem from higher to a two dimension space so that several intersections of a 2-dimensional zonotope with a plane can be computed. Moreover in [1], a method was proposed to compute a family of zonotopes parameterized by a vector λ including the intersection of a zonotope and a strip. A hyperplane can be considered as a particular strip. Two optimization approaches were also proposed to minimize the size criterion of the resulting zonotope. The first approach consists of minimizing the segments of the zonotope whereas the second one minimizes the volume of the intersection. In addition, [67] proposed algorithms for the computation of a zonotope over-approximation of the intersection of a zonotope with a hyperplane using singular value decomposition (SVD) technique.
We will focus on methods directly using the generator representation. Methods making use of polytopes are not considered here. In this section, we describe the methods regarded in [49], [1], [67] in detail. We suggest implementations that improve on efficiency and propose detailed algorithms for the final implementation. Furthermore, inspired from these methods, we develop new approaches and algorithms for handling the problem of zonotope/hyperplane intersection.

5.4.1. Intersection Check Between a Zonotope and a Hyperplane

For the computation of the intersection, we first have to know when the transition is triggered. This demands a check for collision between the reachable sets and the guard expressed in a form of a hyperplane. A check of no-emptiness of the

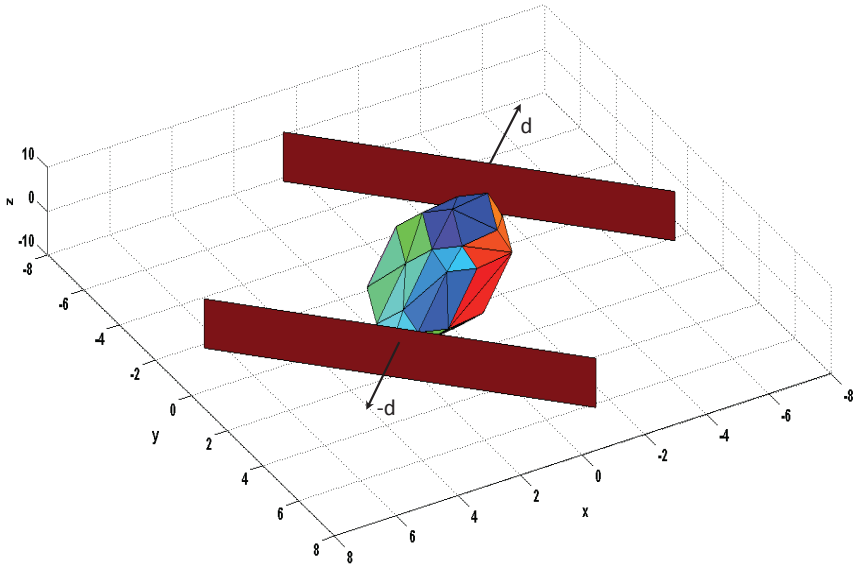


Figure 5.1.: Support hyperplanes of a zonotope in direction d and $-d$.

intersection of a zonotopic reachable set $Z = (c, \langle g_1, \ldots, g_p \rangle)$ and a guard defined as $Hp = \{x \in \mathbb{R}^n : d^T.x = e\}$ is done based on following condition:

$$(Z \cap Hp \neq \varnothing) \Leftrightarrow \left(\exists \alpha = (\alpha_1, \ldots, \alpha_p) \in B^p \mid d^T.c + \textstyle\sum_p^{i=1} d^T.g_i.\alpha_i = e\right). \quad (5.10)$$

This consequently leads to the following detection collision condition between a zonotope and a hyperplane

$$(Z \cap Hp \neq \varnothing) \Leftrightarrow \left(\left(e - d^T.c\right) \in \left[-\textstyle\sum_p^{i=1} \left|d^T.g_i\right|, \sum_p^{i=1} \left|d^T.g_i\right|\right]\right). \quad (5.11)$$

This condition, as formulated in [49], can be retrieved using support hyperplanes. As illustrated in Figure 5.1, the zonotope Z is confined between two supporting hyperplanes. The first corresponds to the support function in direction d and the second is defined by the support function in the opposite direction $-d$. The support function of the zonotope Z in the direction d is equal to $q_u = d^T.c + \sum_p^{i=1} \left|d^T.g_i\right|$ whereas in the direction $-d$ is given by $q_l = d^T.c - \sum_p^{i=1} \left|d^T.g_i\right|$. Hence, an intersection of the zonotope Z with the hyperplane is possible if and only if

$$q_l \leqslant e \leqslant q_u. \quad (5.12)$$

5.4.2. From Dimension n to Dimension 2

In [49], the problem of computing an over-approximation for the intersection of a zonotope $Z = (c, \langle g_1, \ldots, g_p \rangle)$ with a hyperplane $Hp = \{x \in \mathbb{R}^n : d^T.x = e\}$, $c, g_1, \ldots, g_p, d \in \mathbb{R}^n$ and $e \in \mathbb{R}$, is reduced, according to Algorithm 5.1, to a problem

of computing an intersection of a 2-dimensional zonotope, also called *zonogon*, with a plane. This is illustrated in Figure 5.2 for the hyperplane defined by $x = 0$. For this purpose, a set of orthonormal vectors $D = \{l_1, \ldots, l_q\}$ base of Hp is chosen. For each j, $j \in \{1, \ldots, q\}$, we compute the zonogon $Z_j^{(2)}$ resulting from the projection $\Pi_{(d,l_j)}$

$$\begin{array}{rcl} \Pi_{(d,l_j)} & : & \mathbb{R}^n \to \mathbb{R}^2 \\ & & x \longmapsto (d^T.x; l_j^T.x) = \mathbb{M}_{d,l_j} x \end{array} \tag{5.13}$$

of the zonotope Z on the plane defined by the vector pair (d, l_j):

$$Z_j^{(2)} = \big((d^T.c; l_j^T.c), \big\langle (d^T.g_1; l_j^T.g_1), \ldots, (d^T.g_p; l_j^T.g_p) \big\rangle\big) \tag{5.14}$$

where $^{(2)}$ refers to the dimension two. In the same way, we define the line L, projection of the hyperplane Hp on the same plane:

$$L = \big\{(x, y) \in \mathbb{R}^2 : x = e\big\} \tag{5.15}$$

Algorithm 5.1 Transforming a zonotope in several zonogons by 2D-projections

Input: $Z = (c, G = \langle g_1, \ldots, g_p \rangle)$, $Hp = \big\{x \in \mathbb{R}^n : \ d^T x = e\big\}$
Output: $Z_1^{(2)}, ..., Z_r^{(2)}$
1: $[l_1, l_2, ..., l_r] = null(d^T)$ ▸ orthonormal basis of Hp
2: **for** $j = 1$ **to** r **do**
3: $\quad Z_j^{(2)} = \big((d^T.c; l_j^T.c), \big\langle (d^T.g_1; l_j^T.g_1), \ldots, (d^T.g_p; l_j^T.g_p) \big\rangle\big)$
4: **end for**
5: **return** $Z_1^{(2)}, ..., Z_r^{(2)}$

The next step consists in computing in dimension two and for each $j \in \{1, \ldots, q\}$ a supremum point $(e; M_j)$ and an infinimum point $(e; m_j)$ corresponding both to the intersection of the zonogon Z_j^2 with the straight line $x := e$ (see Figure 5.3). As result, we obtain the polyhedron P defined by

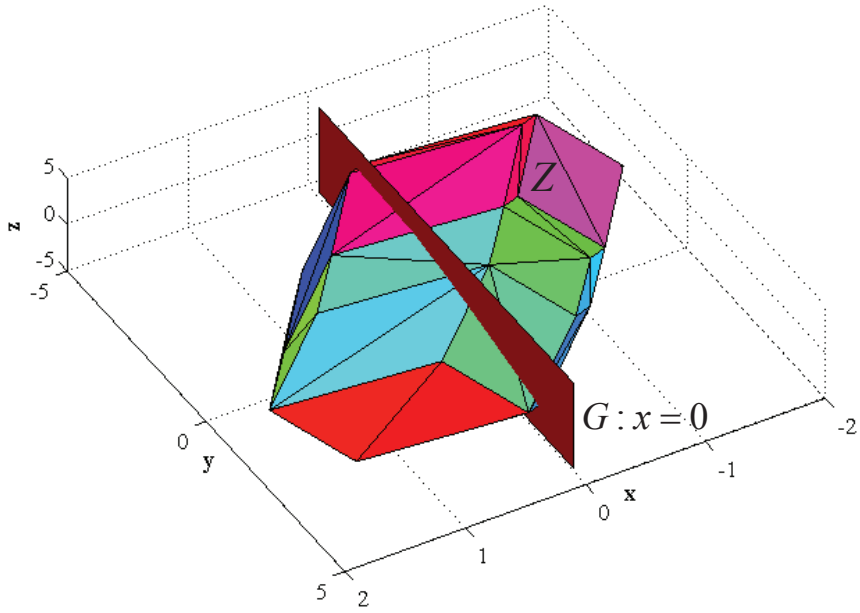
$$P = \big\{x \in \mathbb{R}^n : \ \forall\, j \in \{1, \ldots, q\},\ m_j \leqslant l_j^T.x \leqslant M_j\big\} \tag{5.16}$$

from which a parallelotopic approximation can be computed.

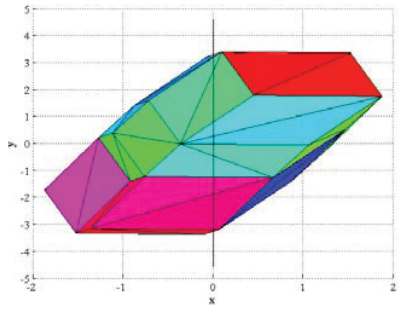
Zonogon/line intersection

A trivial method to compute the intersection of a zonogon with a line is to determine the edges of the zonogon which intersect this line. This involves, in general, two edges of the zonogon. That claims, however, the computation of a list of at most 2^p vertices according to the following equation:

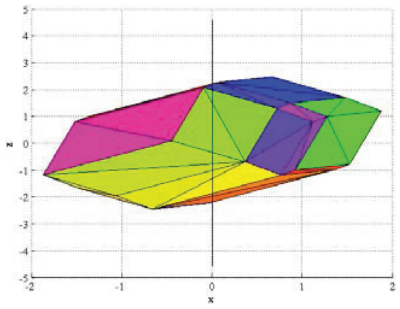
$$v = c + \sum_{i=1}^{p} s_i g_i,\ s_i = \pm 1 \tag{5.17}$$



(a) An example of a zonotope/hyperplane ($x = 0$) intersection in 3D.

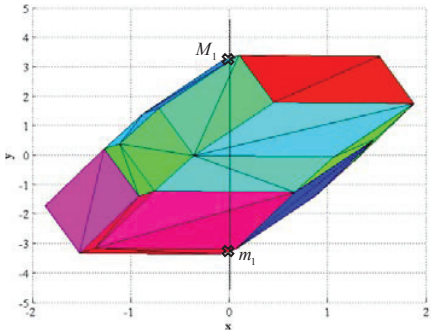


(b) Projection on the plane (x,y) corresponding here to $Z_1^{(2)}$ the xy view of (a).

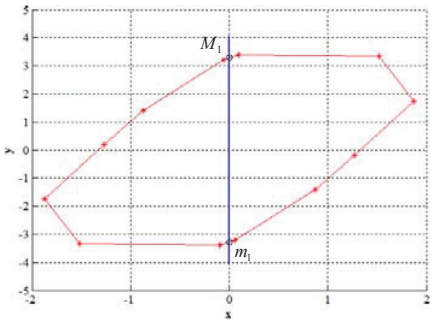


(c) Projection on the plane (x,z) corresponding here to $Z_2^{(2)}$ the xz view of (a) and .

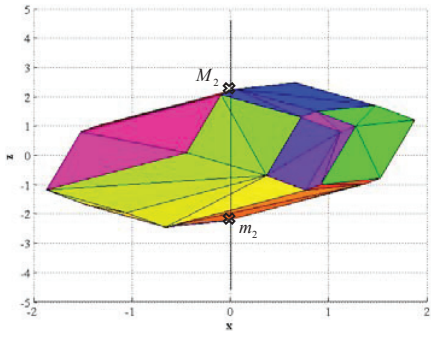
Figure 5.2.: Example of zonotope/hyperplane intersection using the ND-2D transformation.



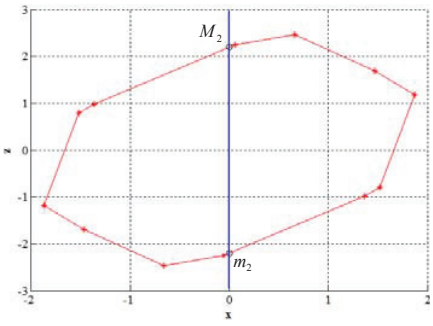
(a) Supremum M_1 and infinimum m_1 of the intersection of the zonogon $Z_1^{(2)}$ with the line $x = 0$.



(b) Computing M_1 and m_1.



(c) Supremum M_2 and infinimum m_2 of the intersection of the zonogon $Z_2^{(2)}$ with the line $x = 0$.



(d) Computing M_2 and m_2.

Figure 5.3.: Computing the supremum and infinimum of the zonotope/hyperplane intersection of the example of Figure 5.2a

However, many of the 2^p combinations are interior points. So that only the $2p$ vertices corresponding to the extreme points of the zonogon must be at the end extracted. Classical algorithms start with a vertex P_0 of the zonogon (say the point with the lowest x-coordinate) and sorts the rest of the points in an angular order. The lowest vertex can be obtained by pointing all generators upwards and subtracting them recursively from the center of the zonogon. The generators are thereby automatically sorted angularly. Taking into account that edges of a zonogon are segments defined by $[P, P + 2g]$, where P is a vertex and g a generator from the sorted generator list, consecutive edges are computed and tested for intersection with the line L. The drawback of this method is that a scan of all vertices and a test of intersection for all edges is inavoidable. We propose in the next sections two different zonogon/line intersection methods. We first begin with our method based on a vertices sort according to their distance and their position to the line L. The second method proposed in [49] uses dichotomous method for the search of edges intersecting the line L.

Sorting the vertices of the zonogon

The idea behind this algorithm is to sort the $2p$ vertices of the zonogon $Z_j^{(2)} = (c, \langle g_1, \ldots, g_p \rangle)$ beginning with the one closer to the line L. The method proposed here is valid for lines defined by $L = \left\{u = (x; y) \in \mathbb{R}^2 : \ d^T.u = e.\right\}$ unlike the dichotomous method, which is restricted to lines parallel to the y-axis.

We define the set $zb = (v_1, v_2, ..., v_{2p})$ of the extreme points v_k of $Z_j^{(2)}$ where $k \in \{1, \ldots, 2p\}$. The sort criterion for the elements of zb is $min(\left|abs(d^T.zb - e)\right|)$. As shown in Figure 5.4, if we sort the elements of zb beginning with the one nearest the line L, the first point of the sorted set will be the first point P_{l1p1} of the first intersecting line l_1. The second point of the same intersecting line P_{l1p2}, is then the nearest point to the line but from the other side (see Algorithm 5.2 lines 9, 12, 19 and 22). For this reason, we divide the set zb in two subsets: the set of points located on the right of the line zb_R and the rest in zb_L. We therefore distinguish between following cases.

- Cases in which the first point and the second point in the sorted array are situated on different sides of the line (Figure 5.4 black points) contribute directly to the determining of the first intersecting line l_1 of the zonogon. The first point P_{l2p1} of l_2, however, will be

$$\text{the last element of: } \begin{cases} zb_R, & \text{if } P_{l1p2} \in zb_R \\ zb_L, & \text{if } P_{l1p2} \in zb_L \end{cases} \tag{5.18}$$

  The second point P_{l2p2} of the second intersecting line l_2 (Figure 5.4 3) corresponding to P_6) is then the next point in the original sorted set.

Algorithm 5.2 Compute zonogon/line intersection points

Input: $Z_j^{(2)} = (c, \langle g_1, \ldots, g_p \rangle)$, $L = \left\{u = (x; y) \in \mathbb{R}^2 : \ d^T.u = e\right\}$
Output: Extreme points of intersecting edges with line L

1: Compute the array zb of the extreme points v of $Z_j^{(2)}$
2: $npts = length(zb)$
3: $dzb = d^T.zb$
4: Sort vertices starting from nearest to the line L
5: **if** $dzb(1) - e < 0$ **then**
6: $\quad il = find(dzb - e < 0)$ ▸ indexes of points in zb on the left of L
7: $\quad nil = length(il)$ ▸ their number
8: $\quad$ **if** $dzb(2) - e < 0$ **then**
9: $\quad\quad il1p2 = npts$ ▸ index of point 2 of line 1
10: $\quad\quad il2p1 = nil$ ▸ index of point 1 of line 2
11: $\quad$ **else**
12: $\quad\quad il1p2 = 2$
13: $\quad\quad il2p1 = npts - nil + 1$ ▸ index of point 1 of line 2
14: $\quad$ **end if**
15: **else**
16: $\quad il = find(dzb - e > 0)$ ▸ indexes of points in zb on the right of
17: $\quad nil = length(il)$ ▸ their number
18: $\quad$ **if** $dzb(2) - e > 0$ **then**
19: $\quad\quad il1p2 = npts$ ▸ index of point 2 of line 1
20: $\quad\quad il2p1 = nil$ ▸ index of point 1 of line 2
21: $\quad$ **else**
22: $\quad\quad il1p2 = 2$
23: $\quad\quad il2p1 = npts - nil + 1$ ▸ index of point 1 of line 2
24: $\quad$ **end if**
25: **end if**
26: **if** $nil = 1$ **then**
27: $\quad il2p2 = 1$ ▸ if only 1 point
28: **else**
29: $\quad il2p2 = il2p1 + 1$ ▸ index of point 2 of line 2
30: **end if**
31: **return** $P_{l1p1} = zb(1:2, 1)$, $P_{l1p2} = zb(1:2, il1p2)$, $P_{l2p1} = zb(1:2, il2p1)$, $P_{l2p2} = zb(1:2, il2p2)$

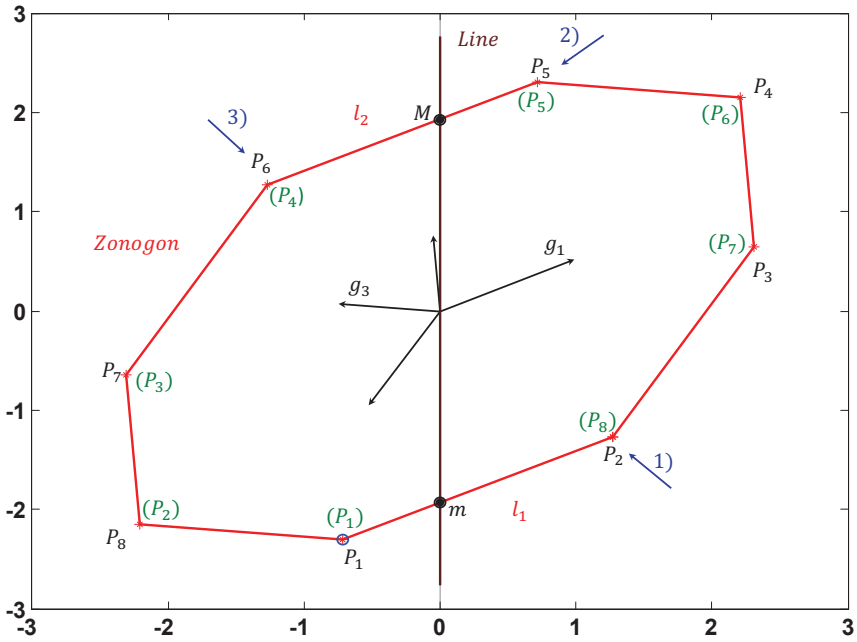


Figure 5.4.: Principle of computing the intersection points according to Algorithm 5.2 adopting a vertex sorting method.

- Otherwise, if the first point and the second point in the sorted array are situated on same side of the line (see Figure 5.4 green points), then P_{l1p2} will be the last point in the sorted set. The points P_{l2p1} and P_{l2p2} defining the second intersecting line are determined in the same way as before.

We propose the Algorithm 5.2 for the implementation of this method. We do not directly manipulate the elements of the set zb, but only their indexes. As a result, we obtain four points $P_{l1p1}, P_{l1p2}, P_{l2p1}, P_{l2p2}$ defining both intersecting lines l_1 and l_2 of the zonogon $Z_j^{(2)}$ with the line L and that for each direction l_j.

Dichotomous search of intersecting edges

The naive way to search for an element in an ordered set is to look at the elements in the order they appear, starting from the first element. However, the efficiency of the search may be enhanced by picking one element of the set (often the middle) as a pivot, and then deciding according to the search criteria which side of the pivot is of interest. The half set on the other side will be then automatically pruned. This procedure is therefore applied on the remainder set recursively. This is called the dichotomous search. In [49], the authors adapted it to search for the edges of the zonogon $Z_j^{(2)}$ intersecting a line L.

The goal is, in fact, to find the vertices of the two edges of the zonogon $Z_j^{(2)}$ inter-

Algorithm 5.3 Compute the minimum zonogon/line intersection point

Input: $Z_j^{(2)} = (c, \langle g_1, \ldots, g_p \rangle)$, $L = \{(x; y) \in \mathbb{R}^2 : \ x = e\}$
Output: m the minimum y-coordinate of the zonogon/line intersection points

1: $P_{min} = c$
2: **for** $i = 1$ **to** p **do**
3: **if** $(y_{g_i} < 0)$ **or** $(y_{g_i} = 0$ **and** $x_{g_i} < 0))$ **then**
4: $g_i = -g_i$ ▸ force generators to point upwards
5: **end if**
6: $P_{min} = P_{min} - g_i$ ▸ lowest vertex
7: **end for**
8: **if** $x_{P_{min}} < e$ **then**
9: $G = \{g_k | x_{g_k} >= 0\}$ ▸ take only vertices on the right of P_{min}
10: **else**
11: $G = \{g_k | x_{g_k} <= 0\}$ ▸ take only vertices on the left of P_{min}
12: **end if**
13: $s = \sum_{g \in G} 2g$ ▸ vector chosen as pivot
14: **while** size(G)> 1 **do**
15: so=null(s^T) ▸ orthonormal to s
16: $G_1 = \{g_k \in G \mid g_k^T.so > 0\}$ ▸ generators above the pivot
17: $G_2 = \{g_k \in G \mid g_k^T.so < 0\}$ ▸ generators under the pivot
18: $s_1 = \sum_{g \in G} 2g$
19: $P_{minnext} = P_{min} + s_1$
20: **if** $(x_{P_{min}} \leqslant e$ **and** $x_{P_{minnext}} \geqslant e)$ **or** $(x_{P_{min}} \geqslant e$ **and** $x_{P_{minnext}} \leqslant e)$ **then**
21: $G = G_2$
22: $s = s_1$
23: **else**
24: $G = G_1$
25: $s = s - s_1$
26: $P_{min} = P_{minnext}$
27: **end if**
28: **end while**
29: $P_{minnext} = P_{min} + s$
30: $m = y_{P_{min}} + (e - x_{P_{min}}) * (y_{P_{minnext}} - y_{P_{min}})/(x_{P_{minnext}} - x_{P_{min}})$
31: **return** m

Algorithm 5.4 Compute the maximum zonogon/line intersection point

Input: $Z_j^{(2)} = (c, \langle g_1, \ldots, g_p \rangle)$, $L = \{(x; y) \in \mathbb{R}^2 : \ x = e\}$
Output: M the maximum y-coordinate of the zonogon/line intersection points

1: $P_{max} = c$
2: **for** $i = 1$ **to** p **do**
3: **if** $(y_{g_i} > 0)$ **or** $(y_{g_i} = 0$ **and** $x_{g_i} > 0))$ **then**
4: $g_i = -g_i$ ▸ generators to point downwards
5: **end if**
6: $P_{max} = P_{max} - g_i$ ▸ highest vertex
7: **end for**
8: **if** $x_{P_{max}} < e$ **then**
9: $G = \{g_k | x_{g_k} >= 0\}$ ▸ take only vertices on the right of P_{max}
10: **else**
11: $G = \{g_k | x_{g_k} <= 0\}$ ▸ take only vertices on the left of P_{max}
12: **end if**
13: $s = \sum_{g \in G} 2g$ ▸ vector chosen as pivot
14: **while** $size(G) > 1$ **do**
15: so=null(s^T) ▸ orthonormal to s
16: $G_1 = \{g_k \in G \mid g_k^T.so > 0\}$ ▸ generators over the pivot
17: $G_2 = \{g_k \in G \mid g_k^T.so < 0\}$ ▸ generators under the pivot
18: $s_1 = \sum_{g \in G} 2g$
19: $P_{maxnext} = P_{max} + s_1$
20: **if** $(x_{P_{max}} \leqslant e$ **and** $x_{P_{maxnext}} \geqslant e)$ **or** $(x_{P_{max}} \geqslant e$ **and** $x_{P_{maxnext}} \leqslant e)$ **then**
21: $s = s_1$
22: **else**
23: $G = G_2$
24: $s = s - s_1$
25: $P_{max} = P_{maxnext}$
26: **end if**
27: **end while**
28: $P_{maxnext} = P_{max} + s$
29: $M = y_{P_{max}} + (e - x_{P_{max}}) * (y_{P_{maxnext}} - y_{P_{max}})/(x_{P_{maxnext}} - x_{P_{max}})$
30: **return** M

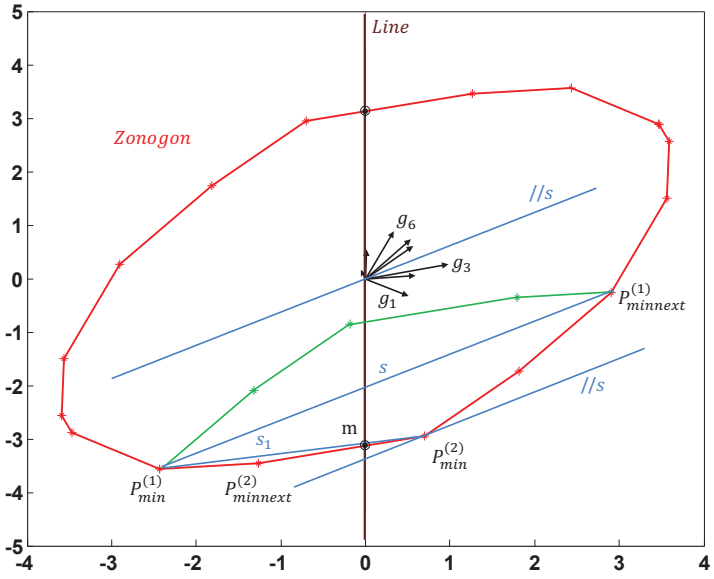


Figure 5.5.: Principle of the dichotomous search for the minimum intersection point according to algorithm 5.3 adopting the method proposed in [49].

secting the line. For computing the minimum intersection point, the lowest vertex of the zonogon is first determined. For this purpose, the generators are forced to point upwards and then subtracted consecutively from the center of the zonogon. Thereafter, only the set of generators driving the vertices of the zonogon to the other side of the line are sought out to build the set G. To avoid a full scan of the set of all the generators $(g_1, \ldots, g_q,\ q \leqslant p)$ in G, this one is hence split into a set G_1 of generators on the left of a pivot s and G_2 on the right of it. For the choice of the pivot s, it is important to remark, as explained in [49] and shown in Figure 5.5, that the line L intersects either the edge $\left[P_{min}; P_{min} + \sum_{g\in G_1} 2g\right]$ or the edge $\left[P_{min} + \sum_{g\in G_1} 2g; P_{min} + \sum_{g\in G} 2g\right]$ of the zonogon, where P_{min} is the currently computed vertex of the zonogon. Consequently that justifies the choice of $s = \sum_{g\in G_1} 2g$ as pivot. In [49], this choice was therefore justified by the absence of correlation with the position of the intersecting line. At the end of the search only one generator remains. The different computation steps of the minimum intersection point are elaborated in Algorithm 5.3. Nevertheless, the worst case complexity remains quadratic.

A new start with the highest vertex of the zonogon therefore leads to the computation of the maximum intersection point. According to Algorithm 5.3, the generators are, in this case, forced to point downwards. We also have to change the set of generators in the dichotomous search accordingly.

5.4.3. Zonotope/Strip Intersection

In [1], methods were proposed to bound the intersection of a zonotope,a strip with a family of zonotopes parameterized by the vector λ. The choice of the parameter λ affects the size and the bounds of the intersection. In [1], the following property was proved:

Property

Given a zonotope $Z = c \oplus GB^p$ and a strip $S = \left\{x \in \mathbb{R}^n : \left|b^T x - d\right| \leqslant \sigma\right\}$ and the vector $\lambda \in \mathbb{R}^n$, if

- $\widehat{c}(\lambda) = c + \lambda(d - b^T c)$,
- $\widehat{G}(\lambda) = \left[(I - \lambda b^T)G \quad \sigma\lambda\right]$.

then $Z \cap S \subseteq \widehat{Z}(\lambda) = \widehat{c}(\lambda) \oplus \widehat{G}(\lambda)B^{p+1}$.
Henceforth, if we take $\sigma = 0$ in the definition of a strip S, we obtain the classical definition of a hyperplane $H = \left\{x \in \mathbb{R}^n : b^T x = d\right\}$. Therefore, we use this property to compute the intersection of a zonotope with a hyperplane.
To minimize the size criterion of the obtained zonotope, two approaches were proposed. The first one minimizes the volume of the intersection given by the following equation:

$$\begin{aligned} Vol(\widehat{Z}(\lambda)) \quad = \quad & 2^n \sum_{i=1}^{N(n,p)} \left|\mathbf{1} - b^T\lambda\right| \ |det(A_i)| \\ & +2^n \sum_{i=1}^{N(n-1,p)} \sigma \, |det[B_i \ \ v_i]| \ \left|v_i^T\lambda\right| \end{aligned} \tag{5.19}$$

where

- $N(n, m)$ denotes the number of ways to choose n elements from a set of m elements.
- A_i denotes each of the different matrices that can be obtained by choosing n columns from matrix G.
- B_i denotes each of the different matrices that can be obtained by choosing $n - 1$ columns from matrix G.
- v_i is a vector orthonormal to the image of B_i.

The second proposed approach minimizes the size of $\widehat{Z}(\lambda)$ by reducing the norm of the generators given by the columns of the matrix $\widehat{G}(\lambda)$. Taking the Frobenius norm of the matrix $\widehat{G}(\lambda)$ leads to the optimum λ^* computed as follows:

$$\lambda^* = \frac{G\,G^T\,b}{b^T\,G\,G^T\,b + \sigma^2} \tag{5.20}$$

We opt for the second approach in our study, because of the direct availability of different parameters and its relative simplicity in formulation compared to the first approach.

5.4.4. Zonotope/Hyperplane Intersection using Singular Value Decomposition (SVD)

In [67], an algorithm for the computation of the intersection of a zonotope $Z = c \oplus GB^p$ with a hyperplane $H = \left\{x \in \mathbb{R}^n : b^T x = d\right\}$ based on *singular value decomposition* of matrices was proposed.
Let $x \in \mathbb{R}^n$,

$$\begin{aligned} x \in Z \cap H &\Leftrightarrow \exists s \in [-1,1]^p \ / \ x = c + Gs \wedge b^T x = d \\ &\Leftrightarrow \exists s \in [-1,1]^p \ / \ x = c + Gs \wedge v^T s = e \end{aligned} \tag{5.21}$$

where $v = G^T b$ and $e = d - b^T c$. Therefore computing an approximation for the intersection of a zonotope with a hyperplane can be regarded as the problem of finding an outer approximation set for the values $s \in [-1,1]^p$ verifying $As = e$ where A is the line vector v^T. A solution for this problem can be computed using SVD, even if the matrix A is singular or close to singular.

Definition

A *singular value decomposition* of a $m \times n$ matrix A with $m \geqslant n$ is a factorization $A = USV^T$ where U is a $m \times m$ matrix satisfying $U^T U = I$, V is a $n \times n$ satisfying $V^T V = I$ and S an $m \times n$ diagonal matrix S satisfying $S = diag(\alpha_1, \alpha_2, \ldots, \alpha_n)$, where $\alpha_1 \geqslant \alpha_2 \geqslant \ldots \geqslant \alpha_n \geqslant 0$ are *the singular values* of A. ■
It can be proved that if $rank(A) = r$ then $\alpha_1 \geqslant \alpha_2 \geqslant \ldots \geqslant \alpha_r > 0$, and $\alpha_{r+1} = \alpha_{r+2} = \ldots \alpha_n = 0$.
The singular values are the nonzero square roots of the eigenvalues of one of the matrices AA^T or $A^T A$. The eigenvectors of AA^T, called the left singular vectors, define the columns of matrix U while the eigenvectors of $A^T A$ are the right singular vectors which specify the rows of matrix V.
The base of the abstract space defined by the rows of the matrix V can be split in two sets. The first set, given by the matrix V_0, defines the kernel of the matrix A. The supplement V_1 of V_0 defines the complement subspace. The SVD can then be rewritten as follows:

$$A = USV^T = \begin{pmatrix} U_1 & U_0 \end{pmatrix} \begin{pmatrix} S_1 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} V_1^T \\ V_0^T \end{pmatrix} \tag{5.22}$$

where $U = \begin{pmatrix} U_1 & U_0 \end{pmatrix}$, $V = \begin{pmatrix} V_1 & V_0 \end{pmatrix}$ and S_1 is the diagonal matrix of the nonzero singular values of A. Let σ_1 and σ_0 be the coordinates of s in the new base, then

$$s = V_1 \sigma_1 + V_0 \sigma_0. \tag{5.23}$$

The equation $As = b$ can again be expressed as

$$U_1 S_1 V_1^T (V_1 \sigma_1 + V_0 \sigma_0) = b \tag{5.24}$$

so that $\sigma_1 = S_1^{-1} U_1^T b$ is hold. Owing to this fact that $\sigma_0 = V_0^T s$ and $s \in [-1;1]^p$, σ_0 must be in the zonotope with the origin as center and having the columns of

the matrix V_0^T as generators. Consequently, an approximative set of s solution of $As = b$ with the constraint $s \in [-1;1]^p$ is the zonotope Z_s defined by its center $c_s = V_1 S_1^{-1} U_1^T b$ and its generators given by the matrix $V_0 V_0^T$. For the special case $A = v^T$, the SVD of the matrix A takes a particular form with

$$U_1 = 1,\ U_0 = \varnothing,\ S_1 = \|v\|_2,\ V_1 = v/\|v\|_2,\ V_0 = \aleph(v) \tag{5.25}$$

where $V_0 = \aleph(v)$ is the orthonormal base of the kernel of v^T.
The intersection of the zonotope $Z = c \oplus GB^p$ with the hyperplane $H = \{x \in \mathbb{R}^n :\ b^T x = d\}$ is hence approximated by the zonotope

$$\begin{gathered} Z_{int} = \langle c_{int}, R_{int} \rangle, \\ c_{int} = c + Gc_s,\ R_{int} = GR_s,\ c_s = v.(d - b^T c)/v^T v,\ R_s = V_0 V_0^T. \end{gathered} \tag{5.26}$$

The collision detection check can be integrated in the algorithm for computing the intersection when using SVD since the consistency check according to condition (5.12) is directly derived with the computation of the vector $v = G^T b$. That results in the Algorithm 5.5 as proposed in [67].

Algorithm 5.5 Zonotope/hyperplane intersection using SVD

Input: $Z = c \oplus GB^p$, $H_p = \{x \in \mathbb{R}^n :\ b^T x = d\}$
Output: Z_{int}

```
v = G^T b
e = d − b^T c
M = v^T sign(v)
collision = (−M ⩽ e) ∧ (e ⩽ M)
if collision then
    V_0 = ℵ(v)                          ▸ orthonormal base of the kernel of v^T
    c_int = c + G.v.(d − b^T c)/v^T v
    R_int = G V_0 V_0^T
else
    c_int = ∅, R_int = ∅
end if
result: c_int, R_int, collision
```

5.4.5. Comparison of Zonotope/Hyperplane Intersection Methods

In this section, we carry out an efficiency comparison of the abovementioned methods implemented with MATLAB for the computation of the intersection of a zonotope and a hyperplane. We use random generated 3D-zonotopes with a growing number of generators and the hyperplane $H_p = \{x \in \mathbb{R}^3 :\ (1\,0\,0)\,.x = 0\}$. The time complexity results are summarized in Table 5.1.

The results reveal that the time complexity for both methods based on 2D-projections grow significantly with the number of generators. However, the time

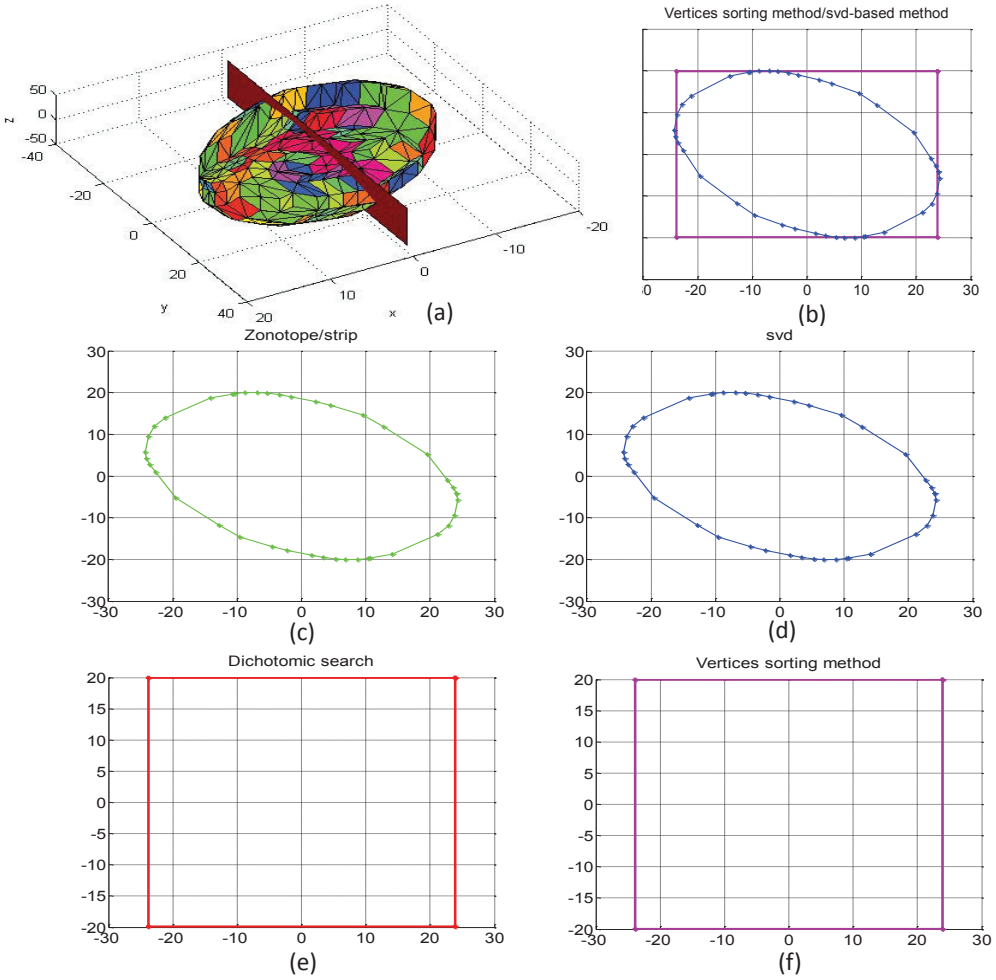


Figure 5.6.: Zonotopic approximation of $Z_1 \cap H$ resulting from (c) zonotope/strip, (d) SVD-based, (e) dichotomous search and (f) vertices sorting algorithms.

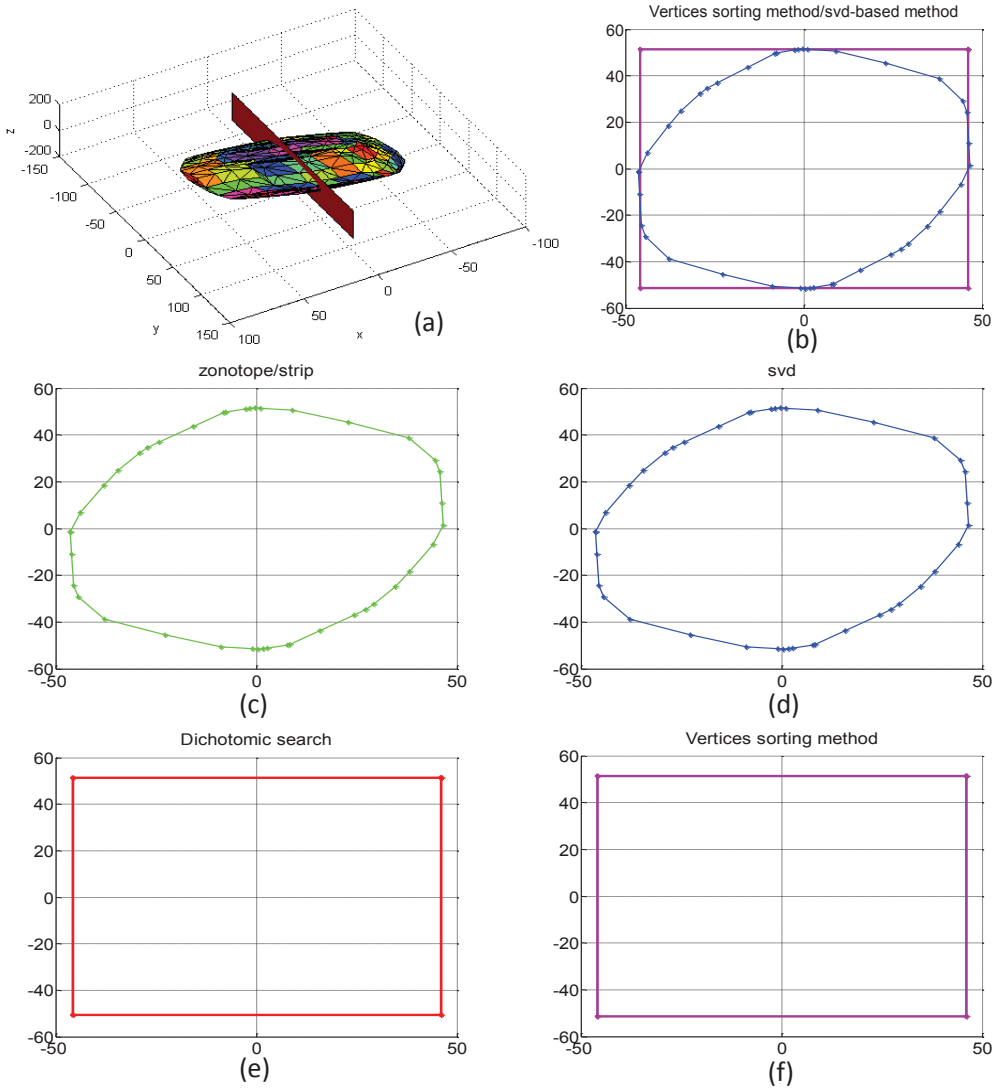


Figure 5.7.: Zonotopic approximation of $Z_2 \cap H$ resulting from (c) zonotope/strip, (d) SVD-based, (e) dichotomous search and (f) vertices sorting algorithms.

Num. of genarators	SVD	Zon./Strip	Dichotomous search	Vertices sorting
20	0.043	0.043	0.083	0.077
30	0.045	0.043	0.077	0.072
40	0.045	0.044	0.087	0.078
80	0.045	0.045	0.127	0.091
160	0.045	0.044	0.214	0.140
320	0.047	0.046	0.598	0.325
640	0.069	0.048	4.705	2.355
1280	0.171	0.050	43.839	22.174
2560	0.883	0.052	425.668	214.953
5120	5.069	0.055	3867.330	1990.890

Table 5.1.: Computation time (in s) of the four proposed zonotope/hyperplane intersection methods with a growing number of generators.

complexity of our vertices sorting algorithm seems to be better as that of the dichotomous search. For a number of generators larger than 1280, we observed an increase in the computation time of the SVD-based method. Hence, the effect of the number of generators on the time complexity in the SVD method appears to be less pronounced as in the 2D-projection methods. The Zonotope/strip intersection method also appears not to be heavily affected by the number of generators.

We next compare the tightness of the zonotopic approximation of the zonotope/hyperplane intersection resulting from different methods. For illustration purposes, we randomly created two different 3D-zonotopes Z_1 and Z_2 with 20 generators and use the hyperplane as defined above. The result shown in Figure 5.7, 5.6 reveals an exact equality of the sets obtained by the 2-D projection methods. The same observation can be made for the zonotope/strip and SVD-based method too. Nevertheless, it should be noted that these latter methods yielded tighter approximates than those of the 2D projection methods (see Figure 5.7(b), 5.6(b)). This is due to our choice of the projection directions. In fact, for this comparison particular choices of the hyperplane and the orthonormal base are made. As later shown in Section 5.11, other choices might result in a tighter, but not necessarily hyper-rectangle approximation.

5.5. Zonotope/Halfspace Intersection

We now consider the intersection of a zonotope $Z = c \oplus GB^p$ with a halfspace $H_s = \left\{x \in \mathbb{R}^n : d^T x \leqslant e\right\}$. We adopt the method proposed in [106] which first determine a tight strip over-approximation of the zonotope S_Z in direction d. As explained in Section 5.4.1, the support functions $q_u = d^T.c + \sum_p^{i=1} \left|d^T.g_i\right|$ in direction d and $q_l = d^T.c - \sum_p^{i=1} \left|d^T.g_i\right|$ in the opposite direction $-d$ are computed resulting in

$$S_Z = \left\{x \in \mathbb{R}^n : q_l \leqslant d^T x \leqslant q_u\right\}. \tag{5.27}$$

For the intersection check, three cases illustrated in Figure 5.8 can be differentiated:

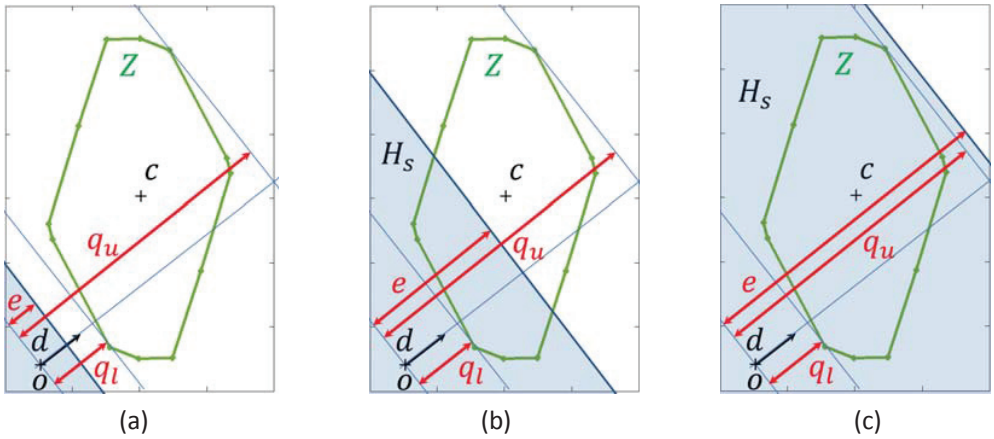


Figure 5.8.: Collision check zonotope/halfspace. (a) No intersection (b) Partial enclosure (c) Total enclosure.

- $Z \cap H_s = \varnothing$ if $q_l > e$ (see Figure 5.8(a)),
- $Z \cap H_s = Z$ if $q_u \leqslant e$ (see Figure 5.8(c)) and
- $Z \cap H_s = Z_{int} \wedge Z_{int} \neq \varnothing \wedge Z_{int} \neq Z$ if $q_l \leqslant e < q_u$ (see Figure 5.8(b)).

We note that a non-empty intersection, corresponding to the last two cases, is always confined between two halfspaces H_s and $H = \left\{x \in \mathbb{R}^n : -d^T x \leqslant -q_l\right\}$ (see Figure 5.8) which consequently limit the supporting strip approximating the intended intersection. This is then expressed as follows:

$$\begin{aligned} S_{Z \cap H_s} &= \left\{x \in \mathbb{R}^n : q_l \leqslant d^T x \leqslant e\right\} \\ &= \left\{x \in \mathbb{R}^n : \left|d^T x - \tfrac{e+q_l}{2}\right| \leqslant \tfrac{e-q_l}{2}\right\}. \end{aligned} \tag{5.28}$$

The original problem of computing an approximation for the intersection $Z \cap H_s$ is thereby reformulated to find an intersection between a zonotope Z and the strip $S_{Z \cap H_s}$. This problem has been already treated in Section 5.4.3.

5.6. Zonotope/Polyhedron Intersection

With the knowledge that a polyhedron $P = \{x \in \mathbb{R}^n : D.x \leqslant E\}$ with $D \in \mathbb{R}^{m \times n}$ and $E \in \mathbb{R}^m$ is none other as the intersection of m halfspaces $\mathcal{H}_s^i = \{x \in \mathbb{R}^n : \langle d_i, x \rangle = e_i\}$, the intersection $Z \cap P$ is computed by calling the zonotope/halfspace intersection method m times as explained in the previous section.

5.7. Zonotope/Zonotope Intersection

In this section, we are concerned with the computation of the intersection between guards given in form of zonotopes and reachable set. This involves the problem of

the intersection between two zonotopes which results, in general, in a non-zonotope and introduces new complexity issues. However, recent works [53], [67] have proposed two different approaches for the approximation of this intersection with a zonotope. The approaches in [53] appeals optimization techniques. The algorithm proposed in [67], however, uses SVD. We intend to overview all available algorithms for zonotopic operations. We first begin with the collision detection check between two zonotopes. This particular test, unlike the cases with hyperplanes or halfspaces, is not obvious. This appeals for more elaborated geometric techniques.

5.7.1. Zonotope/Zonotope Collision Detection

Various approaches were proposed to check for the intersection between convex bodies in general. Hence, most of them are restricted to 3D-problems [107, 108]. An adapted version of the Gilbert-Johnson-Keerthi (GJK) algorithm as well as an approach based on proper values and vectors were suggested in [67, 68] to solve the fault diagnostic correction problem with the use of zonotopes.

The Gilbert-Johnson-Keerthi (GJK) algorithm

The GJK algorithm is an algorithm originally proposed for the computation of the distance between two 3D-objects [28, 46]. In [83], the method was extended to multidimensional convex polyhedra by using support vectors. The distance between two zonotopes Z_1 and Z_2 is given by

$$d(Z_1, Z_2) = min\left\{\|v_1 - v_2\| \mid v_1 \in Z_1 \wedge v_2 \in Z_2\right\}. \tag{5.29}$$

The collision detection problem between two zonotopes is transformed into an equivalent problem checking for the inclusion of the origin in the Minkowski difference $Z = Z_1 \ominus Z_2$. This latter problem is in turn equivalent to the problem of computing the distance of Z to the origin expressed as follows:

$$d(Z_1, Z_2) = \|v(Z_1 \ominus Z_2)\| = \|v(Z)\| \tag{5.30}$$

where $v(Z)$ denotes the nearest vertex to the origin in Z.
As we are only interested in the collision detection between two zonotopes and not in the computation of the distance between them, we use the separating-axis version of the GJK-algorithm (SA-GJK) proposed in [83]. The algorithm constructs in each new iteration k a new simplex $S_k \subseteq Z = c \oplus GB^p$ by adding the nearest support vector w_k to the origin with a new computed direction $-v_k$ and dropping up the furthest. Support vectors are recursively computed according to $w_k = c + G^T.sign\left(-{v_k}^T G^T\right)$. The new supporting direction v_k is the solution of the quadratic optimization problem formulated for searching the nearest vertex to the origin in S_k. The formulation uses the fact that the closest vertex v to the origin is a linear combination of all vertices in S_k where the sum of the coefficients is equal to 1. As shown in Algorithm 5.6, the recursion is repeated until

$$v^T.w > 0 \;\Leftrightarrow\; 0 \notin Z. \tag{5.31}$$

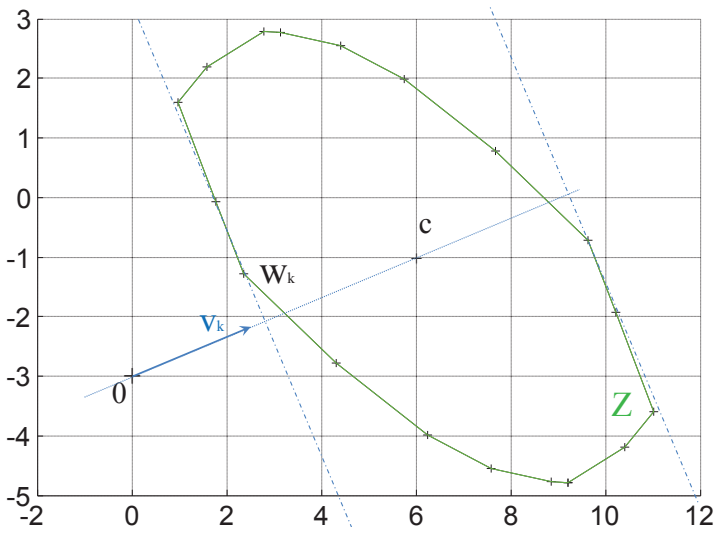


Figure 5.9.: The algorithm 5.6 ends with the extreme vertex w_k of Z in direction v_k with a fulfilled stop criterion ${v_k}^T.w_k > 0$ [67].

This means, as illustrated in Figure 5.9, that the origin is outside Z. Consequently, if we revert to the original problem, this proves the existence of a separating plane between Z_1 and Z_2. We adopt the adaptation of the GJK algorithm proposed in [67] for which a MATLAB implementation has been suggested. This adaptation is described by Algorithm. 5.6. The drawbacks of the GJK algorithm reported in [82, 105] like the termination condition, the impact of the finite precision arithmetic and some geometric aspects must be taken into consideration. In [82, 105], these problems were discussed and enhancement were suggested. However the recommended SA-GJK implementation adapted for zonotopes [67] is open to improvement with regards to the termination conditions and limitation of finite precision error propagation.

Regarding the geometric aspect, two constellations causes the algorithm to loop infinitely according to [107]. The first constellation occurs when the simplex S is oblong shaped and $v(Z)$ is an internal point of S. In this case, two vertices of S may be so close to another that the algorithm will compute the same support point in successive iterations. The second problem occurs if $v(Z)$ lies close to the diagonal of an oblong facet of Z causing the algorithm to alternate also between the same two points. Those problems happen potentially when Z_1 and Z_2 are closed to each other and have almost the same size [82, 107]. That consequently results to oblong-shaped Minkowski difference. In the case of guard intersection, however, special situations are unlikely to be encountered owing to the difference in shape and complexity between the zonotope describing the guard or the invariant and the zonotope enclosing the reachable set.

Algorithm 5.6 An adapted version of the GJK algorithm for the zonotope/-zonotope collision detection [67].

Input: $Z_1 = c_1 \oplus G_1 B^{p1}$, $Z_2 = c_2 \oplus G_2 B^{p2}$
Output: nv
1: $G = [G_1 \quad -G_2]$
2: $c = (c_2 - c_1)$
3: $p = p_1 + p_2$
4: $v = c$
5: $S = \varnothing$
6: **repeat**
7: $w = c + G\,\mathrm{sign}(G^T.(-v))$ ▸ the extreme vertex in the direction $-v$
8: **if** $v^T.w > 0$ **then**
9: $nv = 0$
10: **end if**
11: $v = v(CH(S \cup \{w\}))$ ▸ the nearest vertex to the origin in $CH(S \cup \{w\})$
12: $S = \text{smallest}\,\{X \subseteq S \cup \{w\} \mid v \in CH(X)\}$
13: **until** $v = 0$
14: $nv = 1$
15: **return** nv

A sub-optimal algorithm for zonotope/zonotope collision detection

In [67], an alternative to the GJK algorithm was proposed. The author aimed to overcome the iterative aspect of the GJK algorithm with this alternative. We consider two zonotopes $Z_1 = c_1 \oplus G_1 B^{p1}$ and $Z_2 = c_2 \oplus G_2 B^{p2}$ with $B = [-1, 1]$. The Minkowski difference is equal to $Z = Z_1 \ominus Z_2$ and $Z = c \oplus GB^p$ with $c = c_1 - c_2$, $G = [G_1 - G_2]$ and $p = p_1 + p_2$. Based on the convexity of zonotopes following equivalences are deduced:

$$\begin{aligned} Z_1 \cap Z_2 = \varnothing &\Leftrightarrow 0 \notin c \oplus GB^p \\ &\Leftrightarrow \exists l \mid 0 \notin l^T c \oplus l^T G B^p \\ &\Leftrightarrow \exists l \mid -l^T c \notin l^T G B^p \\ &\Leftrightarrow \exists l \mid -l^T c \notin \left[-\left\|l^T G\right\|_1, \left\|l^T G\right\|_1\right] \text{(see also (5.11))} \\ &\Leftrightarrow \exists l \mid \left|l^T c\right| \notin \left[0, \left\|l^T G\right\|_1\right] \\ &\Leftrightarrow \exists l \mid \left|l^T c\right| > \left\|l^T G\right\|_1 \end{aligned} \tag{5.32}$$

The problem of collision detection is, hence, formulated as an optimization problem aimed at finding a direction l^* maximizing the following function:

$$f_1(l) = \frac{\left|l^T c\right|}{\left\|l^T G\right\|_1}. \tag{5.33}$$

In [67], however, a suboptimal solution was proposed optimizing the following function instead of the function f_1 :

$$f_2(l) = \frac{\|l^T c\|_2}{\|l^T G\|_2} = \frac{l^T c c^T l}{l^T G G^T l}. \tag{5.34}$$

This comes back to the problem of finding a maximum proper value of the matrix $[cc^T \ GG^T]$, the proper vector of which is the intended solution l^*. The collision test consists then in verifying the condition:

$$\left|l^{*T} c\right| > \left\|l^{*T} G\right\|_1. \tag{5.35}$$

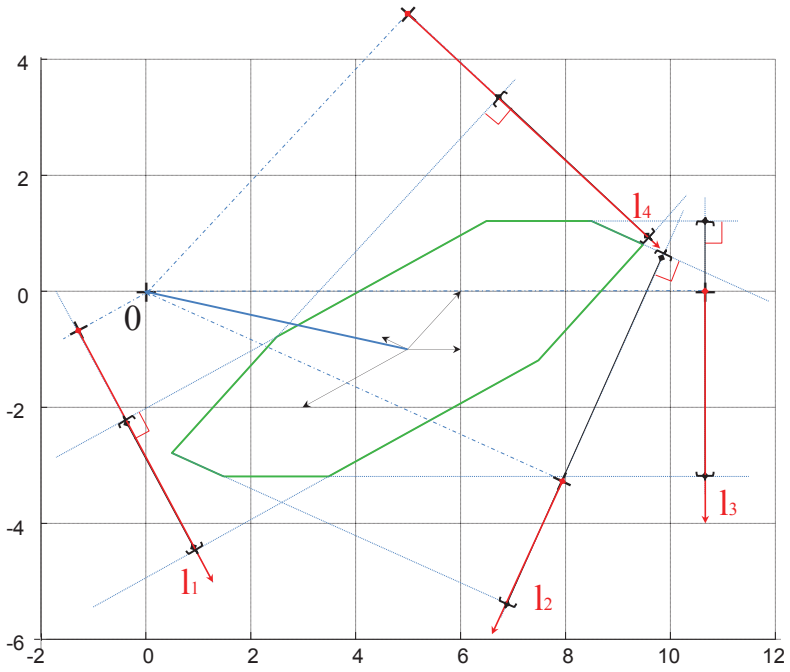


Figure 5.10.: Method to find an l_i (here l_1 or l_4) verifying (5.32) [67].

Figure 5.10 shows this condition to be fulfilled for two directions l_1 and l_4.

Comparison and evaluation of both detection collision methods

We aim in this section to evaluate and compare the performances of the overviewed zonotope/zonotope collision detection methods. We tested the algorithm in cases of no intersection (Table 5.2) as well as by intersections (Table 5.3).

Number of genarators		GJK itera tions	GJK average Time(s)	GJK min Time(s)	GJK max Time(s)	sub-optimal average Time(s)	sub-optimal min Time(s)	sub-optimal max Time(s)
1.	**3**	7	0.0150	0.0131	0.0405	$7.2547e^{-5}$	$6.4926e^{-5}$	0.0029
2.	**4**	5	0.0077	0.0071	0.0184	$6.8522e^{-5}$	$6.4926e^{-5}$	$5.5475e^{-4}$
3.	**6**	5	0.0115	0.0107	0.0206	$7.5608e^{-5}$	$6.5337e^{-5}$	0.0036
4.	**20**	4	0.0059	0.0051	0.0175	$7.6822e^{-5}$	$6.5337e^{-5}$	0.0024
5.	**30**	5	0.0077	0.0071	0.0191	$6.8560e^{-5}$	$6.5337e^{-5}$	$5.7735e^{-4}$
6.	**40**	4	0.0054	0.0051	0.0726	$6.8914e^{-5}$	$6.5337e^{-5}$	$4.9927e^{-4}$
7.	**80**	5	0.0089	0.0081	0.0243	$8.0463e^{-5}$	$6.4515e^{-5}$	0.0038
8.	**160**	11	0.0365	0.0334	0.0764	$7.2075e^{-5}$	$6.7803e^{-5}$	0.0015
9.	**320**	5	0.0105	0.0086	0.0311	$8.2204e^{-5}$	$6.8624e^{-5}$	0.0052
10.	**640**	5	0.0078	0.0071	0.0191	$8.7771e^{-5}$	$7.4788e^{-5}$	0.0032
11.	**640**	5	0.0094	0.0087	0.208	$7.9087e^{-5}$	$7.4377e^{-5}$	$5.4817e^{-4}$
12.	**1280**	5	0.0079	0.0072	0.0174	$9.1605e^{-5}$	$8.4240e^{-5}$	$7.4049e^{-4}$
13.	**2560**	5	0.0092	0.0072	0.0458	$1.1813e^{-4}$	$1.1342e^{-4}$	$6.8542e^{-4}$
14.	**5120**	6	0.0166	0.0133	0.0600	$1.9954e^{-4}$	$1.4917e^{-4}$	0.0048

Table 5.2.: Estimation of the average, min and max computation times of the GJK and the sub-optimal MATLAB implementations for collision detection of two intersecting zonotopes $Z_1(c_1 = [-1; 0; 1], G_1 = [1\ 1\ 2; 1\ 1\ 1; 1\ -\ 2\ 1])$ and a randomly generated zonotope Z_2 with a growing number of generators.

We supposed the guard conditions to be represented by the zonotope Z_1 and the reachable set to be approximated by the zonotope Z_2. This latter is chosen randomly and the number of its generators could be arbitrary augmented. We gave an estimation of the average computation time by running each algorithm 10000 times with the same inputs. We noted also the minimum and the maximum time values. It should be pointed out here that the computation time increase slightly with the number of generators particularly by the sub-optimal algorithm. The results of the GJK algorithm show a significant increase of the computation time with the number of iterations. This in turn is clearly shown to depend more on the form, the orientation and the position of the zonotope Z to the origin as on its complexity. To justify this observations, we introduced the zonotopes (Table 5.210.,11., Table 5.3 10.,11.) with both 640 but different generators. We remark that the number of iterations of the GJK algorithm by the no intersection case are different. That effected accordingly the computation time. We noted therefore, that the case (Table 5.3 10.) is really not an intersecting case. However, the sub-optimal algorithm recognized exactly the opposite. This is due to the sub-optimality aspect of this method. In general, the sub-optimal algorithm is clearly faster. But the result of this algorithm are concluding only by no intersection cases.

Number of genarators		GJK itera tions	GJK average Time(s)	GJK min Time(s)	GJK max Time(s)	sub-optimal average Time(s)	sub-optimal min Time(s)	sub-optimal max Time(s)
1.	**3**	1	$5.3123e^{-5}$	$4.7667e^{-5}$	0.0017	$6.9138e^{-5}$	$6.4926e^{-5}$	$6.1721e^{-4}$
2.	**4**	3	0.0018	0.0017	0.0124	$6.8001e^{-5}$	$6.3282e^{-5}$	0.0015
3.	**6**	3	0.0018	0.0017	0.0053	$7.7129e^{-5}$	$6.3282e^{-5}$	0.0040
4.	**20**	3	0.0018	0.0017	0.0052	$6.9318e^{-5}$	$6.5337e^{-5}$	0.0013
5.	**30**	5	0.0057	0.0054	0.0149	$7.2341e^{-5}$	$6.5337e^{-5}$	$5.5419e^{-4}$
6.	**40**	3	0.0019	0.0017	0.0050	$7.3441e^{-5}$	$6.5337e^{-5}$	$7.1706e^{-4}$
7.	**80**	1	$5.2521e^{-5}$	$4.9311e^{-5}$	0.0043	$7.2320e^{-5}$	$6.6159e^{-5}$	$8.0171e^{-4}$
8.	**160**	1	$5.9509e^{-5}$	$5.0133e^{-5}$	0.0051	$7.0706e^{-5}$	$6.7803e^{-5}$	$6.2830e^{-4}$
9.	**320**	1	$6.3030e^{-5}$	$5.2187e^{-5}$	0.0030	$8.8267e^{-5}$	$7.0679e^{-5}$	0.0024
10.	**640**	5	0.0055	0.0054	0.0140	$8.0225e^{-5}$	$7.4377e^{-5}$	0.0029
11	**640**	1	$5.7674e^{-5}$	$5.5886e^{-5}$	0.0019	$7.6821e^{-5}$	$7.2734e^{-5}$	$5.8269e^{-4}$
12.	**1280**	1	$8.1193e^{-5}$	$6.5748e^{-5}$	0.0041	$9.2673e^{-5}$	$8.6294e^{-5}$	$5.7077e^{-4}$
13.	**2560**	3	0.0020	0.0019	0.0088	$1.1929e^{-4}$	$1.1424e^{-4}$	$7.6966e^{-4}$
14.	**5120**	4	0.0043	0.0039	0.0312	$1.1820e^{-4}$	$1.43670e^{-4}$	0.0012

Table 5.3.: Estimation of the average, min and max computation times of the GJK and the sub-optimal MATLAB implementations for collision detection of two no intersecting zonotopes $Z_1(c_1 = [-6; 0; 1], G_1 = [1\ 1\ 2; 1\ 1\ 1; 1\ -\ 2\ 1])$ and a randomly generated zonotope Z_2 with a growing number of generators.

5.7.2. Intersection of Two Zonotopes using Optimization Techniques

Let $Z_1 = c_1 \oplus G_1 B^{p_1}$, $Z_2 = c_2 \oplus G_2 B^{p_2}$ be two zonotopes and E a matrix in $\mathbb{R}^n$ and let

$$\begin{array}{rcl} \widehat{c}(E) & = & Ec_1 + (I - E)c_2 \\ \widehat{G}(E) & = & [EG_1 \quad (I - E)G_2] \end{array} \tag{5.36}$$

then

$$\begin{array}{l} Z_1 \cap Z_2 \subseteq \widehat{Z}(E) \\ \widehat{Z}(E) = \widehat{c}(E) \oplus \widehat{G}(E) B^{p_1+p_2} \end{array} \tag{5.37}$$

A further problem remains, namely the choice of the matrix E. It is obvious that we are interested in reducing the size of $\widehat{Z}(E)$. For this reason, [53] suggested the following minimization function

$$f(E) = \sum_{i=1}^{p_1} (EG_{1i})^T (EG_{1i}) + \sum_{j=1}^{p_2} (G_{2j} - EG_{2j})^T (G_{2j} - EG_{2j}) \tag{5.38}$$

where G_{1i}, G_{2j} are the columns of the matrices G_1, G_2 and the generators of Z_1, Z_2 respectively. For illustration Figure 5.11 shows the intersection of zonotope Z_1 : $c_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$, $G_1 = \begin{pmatrix} 1 & 1 & -0.5 & -2 \\ 0 & 1 & 0.2 & -1 \end{pmatrix}$ in blue and zonotope Z_2 : $c_2 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$, $G_2 = \begin{pmatrix} 1 & 1 & -0.5 \\ 0 & 1 & 0.2 \end{pmatrix}$ in red. The zonotope Z_{int} in black is the computed over-approximation of the intersection.

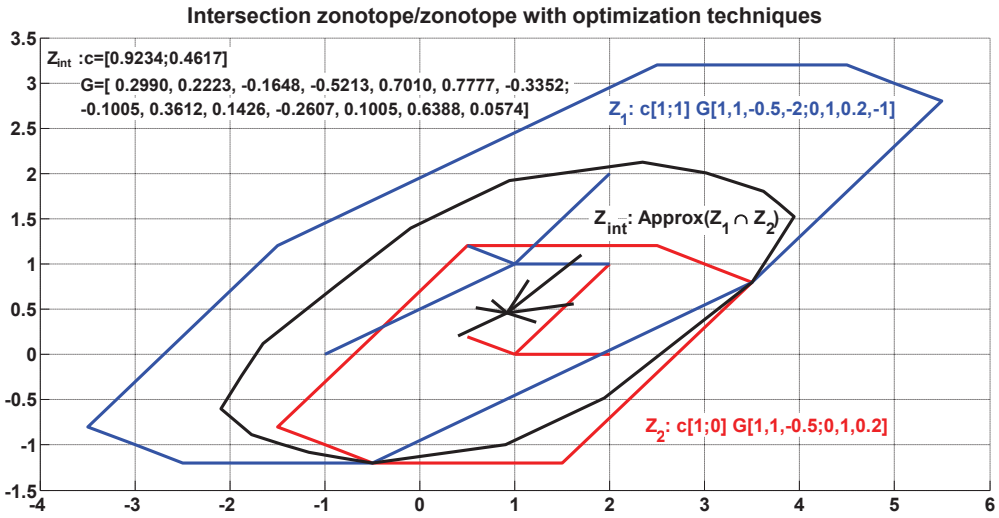


Figure 5.11.: Illustration of the zonotope/zonotope intersection method using optimization.

5.7.3. Intersection of Two Zonotopes using SVD

To use this technique, we have first to rewrite our problem in the form $As = e$ where $s \in [-1, 1]^p$ (see Section 5.4.4). In fact,
$\forall x \in Z_1 \cap Z_2$, $\exists s_1 \in [-1, 1]^{p_1}$ and $\exists s_2 \in [-1, 1]^{p_2}$ / $x = c_1 + G_1 s_1 = c_2 + G_2 s_2$.
Therefore, the problem of computing an approximation of the intersection between these two zonotopes can be formulated as a problem of finding an approximation $[s]$ of the set of $s = [s_1; s_2] \in [-1; 1]^p$ such that
$As = b$ with $A = [G_1 \;\; G_2]$, $b = (c_2 - c_1)$ and $p = p_1 + p_2$.
The SVD results, as elaborated in Section 5.4.4, in the unitary matrices $U = \begin{pmatrix} U_1 & U_0 \end{pmatrix}$, $V = \begin{pmatrix} V_1 & V_0 \end{pmatrix}$ with $U^T U = V^T V = I$ and the diagonal matrix S_1 of the nonzero singular values of A. The zonotopic approximation of $[s] = \langle c_s, G_s \rangle$ is then given by

$$c_s = V_1 S_1^{-1} U_1^T \;\text{ and }\; R_s = V_0 V_0^T. \tag{5.39}$$

Let now c_{s1} be the vector of the p_1 first lines of c_s and R_{s1} the matrix containing just the p_1 first lines of the matrix R_s. An approximation set $Z_{int} = \langle c_{int}, G_{int} \rangle$ of the intersection between Z_1 and Z_2 is then computed according to Algorithm 5.7 as:

$$c_{int} = c_1 + R_1 c_{s1} \;\text{ and }\; R_{int} = R_1 R_{s1}. \tag{5.40}$$

We note here that an another over-approximation can be obtained if we extract the last p_2 components in respectively c_{s2} and R_{s2}, instead of taking the p_1 first

Algorithm 5.7 Zonotope/Zonotope intersection using SVD

algorithmic[htbp]

Input: $Z_1 = c_1 \oplus G_1 B^{p1}$, $Z_2 = c_2 \oplus G_2 B^{p2}$,
Output:
1: $A = [G_1 \;\; G_2]$
2: $b = (c_2 - c_1)$
3: $p = p_1 + p_2$
4: $[U_1, U_0, S_1, V_1, V_0]$=SVD$(A)$
5: $c_s = V_1 S_1^{-1} U_1^T$
6: $R_s = V_0 V_0^T$
7: $P_1 = [I_{p1} \;\; 0_{p1 \times p2}]$
8: $c_{s1} = P_1 c_s$
9: $R_{s1} = P_1 R_s$
10: $c_{int} = c_1 + R_1 c_{s1}$
11: $R_{int} = R_1 R_{s1}$
12: **return** c_{int}, R_{int}

lines of c_s and R_s. We accordingly obtain

$$c_{int} = c_2 + R_2 c_{s2} \;\text{ and }\; R_{int} = R_2 R_{s2}. \tag{5.41}$$

Tests of the SVD-method with the same choices of zonotopes have resulted in the same over-approximations for the intersection set as with optimization.

5.7.4. Comparison of Both Zonotope/Zonotope Intersection Methods

We compare the performances of the zonotope/zonotope intersection methods with the same suite of zonotopes used for testing the collision detection approaches in the case of intersection. The results are summarized in Table 5.4. We note that both methods are reliable and that for zonotopes with more than 40 generators time consumption and number of generators are proportional for both methods. However, the Table 5.4 shows the SVD method to be the fastest. With regard to the tightness of the intersection, both methods practically provide the same over-approximation.

Number of genarators		SDV			Opt.		
		average	min	max	average	min	max
		Time(s)			Time(s)		
1.	**3**	$1.2854e^{-4}$	$9.9031e^{-5}$	0.0011	0.1561	0.0807	0.2373
2.	**4**	$1.2983e^{-4}$	$9.9442e^{-5}$	0.0013	0.1490	0.0805	0.2259
3.	**6**	$1.1885e^{-4}$	$1.0109e^{-4}$	0.0013	0.1505	0.0736	0.2320
4.	**20**	$1.7229e^{-4}$	$1.2286e^{-4}$	0.0018	0.2689	0.1095	0.4732
5.	**30**	$1.9816e^{-4}$	$1.3273e^{-4}$	0.0016	0.3037	0.1406	0.4523
6.	**40**	2.3394^{-4}	$1.3273e^{-4}$	0.0019	0.3945	0.1846	0.6282
7.	**80**	$4.0423e^{-4}$	$1.3273e^{-4}$	0.0034	0.6540	0.3334	1.0995
8.	**160**	$9.7902e^{-4}$	$1.3273e^{-4}$	0.0037	1.1903	0.6928	1.9531
9.	**320**	0.0044	$1.3273e^{-4}$	0.0091	2.3109	1.2639	3.8150
10.	**640**	0.0276	$1.3273e^{-4}$	0.0389	4.4174	2.1154	6.5398
11.	**640**	0.0282	$1.3273e^{-4}$	0.0407	4.2159	2.3018	6.4668
12.	**1280**	0.1411	$1.3273e^{-4}$	0.2046	9.1871	3.9015	12.8966
13.	**2560**	0.8445	$1.3273e^{-4}$	1.0849	17.8006	8.3248	30.4598
14.	**5120**	6.1378	$1.3273e^{-4}$	7.6836	38.1476	17.4229	67.2161

Table 5.4.: Estimation of the average, min and max computation times of the MATLAB implementations of the SDV and the optimization based methods for the intersection of two intersecting zonotopes $Z_1(c_1 = [-1; 0; 1], G_1 = [1\ 1\ 2; 1\ 1\ 1; 1\ -2\ 1])$ and a randomly generated zonotope Z_2 with a growing number of generators.

5.8. Computing Reachable Sets within Discrete Modes

The computation of reachable sets with zonotopes is built on the top of the fundamental recursion scheme given by equation (3.17) in Chapter 3: $\Omega_k = e^{rA}\Omega_{k-1} \oplus \mathcal{V}_r$. For the input contribution, we choose the over-approximation of equation (3.27) based on the constant piecewise assumption of the input

$$\mathcal{V}_r = \mathbb{B}U \tag{5.42}$$

where $\mathbb{B} = \int_0^r e^{A(r-s)}Bds$.

In addition, we consider the approximation of equation (3.18) based on the norm-bounded input assumption

$$\mathcal{V}_r = \mathcal{B}(\beta_r) \tag{5.43}$$

where $\mathcal{B}(\beta_r)$ is the ball of radius $\beta_r = \mu\frac{e^{r\|A\|}-1}{\|A\|}$ and $\mu = sup_{u\in U}\|Bu\|$.
For the over-approximation of the initial set Ω_0, we opt for the method of Section 3.7.1 described in Chapter 3 using the expression $\Omega_0 = CH(X_0 \cup e^{rA}X_0) \oplus \mathcal{B}(\alpha_r) \oplus \mathcal{V}_r$, where $\mathcal{B}(\alpha_r)$ is the ball of radius $\alpha_r = \left(e^{r\|A\|} - 1 - r\|A\|\right) sup_{x\in X_0}\|x\|$ and X_0 is the initial set.

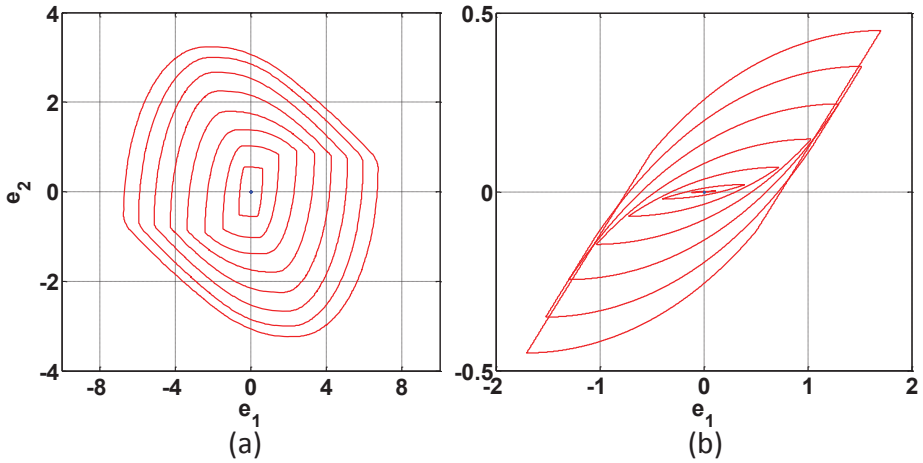


Figure 5.12.: Projections of the reachable sets of the one mode platoon example each 500th iteration with $U = [-1, 1]$, $N = 4000$, $r = 0.001$ and $X_0 = \{0\}^2 \times [-0.0001, 0.0001] \times \{0\}^6$. (a) e_1, e_2 [m] using the norm-bounded assumption $\|Bu\| \leqslant \mu$ (max. zonotope order $q = 1000$). (b) e_1, e_2 [m] using the piecewise constant assumption (max. zonotope order $q = 20$).

To compute Ω_0 using zonotopes, an approximation consisting of three steps was proposed in [47]. The convex hull $\Omega_0 = CH(X_0 \cup e^{rA} X_0)$ is first approximated by a zonotope P according to equation (5.9). In a second step, P is bloated with a ball of radius α_r to obtain the set $S \subseteq P \oplus \blacksquare(\alpha_r)$ enclosing all states reachable within the time interval $[0, r]$. Finally, the input contribution is added.

We tested both initialization methods on the one mode platoon described in Appendix A. Figure 5.12 shows the over-approximation of the initial set based on the piecewise assumption to be significantly tighter as the initial set based on the norm-bounded assumption. Therefore, only the former will be adopted in the following sections.

For systems described by $\dot{x} = Ax + Bu$ with a convex set $u \in U$, the reachable sets are computed from an initial convex set X_0 for a duration $T = Nr$, where r is the chosen time step and N the number of iterations according to Algorithm 5.8. A preliminary computation step is required to obtain both matrices $\Phi = e^{rA}$ and $\mathbb{B} = \int_0^r e^{A_{ij}(r-s)} B_{ij} ds$. We hence use the method proposed in [109] as it does not impose restrictions on the matrices A or B and allows the simultaneous calculation of both intended matrices by evaluating the exponential of the following matrix:

$$C = \begin{pmatrix} A & B \\ 0 & 0 \end{pmatrix} \quad \Rightarrow \quad e^{Cr} = \begin{pmatrix} \Phi & \mathbb{B} \\ 0 & I \end{pmatrix} \tag{5.44}$$

We note that the internal loop of Algorithm 5.8 calls only the linear and Minkowski sum operations twice each in different stages. This loop conserves therefore the

Algorithm 5.8 Iterative computation of reachable sets within discrete modes

Input: α_r, r, N, Φ, $\mathbb{B}$, U, $X_0 = (c, \langle g_1, \ldots, g_p \rangle)$;
Output: $\Omega_0, \ldots, \Omega_N$;
1: $P = (\frac{c+\Phi c}{2}, < \frac{g_1+\Phi g_1}{2}, \ldots, \frac{g_p+\Phi g_p}{2}, \frac{c-\Phi c}{2}, \frac{g_1-\Phi g_1}{2}, \ldots, \frac{g_p-\Phi g_p}{2} >)$
2: $X_0 = P \oplus \blacksquare(\alpha_r)$
3: $V_0 = \mathbb{B}U$
4: $S_0 = \{0\}$
5: **for** $k = 0$ **to** $N - 1$ **do**
6: $\quad X_{k+1} = \Phi X_k$
7: $\quad S_{k+1} = S_k \oplus V_k$
8: $\quad V_{k+1} = \Phi V_k$
9: $\quad R_{k+1} = X_{k+1} \oplus S_{k+1}$
10: $\quad \Omega_{k+1} = reduced - zonotope - order(\Omega_{k+1})$
11: **end for**
12: **return** $\{\Omega_0, \ldots, \Omega_N\}$

zonotopic structure. The problem, hence, resides in the increasing number of the generators after each Minkowski sum call which consequently can have a crucial impact on the complexity of the computation. To control this problem, a maximum number of generators is fixed a-priori and the order reduction method of Section 5.3 is involved at the end of the loop. However, this latter operation is over-approximating and can drastically affect the tightness of the resulting reachable sets.
The following section is dedicated to transitions. Guards expressed as hyperplanes, halfspaces, polyhedra and also as zonotopes are considered for the intersection described in Sections 5.5, 5.4, 5.6 and 5.7. We introduce various strategies for handling the bundle of zonotopes intersecting the guard and discuss their combination with various intersection methods.

5.9. Handling Invariants

In each iteration of Algorithm 5.8, a computation step is integrated for checking and computing the intersection with the invariant. For invariant conditions formulated as halfspaces, polyhedra or zonotopes, the approaches presented in Sections 5.5, 5.6 and 5.7 respectively are applied.

5.10. Handling Transitions

Besides the integration of invariant handling step, a second step is necessary for handling guards in each new recursion step of Algorithm 5.8. Furthermore, guards in form of equalities, inequalities and zonotopes can be treated using the previously introduced zonotope intersection methods. We therefore know that because of the

time discretization, the intersection of the flowpipe with the guard may involve a bundle of successively computed reachable sets. This bundle of zonotopes is determined by the instance of the first intersection detection k_{min} and the last k_{max} respectively.

$$\mathcal{B}^Z = \bigcup_{k=k_{min}}^{k_{max}} \Omega_k. \tag{5.45}$$

A naive approach is to treat each zonotope intersecting the guard separately before and after the transitions. That can result in a computation of significant number of flowpipes. Such an approach is practically infeasible especially for systems with many transitions. Figure 5.13 illustrates this approach on the two-tank system as described in Figure 4.22(b). Each transition is identified by a different color for easy

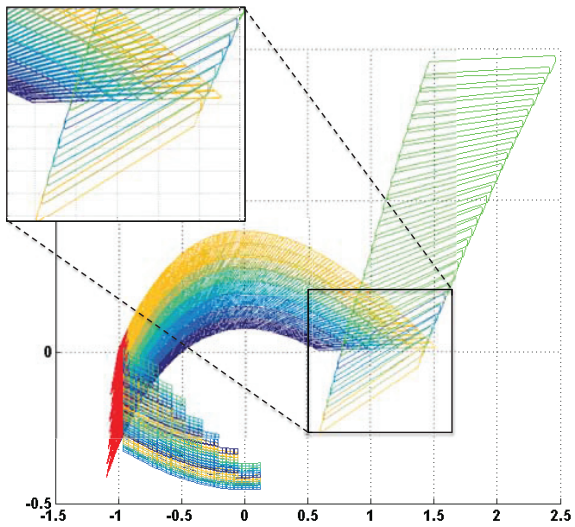


Figure 5.13.: Each zonotopic reachable set intersecting the guard is considered separately before and after the transition.

visualization. Zonotopes in red represent the intersection bundles with the respective guard. It is noteworthing that even for this sample benchmark, the computation complexity grows with the number of maximal allowed transitions and with the existence of multiple transitions outgoing from the same discrete mode. The following section presents some clustering strategies that could circumvent this problem. Although clustering has been suggested as a potential solution to this problem, these methods tend to suffer from rough and increasing over-approximations.

5.10.1. Taking the First Intersection

The simplest and naive approach is to spring to the next transition as soon as an intersection is found. This means that only the first zonotope of the intersecting

bundle is in this case regarded.

5.10.2. The Over-approximative Convex Hull Method

We use the method for computing a zonotope approximation of the initial set [47] to over-approximate the intersecting bundle as the last is constructed by successive computed zonotopes. Beginning with the set $\Omega_{k_{min}}$, the method is recalled $(k_{max} - k_{min})$ times to acquire an over-approximation of the intersecting bundle before computing the intersection with the intended guard. We note that only $\Omega_{k_{min}}$ is involved in this computation. The issued approximation is also expected to be large due to the rough approximation.

5.10.3. Pre- and Post-clustering

Clustering by means of the convex hull operation given in Section 5.3 can be applied either before or after the computation of the intersection. The pre-clustering strategy computes first iteratively an over-approximation $\overline{\Omega}$ of the intersecting bundle as follows

$$CH\left(...CH\left(\Omega_{k_{min}} \cup \Omega_{k_{min}+1}\right) \cup ... \cup \Omega_{k_{max}}\right) \subseteq \overline{\Omega}. \tag{5.46}$$

In a second step, the final intersection with the guard Ω^I is computed using one of the aforementioned intersection methods.

The post-clustering strategy, however, computes, for each intersecting zonotope Ω_k, an over-approximation Ω_k^I of the intersection with the guard. This results in $(k_{max} - k_{min} + 1)$ new zonotopes which are then clustered in the same way as in (5.46) to obtain the final intersection Ω^I.

$$\Omega^I \supseteq CH\left(...CH\left(\Omega_1^I \cup \Omega_2^I\right) \cup ... \cup \Omega_{k_{max}-k_{min}+1}^I\right). \tag{5.47}$$

5.10.4. Finding the Min/Max at a Guard Transition

This method is valid only for the ND-Projection approach and exploits the fact that the supremum and infinimum points for each two-dimensional projection are already available such that the global supremum and infinimum can be extracted. These values allow for the construction of a hyperrectangle or a hyperparallelotope of dimensionality n to over-approximate the final intersection set of the bundle with the guard.

5.10.5. Fixpoint-/Time-Triggered Transition

In addition to the aforementioned guard transitions, we also consider fixpoint- and time-triggered transitions. The former takes place immediately upon reaching a fixpoint while the later is triggered after a predefined time period.
A fixpoint is reached if successive computed reachable sets are equal within a certain margin of tolerance ϵ. To check for the reachability of a fixpoint, we opt for the

zonotope support function formulation as a practical criteria for comparison of sets instead of using the volume owing to its simplicity and its computational efficiency. In fact, for a given direction $l_j \in D = \{l_1, \ldots, l_m\} \subset \mathbb{R}^n$, the support function of the reachable set $\Omega_k = c_k \oplus G_k B^{p_k}$ is given by $\rho_{\Omega_k}(l_j) = {l_j}^T.c_k + \left\|{l_j}^T.G_k\right\|_1$. The sets Ω_k and Ω_{k+1} are ϵ-equal if:

$$\left|\rho_{\Omega_k}(l_j) - \rho_{\Omega_{k+1}}(l_j)\right| \leqslant \epsilon. \tag{5.48}$$

The directions in template D are generally chosen to be uniformly distributed in the unit ball. The ϵ-equality check must be repeated a given number of times for a clearer certainty of a fixpoint.

5.11. Implementation

A prototypical implementation of the above described algorithms and methods was first done in MATLAB. Within the HyPro-project [61], a new implementation in C++ based on the previous MATLAB implementation was required as a part of the Hypro C++ geometric set library. Compared to the MATLAB implementation, the C++ implementation has been extended with the halfspace/polyhedron guard intersection and invariants. We furthermore integrated the above-mentioned strategies for handling transitions as well as picking transition strategies to deal with cases where more as one outgoing transitions are possible. The architecture and the setting parameters of the C++ implementation is illustrated in Figure 5.14. The Algorithm 5.8 upgraded with the invariant computation method described in

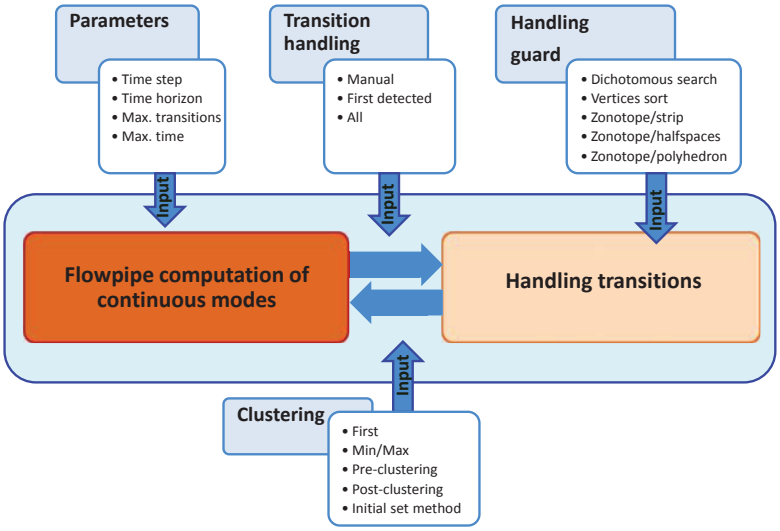


Figure 5.14.: Architecture and setting of the C++ implementation.

Section 5.9 constitutes the core of the flowpipe computation in continuous modes.

The transition handling component includes the intersection methods with hyperplanes, halfspaces and poyhedra. Upon this, the clustering methods: first detected, Min/Max, post-clustering, pre-clustering as well as the initial set based convex hull methods have been integrated and offered as user choices. In addition, the transition choice strategies: *first detected* and *manual* have been made available.

5.12. Experimental Results and Performance Evaluation

We reserve this section for experiments with the C++ implementation. The results of the previous sections are results of the MATLAB implementation.

We begin with a new evaluation of the C++ implementation of zonotope/hyperplane intersection methods as we have done with the MATLAB implementation, albeit with new test suite of zonotopes. We first compare the time computation as well as the tightness of the intersection approximation. We second investigate their performances in the context of guard intersection when combined with different clustering approaches. All benchmarks and tests were run on a Core i5 2.5 GHz system with 8 GB RAM.

5.12.1. Comparison of Intersection Methods

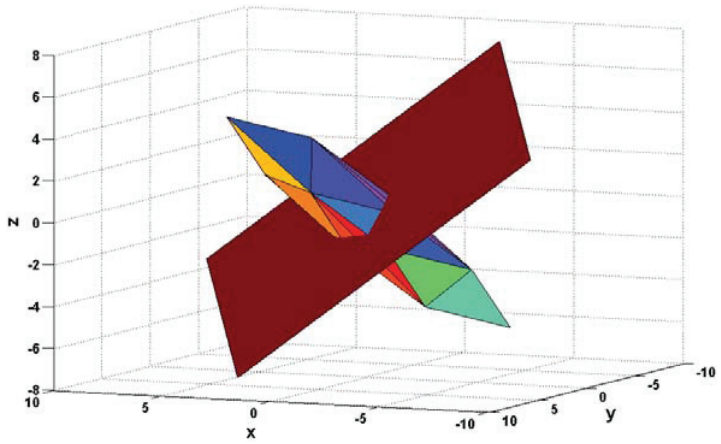


Figure 5.15.: Illustration of the zonotope/hyperplane intersection in 3D.

The test suite is composed of randomly generated 3-dimensional zonotopes with an increasing number of generators and hyperplanes. During this test, the form of the zonotopes and hyperplanes was assumed not to have an impact on the computational complexity. The computation times are listed in Table 5.5. A first observation shows stark differences between the zonotope/strip intersection method and the other methods. The former performs considerably faster than the rest even as

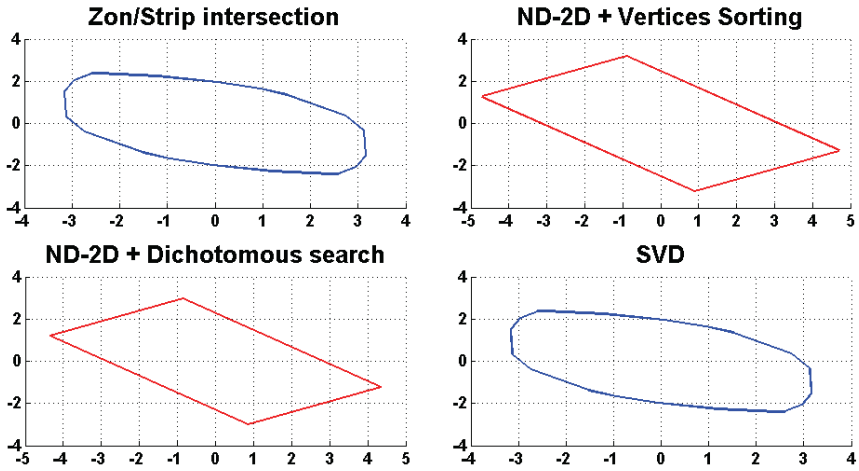


Figure 5.16.: 3-dimensional zonotope/hyperplane intersection with different methods.

Num. of genarators	Zon./Strip	SVD	Vertices sorting	Dichotomous search
4	0.06626	0.1035	0.47391	0.50093
10	0.08078	0.12059	0.55941	0.67121
50	0.18425	0.3525	2.09554	1.97534
100	0.28768	0.99743	4.39717	3.33773
200	0.47578	4.50452	8.94314	5.64377
500	1.28351	27.7316	22.2223	12.5497
1000	2.37012	143.644	37.3283	23.9758
2000	4.37552	599.267	78.5401	43.5329

Table 5.5.: Computation time (in ms) of the zonotope/hyperplane intersection methods implemented in C++.

the number of generators increases. The SVD method is the second best method for fewer than 200 generators. However, the computation drastically slows down for more than 500 generators. The SVD method turns out ultimately as the slowest method for large number of generators. Both ND-2D methods exhibit no significant difference with fewer than 200 generators. However, the dichotomous search method is faster than the vertices sorting method at higher numbers of generators. This probably returns from the implementation efficiency of the sorting algorithm required for vertices sorting methods. It is also important at this stage to recall the results obtained with the MATLAB implementation. We note that both methods performs differently. The MATLAB implementation of the vertices sorting method performed better as the dichotomous method. This is likely attributable to different sorting algorithm and SVD implementations. The sorting algorithm seems to be efficiently implemented in MATLAB than that offered by the C++ Eigen library. In general depending on the application, MATLAB averages a processing speed that is over 100 times slower than C++ code.
Figure 5.16 allows a visual comparison of different approximations computed with all four methods for the zonotope/hyperplane 3D-intersection shown Figure 5.15. The zonotope and the hyperplane of Figure 5.15 are described as follows

$$Z = \left(\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \left\langle \begin{bmatrix} 1.343 \\ 0.8865 \\ 1.369 \end{bmatrix}, \begin{bmatrix} 1.156 \\ 1.794 \\ 0.6454 \end{bmatrix}, \begin{bmatrix} 0.1341 \\ 1.692 \\ 1.571 \end{bmatrix}, \begin{bmatrix} 1.650 \\ 0.6290 \\ 1.793 \end{bmatrix} \right\rangle \right) \tag{5.49}$$

$$\mathcal{H} = \left\{ x \in \mathbb{R}^n : \left\langle \begin{bmatrix} 0.639982 \\ 0.59482 \\ 0.923346 \end{bmatrix}, x \right\rangle = 0 \right\}. \tag{5.50}$$

With regards to the tightness of approximation, we note, in Figure 5.16, differences in the shape and tightness between two groups zonotope/strip intersection and SVD methods, as a group and the dichotomous search and vertices sorting as another. The approximations resulting from the ND-2D methods have been shown in Figure 5.16 to be identical or very similar. The same observation can be made concerning the results of the zonotope/strip intersection and SVD methods. As a general remark, it should be noted that both former methods seem to deliver the tightest approximation, but this is done at the cost of the computation complexity. The resulting zonotopes possess a high number of generators in comparison to the two generators zonotopes obtained with the ND-2D methods.

5.12.2. Experimental Evaluation at Guard Intersection

We discuss, in this section, the results of the reachability analysis on the classic two-tank and the colliding masses systems. We consider invariants and guards on both systems. We aim at investigating the performances of the aforementioned zonotope/hyperplane intersection methods in combination with different clustering strategies.

Flowpipes of the the two-tank system with 100 iterations at the first transition are shown in Figure 5.17 to 5.20 while those of the colliding masses system with 150

iterations are illustration in Figure 5.21 to 5.24. The intersecting zonotope bundle is plotted in green. It should first be noted that the ND-2D projection method using dichotomous search works only for 3-dimensional systems or larger. Consequently, it cannot be used for the two-tank system.

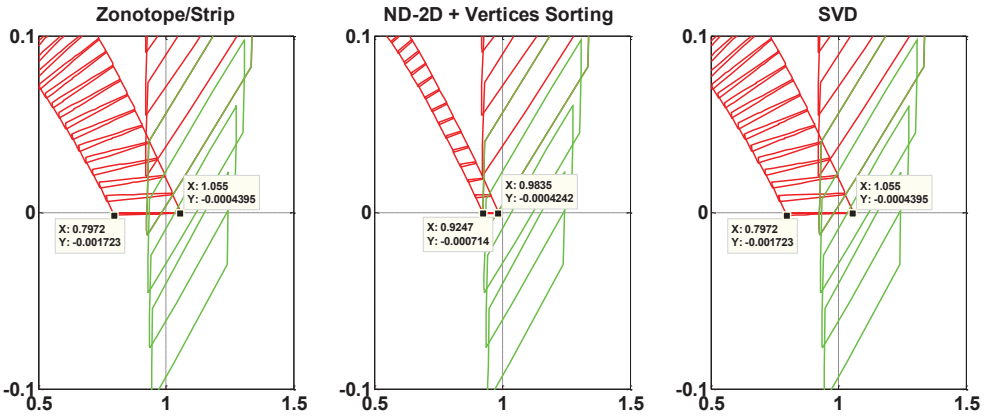


Figure 5.17.: Taking the first intersection found at guard transitions for two-tank system.

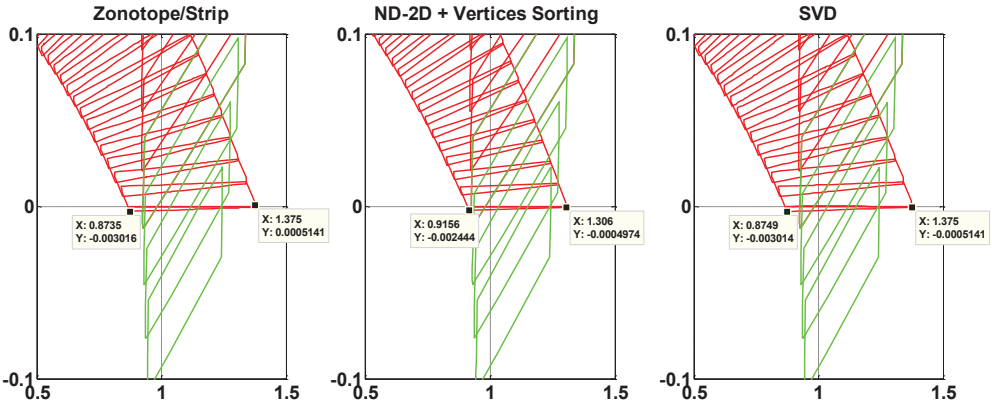


Figure 5.18.: Pre-clustering of intersection bundles for two-tank system.

We generally note that the pre-clustering results in tighter approximations than post-clustering across all zonotope/hyperplane intersection computation methods. We also observe that clustering based on the initial set approximation method provides tight sets in comparison with post-clustering. Among all methods, the ND-2D projection methods perform the best approximation particularly with Min/Max clustering strategy. We furthermore measured computation times which we listed in Table 5.6. Especially noteworthy is the advantage of the ND-2D projection methods as an intersection computation method across all methods despite its apparent

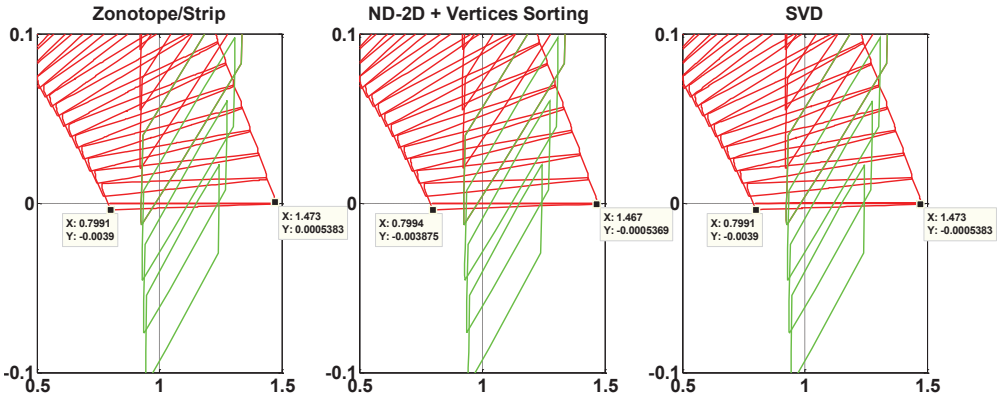


Figure 5.19.: Post-clustering of resulting intersections for two-tank system.

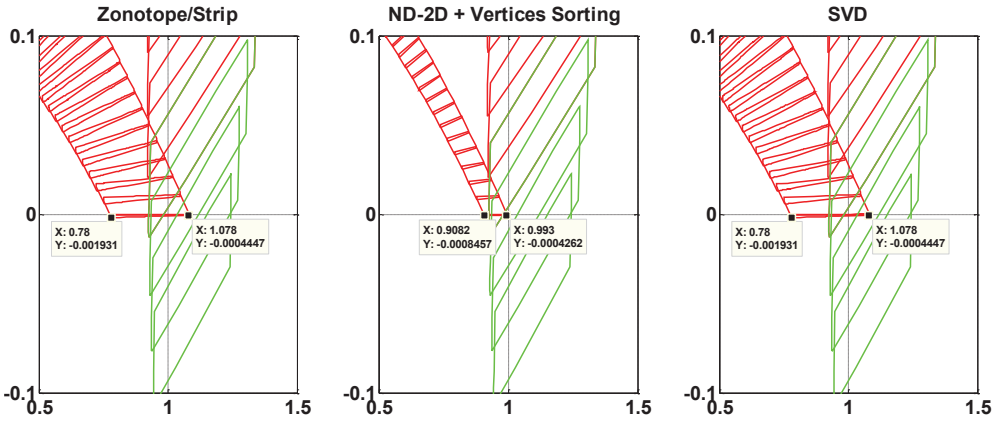


Figure 5.20.: Over-approximation using an approximative convex hull for two-tank system.

sluggishness for intersection computations as show in Table 5.5. In fact, the ND-2D method keeps at most $D-1$ generators for a system of dimension D. This contributes to complexity reduction of zonotopes while computing guard intersections nonetheless at the cost of tightness. However, the zonotope/strip, and to a certain extent, the SVD intersection computation methods, while faster in intersection computation, do not reduce the number of generators of the resulting zonotope. This consequently leads to a steadily increasing complexity. This justifies the disparity in computation times found in the colliding masses system. In fact, the intersection is computed at the guard transition with a zonotopic reachable set of more than 4000 generators. Furthermore, Figure 5.25 shows the number of generators increasing to nearly 60,000 generators due to the initialization of the resulting zonotope for the subsequent continuous dynamics. On the contrary, the two-tank system is shown in Figure 5.25 to deal with a zonotopic reachable set of fewer than 15 generators. It is consequently not significantly affected by the problem of computation with many generators in the continuous dynamics after the guard transition is taken.

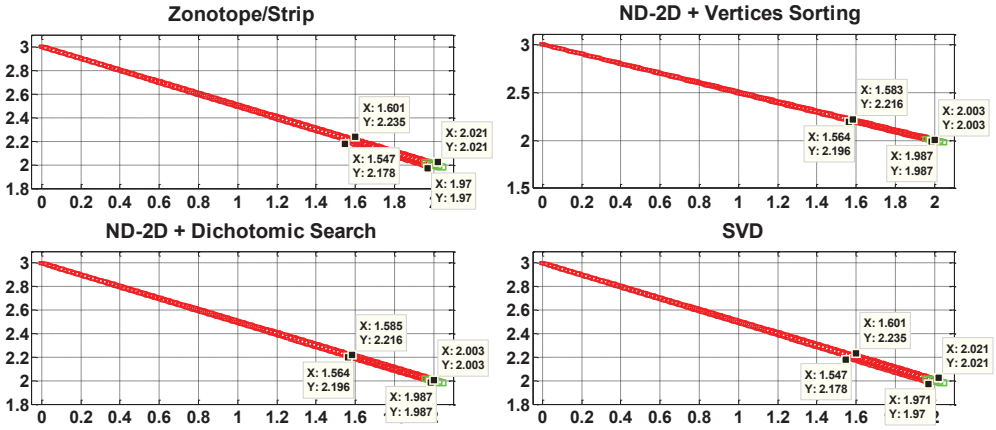


Figure 5.21.: Over-approximation using an approximative convex hull for colliding masses system.

	Two-Tank System (100 iterations)			colliding masses System (120 iterations)			
	Overapprox.	Preclust.g	Postclust.	Overapprox.	Preclust.	Postclust.	Min-Max
Zon./Strip	35603	683.9	768.1	37482	10591	39412	N.A.
Dich. search	N.A.	N.A.	N.A.	6699	4125	3825	3739
Vertices sort.	1133.9	372.2	442.3	9019	5483	4740	5188
SVD	40687	686.1	777.2	844567	60116	73090	N.A.

Table 5.6.: Computation time for reachability analysis on different systems (in ms).

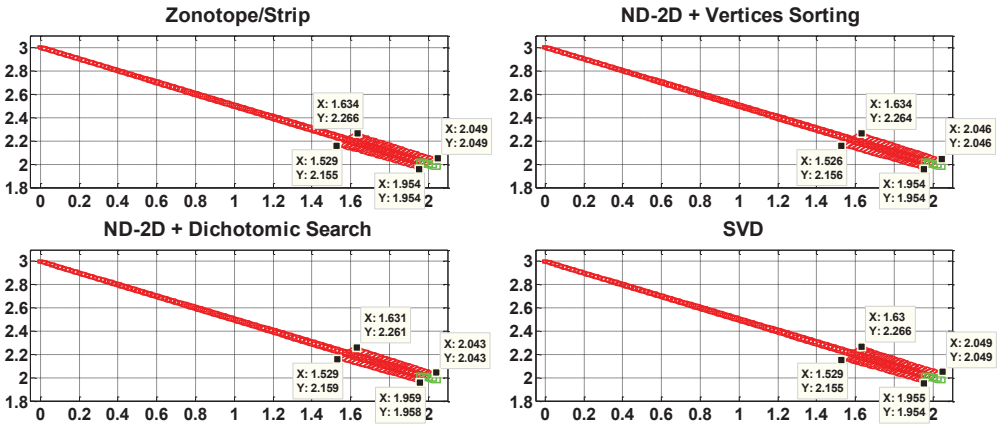


Figure 5.22.: Pre-clustering of intersection bundles for colliding masses system. .

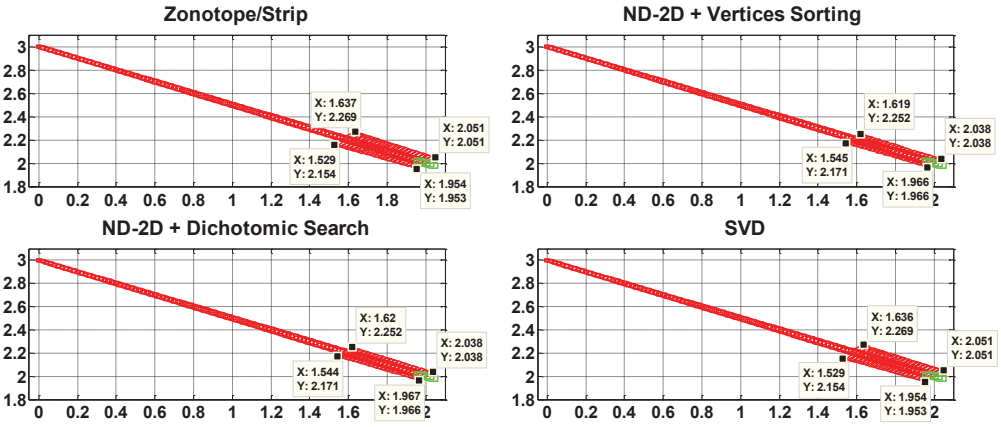


Figure 5.23.: Post-clustering of resulting intersections for colliding masses system.

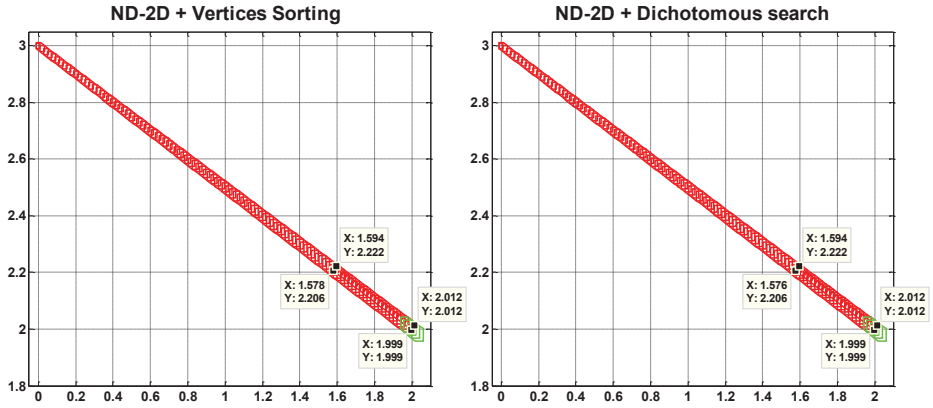


Figure 5.24.: Min/Max of resulting intersections for colliding masses system.

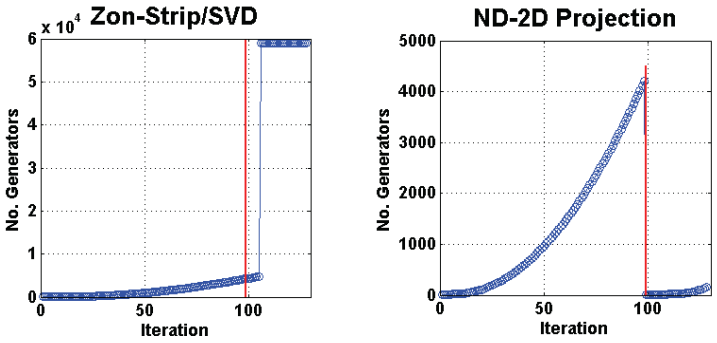


Figure 5.25.: Number of generators in the course of reachability analysis on the colliding masses system varies with the intersection method used. The red vertical lines indicate the iteration in which an intersection with the guard was detected.

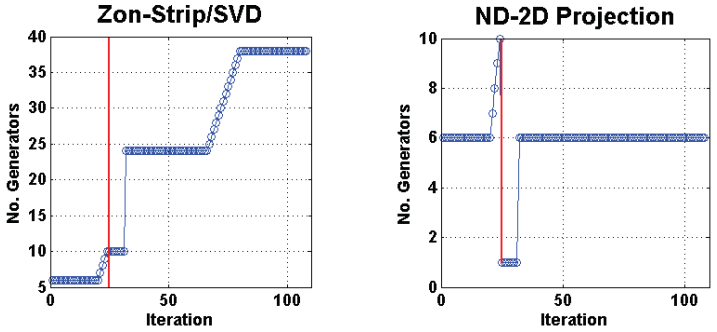


Figure 5.26.: Number of generators in the course of reachability analysis on the Two-Tank system varies with the intersection method used. The red vertical lines indicate the iteration in which an intersection with the guard was detected.

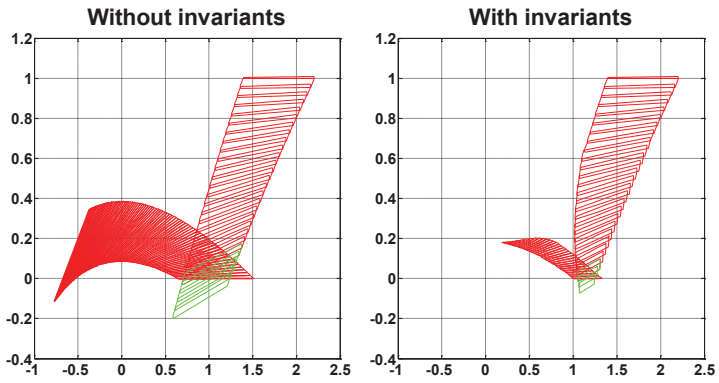


Figure 5.27.: The two-tank system without and with the consideration of invariants respectively.

5.12.3. Handling Invariant

We adopted the zonotope/polyhedron intersection method described in Section 5.6 for handling invariants in discrete mode. To highlight its performance with regard to the tightness, a visual comparison is given in Figure 5.27. This figure shows the flowpipes for the two-tank example (Figure 4.22) without invariants (Figure 5.27(a)) and with an invariant condition $x_1 \geqslant 1$ for state q_1 and $x_2 \leqslant 0.2$ for state q_2 for a more pronounced effect regarding the consideration of invariants (Figure 5.27(a)). We note an over-approximating effect at both invariants.

5.13. Conclusion

In this section, we addressed the problem of reachability analysis based on zonotopes. We reviewed their important properties and geometric operations and particularly explored and described methods for zonotope/hyperplane, zonotope/halfspace, zonotope/polyhedron and zonotope/zonotope intersections. These methods were subsequently assessed with regard to tightness and computation time first in MATLAB and second in C++. In addition, we presented different clustering strategies for handling the bundle of zonotopes intersecting a guard. In a further analysis, we carried out a comparative investigation of the intersection methods combined with the different clustering strategies. The results revealed that depending on the dynamical system, a trade-off must be made between the tightness and complexity. Besides the intersection methods, the clustering strategies and their combination, we extended the C++ implementation with invariants by using the zonotope/polyhedron intersection method.
Generally, zonotopes are more attractive for computing the reachable sets within discrete modes than support functions because they do not require optimization algorithm. However, at intersections with guards or invariants, an approximation is required to recover the zonotopic form from the resulting intersection for the

subsequent iteration. This approximation is often larger than that obtained with support functions.

6. Reachability Analysis of a Networked Platoon of Trucks

6.1. Introduction

As an area of applications, we consider the networked platoon of trucks as a case study. The motivation behind this choice lies mainly in its safety-critical nature and its scalability. We firstly investigate the safety of an LMI-based controlled platoon with the previously described zonotope and support function techniques. We are interested in determining the shorted safe gaps between the trucks where rear-end collisions in case of abrupt braking or loss of communication are ruled out.
Secondly, we address the problem of control design of a scalable platoon. We propose an approach to simplify the control design by ignoring some design criteria which are later checked using reachability techniques.
The resulting scalable platoons are then used to test the capability of the zonotope and support function implementations to deal with large-scaled systems.
Additionally, we demonstrate how reachability analysis can be used to assure stringent time and safety critical requirements, if a platoon of trucks approaches an intersection.

6.2. Description of a Networked Cooperative Platoon

A platoon consists of autonomous driving vehicles following a leader. Each vehicle i is equipped with on-board sensors to capture its actual speed and acceleration as well as the distance d_i from the vehicle ahead. This sensor information is transmitted to the other vehicles via a wireless local area network (WLAN) so that each vehicle i receives the data flow $e_j, \dot{e}_j$ and a_j from all vehicles $j \neq i$. The spacing error e_j is defined as the difference between d_j and a reference distance $d_{j,ref}$. Figures 6.1, 6.2 illustrate the structure and the architecture of the networked platoon of trucks.

6.2.1. Platoon Model

For this application, we adopt the model proposed in [78] where the drivetrain dynamics is approximated by a linear first order filter. The effective acceleration of each truck and the difference of acceleration successive vehicles are given by the

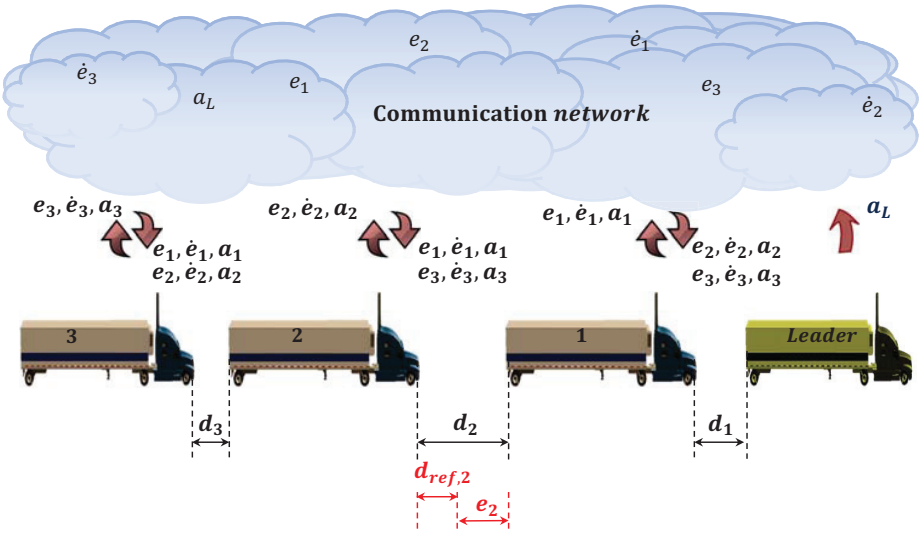


Figure 6.1.: Platoon structure and setup.

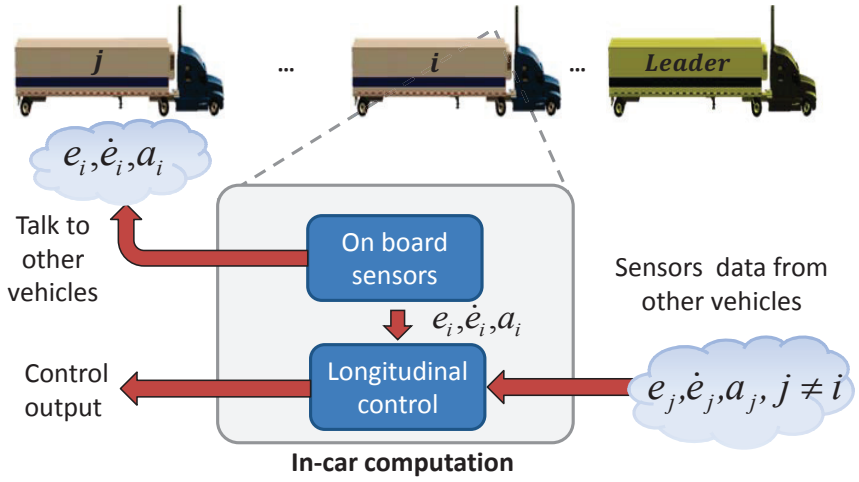


Figure 6.2.: Architecture of the cooperative platoon of vehicles.

following differential equations:

$$\begin{aligned} \dot{a}_i &= -1/T_i \cdot a_i + 1/T_i \cdot u_i, \\ \ddot{e}_i &= a_{i-1} - a_i, \end{aligned} \tag{6.1}$$

where $T_i = \frac{1}{\gamma}$ is the time constant of the drivetrain of vehicle i and u_i the corresponding control input.

If we now take $x = [e_1, \dot{e}_1, a_1, \cdots e_i, \dot{e}_i, a_i \cdots, e_N, \dot{e}_N, a_N]^T \in \mathbb{R}^{3N}$ as state vector and the vector $u = [u_1, \ldots, u_i, \ldots, u_M]^T \in \mathbb{R}^M$ as input for the platoon of N vehicles, its dynamics would then be described with the differential equation

$$\dot{x} = A_s x + B_1 a_L + B_2 u. \tag{6.2}$$

The matrices A_s, B_2 and B_1 are given by the following equation:

$$\begin{bmatrix} \dot{e}_1 \\ \ddot{e}_1 \\ \dot{a}_1 \\ \dot{e}_2 \\ \ddot{e}_2 \\ \dot{a}_2 \\ \dot{e}_3 \\ \ddot{e}_3 \\ \dot{a}_3 \\ \vdots \\ \dot{e}_{N-1} \\ \ddot{e}_{N-1} \\ \dot{a}_{N-1} \\ \dot{e}_N \\ \ddot{e}_N \\ \dot{a}_N \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\gamma & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -\gamma & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\gamma & \cdots & 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & -\gamma \end{bmatrix} \begin{bmatrix} e_1 \\ \dot{e}_1 \\ a_1 \\ e_2 \\ \dot{e}_2 \\ a_2 \\ e_3 \\ \dot{e}_3 \\ a_3 \\ \vdots \\ e_{N-1} \\ \dot{e}_{N-1} \\ a_{N-1} \\ e_N \\ \dot{e}_N \\ a_N \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ \gamma & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ 0 & \gamma & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & \gamma & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & \gamma \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_M \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \ddots \\ 0 \\ 0 \end{bmatrix} a_L \tag{6.3}$$

where a_L is the leader's acceleration.

6.3. Information Flow and Interconnection Topology

The interconnection topology within the platoon is modeled with a directed graph $G = (V, E)$, defined by vertices V and edges E. The i^{th} vertex represents the i^{th}

vehicle and the edge (i, j) indicates that vehicle j receives information from vehicle i. Figure 6.3 describes the general case of bidirectional exchange of information for the platoon of Figure 6.1. This graph is represented by the adjacency matrix $R = [r_{ij}]$ referred to as the *communication matrix* of the platoon. This matrix is defined as follows:

$$r_{ij} := \begin{cases} 1, & \text{if j receives information from i} \\ 0, & \text{otherwise.} \end{cases} \tag{6.4}$$

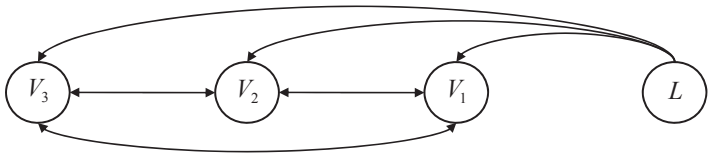


Figure 6.3.: Possible information exchanges inside the platoon.

If we assume sensor data to be always available, the information topology will be then represented by the matrix $T = I + R^T$ where I is the identity matrix.

The interconnection topology is generally time varying because of communication dropouts. The communication matrix is hence a time function $R(t)$. This can be illustrated by Figure 6.4. In this figure, three different interconnection topologies for a platoon of three vehicles are shown. The interconnection topology can spontaneously change in practice. For example, it can jump from (a) to (b) and from (b) to (c) if the communication between V_3 and V_2 and then between V_2 and V_1 are consecutively lost.

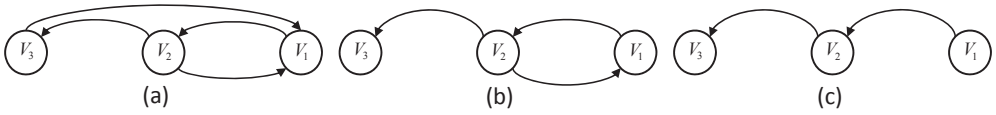


Figure 6.4.: Three different interconnection topologies for a platoon of three vehicles.

6.4. Control Design

The first usual goal of control design is to guarantee the stabilization of the system while assuring a good disturbance rejection in terms of small and safe spacing errors e_i with a reasonable control effort. To mathematically express these control objectives, the following transfer functions and mapping have been defined in [78]

for each vehicle i member of the platoon

$$\begin{array}{l} G_i(s) = \frac{e_i(s)}{e_{i-1}(s)}, \\ F_{v,i}(s) = \frac{(v_i - v_L)(s)}{a_L(s)}, \\ F_{a,i}(s) = \frac{a_i(s)}{a_L(s)}, \\ H : a_L \mapsto (e_1, \ldots, e_N, u_1, \ldots, u_M)^T, \end{array} \tag{6.5}$$

with H being the mapping of a_L to errors and control effort. Apart from the ordinary stability, string stability is required for interconnected vehicle strings to exclude the amplification of the spacing errors upstreams from the start to the end of the platoon. The term $\|G_i\|_\infty < 1$ in (6.5) imposes, however, the stringent condition $e_i \leqslant e_j$ for $i \prec j$, where $\prec$ corresponds to the order of the vehicles in the platoon. Forcing $\|F_{v,i}(s)\|_\infty \leqslant \gamma_v$ and $\|F_{v,i}(s)\|_\infty \leqslant \gamma_a$ assure for bounded velocity error and acceleration of any vehicle as response to any bounded leader acceleration profile. The upper bounds of the overshoots on velocity error and acceleration with respect to the acceleration of the leader a_L remain therefore below $\gamma_v \|a_L\|_\infty$ and $\gamma_a \|a_L\|_\infty$ respectively. The control problem with the previous requirements, is formulated in [78] as a mixed H_2/H_∞ minimization problem:

$$\begin{array}{c} min \ \ (\alpha. \|F\|_\infty + \beta. \|H\|_2 \ ) \ s.t. \\ \|G_i\|_\infty < 1 \ \forall i, \\ \|F_{.,i}\|_\infty < \gamma_. \ \forall i \end{array} \tag{6.6}$$

where F is either $F_{v,i}(s)$, $F_{a,i}(s)$ or a combination of both. The $\|.\|_2$-norm minimization of H, hence, guarantees small distance errors with a low control effort. Solving the problem (6.6) results in a feedback matrix K_i verifying

$$u_i = K_i x \tag{6.7}$$

such that

$$u = Kx. \tag{6.8}$$

Therefore, the closed loop system is described by the following differential equation

$$\begin{array}{rl} \dot{x} & = (A_s + B_2 K)x + B_1 a_L, \\ & = A_{cl}\, x + B_{cl}\, a_L. \end{array} \tag{6.9}$$

6.5. LMI-based Control

In [77, 78], the optimization problem (6.6) is transformed into an LMI formulation by adopting the state space model of the generalized plant P of Figure 6.5.

Assuming $w = a_l$, $z_2 = (e_1, \ldots, e_N, u_1, \ldots, u_M)^T$ and $z_\infty = (a_1, \ldots, a_N)^T$, the system of Figure 6.5 is described with the following equations

$$\begin{array}{rl} \dot{x} & = A_s\, x + B_1\, w + B_2\, u, \\ z_\infty & = C_\infty\, x + D_{\infty,1}\, w + D_{\infty,2}\, u, \\ z_2 & = C_2\, x + D_{2,1}\, w + D_{2,2}\, u. \end{array} \tag{6.10}$$

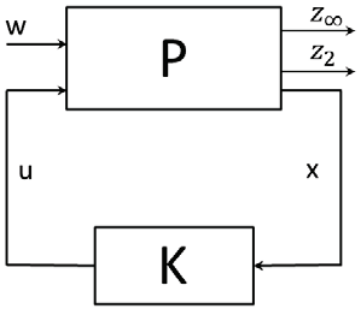


Figure 6.5.: H_2/H_∞ control configuration.

The feedback loop is closed by a controller K solution of the following LMI optimization problem

$$\begin{gathered} min\ \left(\alpha.\gamma^2 + \beta.trace\left(Q\right)\right) s.t. \\ \begin{bmatrix} \left(A_s + B_2K\right)X + X\left(A_s + B_2K\right)^T & B_1 & X\left(C_\infty + D_{\infty,2}K\right)^T \\ B_1^T & -I & D_{\infty,1}^T \\ \left(C_\infty + D_{\infty,2}K\right)X & D_{\infty,1} & -\gamma^2 I \end{bmatrix} < 0, \\ \begin{bmatrix} Q & \left(C_2 + D_{2,2}K\right)X \\ X\left(C_2 + D_{2,2}K\right) & X \end{bmatrix} > 0. \end{gathered} \tag{6.11}$$

where α, β, γ^2 and Q are optimization variables. In [77, 78] a controller for a platoon of three vehicles has been computed based on (6.11). The system matrices are given in Appendix A Section A.8.

To take into account the communication failures in the controller design, the feedback matrix is multiplied by the network topology matrix. This is achieved by forcing zeros in K_i where state information is no longer available. We hence obtain the closed loop system matrices

- for the nominal network communication of Figure 6.3,
- for the flow of information from the start to the end of the platoon shown in Figure 6.6
- for a total outage of communication.

The goal was to determine the conditions under which safety is absolutely guaranteed by carrying out a reachability analysis.

6.5.1. Hybrid Modeling

The main goal of this section is to investigate the impact of the disturbances of the communication network on the performances of the cooperative platoon. We are

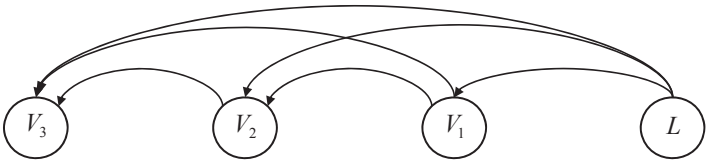


Figure 6.6.: Communication flow in one direction.

particularly interested in worst cases which may occur owing to a loss of communication between two/many or all vehicles. The theory of hybrid systems offers a convenient framework to model this kind of systems. The discrete states describe the continuous dynamics of the controlled platoon generally governed by differential equations in this form

$$\dot{x}(t) = A_q x(t) + B_q u(t) \tag{6.12}$$

where $x(t) \in \mathbb{R}^9$ denotes the state vector, $u(t) = a_L \in [-9, 1]\, ms^2 \subset \mathbb{R}$ is the input vector and $q \in \{1, 2, \ldots\}$ is the discrete mode described by $(A_q, B_q) \in \mathbb{R}^{9\times 9} \times \mathbb{R}^9$.

Communication breakdowns, however, trigger the discrete switches between these states. These discrete events are spontaneous, implying that transitions of this hybrid automaton have no guards. We used our fixpoint triggered transition or alternatively the time-triggered transition to model such kind of transitions. Time-triggered transitions are allowed after about $t = 12s$ to guarantee the enclosure of all reached states. In fact, it was shown in [78] that the controlled platoon reaches its stable state in less than $12s$.

Depending on the topology and the configuration of the communication between vehicles given by the topology matrix T, many communication scenarios are possible. This can consequently lead to a complex hybrid automaton. To simplify the analysis, we focus on some particular and safety-critical worst case scenarios. We firstly consider the scenario of a nominal communication, which then switches to a communication in one direction and finally to a total loss of communication. Secondly, we investigate the case of a direct switching from the nominal communication to complete loss of communication. We are furthermore interested in studying the behavior of the system under arbitrary/chattering between both cases and its impact on the stability of the system. The corresponding hybrid automata are illustrated in Figure 6.7, 6.8 and 6.9 respectively.

6.5.2. Safety Verification of the LMI-based Controlled Platoon

We firstly carried out a reachability analysis using our zonotope implementation with the timed-triggered transition hybrid model. The results of the above-mentioned scenarios with $T = 20s$, $r = 0.01$ and $u \in [-9, 1]\, ms^{-2}$ are shown in Figure 6.10, 6.11 and 6.12. We opted for the time step $r = 0.01s$ as a compromise between precision and time complexity as we noted no significant difference between the

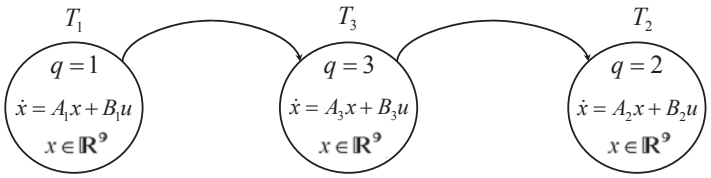


Figure 6.7.: Hybrid automaton: communication → communication in one direction from the start to the end → breakdown of the communication

Figure 6.8.: Hybrid automaton: communication → no communication.

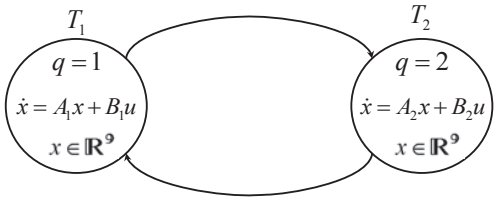


Figure 6.9.: Hybrid automaton: communication → no communication and vice versa.

gap ranges obtained with finer time steps during this study. The outer boundary of each reachable set gives the maximum ranges of the state variables.

As a comparison, we next used the support function based implementation with the fixpoint-triggered transition hybrid model. We chose the *ConstU* scenario with *ConstU initial over-approximation* and the *Oct* set-angle option. The results are illustrated in Figure 6.13, 6.14 and 6.15. The computed fixpoints are plotted in light blue.

The gap ranges resulting from both implementations are summarized in Table 6.1.

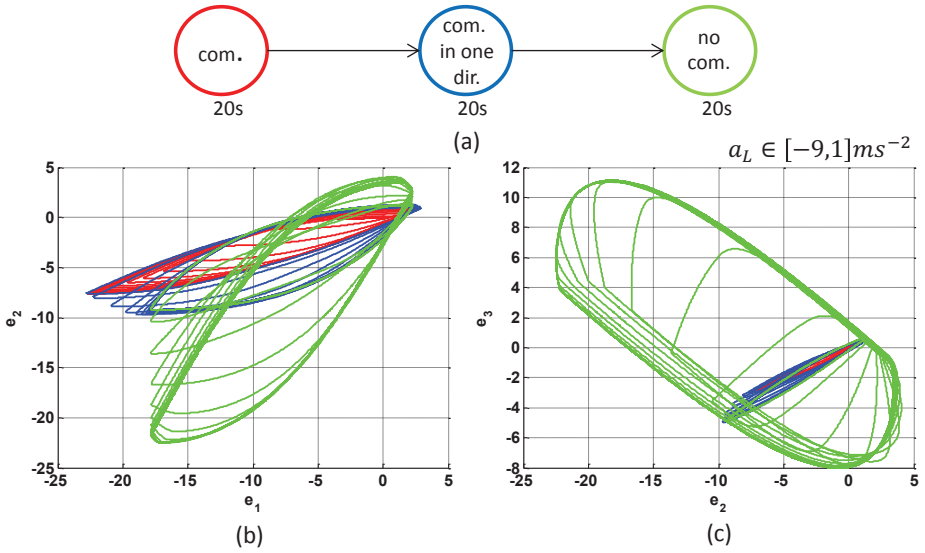


Figure 6.10.: Reachable set projections of the hybrid automaton of Figure 6.7 computed using zonotopes with $T = 20s$ and $r = 0.01$.

Scenario of	Figure 6.7		Figure 6.8		Figure 6.9 (5 transitions)	
Geo. pres.	Zon.	Sp.	Zon.	Sp.	Zon.	Sp.
$e_1[m]$	$[-22.8, 2.9]$	$[-22.8, 3.0]$	$[-24.4, 3.2]$	$[-25.6, 3.8]$	$[-25.6, 3.8]$	$[-28.6, 4.4]$
$e_2[m]$	$[-22.5, 4.1]$	$[-22.5, 5.0]$	$[-22.6, 4.6]$	$[-25.5, 6.2]$	$[-25.6, 6.0]$	$[-25.5, 6.1]$
$e_3[m]$	$[-8.0, 11.2]$	$[-8.5, 11.2]$	$[-8.4, 11.3]$	$[-9.4, 13.0]$	$[-9.7, 13.1]$	$[-10.8, 13.6]$

Table 6.1.: Gap ranges for different scenarios obtained with zonotope and support function based reachability analysis for $T = 20s$ and $r = 0.01s$.

The results of Figure 6.10 and Figure 6.11 show increasing gaps after each loss of information due to outages in the communication between vehicles inside the platoon. The gap between the first vehicle and the leader is the most affected and the second most affected is the gap between the first and the second vehicle. In general, we notice an attenuation of this effect from the start to the end of the

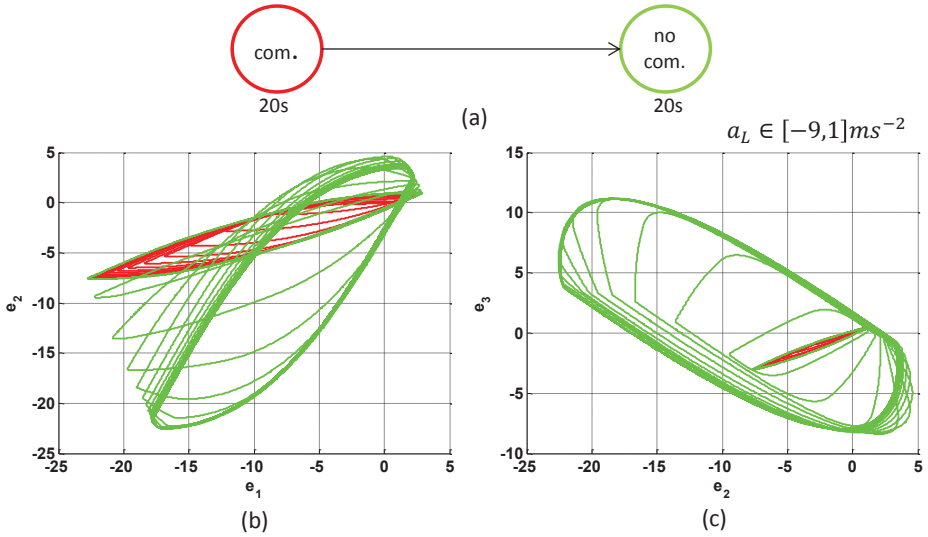


Figure 6.11.: Reachable set projections of the hybrid automaton of Figure 6.8 computed using zonotopes with $T = 20s$ and $r = 0.01$.

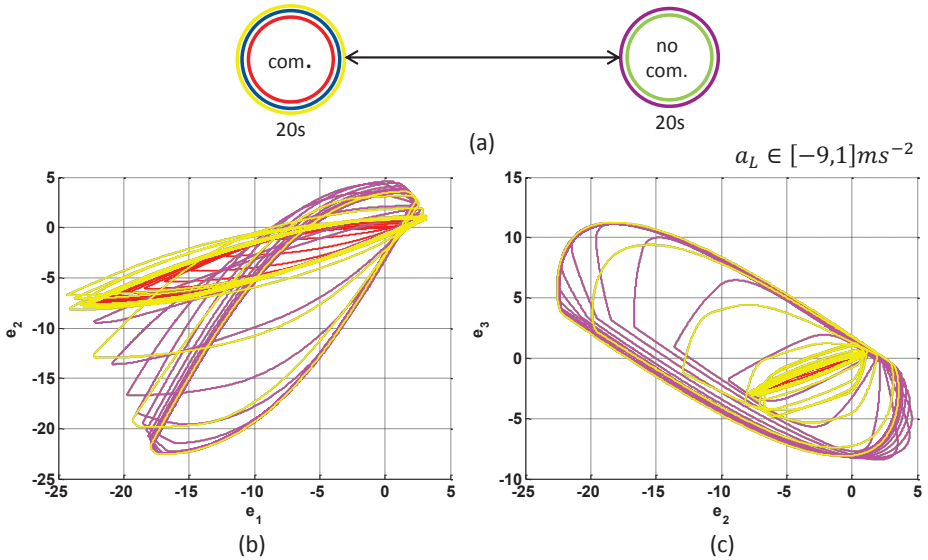


Figure 6.12.: Reachable set projections of the hybrid automaton of Figure 6.8 computed using zonotopes with $T = 20s$, $r = 0.01$ and under several switching.

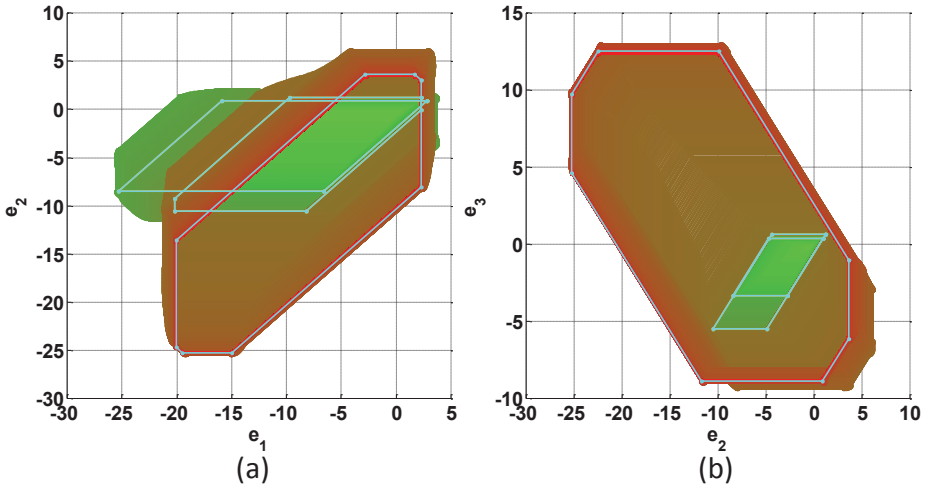


Figure 6.13.: Reachable set projections of the hybrid automaton of Figure 6.7 computed using support functions with $T = 20s$, $r = 0.01s$ and fixpoint-triggered transitions.

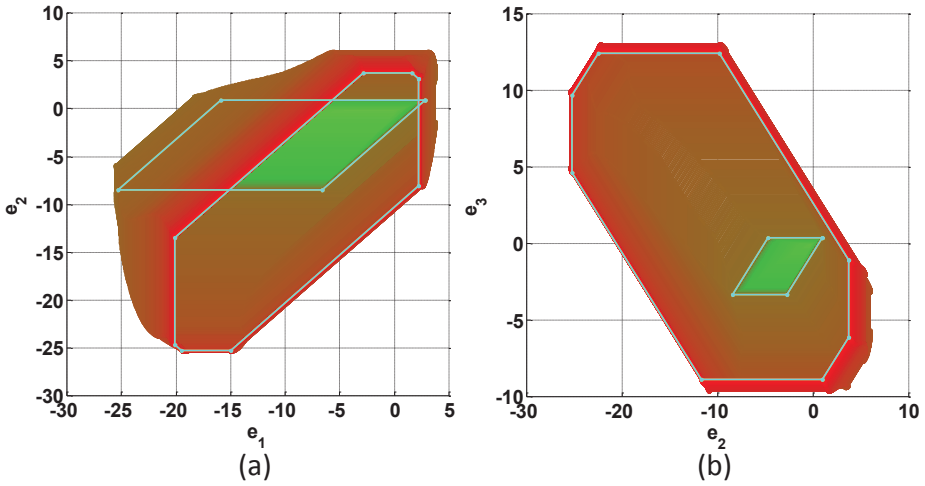


Figure 6.14.: Reachable set projections of the hybrid automaton of Figure 6.8 computed using support functions with $T = 20s$, $r = 0.01s$ and fixpoint-triggered transitions.

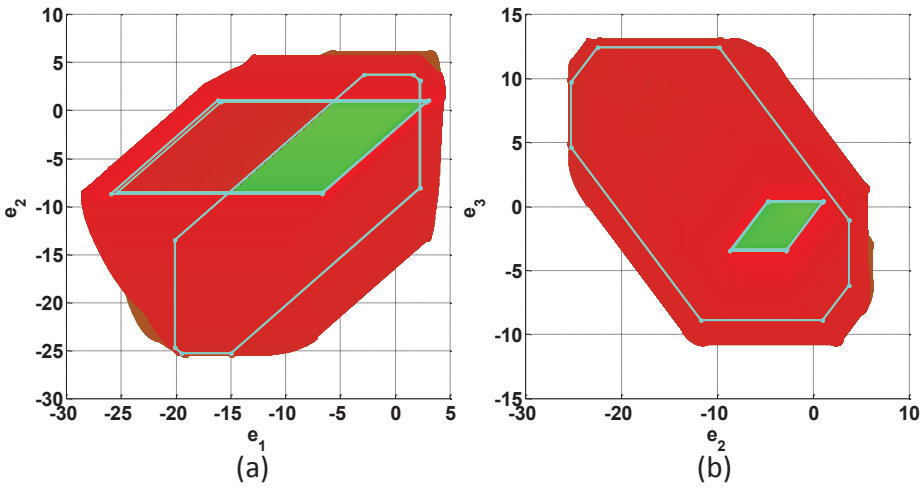


Figure 6.15.: Reachable set projections of the hybrid automaton of Figure 6.8 computed using support functions with $T = 20s$, $r = 0.01s$, fixpoint-triggered transitions and under several switching.

platoon.

Table 6.1 reveals that the zonotope based implementation results are more precise than those obtained from the support function implementation. Zonotopes guarantee collision-free platooning under the condition $d_{1,ref} \geqslant 24.4m$, $d_{2,ref} \geqslant 22.6m$, $d_{3,ref} \geqslant 8.4m$ whereas this condition is $d_{1,ref} \geqslant 28.6m$, $d_{2,ref} \geqslant 25.5m$, $d_{3,ref} \geqslant 10.8m$ for support functions. We also note that chattering between communication and the lack of communication do not really affect the stability of the system. However, in this case, safe gaps must be larger (see Table 6.1, Figure 6.15 and Figure 6.12).

6.6. H_2/H_∞-based Control

In this section, we address the control design of a scalable platoon of vehicles with the help of reachability analysis. As the number of vehicles inside the platoon increases, finding a solution for the corresponding LMI-problem (6.6) becomes more challenging. Our strategy consists in considering only the H_2 [75] or the H_∞ component for the design of the controller which are easily scalable. Other control requirements are then tested using reachability analysis of the closed loop system. Profiting from the block structure of the system description in (6.3), a linear quadratic regulator (LQR)-control design is applied to treat the H_2 component of the control problem and consequently computes the feedback matrix K. LQR-control design is based on the minimization of the following cost function

$$J(u) = \int_0^\infty \left[x^T Q x + u^T R u\right] dt \tag{6.13}$$

where $Q \geqslant 0$ and $R > 0$ are adequately chosen matrices. The control input is then given by the following equation

$$u = -R^{-1}B_2^T Px \tag{6.14}$$

where the matrix P is a positive semidefinite solution of the Riccati equation

$$PA_s + A_s^T P - Q + PB_2R^{-1}B_2^T P = 0. \tag{6.15}$$

The choice of the identity matrix I for weighting matrices Q and R led to the results of Figure 6.16. We note that with this simple choice, the controller met the control requirements for platoons of length 3, 5, 10, 15 and 20. However, testing with different values of the weighting matrix Q led to the conclusion that the best performances are obtained with the controller computed with $Q = 1000.I$ even in the case of a loss of communication as demonstrated in Figure 6.18 for a platoon of 3 vehicles.

Although only the problem of a steep decrease in the errors with a minimum control effort was tackled with the H_2 component of problem (6.6), the results of Figure 6.16 show that an adequate choice of control parameters could limit the overshoots in the distance and in the velocity with an acceptable system time response even for large-scaled platoon like the examples of Figure 6.16(d) or Figure 6.17.

If we now consider only the H_∞ component of the control problem and use the general control configuration of Figure 6.5 described by equations (6.10), a central sub-optimal controller can be computed by solving the so-called H_∞-like Riccati equation.

$$PA_s + A_s^T P + C_\infty^T C_\infty + P(\gamma^{-2}B_1B_1^T - B_2B_2^T)P = 0. \tag{6.16}$$

The state feedback is then

$$u = -B_2^T Px, \tag{6.17}$$

where the matrix P is a positive semidefinite solution of the Riccati equation (6.16).

The results obtained based on H_∞ control design with $\gamma = 8$ are illustrated in Figure 6.19 for a platoon of 3, 5, 10 and 15 vehicles. We note that the overshoots in the distance and velocity remain in an acceptable range with $\gamma = 8$. The convergence time to the steady state is shown, however, to be dependent on the platoon length. It is noteworthy here that testing with different values of γ for each platoon length leads in the most of the cases to the desired controller. For example, Figure 6.20 shows that the controller obtained with $\gamma = 6.6$ performs better with regards to overshoots than those computed with γ =9, 8 or 7 for a platoon of 20 vehicles. However, unlike the H_2-control, the sub-optimal γ depends on the platoon length. Figure 6.21, for example, shows the responses of a platoon of 3 vehicles obtained with $\gamma = 3$ in different communication scenarios. Intensive tests have revealed this choice to be the best performing for a platoon of length 3.

In a next step, reachability analysis is performed to verify if these performance criteria are guaranteed under uncertainties in input as well as disturbances in the communication between the vehicles.

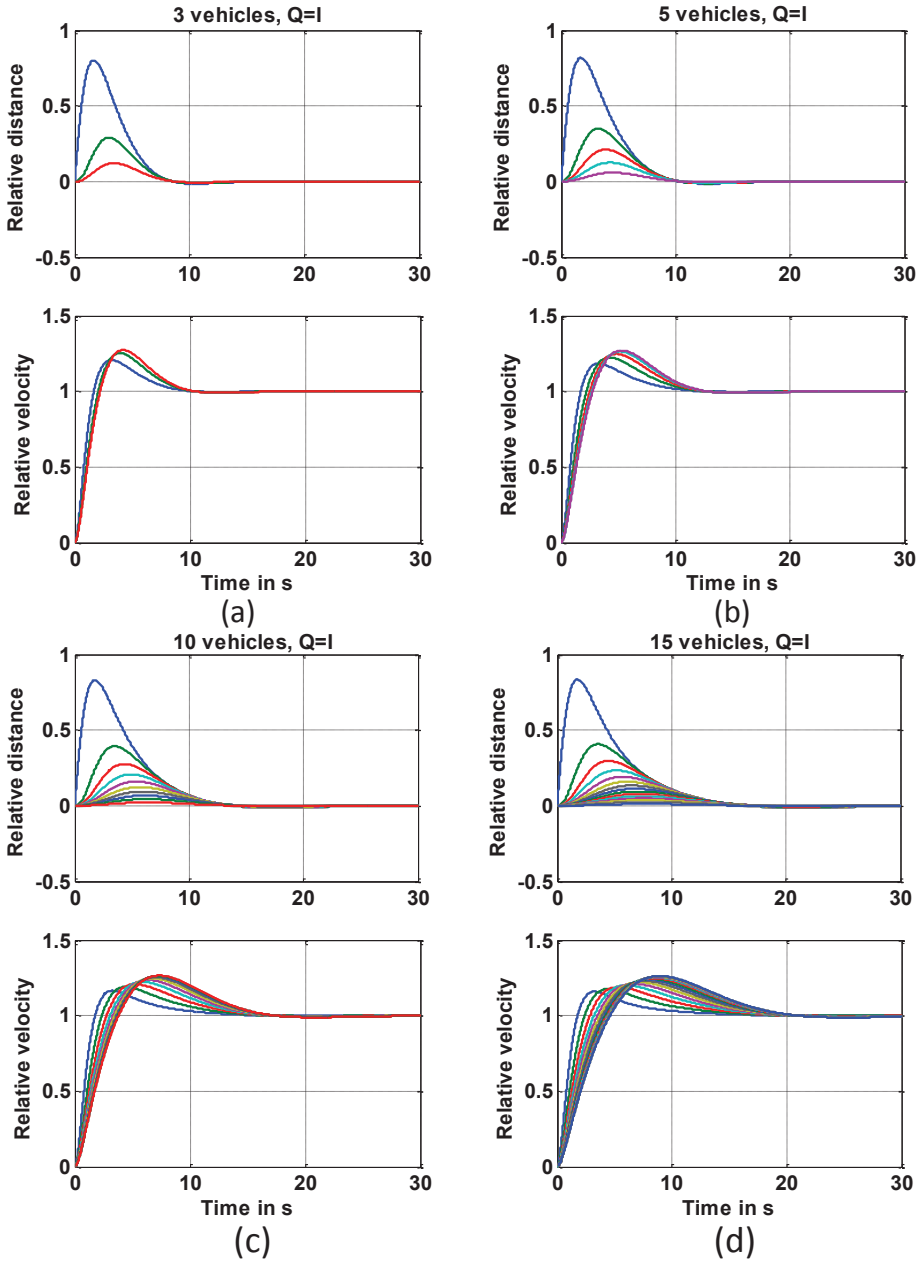


Figure 6.16.: The responses of a H_2-based controlled platoon of 3, 5, 10, 15 trucks to a leader $1m/s$ step form velocity with the $Q = I$.

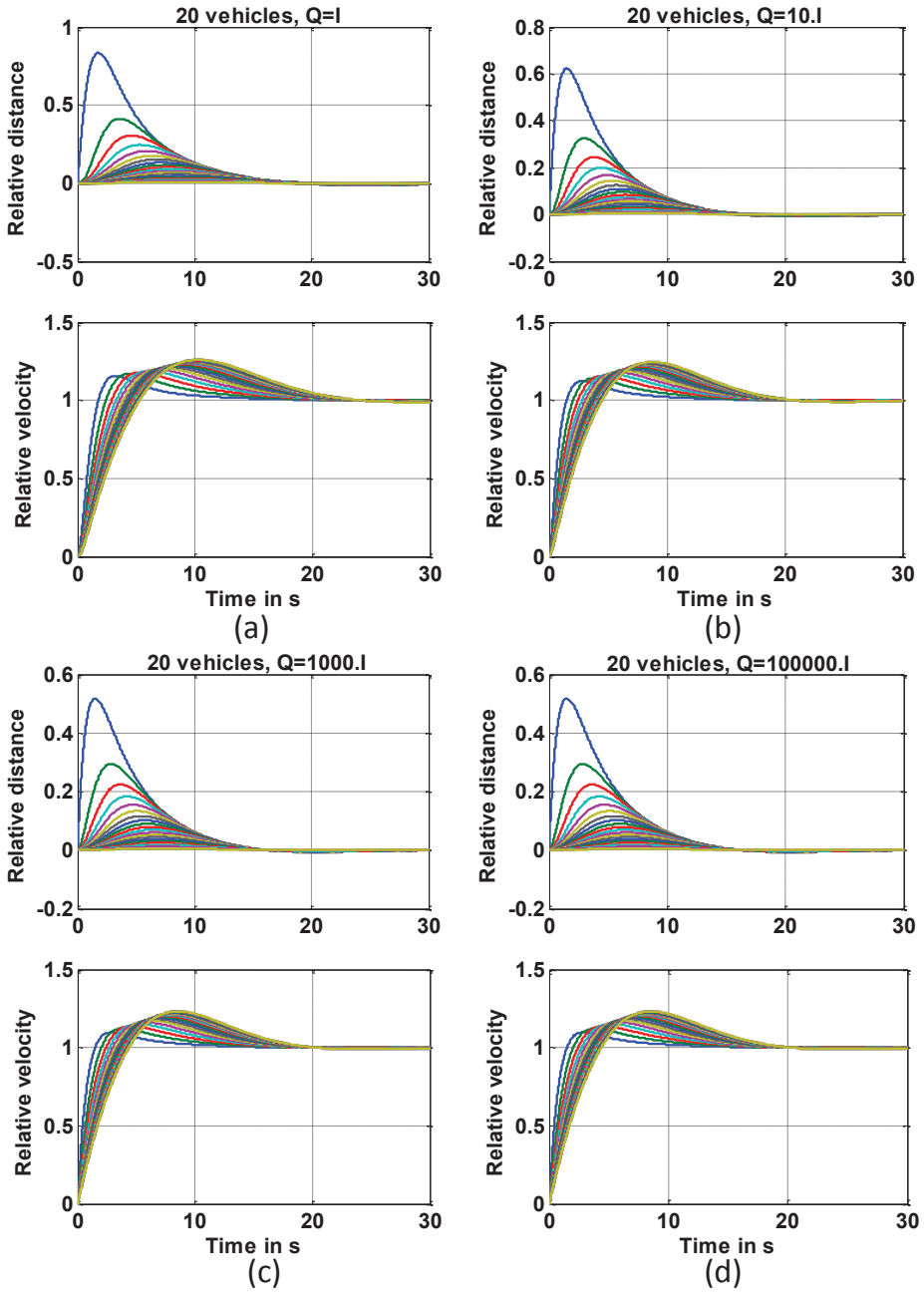


Figure 6.17.: The responses of a H_2-based controlled platoon of 20 trucks to a leader $1m/s$ step form velocity with different weighting matrices Q (a) $Q = I$, (b) $Q = 10.I$, (c) $Q = 1000.I$, (d) $Q = 100000.I$,.

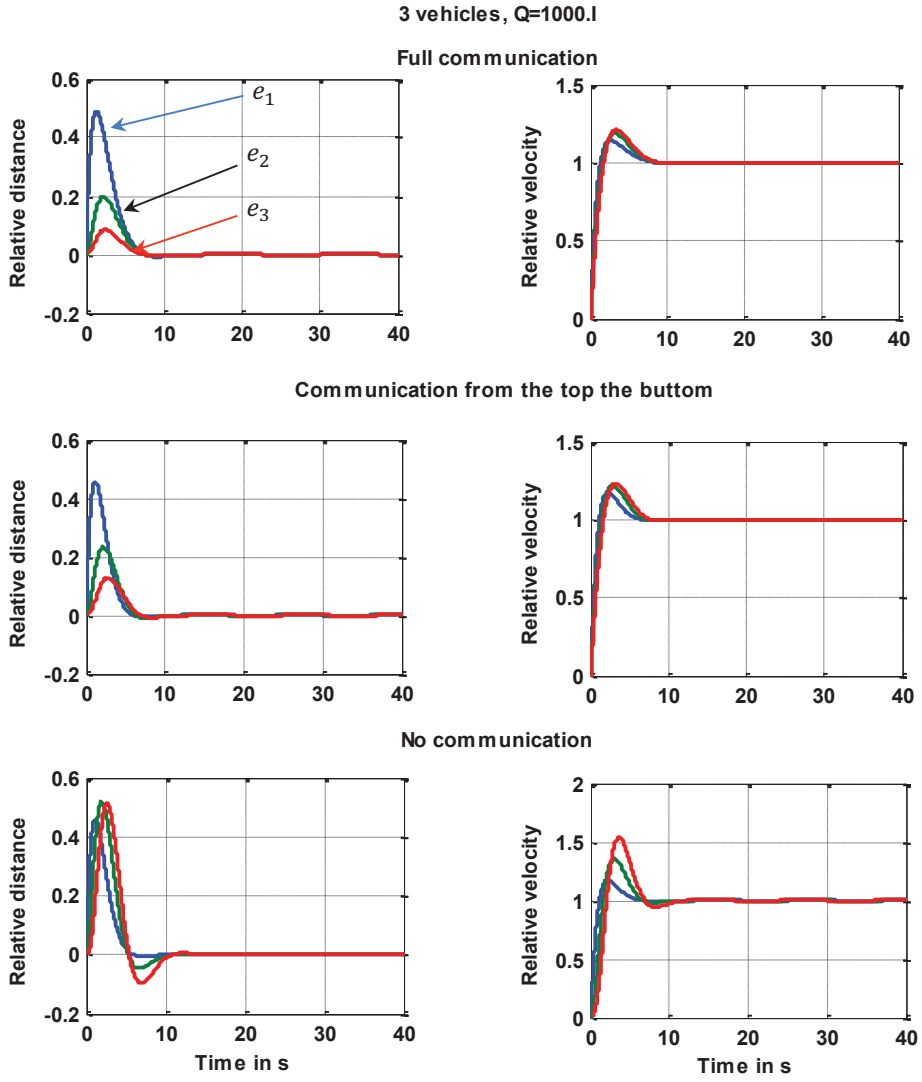


Figure 6.18.: The responses of a H_2-based controlled platoon of 3 trucks computed with $Q = 1000.I$ to a leader $1m/s$ step form velocity in cases of loss of communication.

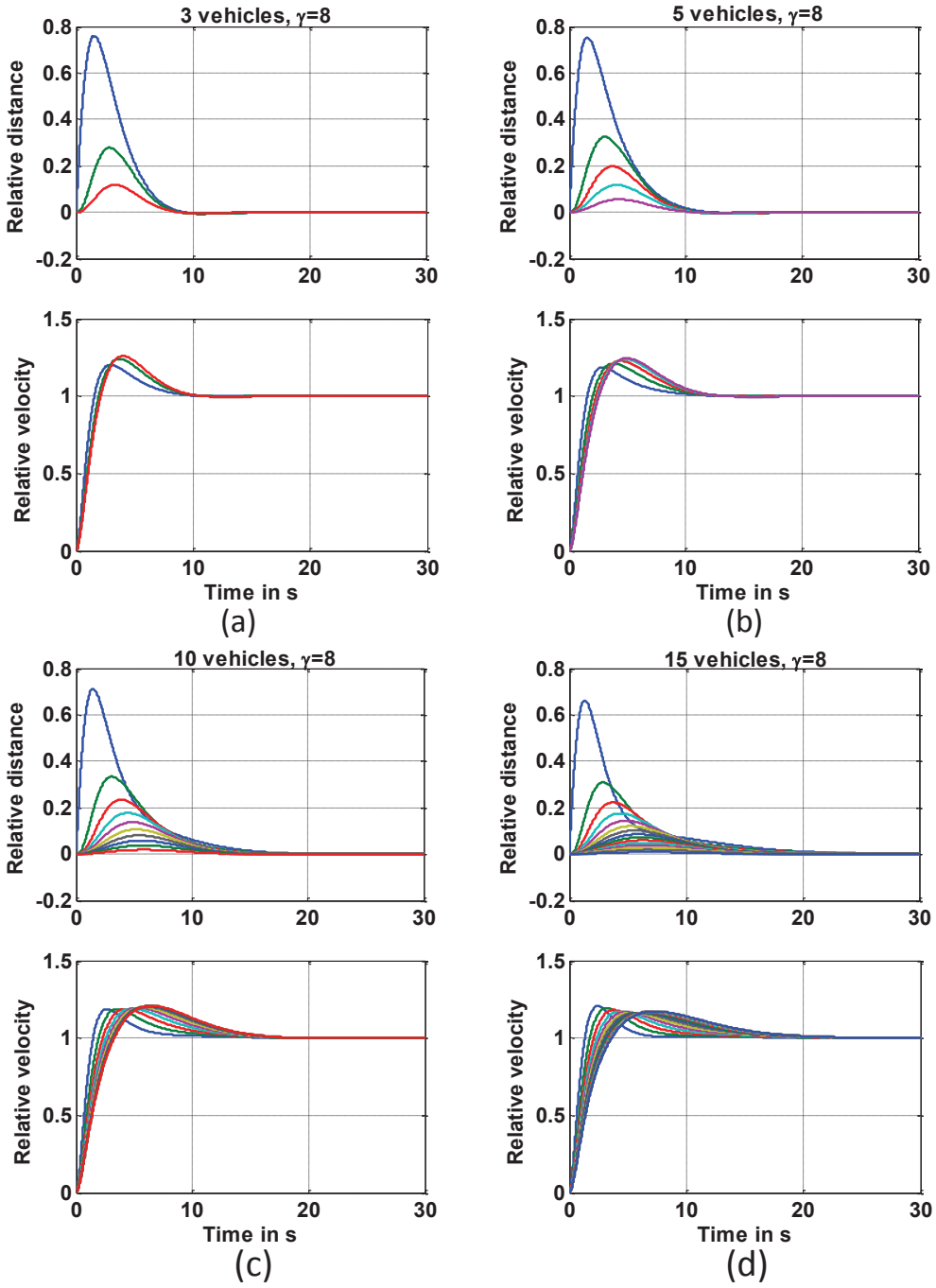


Figure 6.19.: The responses of a H_∞-based controlled platoon of 3, 5, 10, 15 trucks to a leader $1m/s$ step form velocity with a $\gamma = 8$.

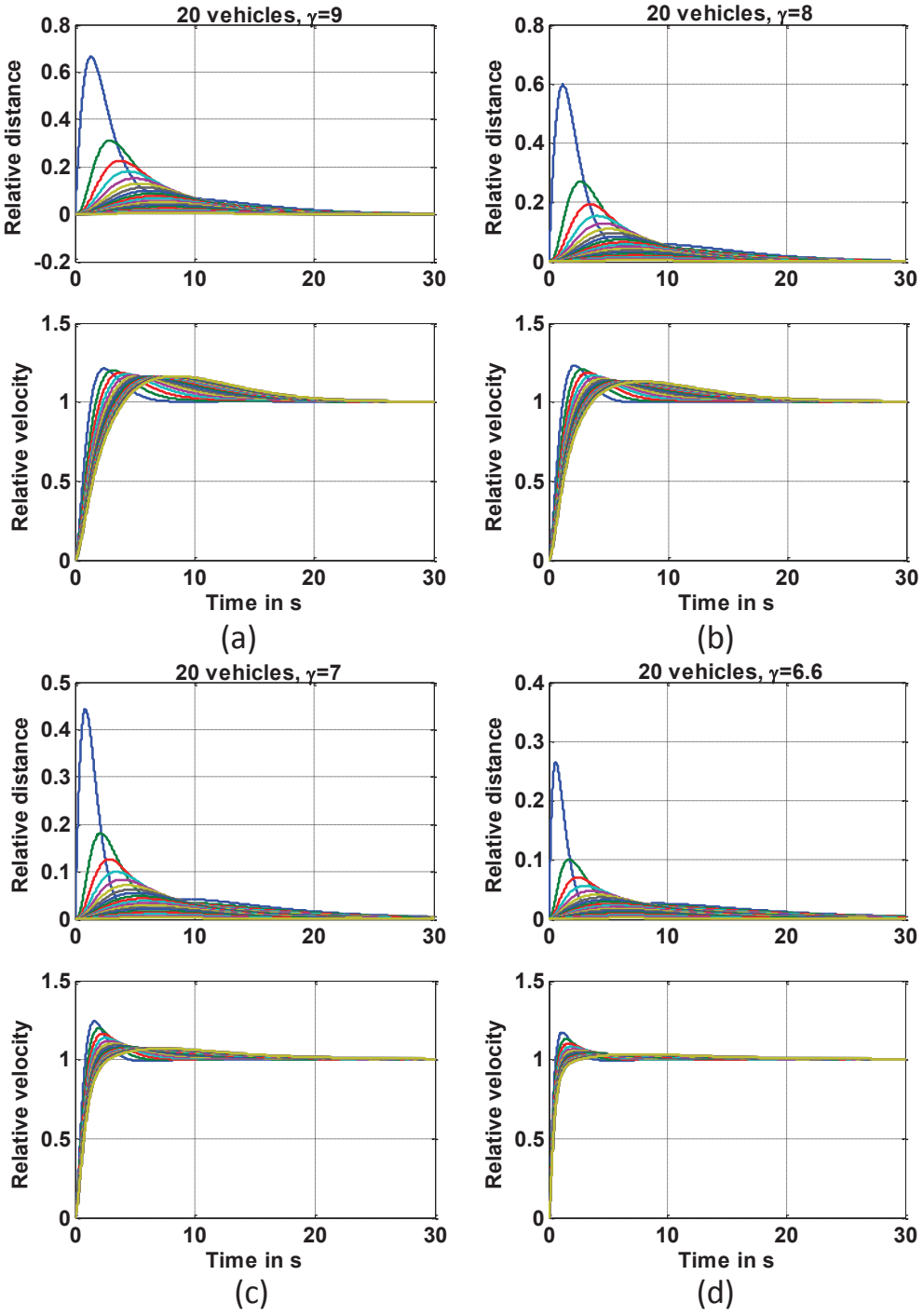


Figure 6.20.: The responses of a H_∞-based controlled platoon of 20 trucks to a leader $1m/s$ step form velocity with different γ values. (a) $\gamma = 9$, (b) $\gamma = 8$, (c) $\gamma = 7$, (d) $\gamma = 6.6$.

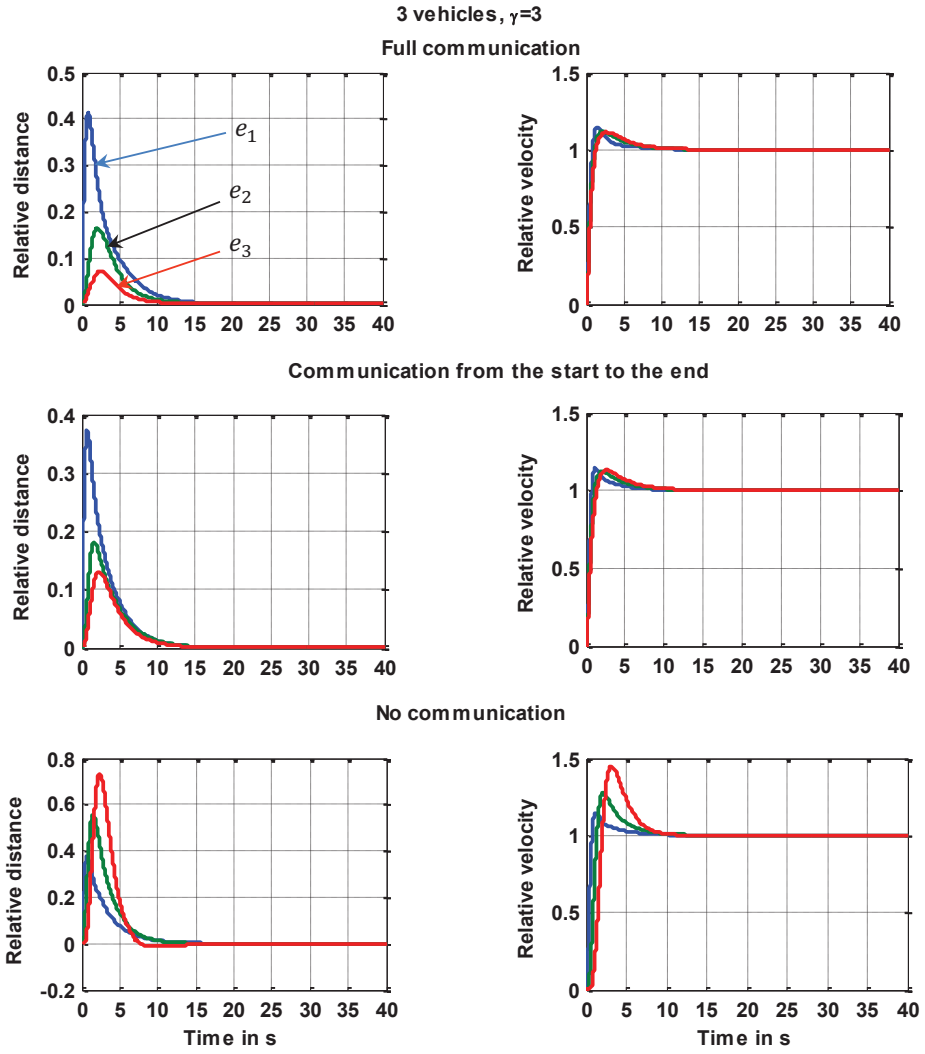


Figure 6.21.: The responses of a H_∞-based controlled platoon of 3 trucks computed with $\gamma = 3$ to a leader $1m/s$ step form velocity in cases of loss of communication.

6.6.1. Reachability Analysis

Clearly, it is difficult to make a better controller choice based only on simulation results, it is impossible to consider for all possible initialization scenarios. However, this is possible through reachability analysis. In addition, like the case of LMI-based controller, declining velocity overshoots and string stability are ignored during the control design to reduce the complexity. The checking of these requirements can furthermore be performed using reachability results of the closed loop controlled platoons.

Q	I	100.I	1000.I	100000.I
x_1	$[-27.4, 3.6]$	$[-15.3, 1.9]$	$[-14.6, 1.8]$	$[-14.3, 1.8]$
x_2	$[-8.1, 8.1]$	$[-5.1, 5.2]$	$[-4.9, 5.0]$	$[-4.8, 4.9]$
x_3	$[-11.1, 3.1]$	$[-10.5, 2.5]$	$[-10.3, 2.4]$	$[-10.3, 2.4]$
x_4	$[-10.8, 1.6]$	$[-6.6, 0.9]$	$[-6.5, 0.9]$	$[-6.4, 0.9]$
x_5	$[-3.0, 3.0]$	$[-2.1, 2.1]$	$[-2.1, 2.1]$	$[-2.0, 2.0]$
x_6	$[-11.6, 3.6]$	$[-2.1, 2.1]$	$[-10.9, 2.9]$	$[-10.8, 2.9]$
x_7	$[-4.7, 0.8]$	$[-2.9, 0.4]$	$[-2.8, 0.4]$	$[-2.8, 0.4]$
x_8	$[-1.3, 1.3]$	$[-0.9, 0.9]$	$[-0.9, 0.9]$	$[-0.9, 0.9]$
x_9	$[-11.8, 3.8]$	$[-11.2, 3.2]$	$[-11.1, 3.1]$	$[-11.0, 3.1]$

Table 6.2.: Gap ranges of a H_2-based controlled platoon of 3 vehicles obtained with support function based reachability analysis for $T = 30s$, $r = 0.01s$ and different weighting matrices Q.

In this section, we present only a small part of the produced results which we deemed to be sufficient to demonstrate how reachability can aid by control design of large-scale systems. For this purpose, we use our support function implementation with the same options as with the LMI-based platoon but with the time horizon $T = 30s$. We choose a narrow neighborhood of the origin as initial set and the input to vary constant-wise in the interval $[-9, 1]\, ms^{-2}$. We first begin with the choice of the weighting matrix Q. Table 6.2 lists the interval hulls of reachable sets for different state variables in case of nominal communication for controllers obtained with the weighting matrices $Q = I$, $100.I$, $1000.I$ and $100000.I$. The variables x_1, x_4 and x_7 thereby correspond to the gaps e_1, e_2 and e_3 in meters $[m]$. We remark that overshoots in the velocity and acceleration remain in acceptable ranges. String stability in the form of distance attenuation from the start to the end of the platoon is also assured. In case of outages of the communication between the vehicles, we show that the gaps increase. This is achieved by testing the scenario depicted by Figure 6.9 with many transitions between nominal and total loss of communication. This is likely due to over-approximation and arithmetic errors and illustrated in Figure 6.22 in comparison with Figure 6.23.

We note that the controllers obtained with $Q = 1000.I$ and $Q = 100000.I$ provide the shortest gaps and that both exhibit similar performances.

The same strategy as above is used to make the best controller parameter choice

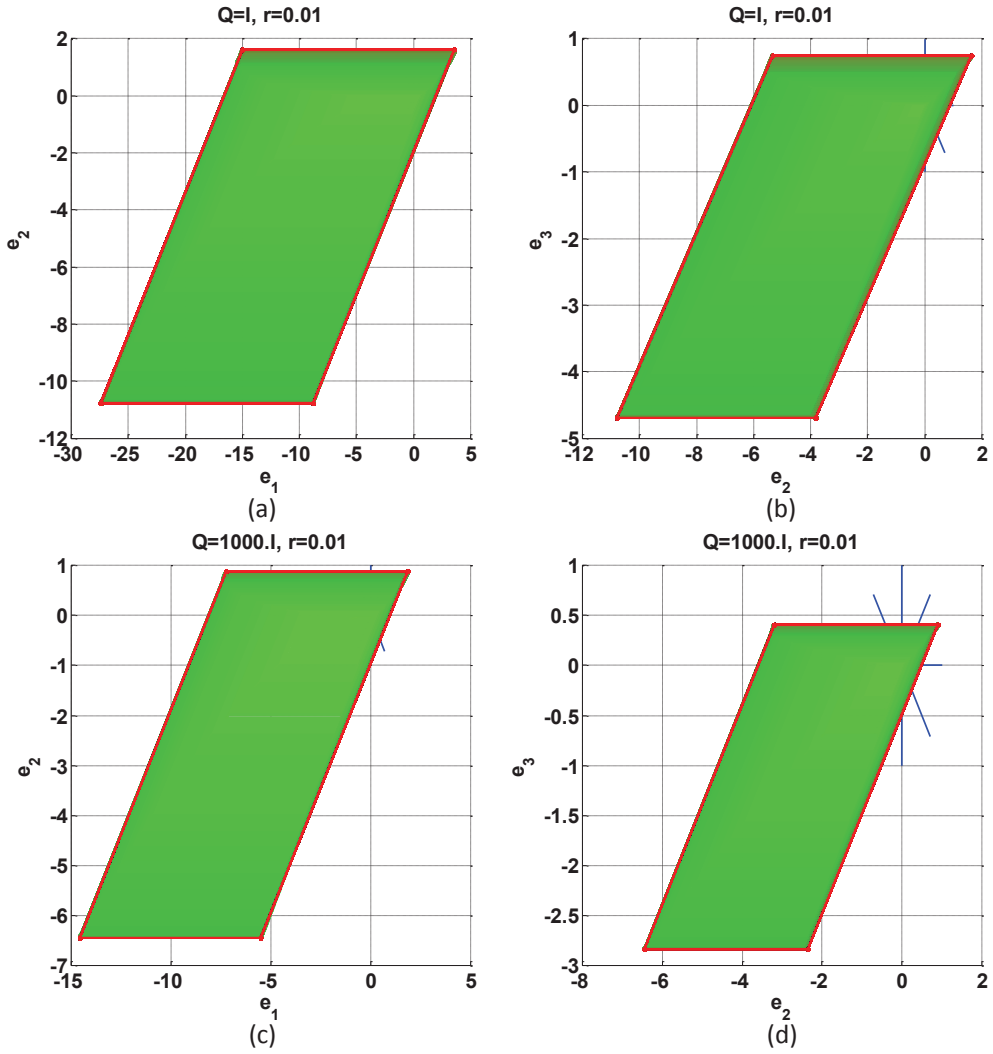


Figure 6.22.: The reachable sets of a H_2-based controlled platoon of 3 trucks computed with $Q = I$ and $Q = 1000.I$ in case of nominal communication scenario.

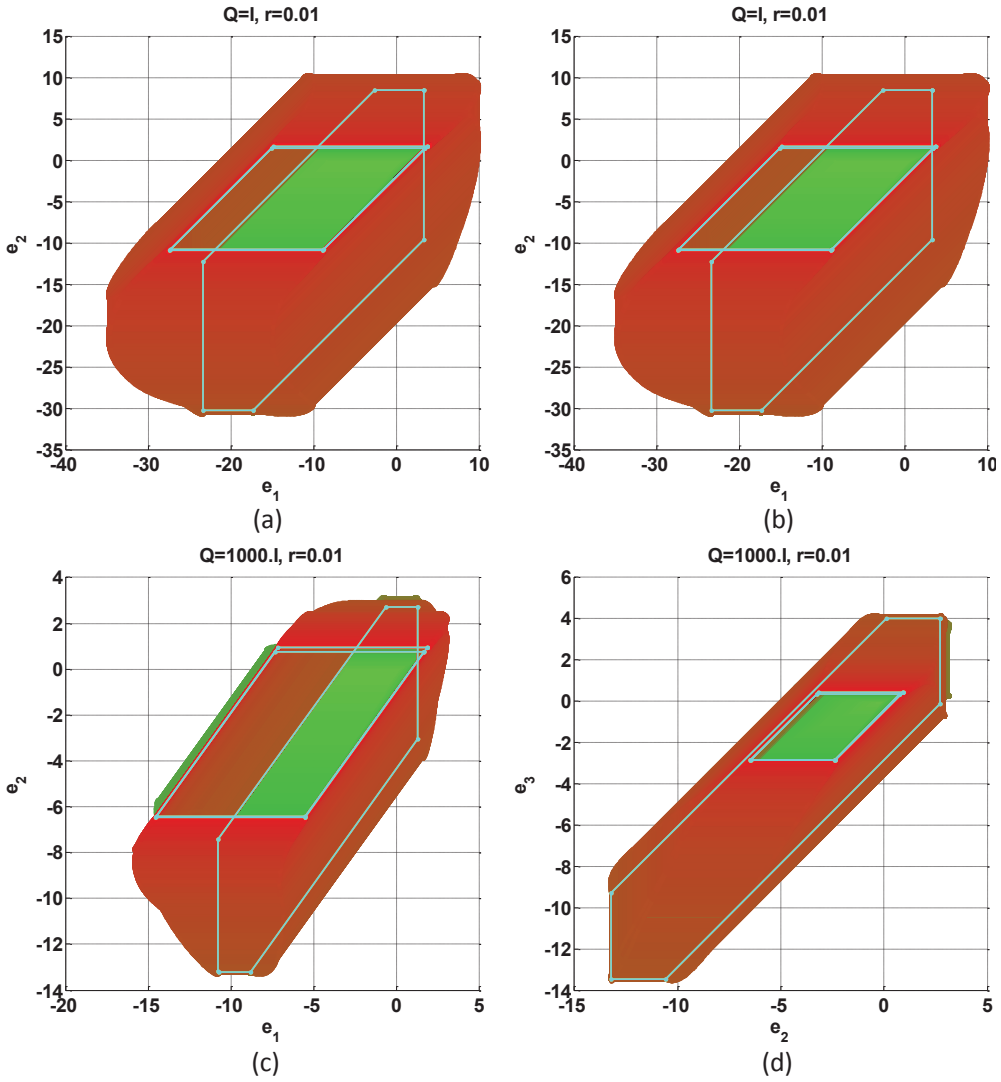


Figure 6.23.: The reachable sets of a H_2-based controlled platoon of 3 trucks computed with $Q = I$ and $Q = 1000.I$ for the communication scenario of Figure 6.9 with fixpoint triggered transitions and a number of transitions equal to 3.

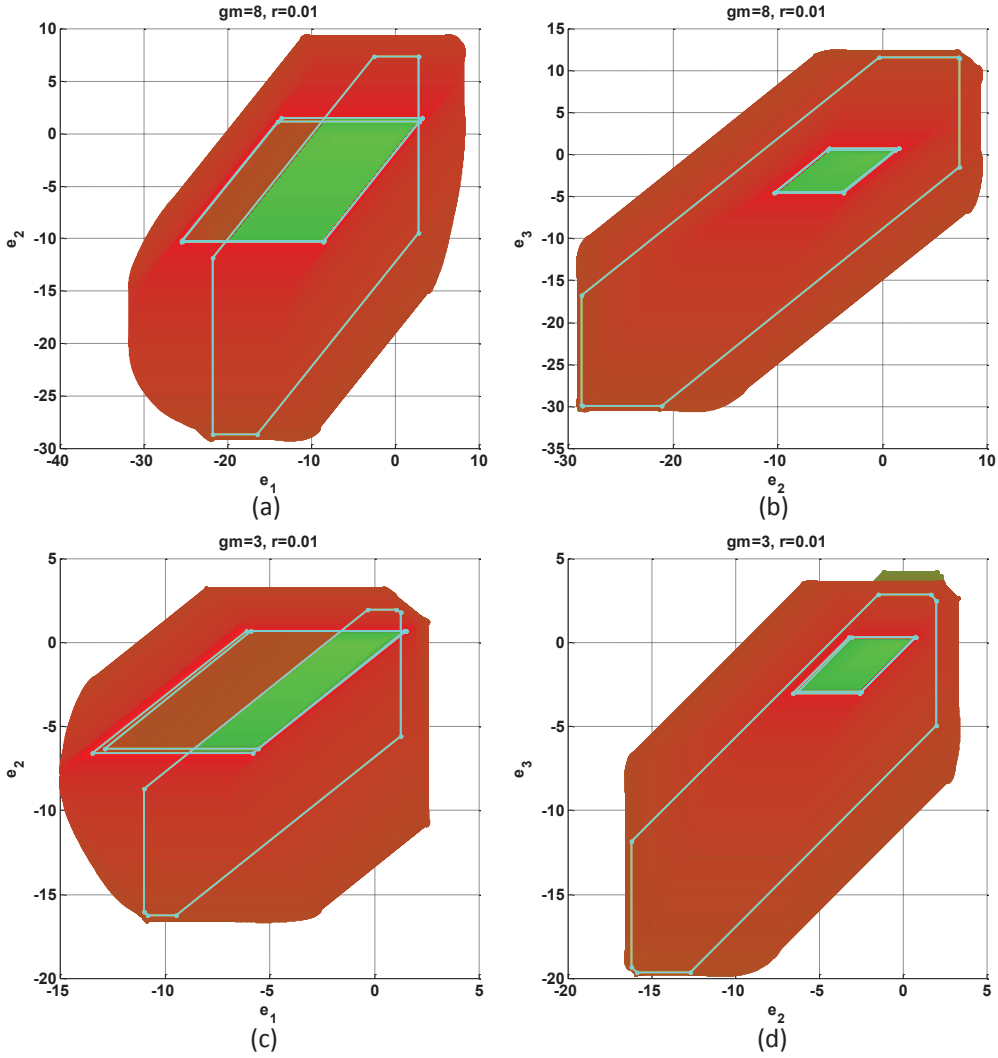


Figure 6.24.: The reachable sets of a H_{infty}-based controlled platoon of 3 trucks computed with γ for the communication scenario of Figure 6.9 with fixpoint triggered transitions and a number of transitions equal to 3.

γ	8	7	6	5	4	3	3.5
x_1	$[-25.4, 3.1]$	$[-24.8, 2.9]$	$[-23.9, 2.7]$	$[-22.4, 2.5]$	$[-19.7, 2.2]$	$[-13.2, 1.5]$	$[-17.3, 1.9]$
x_2	$[-7.6, 7.6]$	$[-7.4, 7.5]$	$[-7.2, 7.3]$	$[-6.8, 6.8]$	$[-6.1, 6.1]$	$[-4.0, 4.1]$	$[-5.4, 5.4]$
x_3	$[-11.0, 3.0]$	$[-11.0, 3.0]$	$[-11.0, 3.0]$	$[-10.9, 2.9]$	$[-10.8, 2.9]$	$[-10.5, 2.5]$	$[-10.8, 2.8]$
x_4	$[-10.2, 1.4]$	$[-10.1, 1.3]$	$[-9.8, 1.2]$	$[-9.3, 1.1]$	$[-8.5, 0.9]$	$[-6.4, 0.7]$	$[-7.8, 0.9]$
x_5	$[-2.8, 2.8]$	$[-2.8, 2.8]$	$[-2.7, 2.7]$	$[-2.6, 2.6]$	$[-2.3, 2.3]$	$[-1.6, 1.6]$	$[-2.1, 2.1]$
x_6	$[-11.4, 3.4]$	$[-11.4, 3.4]$	$[-11.3, 3.3]$	$[-11.2, 0.5]$	$[-10.9, 2.9]$	$[-10.2, 2.2]$	$[-10.7, 2.7]$
x_7	$[-4.5, 0.7]$	$[-4.4, 0.6]$	$[-4.3, 0.6]$	$[-4.1, 0.5]$	$[-3.8, 0.5]$	$[-2.9, 0.3]$	$[-3.5, 0.4]$
x_8	$[-1.2, 1.2]$	$[-1.2, 1.2]$	$[-1.2, 1.2]$	$[-1.1, 1.1]$	$[-1.0, 1.0]$	$[-0.7, 0.7]$	$[-0.9, 0.9]$
x_9	$[-11.6, 3.6]$	$[-11.6, 3.6]$	$[-11.5, 3.5]$	$[-11.3, 3.3]$	$[-11.0, 3.0]$	$[-10.1, 2.1]$	$[-10.7, 2.7]$

Table 6.3.: Gap ranges of a H_∞-based controlled platoon of 3 vehicles obtained with support function based reachability analysis for $T = 30s$, $r = 0.01s$ and different values of γ.

Method	LMI	H_2	H_∞
e_1 in $[m]$	25.6	15.9	15.0
e_2 in $[m]$	25.4	15.4	16.5
e_3 in $[m]$	9.7	13.5	19.8

Table 6.4.: Safe minimum gaps for a platoon of 3 vehicles controlled with different control design methods.

for the H_∞ control design. Table 6.3 lists interval hulls of the reachable sets of a platoon of 3 vehicles obtained with different values of γ in case of nominal communication. For this analysis, we took the same conditions and options as by the H_2-based platoon. We note that $\gamma = 3$ leads to the shortest safe gaps, which correspond to the absolute values of the lower bounds of the intervals corresponding to the state variables x_1, x_4 and x_7. However, we notice thereby that the string stability is no more guaranteed.

A comparative overview of minimum safe gaps provided by different control design methods for a platoon of 3 vehicles is given in Table 6.4. It is particularly noteworthy that the simplest controller is also the best performing controller with regards to the control and safety requirements.

We applied the same procedure to decide on the controller for platoons of length 5, 10, 15 and 20. The choice of $Q = 1000.I$ has led to the best controller for the H_2-based design. For the H_∞-based controller, however, the optimal value of γ varies with the length of the platoon.

6.7. Managing Platoons at Intersections

In this section, we demonstrate how reachability can be applied to derive the time criteria for assuring safety with the management of a platoon at intersection as case study.

It is known that managing intersection is a particularly complicated task in traffic management. The task becomes even more complex when a platoon is approaching an intersection and the goal is to maintain the formation during the crossing process or in the case of a collision risk, to split itself into two sub-platoons.

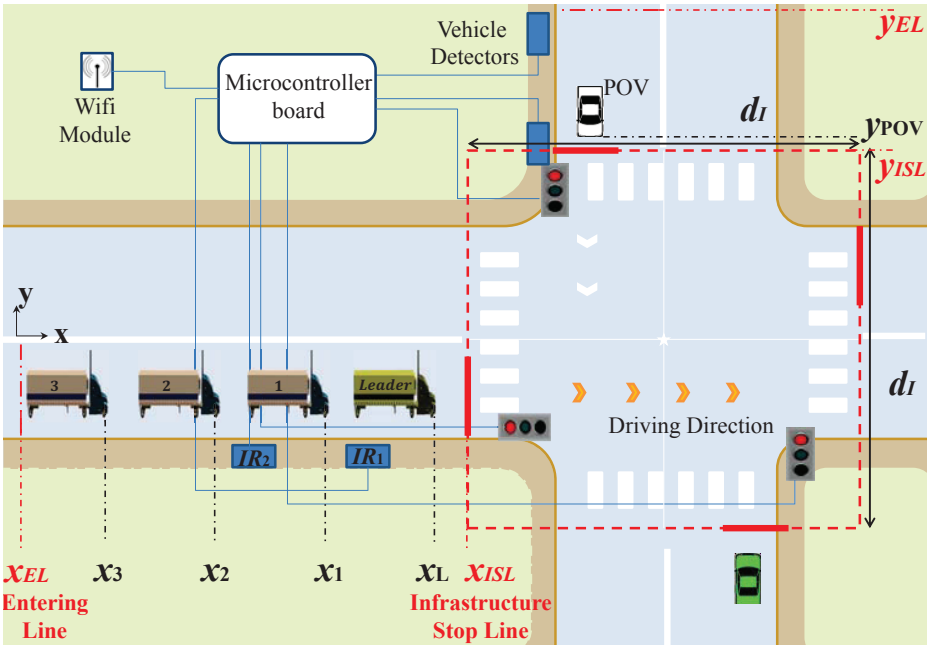


Figure 6.25.: Intersection infrastructure.

6.7.1. The Intersection Infrastructure

The intersection infrastructure is equipped with side sensors, traffic lights, a WiFi module and a microcontroller board. The road sensors in lines IR_1 and IR_2 of Figure 6.25 are two infrared distance sensors. They allow a systematically assessment of position and velocity information. This information is thereby automatically available for safe management and for monitoring of the crossing. The intersection is also equipped with scaled time/phase controllable traffic lights. In this work, we consider signalized as well as unsignalized intersections. The WiFi module assures the communication between the microcontroller unit on board of each truck and the intersection microcontroller board. In addition, an indoor positioning system consisting of two ultrasonic transmitters, a microcontroller and a radio sender module measures independently from the platoon infrastructure the position of each truck entering the intersection. A detailed hardware description of the intersection is available in [35].
Besides lines IR_1 and IR_2, following lines shown in Figure 6.25 are defined:

- The entering line EL with the coordinates x_{EL} and y_{EL} at which the vehicles receive the information that they are entering an intersection,
- The intersection stop line ISL with the coordinates x_{ISL} and y_{ISL} at which the vehicles have to wait for the permission to cross the intersection.

The width d_I of the intersection area is also assumed to be known. In this study, we furthermore assume the intersection to have the same width in both directions. The intersection area and therefore the unsafe region is consequently defined by

$$(x_{IL} < x < x_{IL} + d_I) \wedge (y_{IL} < y < y_{IL} + d_I). \tag{6.18}$$

6.7.2. The Intersection Model

The intersection shown in Figure 6.25 involves the platoon governed by the dynamics described with equation (6.12) and principal other vehicles (POV) on the other side of the intersection. A PI-controller is also introduced to track the reference speed v_{ref}. The positioning system and the on-board sensors collect for each vehicle $k \in \{i, POV\}$ inside the intersection the position x_k, the velocity v_k and the acceleration a_k which verify

$$v_k = \dot{x}_k \text{ and } a_k = \dot{v}_k. \tag{6.19}$$

As a consequence the dynamics of the platoon and the POV is once again described by

$$\dot{x} = Ax + Bu, \tag{6.20}$$

where $x = [x_L, v_L, \cdots, x_i, v_i, a_i, \cdots, x_{POV}, v_{POV}]^T$, u the input of the system, (A, B) are the system matrices deduced from the equation of the underlying PI-controller with the combination of equations (6.12) and (6.19).

6.7.3. Reachability using Zonotopes

The ISL in Figure 6.25 defines the critical or the unsafe area for the verification analysis. In addition, a risk of collision increases if the platoon P and vehicle POV enter simultaneously the intersection. For a platoon of length n_p, this last condition is formally expressed as follows

$$\forall i \in \{1, \ldots, n_p\} \ (y_{IL} < y_{POV} < y_{IL} + d_I) \wedge (x_{IL} < x_i < x_{IL} + d_I)\,. \tag{6.21}$$

However, the size of the vehicles is not considered in this formulation. For greater safety guarantee, a time condition is introduced. Under the following condition, where t^e and t^l are the predicted values of the entering and leaving time respectively,

$$\forall i \in \{1, \ldots, n_p\} \ \left(t_i^e < t_{POV}^e < t_i^l\right) \tag{6.22}$$

a collision may occur. Ensuring the contrary increases the intersection safety.
The simulation has shown that with an initial state vector x_0=[80,30,60,30,0,40,30,

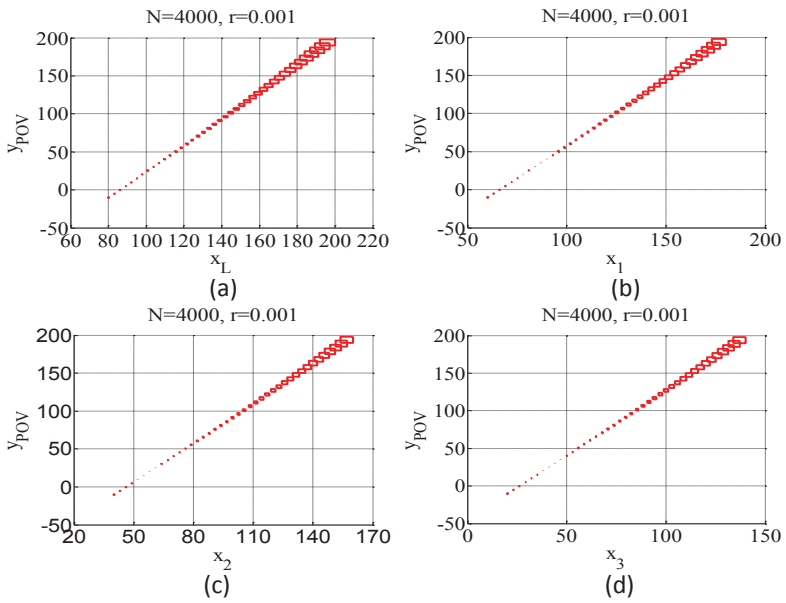


Figure 6.26.: Reachable sets: y_{POV} as a function of x_L, x_1, x_2 and x_3 for X_0=$[x_0, 0.01I_{13}]$ and T =4s.

0,20,30,0,-10,50] and for $x_{ISL} = 200m$, a collision between the POV and the leader of the platoon is possible after $t = 4s$. We use our zonotope implementation to retrieve this result. We initialized our verification first with a zonotope X_0=$[x_0, 0.01I_{13}]$ where I_{13} is the identity matrix in $\mathbb{R}^{13\times13}$. We are concerned with the evolution of the positions of different vehicles involved in the intersection and particularly in the position y_{POV} and the speed v_{POV} of the vehicle POV in relation to current positions of the platoon vehicles. For this reason, we plotted the projections of the reachable sets on the planes (x_L, y_{POV}) and (x_i, y_{POV}) as well as the projections on the planes (x_L, v_{POV}) and (x_i, v_{POV}) for $i \in \{1, 2, 3\}$ and that for a time horizon $T = 4s$. The results are illustrated respectively in Figure 6.26(a)-(d) and Figure 6.27(a)-(d).

Figure 6.26(a) shows that the leader of the platoon and the POV cannot reach the line $200m$ in the interval time $[0, 4]s$. This consequently guarantees a collision free intersection. Figure 6.26(c)-(d), however, depicts the distance each vehicle can reach behind the leader. The movement of the POV in correlation with the states of the platoon participants can also be tracked in this way. Using these results with the POV velocity information of Figure 6.27(a)-(d) help to safely manage the intersection. For example, an emergency stop message to the POV from the intersection management unit after $4s$ can ensure a safe intersection crossing of the whole platoon.

Otherwise the waiting time at the ISL can be set to allow vehicle POV to cross the intersection between the leaving time t_i^f of vehicle i and the entering time t_{i+1}^e

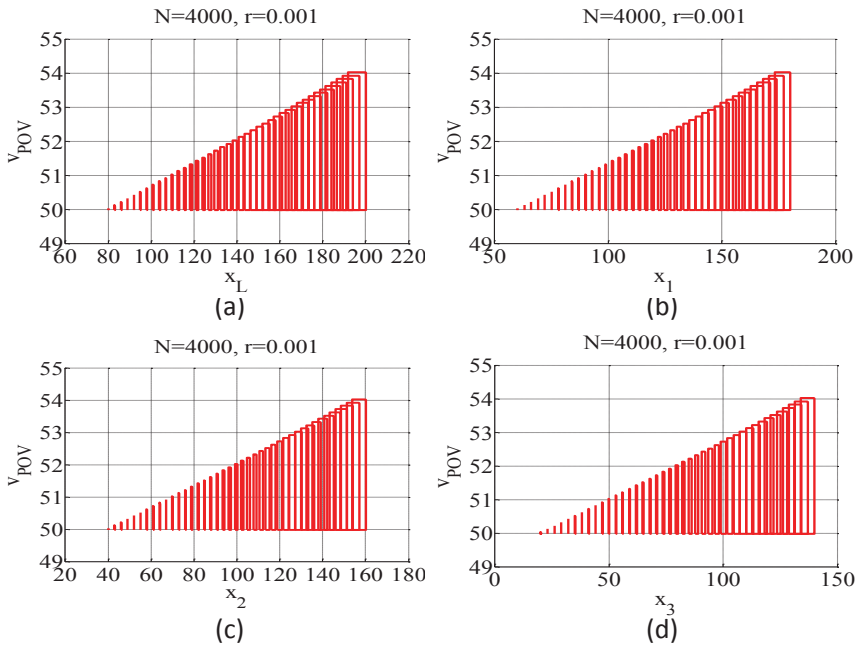


Figure 6.27.: Reachable sets: v_{POV} as a function of x_L, x_1, x_2 and x_3 for X_0=$[x_0, 0.01I_{13}]$ and T =4s.

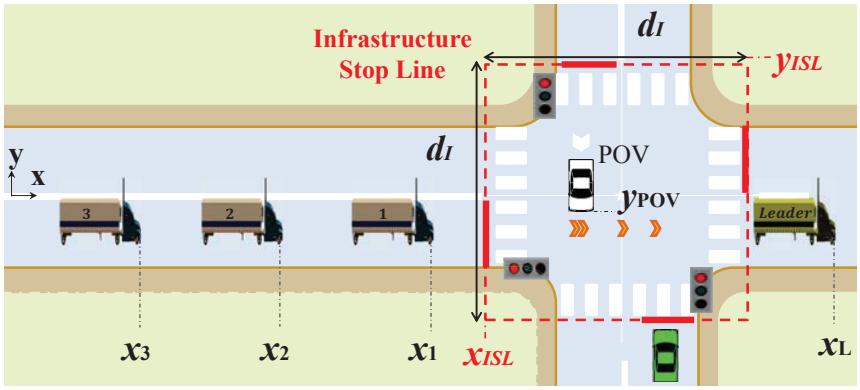


Figure 6.28.: A critical intersection scenario: split the platoon and allow in-between vehicle POV to cross.

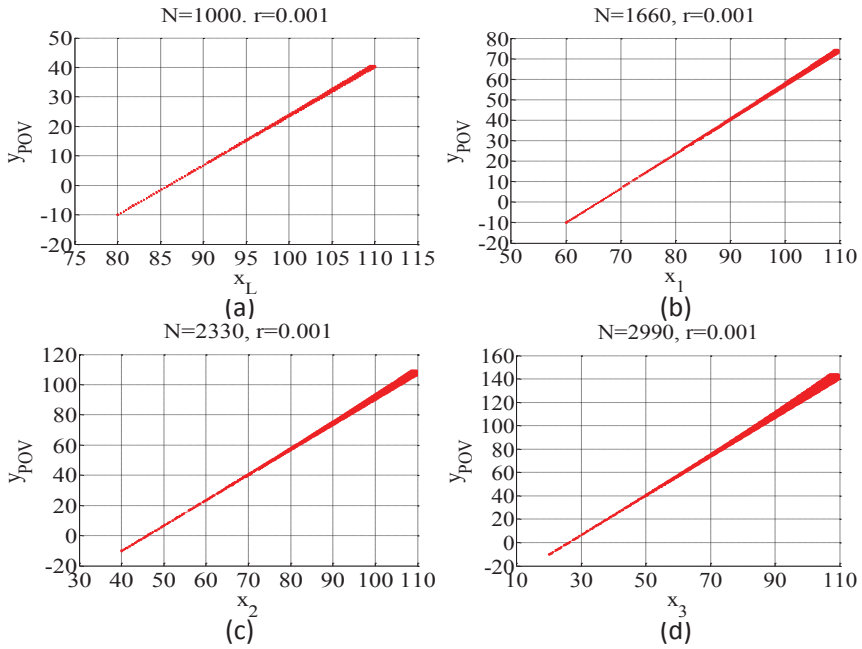


Figure 6.29.: Reachable sets to estimate entering times for different trucks at line x_{ISL} =200m.

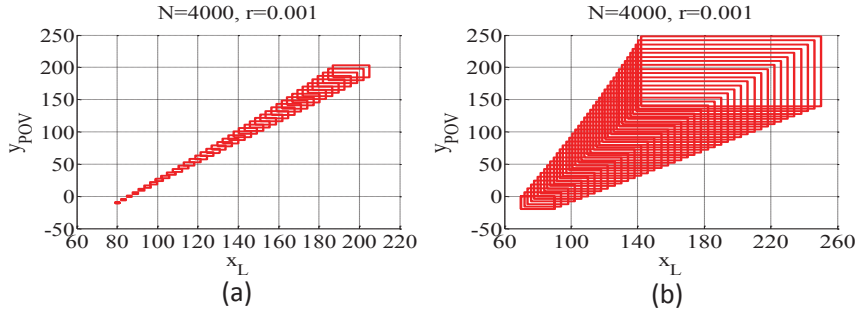


Figure 6.30.: Reachable sets for (a) X_0=$[x_0, I_{13}]$ and (b) X_0=$[x_0, 10I_{13}]$ for T =4s.

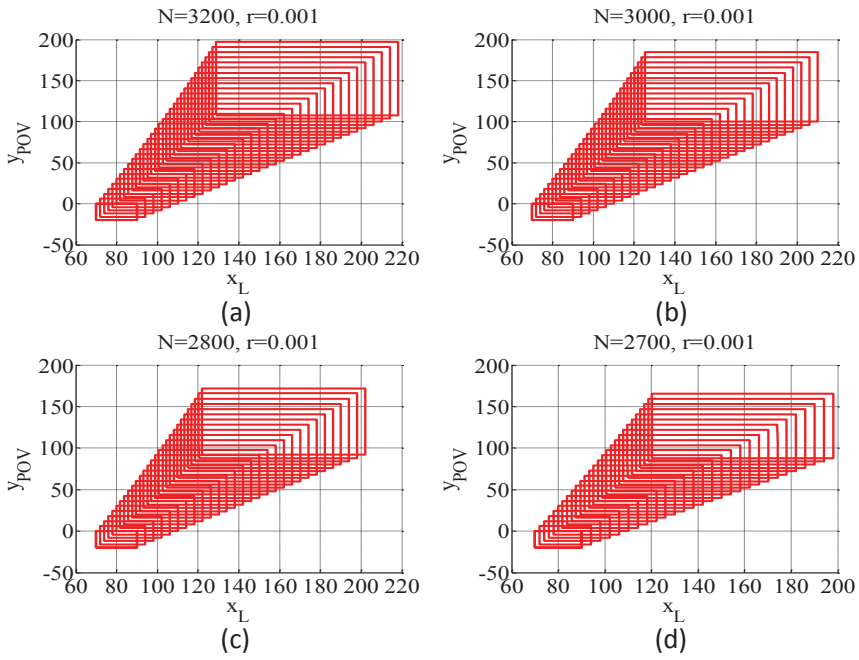


Figure 6.31.: Reachable sets with X_0=$[x_0, 10I_{13}]$ for different T: Entering time estimation for $x_{ISL} = 200m$.

of its follower $i + 1$. An increase of the speed v_{POV} may therefore be necessary to improve the safety aspect related with the critical platoon split decision. To put this idea into practice, the split scenario illustrated in Figure 6.28 is considered. A new ISL is thereby fixed at the distance $x_{ISL} = 110m$. We first examine the results of Figure 6.29(a)-(b). The reachable sets corresponding to the position of the POV as a function of the position of each truck within the platoon are plotted until the ISL was reached. The elapsed time, which is effectively the entering time t^e, is thereby assessed. Different entering times t^e_L=1s, t^e_1=1.66s, t^e_2=2.33s and t^e_3=2.99s can be recorded in this way. We note that the vehicles enter the intersection with a time gap of $0.66s$. The decision on allowing the POV to cross the intersection between two successive platoon vehicles depends on the unsafe condition (6.21), the intersection width d_I and the information on the speed of the POV acquired from Figure 6.27.

The same procedure is also applied to estimate different leaving times t^l. However, in this case, the computation of the reachable sets must be stopped once the line $x_{ISL} + d_I$ is reached. Taking for example x_{ISL}=110m and d_I=50m and based on results of Figure 6.26(c), $t^f_2 = 4s$ can be deduced. This means that truck 2 requires almost $t = 2,67s$ to cross the intersection. For the other vehicles, this crossing time is practically the same because the main platoon control objective is to maintain constant reference speed v_{ref}.

To take advantage of verification in comparison with simulation, we thereafter initialized the reachability using our zonotope implementation with the sets X_0=$[x_0, I_{13}]$ and X_0=$[x_0, 10I_{13}]$. The results are illustrated in Figure 6.30(a),(b). In both cases, we deduced that for $x_{ISL} = 200m$, a collision free intersection under $4s$ could no longer be guaranteed. The choice of the initial set X_0=$[x_0, 10I_{13}]$ allowed large uncertainties in state variables and is hence more relevant in practice. In fact, the above strategy can be adopted to estimate different t^e, t^l and recall all necessary information required for a safe management of the intersection. For example, results of Figure 6.31(a)-(b) lead to the conclusion that no collisions are possible for times under $t = 2.7s$.
In cases of large uncertainties, it is evident that making a decision concerning the safety of the intersection is quite difficult. However, since these decisions are based on reachability analysis, they will consequently guarantee for absolute safety at the intersection.

6.8. Conclusion

In this section, we addressed the problem of safety verification of interconnected platoon of vehicles. The vehicles communicate via a communication network subject to failures. A controller based on LMI control design is proposed to maintain safe and small spacing errors between the vehicles. We modeled the controlled platoon as a hybrid automaton switching spontaneously between different modes corresponding to the changing topology of the communication inside the platoon. We used our zonotope and our support function implementations to get an approximation of the reachable sets and compare their results. These assist in determining the shortest safe gaps between the vehicles inside the platoon which ensure a collision-free path even under a complete outage of communication. We were furthermore able to check for string stability and declining overshoots in the velocity. These requirements have been ignored during the control design with the aim to reduce complexity.

Beside the check for safety conditions, reachability can also help in control design of large scale systems. We suggested a practical approach for the control synthesis of a platoon of vehicles with scalable length. Easy scalable control synthesis approaches like H_2 and H_∞ control design were suggested to compute a feedback controller. Control requirements like string stability and confining overshoots in the velocity or in acceleration under specific thresholds were ignored during control design to simplify the computation. The choice between different computed controllers was made on the basis of the results of the reachability analysis of the closed-loop system. We were able to find controllers for platoons of length 3, 5, 10, 15, 20 with this approach for which control and safety requirements are guaranteed even under loss of communication.

We furthermore show, with the example of a platoon approaching an intersection, how reachability analysis can be useful for safely managing the intersection. It was

demonstrated that reachable sets can provide information required to estimate the collision time, the entering time and the leaving time for each vehicle involved at the intersection. It was also possible to track different state variables and in particular the position and the speed of the vehicle on the other side of the intersection. These results contribute to the basis for crucial decisions like permitting the platoon to pass the intersection, stopping it or splitting it if the collision risk is high.

We illustrated different practical applicability aspects of the reachability analysis with these examples particularly in the field of safety and control of large-scale connected systems.

The problem of conceiving a controller for networked systems is a challenging task because of the complex interaction of its different components with one another and also with the surrounding environment. The design process becomes more difficult if large-scaled systems are involved. We proposed reachability analysis of continuous systems to guarantee control requirements which might not be taken into account during the control design owing to the complexity of the problem. As an example, we suggested a large-scalable platoon of trucks. We investigated the main part of our work on devolving an algorithm for the computation of the reachable set for this kind of hybrid systems. We used our support function and zonotope implementations to check for safety, assess the performances of the obtained controlled platoon and decide on the best performing controller.

7. Conclusion

This thesis explores methods and approaches developed during the last decade for computing reachable sets of linear hybrid systems. After an assessment and a comparative evaluation of some available verification tools, a theoretical overview of the most significant methods was provided. It begun with the skeleton of the main algorithm which consists of a recursion derived from the linear differential equation describing the continuous dynamics of the hybrid system. It then covered different approximation techniques aimed at finding the tightest approximations for the initial set and the input contribution required to complete the construction of the above mentioned algorithm. In addition, we included a survey of different strategies for dealing with invariants. Furthermore, different techniques for handling the discrete dynamics of hybrid systems were include as part of this overview.

We opted for support functions and zonotopes for representing reachable set. We dedicated a large part of this work to a detailed theoretical description as well as practical guides for the implementation of the aforementioned techniques. We also focus on techniques specific to each geometric representation. We elaborated on diverse approaches for handling invariants, guard intersections and transitions which are specially conceived for support functions as well as for zonotopes

We developed two toolboxes integrating these diverse methods in a same framework. The first developed under MATLAB uses support functions. The second is a C++ tool based on zonotopes. Both tools allow for a flexible choice of parameters and combination of methods via an intuitive graphical user interface. Guards in form of hyperplanes, halfspaces and polyhedron are allowed, while invariant given as halfspaces or polyhedron can also be treated. Furthermore, different clustering methods for handling transitions are also available. We furthermore proposed the time-triggered and the novel fixpoint-triggered transition. Therefore, we allowed the user to choose the transition picking strategy for cases where from a source location more as one transition can be fired. The user can decide, during the analysis, which transition can be taken if the manual option is set. He can also choose options, like first/last detected or most/few intersection transition, with the latter referring to the number of reachable sets intersecting the guard.

We then used the tools to assess these methods and carried out a comparative performance analysis. We generally note that in many cases and particularly for complex systems, the choice of methods and their combination can have a crucial impact on the tightness of the approximation and on the efficiency of the computation.

The complexity of the hybrid system essentially depends on its state dimension, the differential equation describing the continuous dynamics, the number of

locations, the number of transitions and the complexity of the logic expressions describing the guard conditions. The presence of invariants, the possibility of Zeno phenomena and the possibility of multiple outgoing transitions can additionally affect this complexity. Our experiments have revealed that it is practically impossible to provide a general approaches which is efficient for all kind of linear hybrid systems. It is therefore impossible to predict in advance which methods are most appropriate to deal with a specific system. Thus, it is very helpful to have many methods inside the same framework at our disposal. This allows for testing with multiple methods and the choice of the best performing. An enormous modeling effort is thereby spared if our implementation is used in place of other different available verification tools.

We finally demonstrated potential applications of the reachability analysis using a networked platoon of vehicles as case study. We first carried out a reachability analysis to determine the shortest safe gaps between vehicles in a platoon controlled using linear matrix inequalities. We than showed how reachability can help by the choice of the best performing platoon controller. The controllers were computed using H_2 or H_∞ optimal control design. Afterwards, we considered a platoon approaching an intersection as application to demonstrate how time and state critical conditions can be determined using reachability analysis. These conditions are decisive for a safe and reliable management of the intersection.

This work covers only a small part of hybrid systems verification techniques, having only focused on reachability techniques for linear hybrid systems. However, our experience reveals that we are far away from proposing a general platform able to deal with different hybrid automaton definitions and models. Even for the class of linear hybrid systems, it is theoretically and practically difficult to find the most suitable techniques for a given system.

Within the HyPro Project [61], we are working on a C++ implementation of a library including geometric sets, like boxes, polytopes, zonotopes and support functions with their corresponding operations and if possible transformation between them. Taylor models are also part of this library. The main goal of this project is to use the library for implementing a reachability toolbox integrating all these geometric sets for the verification of linear as well as nonlinear hybrid systems. Our contribution to the project consisted of collecting and building a suite of benchmarks for testing tools [33], assessing some of them and finally on the C++ implementations of zonotopes and support functions.

Based on our experience with tools, we are working in the context of this ongoing project to overcome the inherent disadvantages of currently available methods where possible. It might therefore be conceivable to change the presentation set in specific computation steps in oder to improve the efficiency and tighten the approximation. However, we have to thereby take into account that transformations between sets may be also potential sources of inefficiency and approximation errors.

Our experience with reachability tools for nonlinear systems, done outside the context of this thesis, has shown these last to suffer from the same problems as tools dedicated for linear systems. However, apart from the problems arising in

handling transitions, which can be generally treated in the same way as by linear systems, computing a tight over-approximation of reachable sets for the nonlinear part depends crucially on the recursion adopted for the iterative computation and particularly on the bounding approach of the remainder term in Taylor series. Splitting is thereby often required to refine the approximation.

Furthermore, our survey of verification tools based on theorem proving or on model checking techniques, such as KeYmaera [88], iSAT [100] and dReach [27], has demonstrated these approaches to have difficulties in dealing with the continuous dynamics.

After these different surveys and the practical implementation experience, the question arises as to how far could set-theoretic and logic-theoretic techniques be combined into the goal of improving the performance of verification tools. Seeking the answer for this question and solutions for the aforementioned open problems calls for an immense experimentation and research effort.

A. Appendix A

This appendix includes a detailed description of the different benchmarks used to assess the performances of the tools.
We use the following notations for the description of the hybrid automata: x_dot corresponds to $\dot{x}$, $x == y$ to $x = y$, & to the logic AND and | to the logic OR .

A.1. Infinity test

This example is a simple two dimensional system consisting of a one location with a flow given by the following system of differential equations

$$\begin{cases} \dot{x} &= x + u \\ \dot{y} &= 2y + u, \end{cases} \tag{A.1}$$

where $u \in [0; 1]$. The initial values for x and y is the origin and the unsafe states are $x < 0$ or $y < 0$.

A.2. The Bouncing Ball Example

The bouncing ball example is commonly the first example used to test tools or demonstrate the validity of new methods. It consists of a simple ball released from

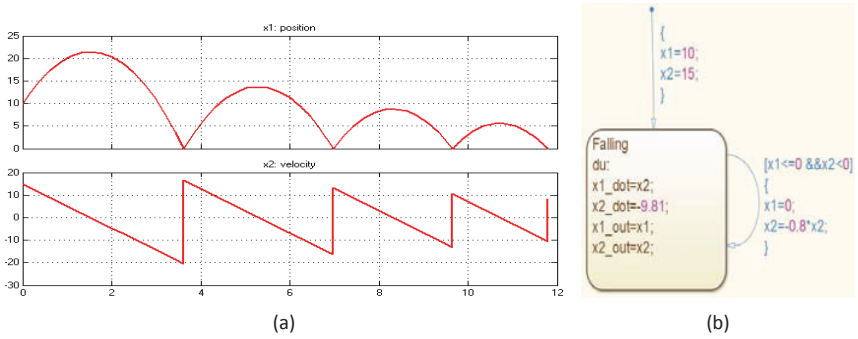


Figure A.1.: Hybrid automaton of the bouncing ball benchmark.

a predefined height and left to bounce on the ground. It is modeled by the unique mode hybrid automaton of Figure A.1(b) with a loop transition. The dynamics of the ball is illustrated in Figure A.1(a). The initial set is $x_1 = 2 \& x_2 = 0$ and the forbidden set $x_1 < 0$.

A.3. The Colliding Masses Benchmark

This example proposed in [50] describes two particles m_1 and m_2 moving in a straight homogeneous motion which then collide and move apart in the opposite direction. This example can be modeled as a hybrid system. The continuous

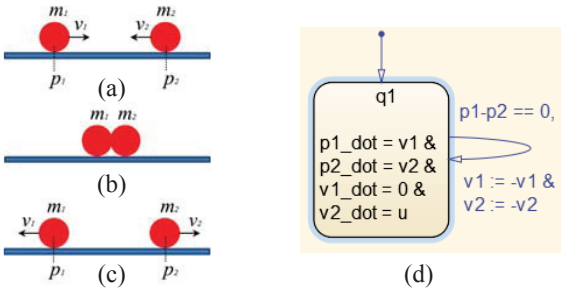


Figure A.2.: Colliding masses example. (a) Before the collision. (b) Collision. (c) After collision (d) The corresponding hybrid automaton.

behavior is thereby governed by the dynamics of the masses before and after the collision (Figure A.2(a), (c)) given by the differential equation describing the flow of the unique location of the corresponding hybrid model of Figure A.2(d). The impact between both particles m_1 and m_2 (Figure A.2(b)), however, determines the discrete event. For the case $m_1 = m_2$, the guard condition $p_1 = p_2$ and the reset equation $v_1 = -v_1 \wedge v_2 = -v_2$ of the unique transition of the hybrid automaton can be thereby deduced (see Figure A.2(d)). The initial set is chosen equal to $p_1 = 0 \& p_2 = 3 \& v_1 = 2 \& v_2 = -1$ whereas $u = 0$ and the forbidden set may be $p_1 > 2 | p_2 = 2$.

A.4. The Transient in Flower Benchmark

We consider the piecewise linear system proposed in [36] consisting of controlled switches between four cells in a 2D-partition. This example is characterized by dynamics that follows a flower-like trajectory. It has been used in [36] as testing example for the ReachLab platform. The continuous dynamics of the four cells is given by the following matrices:

$$A_1 = A_3 = \begin{pmatrix} -0.1 & 5 \\ -1 & -0.1 \end{pmatrix}, \; A_2 = A_4 = \begin{pmatrix} -0.1 & 1 \\ -5 & -0.1 \end{pmatrix} \tag{A.2}$$

The corresponding hybrid automaton is given in Figure A.3. As initial values -1 for x_1 and 0 for x_2 are used. For the unsafe set, we take all states verifying $x_2 > 1$ or $x_2 < -1$.

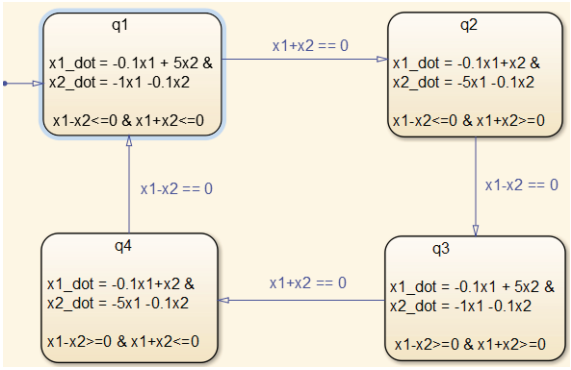


Figure A.3.: The hybrid automaton of the transient in flower example.

A.5. The Two-Tank Benchmark

Different types of two-tank benchmarks have been already proposed as examples of hybrid systems. The most classical ones of them are commonly used in lectures. such as [74], to introduce hybrid automata. We propose the two-tank system, suggested in [59] to tackle the stability problem of limit cycles in hybrid systems, as a benchmark for testing verification tools. This has been also previously used in [47] to test reachability techniques based on zonotopes. The corresponding hybrid

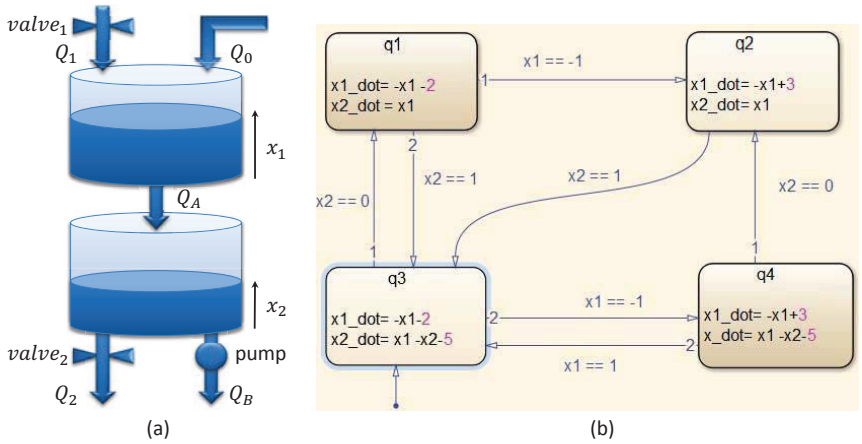


Figure A.4.: The two-tank system: (a) Schematic diagram of the considered two-tank system. (b) The corresponding hybrid automaton.

model has four locations with many possible transitions and nondeterministic jumps. This benchmark has a periodic dynamical behavior which converges to a stable limit cycle depending on the chosen initial set. The benchmark illustrated in Figure A.4(a) consists of two tanks. The liquid in the first tank has two external sources – a constant inflow source with the flow Q_0 and a second source with the flow Q_1

equipped with a controlled valve $valve1$. A drain placed at the button of tank 1 allows the liquid to pass into tank 2 with the flow Q_A. Tank 2 is itself equipped with two drains. In the first drain, a pump is placed to assure a constant liquid outflow Q_B whereas the flow in the second one Q_2 is controlled by an electro-valve $valve2$. Both valves $valve1$ and $valve2$ can take the states On or Off. This results consequently in four possible discrete modes for the hybrid automaton. The liquid levels x_i, $i \in \{1, 2\}$ in both tanks obey to the following differential equations

$$\begin{aligned} \dot{x}_1 &= \begin{cases} -x_1 - 2 & \text{if } valve_1 \text{ is Off} \\ -x_1 + 3 & \text{if } valve_1 \text{ is On} \end{cases} \\ \dot{x}_2 &= \begin{cases} x_1 & \text{if } valve_2 \text{ is Off} \\ x_1 - x_2 - 5 & \text{if } valve_2 \text{ is On} \end{cases} \end{aligned} \tag{A.3}$$

The transitions are governed by the switching strategy of both valves. This latter aims at maintaining the liquid levels in both tanks within predefined safe levels. In this case study, $valve_1$ should be immediately opened once x_1 falls to -1 and closed as soon as x_2 reaches level 1. On the other hand, $valve_2$ is closed if x_2 reaches the level 0. Otherwise, as soon as x_2 goes above 1, $valve_2$ should be promptly opened. The resulting hybrid automaton is given in Figure A.4(b). We

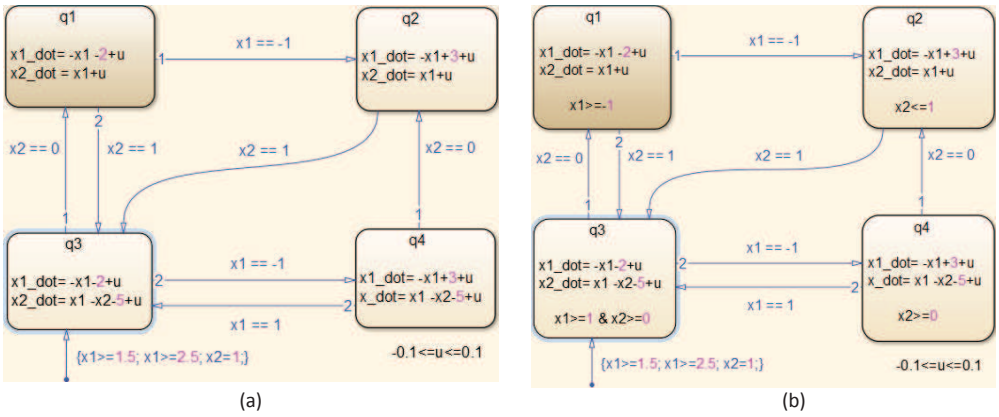


Figure A.5.: Hybrid automata for the two-tank benchmark without invariants (a) and with invariants (b).

test using reachability techniques whether these safety criteria are satisfied or not. We carry out a verification analysis under disturbance by adding an input u to the system as shown in Figure A.5(a). Afterwards, we introduce invariants in the discrete locations like in Figure A.5(b). We furthermore allow nondeterminism and vary the range of the initial set.
Possible initial set may be $1.5 \leqslant x_1 \leqslant 2.5 \& x_2 = 1$. The input can be bounded in $[-0.1, 0.1]$. The unsafe set is described by $-1 > x_1 | x_2 \leqslant -1$

A.6. The Navigation Benchmark

The navigation benchmark was first proposed in [37] as a benchmark for evaluating verification tools for hybrid systems. It describes an object moving in a two dimensional $n \times n$ grid. The grid is divided into a scalable number of cells. To each grid a matrix *map* is associated, the element of which $map(i, j)$ defines the dynamics of the object inside each cell $c(i, j)$, $i, j \in \{1, \ldots, n\}$. Two particular regions inside the grid are preliminary defined: a bad region **B** and a good one **A**. The object has to reach the good region while avoiding the bad one. Figure A.6(a) shows an instance

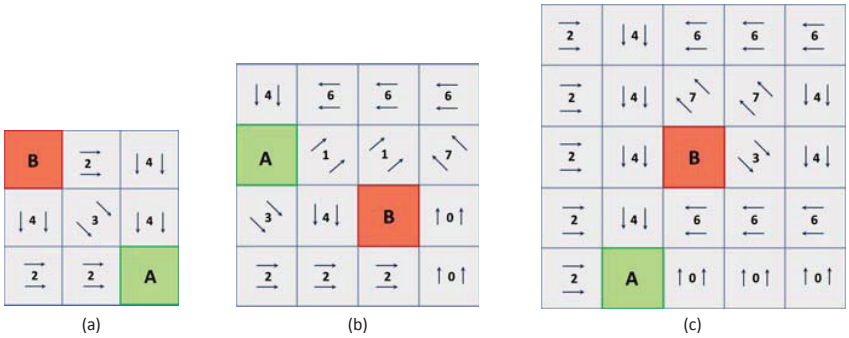


Figure A.6.: Instances and maps of the different navigation benchmark versions.

of the navigation benchmark for a 3×3 grid with the corresponding map matrix and the directions of the desired velocity in each cell. This last has been already used in [17] to test tools like PHAVer, HSolver and KeYmaera. The chosen map of the 4×4-navigation benchmark as well as this of the 5×5 variant are shown respectively in Figure A.6(b) and Figure A.6(c).
The desired velocity v_d of the object in each cell on the grid is given by

$$v_d = (sin(map(i, j) * \frac{\pi}{4}), cos(map(i, j) * \frac{\pi}{4}))^T. \tag{A.4}$$

The actual object velocity v is therefore calculated using the following differential equation

$$\dot{v} = A(v - v_d). \tag{A.5}$$

The matrix A for the 3×3 version is given by $A_{3\times3} = \begin{pmatrix} -1.2 & 0.1 \\ 0.1 & -1.2 \end{pmatrix}$, whereas the matrix for the 5×5 instance is chosen to be equal to $A_{5\times5} = \begin{pmatrix} -0.8 & -0.2 \\ -0.1 & -0.8 \end{pmatrix}$. For further tests, we constructed a 4×4 version with the map $A_{4\times4} = \begin{pmatrix} -0.9 & 0.2 \\ 0.3 & -0.9 \end{pmatrix}$.
The navigation benchmark can be modeled as a hybrid automaton with in general $n \times n - 2$ locations. The flow is given by the following system of differential equations

$$\begin{cases} \dot{x} &= v + u \\ \dot{v} &= A(v - v_d), \end{cases} \tag{A.6}$$

where u is an uncertain input, $x = [x_x, x_y]^T$ and $v = [v_x, v_y]^T$ constitute the state vector. The invariant of each location can eventually be defined by the position constraints. The corresponding transition guards, however, are the constraints describing common boundaries with neighboring cells. As examples, the hybrid automata of the 3×3-navigation benchmark and the 4×4-navigation benchmark are shown in Figure A.7 and Figure A.8. The initial states for our navigation tests

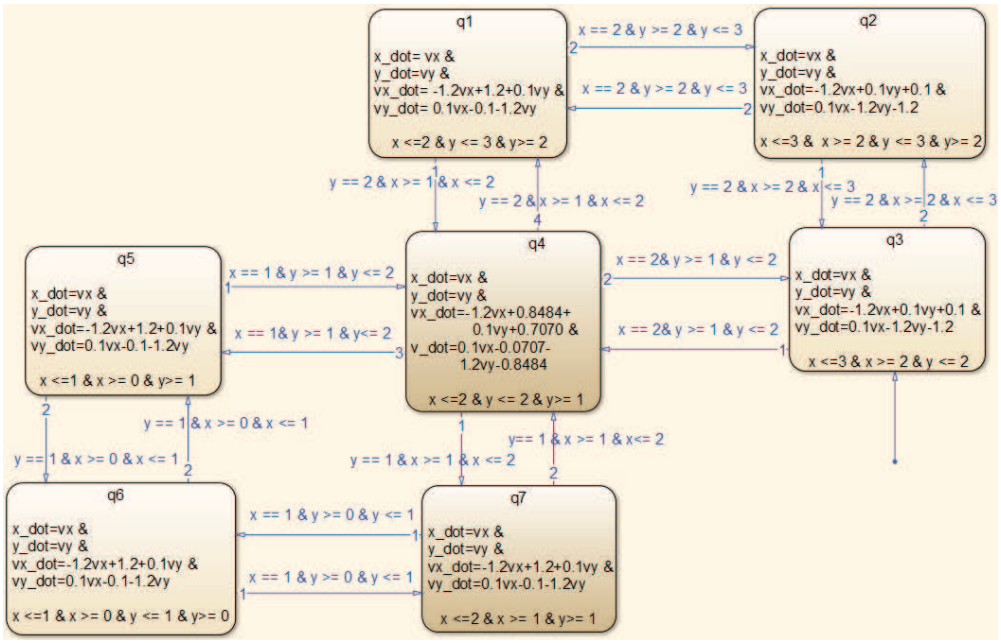


Figure A.7.: Hybrid automata of the 3×3 navigation benchmarks..

with the different mentioned maps and matrices A are given in Table A.1 for initial points and Table A.2 for initial sets. The set of forbidden states for the reachability analysis is the set of states inside and along the borders of the region B in each considered grid.

	3x3-map	4x4-map	5x5-map
xx	2	0.5	3.5
yy	1	1.5	3.5
vx, vy ∈	[-0.3;0]	[-0.3;0.3]	[-0.3;0.3]

Table A.1.: Initial states (points) for the navigation benchmarks

For the assessment of tools in Chapter 2 the input is taken $u == 0$.

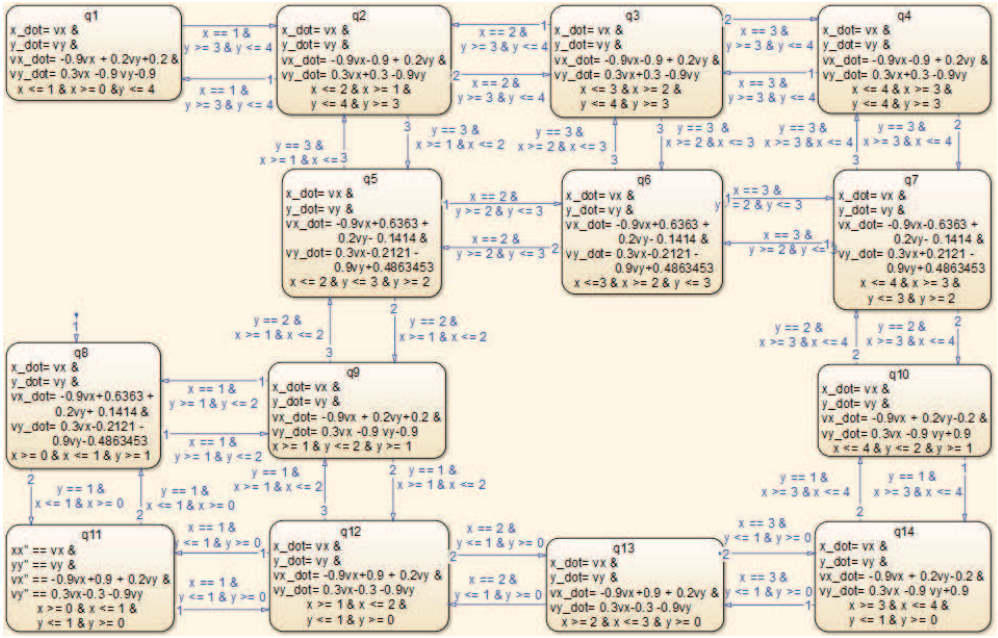


Figure A.8.: Hybrid automata of the 4×4 navigation benchmarks.

	3x3-map	4x4-map	5x5-map
xx	[2;3]	[0;1]	[3;4]
yy	[1;2]	[1;2]	[3;4]
vx, vy $\in$	[-0.3;0.3]	[-0.3;0.3]	[-0.3;0.3]

Table A.2.: Initial states (sets) for the navigation benchmarks

A.7. The Heating Benchmark

The heating benchmark consists of n rooms with temperatures x_i for $i = 1 \dots n$ and m heaters. The temperature changes according to the following differential equation

$$\dot{x_i} = c_i h_i + b_i(u - x_i) + \sum_{i \neq j} a_{i,j}(x_j - x_i), \tag{A.7}$$

where $c_i = 15$ is the heating power, h_i is a boolean variable with value 1 if room i has a heater and 0 else, $a_{i,j} = 1$ for adjacent rooms, u is the outside temperature and b_i is a value describing the intensity of the heat exchange with the outside supposed to be equal to

1. 0.08 for rooms with one common wall with the neighbor rooms,
2. 0.16 for rooms with two common walls with the neighbor rooms and
3. 0.24 for rooms with three common walls with the neighbor rooms.

Two types of heaters could be considered: movable and anchored heaters. The switching conditions will, hence, depend on the type of the heater. For anchored

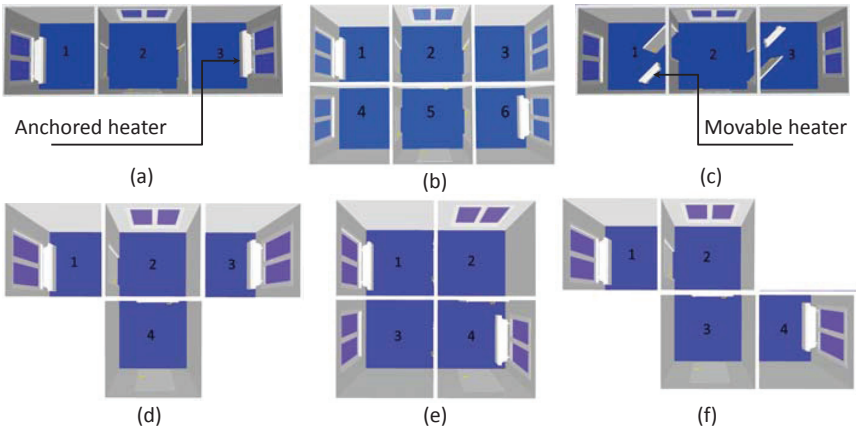


Figure A.9.: Layouts of the different heating benchmark variants.

heaters, we opted for the switching criteria adopted in [5]. An upper and a lower temperature thresholds $T^{high} = 24$ and $T^{low} = 22$ are chosen with a range $\delta T \in [0; 0.05]$ so that

(a The heater in room i is switched on if $x_i < T^{low} + \delta T$ and

(b switched off if $x_i > T^{high} - \delta T$, resulting consequently in a change of the temperature in different rooms.

The external temperature verifies $u \in [0; 0.1]$. As examples for anchored heaters, we consider for our study, the 6 room heating benchmark of [5] shown in Figure A.9(b), the 3 room example of Figure A.9(a) and the three variants of the 4 room heating benchmark of Figure A.9(e), (f), (g).
The initial values of the room temperatures for these benchmarks are listed in Table A.3 and the initial location is chosen to be the location where both heaters are off.

Room	x_1	x_2	x_3	x_4	x_5	x_6
Temperature	23.5	23.5	23	22.5	23	22.5

Table A.3.: Initial states for the proposed heating benchmarks with fixed heaters.

For movable heaters, we adopt the conditions of [37]. If a room i has no heater, it might receive one from another room j if the following conditions are met

1. room i has no heater,

2. room j has a heater,

3. $x_i \leqslant get_i$ and

4. $x_j - x_i \geqslant dif_i$, where get_i and dif_i are constants which may be different for each room.

We choose, like in [37], the setting $u = 4$, $off = (21, 21, 21)^T$, $on = (20, 20, 20)^T$, $get = (18, 18, 18)^T$ and $dif = (1, 1, 1)^T$ for these parameters.
As example for movable heaters, we consider the benchmark with 3 rooms and 2 portable heaters of [17, 95] shown in Figure A.9(c). For the initial state both heaters must be placed in room 1 and room 2 and $x_1 = x_2 = x_3 = 20$. The set of forbidden states is the set of states where the temperature of at least one room is below or equal 14. All considered benchmarks are conceived to be safe under the above considered conditions.

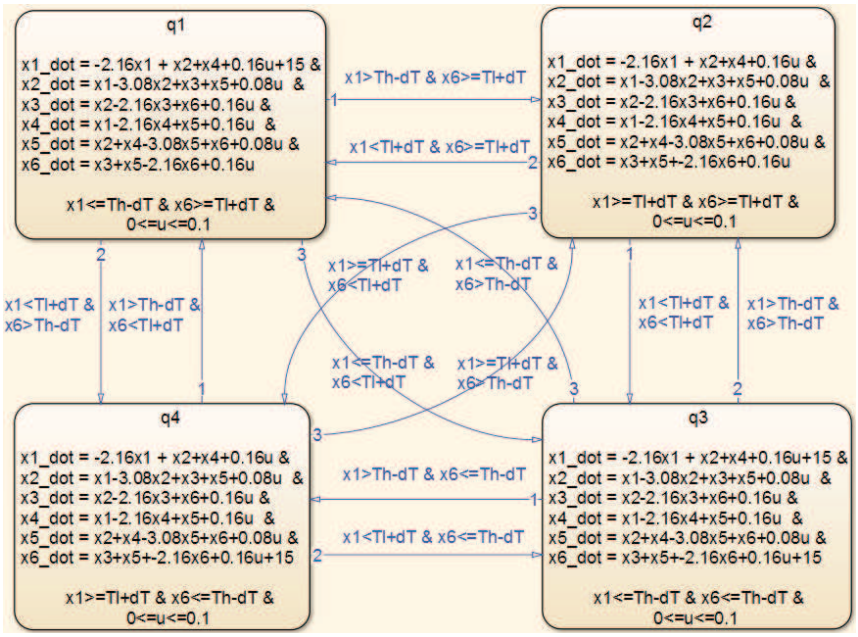


Figure A.10.: The graphical SpaceEx input format for the hybrid automaton of the heating benchmark variant with six rooms and two fixed heaters with the layout of Figure A.9(b) ($Tl = T^{low}$, $Th = T^{high}$ and $dT = \delta T$).

A.8. Cooperative Platoon of Trucks

We consider the cooperative platoon of three autonomously driven trucks with a leader at the start of the convey. This benchmark has been already proposed in [18, 21]. Each truck in the platoon senses its velocity, its acceleration or its actual gap to the vehicle ahead and then sends these values via a wireless network

to the other participants. Concurrently, it collects via wireless the same information about the other vehicles with on-board electronics. Figure 6.1 illustrates the infrastructure of the platoon. The distance between two consecutive trucks i and $i+1$ is given by $e_i = d_i(t) - d_{ref,i}$ where $d_{ref,i}$ is the safe gap. The minimum gaps guaranteeing collision free platooning are evaluated using verification techniques. A centralized controller based on LMI control design ensure besides the control requirements such as stability, short and meanwhile safe gaps between the trucks with a guarantee of string stability. The controlled platoon is a nine dimensional linear system described by the following differential equation $\dot{x} = Ax + Ba_L$, where $x = [e_1, \dot{e_1}, a_1, e_2, \dot{e_2}, a_2, e_3, \dot{e_3}, a_3]$ is the state vector, A and B are respectively the state and the input matrices.

A.8.1. Platoon under full communication (single mode)

We begin first with computing the minimum safe gap $d_{ref,i}$ for $i = 1, 2, 3$ for the platoon under full communication between the trucks. The corresponding hybrid automaton consists in this case in only one mode characterized by the matrices

$$A_c = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1.6050 & 4.8680 & -3.5754 & -0.8198 & 0.4270 & -0.0450 & -0.1942 & 0.3626 & -0.0946 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0.8718 & 3.8140 & -0.0754 & 1.1936 & 3.6258 & -3.2396 & -0.5950 & 0.1294 & -0.0796 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 \\ 0.7132 & 3.5730 & -0.0964 & 0.8472 & 3.2568 & -0.0876 & 1.2726 & 3.0720 & -3.1356 \end{pmatrix}, \tag{A.8}$$

$$B_c = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}^T.$$

The system starts with the origin as initial value or a neighborhood as an initial set. For some verification tools, we impose an unsafe state described by the constraint $e_1 \leqslant -30$.

A.8.2. Platoon under a dropout of communication (2 modes)

This example basically extends the previous one to a system with a second location representing a second operation mode where the communication between all trucks abruptly drops out. [18, 21]. This new mode is described by the matrices $B_n = B_c$ and

$$A_n = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1.6050 & 4.8680 & -3.5754 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.1936 & 3.6258 & -3.2396 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 \\ 0.7132 & 3.5730 & -0.0964 & 0.8472 & 3.2568 & -0.0876 & 1.2726 & 3.0720 & -3.1356 \end{pmatrix}. \tag{A.9}$$

The transitions between both modes is practically a *true* transition, which models simply a spontaneous triggered transition (see Figure A.11(a)). Since only a few tools allow this kind of transition, we specially constructed a hybrid model for them,

in which the transitions are time triggered. The corresponding guard transition then takes the form $t = T$ or $t >= T$, where T is a time horizon after which the closed loop system reaches a stable state. We have to include a clock t to the differential equations of both modes and to add the reset condition $t := 0$ to the transitions (see Figure A.11(b)). To guarantee the enclosure of all reached states from a chosen initial set, we impose T as time horizon for the computation inside both modes. Based on simulation results obtained from the control design, we decide for a time horizon $T = 20s$. The initial values and unsafe states are the

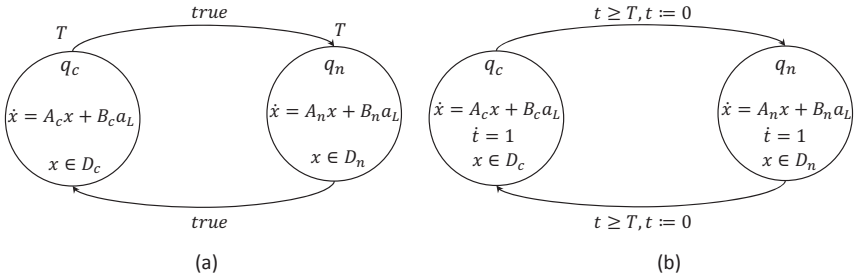


Figure A.11.: Possible hybrid models of the platoon. (a) The transitions are spontaneous. (b) The transitions are time triggered.

same as for the platoon example with one location.

A.9. A Five Dimensional Linear Switching System (5D LSS)

This consists of a hybrid system, for which the linear dynamics in each mode is randomly generated using the *rss* MATLAB function. To guarantee for stabilization and the convergence of the system to a steady state in each mode, we used LQR control design. The transitions between different modes have been determined heuristically by simulations. This results in the hybrid automaton of Figure A.12. For verification purposes, we initialize the hybrid automaton with the state vector $(3, 4, 0, 0, 0)^T$. We choose an input $u \in [-1; 1]$ and an unsafe set defined by $x_1 \geqslant 5$.

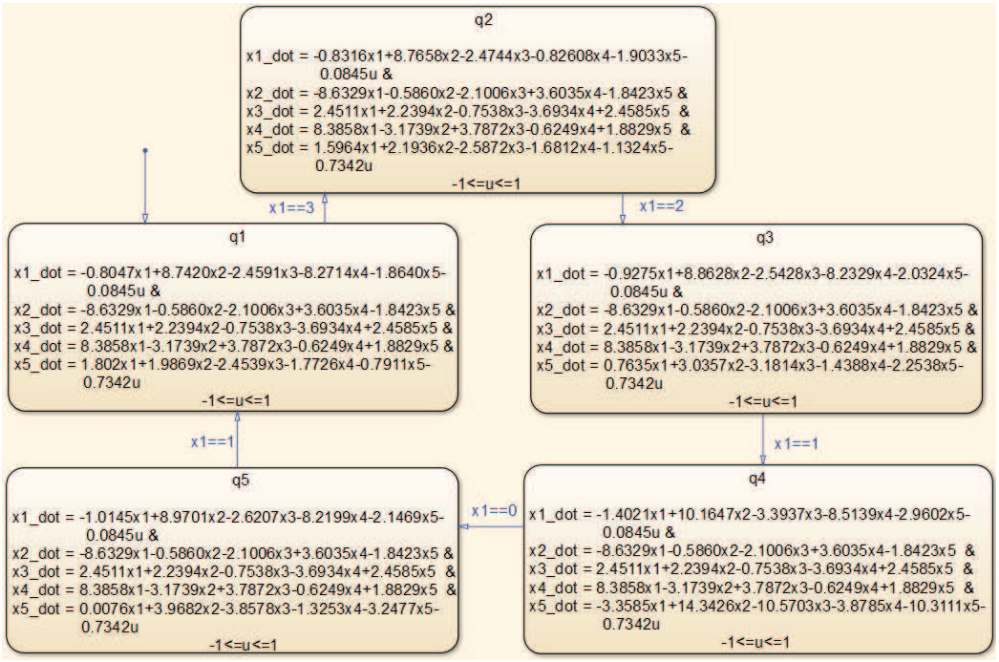


Figure A.12.: The SpaceEx graphical input file for the five dimensional linear switching system.

B. Appendix B

This appendix is reserved to proofs of Chapter 3. Some proofs extend already existing proofs for systems in the form $\dot{x} = Ax + u$ to systems taking the form $\dot{x} = Ax + Bu$ and are derived based on support function properties.

B.1. Proof of Lemma 1: Norm-bounded uncertain input

The input contribution is over-approximated with the following set $\mathcal{V}_r$ if $\mu = sup_{u \in U} \|Bu\|$

$$\mathcal{V}_r = \mathcal{B}(\beta_r). \tag{B.1}$$

where $\beta_r = \mu \frac{e^{r\|A\|}-1}{\|A\|}$.

Proof. (adapted from [47])
Let $x \in X$ and $y \in \mathcal{R}_r(X)$, then $\exists u(.) : \mathbb{R}^+ \to \mathbb{R}^n$ such that

$$y = e^{Ar}x + \int_0^r e^{(r-s)A} Bu(s) ds. \tag{B.2}$$

It follows, accordingly (Banach space, Lebesgue integrable)

$$\begin{array}{ll} \left\| y - e^{Ar}x \right\| & = \left\| \int_0^r e^{(r-s)A} Bu(s) ds \right\| \\ & \leqslant \int_0^r \left\| e^{(r-s)A} Bu(s) \right\| ds \\ & \leqslant \int_0^r e^{(r-s)\|A\|} \left\| Bu(s) \right\| ds \\ & \leqslant \mu \int_0^r e^{(r-s)\|A\|} ds \\ & = \mu \frac{e^{r\|A\|}-1}{\|A\|} \\ & = \beta_r. \end{array} \tag{B.3}$$

□

B.2. Proof of Lemma 3: Toward a tighter approximation of the input contribution

The input contribution is over-approximated with the following set $\mathcal{V}_r$ i

$$\mathcal{V}_r = rBU \oplus \mathcal{E}_U \tag{B.4}$$

with

$$\mathcal{E}_U = \square \left(|A|^{-2} \left(e^{r|A|} - I_n - r\,|A| \right) \right) \boxdot (ABU)\,. \tag{B.5}$$

Proof. (Adapted from [71]) We draw heavily on the demonstration proposed in [71] to derive this proof. We know that each solution $x(t)$ of the differential equation (3.9) can be written in the form given by (3.11). Particularly for $t = r$, we have

$$x(r) = e^{Ar}x_0 + \int_0^r e^{(r-s)A}Bu(s)ds. \tag{B.6}$$

We consider now the set $V_r = \mathcal{R}_r\left(\{0_{\mathbb{R}^n}\}\right)$ which is defined as follows

$$V_r = \left\{ \int_0^r e^{(r-s)A}Bu(s)ds : \forall s \in [0,r], u(s) \in U \right\}. \tag{B.7}$$

The goal is to find an over-approximation $\mathcal{V}_r$ for V_r.
For this purpose the set V_r is characterized with its support functions. Let v be an element of V_r then it exist a function $u(.) : \mathbb{R}^+ \to U$ such that $v = \int_0^r e^{(r-s)A}Bu(s)ds$. Under consideration of Taylor series expansion of the exponential function $e^{(r-s)A} = \sum_{i=0}^{\infty} \frac{(r-s)^i}{i!}A^i$, the vector v can be rewritten as follows

$$\begin{array}{rl} v & = \int_0^r \sum_{i=0}^{\infty} \frac{(r-s)^i}{i!}A^iBu(s)ds \\ & = \sum_{i=0}^{\infty} \int_0^r \frac{(r-s)^i}{i!}A^iBu(s)ds \\ & = \int_0^r Bu(s)ds + \sum_{i=1}^{\infty} \int_0^r \frac{(r-s)^i}{i!}A^{i-1}ABu(s)ds. \end{array} \tag{B.8}$$

Taking the maximum of the inner product of the vector v with a given direction $l \in \mathbb{R}^n$, we obtain the value of the support function $\rho_{V_r}(l) = sup_{v \in V_r}$ of the set V_r in this direction. Consequently, we have

$$\begin{array}{rl} \rho_{V_r}(l) & \leqslant sup_{v \in V_r}\left(\left(\int_0^r Bu(s)ds + \sum_{i=1}^{\infty}\int_0^r \frac{(r-s)^i}{i!}A^{i-1}ABu(s)ds\right).l\right) \\ & \leqslant \int_0^r sup_{u(s)\in U}\left(Bu(s).l\right)ds + \\ & \qquad \sum_{i=1}^{\infty}\left(\int_0^r \frac{(r-s)^i}{i!}\left|A^{i-1}\right| sup_{u(s)\in U}\left|ABu(s)\right|.\left|l(s)\right|ds\right) \\ & \leqslant r\rho_{BU}(l) + \sum_{i=1}^{\infty}\left(\int_0^r \frac{(r-s)^i}{i!}\left|A\right|^{i-1}ds\right) sup_{u(s)\in U}\left|ABu(s)\right|.\left|l\right| \end{array} \tag{B.9}$$

Furthermore, we know that the support function of a symmetric box $\Omega = [-\omega_1, \omega_1] \times \ldots \times [-\omega_n, \omega_n]$ for an $w = (w_1, \ldots, w_n)^T \in \mathbb{R}^n$ in a direction $l = (l_1, \ldots, l_n)^T \in \mathbb{R}^n$ is defined by

$$\rho_{\Omega}(l) = sup_{x\in\Omega}\, x.l = \sum_{i=1}^{n} \left|\omega_i l_i\right|. \tag{B.10}$$

Combining this definition with definition (6) of the symmetric interval hull of sets, we can deduce that the term $sup_{u(s)\in U}\left|ABu(s)\right|.\left|l\right|$ in equation (B.9) represents the support function $\rho_{\boxdot(ABU)}(l)$ of the symmetric interval hull of the set ABU in direction l.
On the other hand, we have

$$\begin{array}{rl} \sum_{i=1}^{\infty}\left(\int_0^r \frac{(r-s)^i}{i!}\left|A\right|^{i-1}ds\right) & = \sum_{i=1}^{\infty}\frac{(r)^{i+1}}{(i+1)!}\left|A\right|^{i-1} \\ & = \left|A\right|^{-2}\left(e^{r|A|} - I_n - r\left|A\right|\right). \end{array} \tag{B.11}$$

Taking the same notation $\mathcal{E}_U$ as in [71] for the set $\mathcal{E}_U = \square\left(\sum_{i=1}^{\infty} \frac{(r)^{i+1}}{(i+1)!} |A|^{i-1} \boxdot (ABU)\right)$, we derive for all chosen directions l the following inequality

$$\rho_{V_r}(l) \leqslant r\rho_{BU}(l) + \rho_{\mathcal{E}_U}(l). \tag{B.12}$$

This in turn leads to the following over-approximation of the input contribution

$$V_r \subset \mathcal{V}_r = r\, BU \oplus \mathcal{E}_U. \tag{B.13}$$

□

B.3. Proof of Lemma 4

The initial set is over-approximated by

$$\Omega_0 = CH(X_0 \cup e^{rA}X_0) \oplus \mathcal{B}(\alpha_r) \oplus \mathcal{V} \tag{B.14}$$

with

$$\alpha_r = \left(e^{r\|A\|} - 1 - r\,\|A\|\right) sup_{x\in X_0} \|x\|\,. \tag{B.15}$$

and $\mathcal{B}(\alpha_r)$ the ball of radius α_r.

Proof. Let $z \in \mathcal{R}^*_{[0,r]}(X_0) \Rightarrow \exists x \in X_0,\, t \in [0,r] \mid z = e^{tA}x$. Let also $y \in S$ given as $y = x + \frac{t}{r}(e^{rA}x - x)$. Using the Taylor expansion of

$$e^{tA} = I_n + At + \textstyle\sum_{k=2}^{\infty} \frac{t^k}{k!}A^k, \tag{B.16}$$

where I_n is the identity matrix, we can write

$$\begin{aligned} z - y &= e^{tA}x - x - \tfrac{t}{r}(e^{rA}x - x) \\ &= \left(e^{tA} - I_n - \tfrac{t}{r}(e^{rA} - I_n)\right) x \\ &= \left(\textstyle\sum_{k=2}^{\infty} \frac{t^k}{k!}A^k - \frac{t}{r}\sum_{k=2}^{\infty} \frac{r^k}{k!}A^k\right) x \\ &= \left(\textstyle\sum_{k=2}^{\infty} \frac{t\left(t^{k-1} - r^{k-1}\right)}{k!}A^k\right) x \end{aligned} \tag{B.17}$$

Otherwise, $t\left(r^{k-1} - t^{k-1}\right) \leqslant r^k$ holds $\forall t \in [0,r]$, which leads to

$$\begin{aligned} \|z - y\| &= \left\| \left(\textstyle\sum_{k=2}^{\infty} \frac{t\left(t^{k-1} - r^{k-1}\right)}{k!}A^k\right) x \right\| \\ &\leqslant \textstyle\sum_{k=2}^{\infty} \left\| \frac{t\left(t^{k-1} - r^{k-1}\right)}{k!}A^k \right\| \|x\| \\ &\leqslant \textstyle\sum_{k=2}^{\infty} \left\| \frac{t\left(t^{k-1} - r^{k-1}\right)}{k!} \right\| \left\|A^k\right\| sup_{x\in X_0} \|x\| \\ &\leqslant \textstyle\sum_{k=2}^{\infty} \frac{r^k}{k!} \left\|A^k\right\| sup_{x\in X_0} \|x\| \\ &\leqslant \textstyle\sum_{k=2}^{\infty} \frac{r^k}{k!} \|A\|^k sup_{x\in X_0} \|x\| \\ &\leqslant \textstyle\sum_{k=2}^{\infty} \frac{r^k}{k!} \|A\|^k sup_{x\in X_0} \|x\| \end{aligned} \tag{B.18}$$

Moreover, taking note of the next Taylor expansion

$$e^{r\|A\|} = I_n + \|A\| r + \sum_{k=2}^{\infty} \frac{r^k}{k!} \|A\|^k, \tag{B.19}$$

the following inequality can be deduced

$$\|z - y\| \quad \leqslant \left(e^{r\|A\|} - I_n - \|A\| r\right) sup_{x \in X_0} \|x\| \tag{B.20}$$

and therewith $\alpha_r = \left(e^{r\|A\|} - I_n - \|A\| r\right) sup_{x \in X_0} \|x\|$. □

B.4. Proof of Lemma 3.7.3

The initial set can also be approximated by

$$\Omega_0 = CH(\bigcup_{\lambda \in [0,1]} \Omega_{0,\lambda}) \tag{B.21}$$

$$with\ \Omega_{0,\lambda} = (1-\lambda) X_0 \oplus \lambda e^{rA} X_0 \oplus \lambda (1-\lambda) \mathcal{E}_{X_0} \oplus \lambda r BU \oplus \lambda^2 \mathcal{E}_U \tag{B.22}$$

and

$$\begin{aligned} \mathcal{E}_{X_0} &= \square \left(|A|^{-1} \left(e^{r|A|} - I_n\right) \boxdot \left(A \left(I_n - e^{rA}\right) X_0\right)\right) \\ &\quad \oplus \square \left(|A|^{-2} \left(e^{r|A|} - I_n - r|A|\right) \boxdot \left(A^2 e^{rA} X_0\right)\right) \\ \mathcal{E}_U &= \square \left(|A|^{-2} \left(e^{r|A|} - I_n - r|A|\right) \boxdot (ABU)\right) \end{aligned} \tag{B.23}$$

Proof. The solution $x(t)$ of the differential equation (3.9) is given by

$$x(t) = e^{At} x_0 + \int_0^t e^{(t-s)A} Bu(s) ds. \tag{B.24}$$

If therefore we consider the following set

$$V_t = \left\{ \int_0^t e^{(t-s)A} Bu(s) ds : \forall s \in [0,t], u(s) \in U \right\}, \tag{B.25}$$

we get

$$R_t(X_0) = e^{tA} X_0 \oplus V_t. \tag{B.26}$$

The initial set, however, is approximated by the following expression

$$R_{[0,r]}(X_0) = CH(\bigcup_{t \in [0,r]} R_t(X_0)). \tag{B.27}$$

The objective of the proof in [71], was to find a tight approximation for the set $R_t(X_0)$ for $t \in [0,r]$. The proof has two parts. The first part, consisting in computing an approximation for the set $e^{tA} X_0$, was detailed in [71], whereas the proof for the approximation of the input contribution was there not really given. We propose, hence, the complete proof of the above lemma. Let's begin with the

approximation of the set $e^{tA}X_0$. We consider an $x_0 \in X_0$ and take into account that $e^{tA}x_0 = e^{(t-r)A}e^{rA}x_0$. Then

$$\begin{aligned}
e^{tA}x_0 &= (1-\tfrac{t}{r})e^{tA}x_0 + \tfrac{t}{r}e^{tA}x_0 \\
&= (1-\tfrac{t}{r})e^{tA}x_0 + \tfrac{t}{r}e^{(t-r)A}e^{rA}x_0 \\
&= \textstyle\sum_{i=0}^{\infty}(1-\frac{t}{r})\frac{t^i}{i!}A^ix_0 + \sum_{i=0}^{\infty}\frac{t}{r}\frac{(t-r)^i}{i!}A^ie^{rA}x_0 \\
&= (1-\tfrac{t}{r})x_0 + \tfrac{t}{r}e^{rA}x_0 + [\textstyle\sum_{i=1}^{\infty}(1-\frac{t}{r})\frac{t^i}{i!}A^ix_0 + \sum_{i=1}^{\infty}\frac{t}{r}\frac{(t-r)^i}{i!}A^ie^{rA}x_0] \\
&= (1-\tfrac{t}{r})x_0 + \tfrac{t}{r}e^{rA}x_0 + \tfrac{t}{r}(1-\tfrac{t}{r})[\textstyle\sum_{i=1}^{\infty} r\frac{t^{i-1}}{i!}A^ix_0 - \\
&\qquad \textstyle\sum_{i=1}^{\infty} r\frac{(t-r)^{i-1}}{i!}A^ie^{rA}x_0] \\
&= (1-\tfrac{t}{r})x_0 + \tfrac{t}{r}e^{rA}x_0 + \tfrac{t}{r}(1-\tfrac{t}{r})[\textstyle\sum_{i=1}^{\infty}(\frac{t}{r})^{i-1}\frac{r^i}{i!}A^ix_0 - \\
&\qquad \textstyle\sum_{i=1}^{\infty}\frac{r^i}{i!}(\frac{t}{r}-1)^{i-1}A^ie^{rA}x_0] \\
&= (1-\tfrac{t}{r})x_0 + \tfrac{t}{r}e^{rA}x_0 + \tfrac{t}{r}(1-\tfrac{t}{r})[\textstyle\sum_{i=1}^{\infty}(\frac{t}{r})^{i-1}\frac{r^i}{i!}A^i(I_n-e^{rA})x_0 + \\
&\qquad \textstyle\sum_{i=1}^{\infty}((\frac{t}{r})^{i-1} - (\frac{t}{r}-1)^{i-1})\frac{r^i}{i!}A^ie^{rA}x_0] \\
&= (1-\tfrac{t}{r})x_0 + \tfrac{t}{r}e^{rA}x_0 + \tfrac{t}{r}(1-\tfrac{t}{r})[\textstyle\sum_{i=1}^{\infty}(\frac{t}{r})^{i-1}\frac{r^i}{i!}A^i(I_n-e^{rA})x_0 + \\
&\qquad \textstyle\sum_{i=2}^{\infty}((\frac{t}{r})^{i-1} - (\frac{t}{r}-1)^{i-1})\frac{r^i}{i!}A^ie^{rA}x_0].
\end{aligned} \tag{B.28}$$

On the other hand, we know that for that $\forall\lambda \in [0,1]$ and $\forall i \in \mathbb{N}$

$$-1 \leqslant \lambda^i - (\lambda-1)^i \leqslant 1 \text{ and } -1 \leqslant \lambda^i \leqslant 1. \tag{B.29}$$

takin $\lambda = \frac{t}{r}$ leads consequently to the following equation

$$\begin{aligned}
e^{tA}x_0 - \left[(1-\tfrac{t}{r})x_0 + \tfrac{t}{r}e^{rA}x_0\right] \in\ & \tfrac{t}{r}(1-\tfrac{t}{r})[\textstyle\sum_{i=1}^{\infty}\frac{r^i}{i!}|A|^{i-1} \boxdot (A(I_n-e^{rA})X_0) \oplus \\
& \textstyle\sum_{i=2}^{\infty}\frac{r^i}{i!}|A|^{i-2} \boxdot (A^2e^{rA}X_0)] \\
e^{tA}x_0 - \left[(1-\tfrac{t}{r})x_0 + \tfrac{t}{r}e^{rA}x_0\right] \in\ & \tfrac{t}{r}(1-\tfrac{t}{r})[|A|^{-1}(e^{r|A|}-I_n) \boxdot (A(I_n-e^{rA})X_0) \oplus \\
& |A|^{-2}(e^{r|A|}-I_n-r|A|) \boxdot (A^2e^{rA}X_0)].
\end{aligned} \tag{B.30}$$

Therewith, we deduce that

$$\begin{aligned}
e^{tA}X_0 \subseteq (1-\lambda)X_0 \oplus \lambda e^{rA}X_0 \oplus \lambda(1-\lambda)[&\square(|A|^{-1}(e^{r|A|}-I_n) \boxdot (A(I_n-e^{rA})X_0)) \\
&\oplus \square(|A|^{-2}(e^{r|A|}-I_n-r|A|) \boxdot (A^2e^{rA}X_0))].
\end{aligned} \tag{B.31}$$

Concerning the input contribution, we have to prove that $V_t \subseteq (\frac{t}{r})rBU \oplus (\frac{t}{r})^2\mathcal{E}_U$ with

$$\mathcal{E}_U = \square\left(|A|^{-2}\left(e^{r|A|}-I_n-r|A|\right) \boxdot (ABU)\right). \tag{B.32}$$

For this purpose, we proceed in the same way as in proof B.2 by defining a vector $v \in V_t$.

$$\begin{aligned}
v &= \textstyle\int_0^t \sum_{i=0}^{\infty}\frac{(t-s)^i}{i!}A^iBu(s)ds \\
&= \textstyle\sum_{i=0}^{\infty}\int_0^t \frac{(t-s)^i}{i!}A^iBu(s)ds \\
&= \textstyle\int_0^t Bu(s)ds + \sum_{i=1}^{\infty}\int_0^t \frac{(t-s)^i}{i!}A^{i-1}ABu(s)ds.
\end{aligned} \tag{B.33}$$

We use the support function technique to find an over-approximation of the set V_t. For each direction $l \in \mathbb{R}^n$, we define the support function of the set V_t in direction

l as follows

$$\begin{aligned}\rho_{V_t}(l) &\leqslant sup_{v\in V_t}\left(\left(\int_0^t Bu(s)ds+\sum_{i=1}^{\infty}\int_0^t\frac{(t-s)^i}{i!}A^{i-1}ABu(s)ds\right).l\right)\\ &\leqslant \int_0^t sup_{u(s)\in U}\left(Bu(s).l\right)ds+\\ &\qquad\qquad \sum_{i=1}^{\infty}\left(\int_0^t\frac{(t-s)^i}{i!}\left|A^{i-1}\right|sup_{u(s)\in U}\left|ABu(s)\right|.\left|l(s)\right|ds\right)\\ &\leqslant t\rho_{BU}(l)+\sum_{i=1}^{\infty}\left(\int_0^t\frac{(t-s)^i}{i!}\left|A\right|^{i-1}ds\right)sup_{u(s)\in U}\left|ABu(s)\right|.\left|l\right|.\end{aligned} \tag{B.34}$$

Furthermore, we have

$$\sum_{i=1}^{\infty}\left(\int_0^r\frac{(t-s)^i}{i!}\left|A\right|^{i-1}ds\right) = \sum_{i=1}^{\infty}(\frac{t}{r})^{i+1}\frac{(r)^{i+1}}{(i+1)!}\left|A\right|^{i-1}. \tag{B.35}$$

We also know that $\forall t\in[0,r]$, $\forall i\in\mathbb{N}$ and $i\geqslant 2$, $(\frac{t}{r})^i\leqslant(\frac{t}{r})^2$, which leads to the following inequality

$$\sum_{i=1}^{\infty}\left(\int_0^r\frac{(t-s)^i}{i!}\left|A\right|^{i-1}ds\right)\leqslant(\frac{t}{r})^2\left|A\right|^{-2}\left(e^{r|A|}-I_n-r\left|A\right|\right). \tag{B.36}$$

We get finally

$$\rho_{V_t}(l) \leqslant t\rho_{BU}(l)+(\frac{t}{r})^2\left|A\right|^{-2}\left(e^{r|A|}-I_n-r\left|A\right|\right)sup_{u(s)\in U}\left|ABu(s)\right|.\left|l\right|. \tag{B.37}$$

That leads us to the following conclusion

$$V_t\subseteq(\frac{t}{r})\ rBU+(\frac{t}{r})^2\square\left(\left|A\right|^{-2}\left(e^{r|A|}-I_n-r\left|A\right|\right)\boxdot(ABU)\right). \tag{B.38}$$

□

Bibliography

[1] T. Alamo, J.M. Bravo, and E.F. Camacho. Guaranteed state estimation by zonotopes. *Automatica*, 41(6):1035–1043, 2005.

[2] M. Althoff. Continuous reachability analyzer (CORA). http://www6.in.tum.de/Main/SoftwareCORA, 2015. Software package.

[3] M. Althoff, O. Stursberg, and M. Buss. Reachability analysis of linear systems with uncertain parameters and inputs. In *Proc. of the 46th IEEE Conference on Decision and Control*, pages 726–732, 2007.

[4] M. Althoff, O. Stursberg, and M. Buss. Verification of uncertain embedded systems by computing reachable sets based on zonotopes. In *Proc. of the 17th IFAC World Congress*, 2008.

[5] M. Althoff, O. Stursberg, and M. Buss. Computing reachable sets of hybrid systems using a combination of zonotopes and polytopes. In *Nonlinear Analysis: Hybrid Systems*, volume vol. 4, issue 2, pages 233–249, May 2010.

[6] E. Asarin and G. Schneider. Widening the boundary between decidable and undecidable hybrid systems. In L. Brim, P. Jancar, M. Kret'inský, and A. Kucera, editors, *CONCUR 2002 - Concurrency Theory, 13th International Conference*, volume 2421 of *Lecture Notes in Computer Science*, pages 193–208, 2002.

[7] E. Asarin, T. Dang, and O. Maler. d/dt: A tool for reachability analysis of continuous and hybrid systems. In *the 5th IFAC Symposium Nonlinear Control Systems (NOLCOS)*, pages 3–34, 2001.

[8] E. Asarin, G. Schneider, and S. Yovine. On the decidability of the reachability problem for planar differential inclusions. In *HSCC'01 number 2034 in LNCS*, pages 89–104. Springer, 2001.

[9] E. Asarin, V. P. Mysore, A. Pnueli, and G. Schneider. Low dimensional hybrid systems - decidable, undecidable, don't know. *Inf. Comput.*, 211:138–159, Feb. 2012.

[10] R. Bagnara, P. M. Hill, and E. Zaffanella. The Parma Polyhedra Library: Toward a complete set of numerical abstractions for the analysis and verification of hardware and software systems. Quaderno 457, Dipartimento di Matematica, Universitá di Parma, Italy, 2006.

[11] R. Baier. *Mengenwertige Integration und die diskrete Approximation erreichbarer Mengen*. PhD thesis, Department of Mathematics, University of Bayreuth, Germany, 1994.

[12] R. Baier and M. Gerdts. Set-valued numerical analysis and optimal control. Lecture Notes for the DAAD Intensive Course Optimization - Theory and Applications, Jul. 05-17 2005.

[13] S. Bak. HyCreate: A tool for overapproximating reachability of hybrid automata. http://stanleybak.com/projects/hycreate/hycreate.html, 2013. Software package.

[14] A. Balluchi, A. Casagrande, P. Collins, A. Ferrari, T. Villa, and A. L. Sangiovanni-Vincentelli. Ariadne: a framework for reachability analysis of hybrid automata. In *Proc. of the international symposium on mathematical theory of networks and systems*, 2006.

[15] A. Balluchi, D. Bresolin, A. Casagrande, P. Collins, A. Ferrari, E. Mazzi, T. Villa, and A. L. Sangiovanni-Vincentelli. Reachability computation for hybrid systems with ariadne. In *Proc. of the 17th IFAC World Congress (IFAC'08), Seoul (Korea)*, Jul. 2008.

[16] H. H. Bauschke and P. L. Combettes. *Convex analysis and monotone operator theory in Hilbert spaces*. CMS Books in mathematics. Springer, New York, Dordrecht, Heidelberg, 2011. ISBN 978-1-4419-9466-0.

[17] I. Ben Makhlouf and S. Kowalewski. An evaluation of two recent reachability analysis tools for hybrid systems. In *the 2nd IFAC Conference on Analysis and Design of Hybrid Systems (ADHS'06)*, 2006.

[18] I. Ben Makhlouf and S. Kowalewski. Networked cooperative platoon of vehicles for testing methods and verification tools. In *ARCH14 CPSWeek Berlin*, 2014.

[19] I. Ben Makhlouf, S. Kowalewski, M. Chávez Grunewald, and D. Abel. Safety assessment of networked vehicle platoon controllers - practical experiences with available tools. In *the 3rd IFAC Conference on Analysis and Design of Hybrid Systems, Zaragoza, Spain (ADHS'09)*, 2009.

[20] I. Ben Makhlouf, J. P. Maschuw, P. Hänsch, H. Diab, S. Kowalewski, and D. Abel. Safety verification of a cooperative vehicle platoon with uncertain inputs using zonotopes. In *the 18th IFAC World Congress, 2011, Milano, Italy*, pages 9769–9774, 2011.

[21] I. Ben Makhlouf, Hilal H. Diab, and S. Kowalewski. Safety verification of a controlled cooperative platoon under loss of communication using zonotopes. In *Proc. of the 4th IFAC Conference on Analysis and Design of Hybrid Systems (ADHS'12), Eindhoven, NL*, pages 333–338, 2012.

[22] I. Ben Makhlouf, P. Hänsch, and S. Kowalewski. Comparison of reachability methods for uncertain linear time-invariant systems. In *ECC13: European Control Conference , Zurich, Switzerland Jul. 17-19*, pages 1101–1106, 2013.

[23] L. Benvenuti, D. Bresolin, P. Collins, A. Ferrari, L. Geretti, and T. Villa. Ariadne: Dominance checking of nonlinear hybrid automata using reachability analysis. In A. Finkel, L. Leroux, and I. Potapov, editors, *Reachability Problems*, volume 7550 of *Lecture Notes in Computer Science*, pages 79–91. Springer Berlin Heidelberg, 2012.

[24] O. Botchkarev and S. Tripakis. Verification of hybrid systems with linear differential inclusions using ellipsoidal approximations. In *HSCC'00. Volume 1790 of LNCS*, pages 73–88. Springer, 2000.

[25] O. Bournez, O. Maler, and A. Pnueli. Orthogonal polyhedra: Representation and computation. In *Schuppen (Eds.), Hybrid Systems: Computation and Control, LNCS 1569*, pages 46–60. Springer, 1999.

[26] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004. ISBN 0521833787.

[27] D. Bryce, J. Sun, W. Chen, P. Zuliani, Q. Wang, S. Gao, F. Shmarov, and S. Kong. dReach. https://dreal.github.io/dReach/, 2013. Software package.

[28] S. Cameron. Enhancing GJK: computing minimum and penetration distances between convex polyhedra. In *Proc. of the IEEE International Conference on Robotics and Automation*, volume 4, pages 3112–3117, Apr. 1997.

[29] 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, Jan. 2006.

[30] X. Chen, E. Abraham, and S. Sankaranarayanan. Taylor model flowpipe construction for non-linear hybrid systems. In *Proc. of the 33rd IEEE Real-Time Systems Symposium (RTSS'12)*, pages 183–192. IEEE Computer Society, 2012.

[31] X. Chen, E. Abraham, and S. Sankaranarayanan. Taylor model over-approximations for flowpipe/guard intersections. In *5th Int. Workshop on Numerical Software Verification (NSV'12)*, 2012.

[32] X. Chen, E. Abraham, and S. Sankaranarayanan. Flow*: An analyzer for non-linear hybrid systems. In *Proc. of the 25th Int. Conf. on Computer Aided Verification (CAV'13)*, volume 8044 of *LNCS*, pages 258–263. Springer-Verlag, 2013.

[33] X. Chen, S. Schupp, I. Ben Makhlouf, E. Abraham, G. Frehse, and S. Kowalewski. A benchmark suite for hybrid systems reachability analysis. In *Proc. of the 7th NASA Formal Methods Symposium (NFM'15)*, LNCS 9058, pages 408–414. Springer, 2015.

[34] T. Dang. *Vérification et synthèse des systèmes hybrides*. PhD thesis, INPG, Oct. 2000.

[35] H. Diab, I. Ben Makhlouf, and S. Kowalewski. A platoon of vehicles approaching an intersection: A testing platform for safe intersections. In *the 15th IEEE Intelligent Transportation Systems Conference*, pages 1918–1923. IEEE, 2012.

[36] A. Dubey, X. Wu, H. Su, and T. J. Koo. Computation platform for automatic analysis of embedded software systems using model based approach. In *Automated Technology for Verification and Analysis: Third International Symposium (ATVA 2005), volume 3707 of Lecture Notes in Computer Science*. Springer-Verlag, 2005.

[37] A. Fehnker and F. Ivancic. Benchmarks for hybrid systems verification. In *Proc. Hybrid Systems: Computation and Control (HSCC'04)*, pages 326–341. Springer, 2004.

[38] J. A. Ferrez, K. Fukuda, and Th.M. Liebling. Cuts, zonotopes and arrangements. Technical report, EPFL, 2001. RO 2001.11.05, chapter 10.

[39] M. Fränzle and C. Herde. HySAT: An efficient proof engine for bounded model checking of hybrid systems. *Formal Methods in System Design*, 30(3): 179–198, 2007.

[40] M. Fränzle, C. Herde, T.Teige, S. Ratschan, and T. Schubert. Efficient solving of large non-linear arithmetic constraint systems with complex boolean structure. *JSAT*, 1(3-4):209–236, 2007.

[41] G. Frehse. PHAVer: Algorithmic verification of hybrid systems past hytech. In *HSCC'05, LNCS 3414*, pages 258–273. Springer-Verlag, 2005.

[42] G. Frehse. An introduction to SpaceEx v0.8. http://spaceex.imag.fr/documentation/user-documentation/introduction-spaceex-27, 2010.

[43] G. Frehse and R. Ray. Flowpipe-guard intersection for reachability computations with support functions. In *IFAC Conference Analysis and Design of Hybrid Systems (ADHS'12)*, pages 94–101, 2012.

[44] G. Frehse, C. Le Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler. SpaceEx: Scalable verification of hybrid systems. In Shaz Qadeer Ganesh Gopalakrishnan, editor, *Proc. of the 23rd*

International Conference on Computer Aided Verification (CAV'11), LNCS. Springer, 2011.

[45] P. K. Ghosh and K. V. Kumar. Support function representation of convex bodies, its application in geometric computing, and some related representations. *Computer Vision and Image Understanding*, 72(3):379–403, 1998.

[46] E.G. Gilbert, D.W. Johnson, and S.S. Keerthi. A fast procedure for computing the distance between complex objects in three-dimensional space. *Journal of Robotics and Automation*, 4(2):193–203, Apr. ISSN 0882-4967.

[47] A. Girard. Reachability of uncertain linear systems using zonotopes. *Hybrid Systems Computation and Control*, 3414(206):291–305, 2005.

[48] A. Girard and C. Guernic. Zonotope/hyperplane intersection for hybrid systems reachability analysis. In *Proc. of the 11th international workshop on Hybrid Systems: Computation and Control*, HSCC'08, pages 215–228, Berlin, Heidelberg, 2008. Springer-Verlag.

[49] A. Girard and C. Le Guernic. Efficient reachability analysis for linear systems using support functions. *the 17th IFAC World Congress*, 2008.

[50] R. Goebel, R.G. Sanfelice, and A.R. Teel. Hybrid dynamical systems. *Control Systems*, 29(2):28–93, 2009.

[51] M. Goyal. Reachability analysis of hybrid systems: An experience report. In *International Journal of Modeling and Optimization*, volume Vol.2, No.6, pages 681–682. IACSI Press, 2012.

[52] M. R. Greenstreet and I. Mitchell. Reachability analysis using polygonal projections. In *In hybrid systems: computation and control*, pages 103–116. Springer, 1999.

[53] P. Guerra and V. Puig. Passive robust fault detection using interval MA parity equations: Inverse vs direct image tests. In Aarti Gupta and Sharad Malik, editors, *Proc. of the 17th IFAC World Congress*, volume 17, pages 4523–4528, 2008.

[54] L. J. Guibas, A. Nguyen, and L. Zhang. Zonotopes as bounding volumes. In *Proc. of 14th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 803–812, 2003.

[55] Z. Han and B. H. Krogh. Reachability analysis of large-scale affine systems using low-dimensional polytopes. In *HSCC'06*, pages 287–301, 2006.

[56] T. A. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya. What's decidable about hybrid automata? In *Journal of Computer and System Sciences*, pages 373–382. ACM Press, 1995.

[57] T.A. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya. What's decidable about hybrid automata. *Journal of Computer and Systems Sciences*, 57:94–124, 1998.

[58] J.B. Hiriart-Urruty and C. Lemaréchal. *Fundamentals of Convex Analysis*. Grundlehren Text Editions. Springer Berlin Heidelberg, 2001. ISBN 9783540422051.

[59] I. A. Hiskens. Stability of limit cycles in hybrid systems. In *Proc. of the 34th Hawaii International Conf. on System Sciences*, pages 163–328, 2001.

[60] I. Hwang, D. Stipanovic, and C. J. Tomlin. Polytopic approximations of reachable sets applied to linear dynamic games and a class of nonlinear systems. In E.H. Abed, editor, *Advances in Control, Communication Networks, and Transportation Systems*, Systems and Control: Foundations and Applications, pages 3–19. Birkhäuser Boston, 2005.

[61] HyPro. A toolbox for the reachability analysis of hybrid systems using geometric approximations. http://ths.rwth-aachen.de/research/hypro/, 2014.

[62] S. Kong, S. Gao, W. Chen, and E. Clarke. dReach: δ-reachability analysis for hybrid systems. In C. Baier and C. Tinelli, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, volume 9035 of *Lecture Notes in Computer Science*, pages 200–205. Springer Berlin Heidelberg, 2015.

[63] A. B. Kurzhanski and P. Varaiya. Ellipsoidal techniques for reachability analysis. In *Proc. of the Third International Workshop on Hybrid Systems: Computation and Control (HSCC'00)*, pages 202–214, London, UK, 2000. Springer-Verlag.

[64] M. Kvasnica, P. Grieder, M. Baotic, and M. Morari. Multi-parametric toolbox (mpt). In R. Alur and G. J. Pappas, editors, *HSCC'04*, volume 2993 of *Lecture Notes in Computer Science*, pages 448–462. Springer, 2004.

[65] G. Lafferriere, G. J. Pappas, and S. Yovine. Reachability computation for linear hybrid systems. In *Proc. of the 14th IFAC World Congress, volume E*, pages 7–12. Elsevier Science Ltd, 1998.

[66] G. Lafferriere, G. J. Pappas, and S. Yovine. A new class of decidable hybrid systems. In *Proc. Hybrid Systems Computation and Control (HSCC'99)*, pages 137–151. Springer, 1999.

[67] A. Lalami. *Diagonostic et approaches ensemblistes à base des zonotopes*. PhD thesis, Université de Cergy-Pontoise, 2008.

[68] A. Lalami and C. Combastel. Generation of set membership tests for fault diagnosis and evaluation of their worst case sensitivity. In *Proc. of the 6th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes*, pages 569–574, Aug. 29 2006.

[69] V.T.H. Le, C. Stoica, T. Alamo, E.F. Camacho, and D. Dumur. Zonotope-based set-membership estimation for multi-output uncertain systems. In *the IEEE International Symposium on Intelligent Control (ISIC'13)*, pages 212–217, Aug. 2013.

[70] C. Le Guernic. *Analyse algorithmique des systèmes hybrides*. PhD thesis, VERIMAG, Sep. 2009.

[71] C. Le Guernic. *Reachability Analysis of Hybrid Systems with Linear Continuous Dynamics*. PhD thesis, VERIMAG, Oct. 2009.

[72] C. Le Guernic and A. Girard. Reachability analysis of hybrid systems using support functions. In *Proc. of the 21st International Conference on Computer Aided Verification (CAV'09)*, pages 540–554, Berlin, Heidelberg, 2009. Springer-Verlag.

[73] C. Le Guernic and A. Girard. Reachability analysis of linear systems using support functions. *Nonlinear Analysis: Hybrid Systems*, 4(2):250–262, 2010.

[74] J. Lygeros. Lecture notes on hybrid systems. Technical report, Department of Electrical and Computer Engineering University of Patras, Rio, Patras, GR-26500, Greece, 2004.

[75] I. Ben Makhlouf and S. Kowalewski. Optimizing safe control of a networked platoon of trucks using reachability. In *ARCH'15 CPS Week, Seatle, WA, USA April 13*, 2015.

[76] K. L. Man, T. Krilavicius, and Kaiyu Wan. Recent advanced languages and tools for hybrid systems. *International Journal of Computer Science*, 37(3): 224–233, 2010.

[77] J. P. Maschuw and D. Abel. Longitudinal vehicle guidance in networks with changing communication topology. In *the IFAC-Symposium Advances in Automotive Control (AAC 2010)*, München, Jul. 2010.

[78] J. P. Maschuw, G. C. Keßler, and D. Abel. LMI-based control of vehicle platoons for robust longitudinal guidance. In *the IFAC World Congress 2008*, Seoul, Jul. 2008.

[79] Mathworks. *Optimization Toolbox Documentation*, 2014.

[80] W. Min, Y. Gangfeng, L. Zhiyun, and L. Meiqin. Characterization of backward reachable set and positive invariant set in polytopes. In *American Control Conference (ACC '09)*, pages 4351–4356, June 2009.

[81] I. M. Mitchell. A toolbox of level set methods. http://www.cs.ubc.ca/ mitchell/ToolboxLS/, June 2007. UBC Department of Computer Science Technical Report TR-2007-11.

[82] J. Nagel. GJK collision detection algorithm wanted. comp.graphics.algorithms newsgroup, Apr. 1998.

[83] L. Olvång. Real-time collision detection with implicit objects. Uppsala University, Department of Information Technology, 2010.

[84] A. Pinto, L. P. Carloni, R. Passerone, and A. Sangiovanni-Vincentelli. Interchange format for hybrid systems: Abstract semantics. In J. Hespanha and A. Tiwari, editors, *Hybrid Systems Computation and Control*, volume 3927 of *Lecture Notes in Computer Science*, pages 491–506. Springer Berlin Heidelberg, 2006.

[85] A. Platzer. Differential dynamic logic for hybrid systems. *Journal of Automated Reasoning*, 41(2):143–189, 2008.

[86] A. Platzer and E.M. Clarke. Computing differential invariants of hybrid systems as fixedpoints. In Aarti Gupta and Sharad Malik, editors, *Proc. of Computer-Aided Verification (CAV'08) , Princeton, USA*, volume 5123 of *LNCS*, pages 176–189. Springer, 2008.

[87] A. Platzer and J.D. Quesel. KeYmaera: A hybrid theorem prover for hybrid systems. In Alessandro Armando, Peter Baumgartner, and Gilles Dowek, editors, *Proc. of Third International Joint Conference Automated Reasoning (IJCAR) 2008, Sydney, Australia*, volume 5195 of *LNCS*, pages 171–178. Springer, 2008.

[88] A. Platzer and J.D. Quesel. KeYmaera. http://symbolaris.com/download.html, 2008. Software package.

[89] P. Prabhakar, V. Vladimerou, M. Viswanathan, and G. E. Dullerud. A decidable class of planar linear hybrid systems. In Magnus Egerstedt and Bud Mishra, editors, *HSCC'08*, volume 4981 of *Lecture Notes in Computer Science*, pages 401–414. Springer, 2008. ISBN 978-3-540-78928-4.

[90] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, New York, NY, USA, 3 edition, 2007. ISBN 0521880688, 9780521880688.

[91] A. Puri and P. Varaiya. Decidability of hybrid systems with rectangular differential inclusions. In *Proc. of the 6th International Conference on Computer Aided Verification (CAV 1994), LNCS 818*, pages 95–104. Springer-Verlag, 1994.

[92] A. Puri, P. Varaiya, and V. Borkar. ϵ-approximation of differential inclusions. In *Proc. of the 34th IEEE Conference on Decision and Control*, volume 3, pages 2892–2897, 1995.

[93] S. Ratschan and Z. She. Safety verification of hybrid systems by constraint propagation based abstraction refinement. In *HSCC'05, LNCS 3414*, pages 573–589. Springer-Verlag, 2005.

[94] S. Ratschan and Z. She. Constraints for continuous reachability in the verification of hybrid systems. In *Proc. of the 8th Int. Conf. on Artif. Intell. and Symb. Comp., AISC'2006*, number 4120 in LNCS, pages 196–210. Springer, 2006.

[95] S. Ratschan and Z. She. Safety verification of hybrid systems by constraint propagation based abstraction refinement. *ACM Transactions in Embedded Computing Systems*, 6(1), 2007.

[96] R. Ray. *Reachability Analysis of Hybrid Systems using Support Functions*. PhD thesis, Grenoble University, May 2012.

[97] R. T. Rockafellar, R. J.-B Wets, and M. Wets. *Variational analysis*. Grundlehren der mathematischen Wissenschaften. Springer, Berlin, Heidelberg, New York, 1998. ISBN 3-540-62772-3.

[98] R.T. Rockafellar. *Convex Analysis*. Convex Analysis. Princeton University Press, 1997. ISBN 9780691015866.

[99] K. Scheibler, S. Kupferschmid, and B. Becker. Recent improvements in the SMT solver iSAT. In C. Haubelt and D. Timmermann, editors, *MBMV*, pages 231–241. Institut für Angewandte Mikroelektronik und Datentechnik, Fakultät für Informatik und Elektrotechnik, Universität Rostock, 2013.

[100] K. Scheibler, A. Mahdi, A. Disterhof, F. Neubauer, L. Winterer, and T. Paxian. iSAT3. https://projects.informatik.uni-freiburg.de/projects/isat3/, 2013. Software package.

[101] B. I. Silva and B. H. Krogh. Formal verification of hybrid systems using CheckMate: a case study. In *Proc. of the American Control Conference*, volume 3, pages 1679–1683 vol.3, 2000.

[102] SPP1305. DFG-priority program 1305, control theory of digitally networked dynamical systems. http://spp-1305.atp.ruhr-uni-bochum.de/index.php?site=startseite, 2007-2011. http://spp-1305.atp.ruhr-uni-bochum.de/index.php?site=abel.

[103] J. Sprinkle. Generative components for hybrid systems tools. *Journal of Object Technology*, 4(3):33–38, Apr. 2005. ISSN 1660-1769. GPCE Young Researchers Workshop 2004.

[104] O. Stursberg and B. H. Krogh. On efficient representation and computation of reachable sets for hybrid systems. In *HSCC'03, LNCS 2289*, pages 482–497. Springer, 2003.

[105] K. Sundaraj, D. d'Aulignac, and E. Mazer. A new algorithm for computing minimum distance. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000)*, volume 3, pages 2115–2120, 2000.

[106] S. M. Tabatabaeipour and J. Stoustrup. Set-membership state estimation for discrete time piecewise affine systems using zonotopes. In *ECC13: European Control Conference , Zürich, Switzerland*, pages 3143–3148, Jul. 17-19, 2013.

[107] G. Van den Bergen. A fast and robust GJK implementation for collision detection of convex objects. *Journal of Graphics Tools*, 4(2):7–25, Mar. 1999. ISSN 1086-7651.

[108] G. Van den Bergen. Proximity queries and penetration depth computation on 3D game objects. In *Proc. of Game Developers Conference*, San Jose, CA, Mar. 2001.

[109] C. Van Loan. Computing integrals involving the matrix exponential. In *the IEEE Trans. Autom. Control 23*, pages 395–404, 1978.

[110] P. Varaiya. Reach set computation using optimal control. In *Proc. KIT Workshop*, pages 377–383, 1997.

Own Publication

[1] I. Ben Makhlouf, J. Gan and S. Kowalewski, A Study on Solving Guard and Invariant Set Intersection in Zonotope-based Reachability of Linear Hybrid Systems, in Proc. 5th IFAC Conference on Analysis and Design of Hybrid Systems (ADHS'15) October 14-16, Georgia Tech, Atlanta, USA., 2015

[2] I. Ben Makhlouf and S. Kowalewski, Optimizing Safe Control of a Networked Platoon of Trucks Using Reachability, in Proc. ARCH'15 CPS Week, Seatle, WA, USA April 13, 2015

[3] X. Chen, S. Schupp, I. Ben Makhlouf, E. Abraham, G. Frehse and S. Kowalewski, A benchmark suite for hybrid systems reachability analysis, in Proc. NASA Formal Methods - 7th International Symposium, NFM 2015, Pasadena, CA, USA, April 27-29, 2015, Proceedings, 2015, vol. 9058 in Lecture Notes in Computer Science, Springer, pp. 408-414

[4] S. Schupp, E. Abraham, X. Chen, I. Ben Makhlouf, G. Frehse, S. Sankaranarayanan and S. Kowalewski, Current Challenges in the Verification of Hybrid Systems, in Proc. Fifth Workshop on Design, Modeling and Evaluation of Cyber Physical Systems (CyPhy'15), Held in conjunction with ESWEEK 2015 on October 8, 2015, Amsterdam, The Netherlands

[5] I.Ben Makhlouf and S. Kowalewski, Networked Cooperative Platoon of Vehicles for Testing Methods and Verification Tools, in Proc. ARCH14 CPSWeek Berlin, 2014

[6] I. Ben Makhlouf, H. Diab and S. Kowalewski, Reachability Analysis for Managing Platoons at Intersections, in Proc. 21st Mediterranean Conference on Control and Automation (MED'13) Platanias-Chania, Crete, Greece, June 25-28, 2013, IEEE, pp. 1141-1147

[7] I. Ben Makhlouf, P. Hänsch and S. Kowalewski, Comparison of Reachability Methods for Uncertain Linear Time-Invariant Systems, in Proc. ECC13: European Control Conference, Zurich, Switzerland July 17-19, 2013, EUCA, pp. 1101-1106

[8] I. Ben Makhlouf, H. Diab and S. Kowalewski, Safety Verification of a Controlled Cooperative Platoon Under Loss of Communication Using Zonotopes, in Proc. ADHS 2012, Eindhoven, NL, 2012, In proc. of the 4th IFAC Conference on Analysis and Design of Hybrid Systems (ADHS 12), pp. 333-338

[9] H. Diab, I. Ben Makhlouf and S. Kowalewski, A Platoon of Vehicles Approaching an Intersection: A Testing Platform for Safe Intersections, in Proc. 15th IEEE Intelligent Transportation Systems Conference, 2012, IEEE, pp. 1918-1923

[10] I. Ben Makhlouf, J. P. Maschuw, P. Hänsch, H. Diab, S. Kowalewski and D. Abel, Safety Verification of a Cooperative Vehicle Platoon with Uncertain Inputs Using Zonotopes, in Proc. 18th IFAC World Congress, 2011, Milano, Italy, 2011, IFAC, pp. 9769-9774

[11] P. Hänsch, H. Diab, I. Ben Makhlouf and S. Kowalewski, Reachability Analysis of Linear Systems with Stepwise Constant Inputs, in Proc. Electronic Notes in Theoretical Computer Science, 2011, vol. 297, Elsevier, pp. 61-74, Proceedings of the first workshop on Hybrid Autonomous System (HAS2011)

[12] M. Chávez Grunewald, I. Ben Makhlouf, H. Diab, V. Mut, S. Kowalewski and D. Abel, Regelung und Sicherheitsanalyse einer Gruppe Massenpunktfahrzeuge mit Hilfe energiebasierter Methoden, at - Automatisierungstechnik, vol. 58, iss. 4, pp. 227-235, 2010

[13] M. Chávez Grunewald, I.Ben Makhlouf, H. Diab, D. Abel and S. Kowalewski, On the Effects of Network Delays on an Energy-based Controller, in Proc. 2nd IFAC Workshop on Distributed Estimation and Control in Networked Systems (NecSys' 2010), 2010, pp. 169-174

[14] H. Diab, M. Chávez Grunewald, I. Ben Makhlouf, D. Abel and S. Kowalewski, A Testing Platform for Cooperative Vehicle Platoon Controllers, in Proc. 13th International IEEE Conference on Intelligent Transportation Systems (ITSC 2010), 2010, IEEE, pp. 1718-1723

[15] I. Ben Makhlouf, S. Kowalewski, M. G. Chávez Grunewald and D. Abel, Safety Assessment of Networked Vehicle Platoon Controllers - Practical Experiences With Available Tools, in Proc. 3rd IFAC Conference on Analysis and Design of Hybrid Systems, Zaragoza, Spain 2009

[16] I. Ben Makhlouf and S. Kowalewski, S., An Evaluation of Two Recent Reachability Analysis Tools for Hybrid Systems, in Proc. 2nd IFAC Conference on Analysis and Design of Hybrid Systems, 2006

Aachener Informatik-Berichte

This list contains all technical reports published during the past five years. A complete list of reports dating back to 1987 is available from

http://aib.informatik.rwth-aachen.de/.

To obtain copies consult the above URL or send your request to: Informatik-Bibliothek, RWTH Aachen, Ahornstr. 55, 52056 Aachen, Email: biblio@informatik.rwth-aachen.de

2013-01 *	Fachgruppe Informatik: Annual Report 2013
2013-02	Michael Reke: Modellbasierte Entwicklung automobiler Steuerungssysteme in Klein- und mittelständischen Unternehmen
2013-03	Markus Towara and Uwe Naumann: A Discrete Adjoint Model for OpenFOAM
2013-04	Max Sagebaum, Nicolas R. Gauger, Uwe Naumann, Johannes Lotz, and Klaus Leppkes: Algorithmic Differentiation of a Complex C++ Code with Underlying Libraries
2013-05	Andreas Rausch and Marc Sihling: Software & Systems Engineering Essentials 2013
2013-06	Marc Brockschmidt, Byron Cook, and Carsten Fuhs: Better termination proving through cooperation
2013-07	André Stollenwerk: Ein modellbasiertes Sicherheitskonzept für die extrakorporale Lungenunterstützung
2013-08	Sebastian Junges, Ulrich Loup, Florian Corzilius and Erika Ábrahám: On Gröbner Bases in the Context of Satisfiability-Modulo-Theories Solving over the Real Numbers
2013-10	Joost-Pieter Katoen, Thomas Noll, Thomas Santen, Dirk Seifert, and Hao Wu: Performance Analysis of Computing Servers using Stochastic Petri Nets and Markov Automata
2013-12	Marc Brockschmidt, Fabian Emmes, Stephan Falke, Carsten Fuhs, and Jürgen Giesl: Alternating Runtime and Size Complexity Analysis of Integer Programs
2013-13	Michael Eggert, Roger Häußling, Martin Henze, Lars Hermerschmidt, René Hummen, Daniel Kerpen, Antonio Navarro Pérez, Bernhard Rumpe, Dirk Thißen, and Klaus Wehrle: SensorCloud: Towards the Interdisciplinary Development of a Trustworthy Platform for Globally Interconnected Sensors and Actuators
2013-14	Jörg Brauer: Automatic Abstraction for Bit-Vectors using Decision Procedures
2013-16	Carsten Otto: Java Program Analysis by Symbolic Execution
2013-19	Florian Schmidt, David Orlea, and Klaus Wehrle: Support for error tolerance in the Real-Time Transport Protocol

2013-20 Jacob Palczynski: Time-Continuous Behaviour Comparison Based on Abstract Models

2014-01 * Fachgruppe Informatik: Annual Report 2014

2014-02 Daniel Merschen: Integration und Analyse von Artefakten in der modellbasierten Entwicklung eingebetteter Software

2014-03 Uwe Naumann, Klaus Leppkes, and Johannes Lotz: dco/c++ User Guide

2014-04 Namit Chaturvedi: Languages of Infinite Traces and Deterministic Asynchronous Automata

2014-05 Thomas Ströder, Jürgen Giesl, Marc Brockschmidt, Florian Frohn, Carsten Fuhs, Jera Hensel, and Peter Schneider-Kamp: Automated Termination Analysis for Programs with Pointer Arithmetic

2014-06 Esther Horbert, Germán Martín García, Simone Frintrop, and Bastian Leibe: Sequence Level Salient Object Proposals for Generic Object Detection in Video

2014-07 Niloofar Safiran, Johannes Lotz, and Uwe Naumann: Algorithmic Differentiation of Numerical Methods: Second-Order Tangent and Adjoint Solvers for Systems of Parametrized Nonlinear Equations

2014-08 Christina Jansen, Florian Göbe, and Thomas Noll: Generating Inductive Predicates for Symbolic Execution of Pointer-Manipulating Programs

2014-09 Thomas Ströder and Terrance Swift (Editors): Proceedings of the International Joint Workshop on Implementation of Constraint and Logic Programming Systems and Logic-based Methods in Programming Environments 2014

2014-14 Florian Schmidt, Matteo Ceriotti, Niklas Hauser, and Klaus Wehrle: HotBox: Testing Temperature Effects in Sensor Networks

2014-15 Dominique Gückel: Synthesis of State Space Generators for Model Checking Microcontroller Code

2014-16 Hongfei Fu: Verifying Probabilistic Systems: New Algorithms and Complexity Results

2015-01 * Fachgruppe Informatik: Annual Report 2015

2015-02 Dominik Franke: Testing Life Cycle-related Properties of Mobile Applications

2015-05 Florian Frohn, Jürgen Giesl, Jera Hensel, Cornelius Aschermann, and Thomas Ströder: Inferring Lower Bounds for Runtime Complexity

2015-06 Thomas Ströder and Wolfgang Thomas (Editors): Proceedings of the Young Researchers' Conference "Frontiers of Formal Methods"

2015-07 Hilal Diab: Experimental Validation and Mathematical Analysis of Cooperative Vehicles in a Platoon

2015-08 Mathias Pelka, Jó Agila Bitsch, Horst Hellbrück, and Klaus Wehrle (Editors): Proceedings of the 1st KuVS Expert Talk on Localization

2015-09 Xin Chen: Reachability Analysis of Non-Linear Hybrid Systems Using Taylor Models

2015-11 Stefan Wüller, Marián Kühnel, and Ulrike Meyer: Information Hiding in the Public RSA Modulus

2015-12 Christoph Matheja, Christina Jansen, and Thomas Noll: Tree-like Grammars and Separation Logic

2015-13 Andreas Polzer: Ansatz zur variantenreichen und modellbasierten Entwicklung von eingebetteten Systemen unter Berücksichtigung regelungs- und softwaretechnischer Anforderungen

2015-14 Niloofar Safiran and Uwe Naumann: Symbolic vs. Algorithmic Differentiation of GSL Integration Routines

* These reports are only available as a printed version.

Please contact emailbiblio@informatik.rwth-aachen.de to obtain copies.