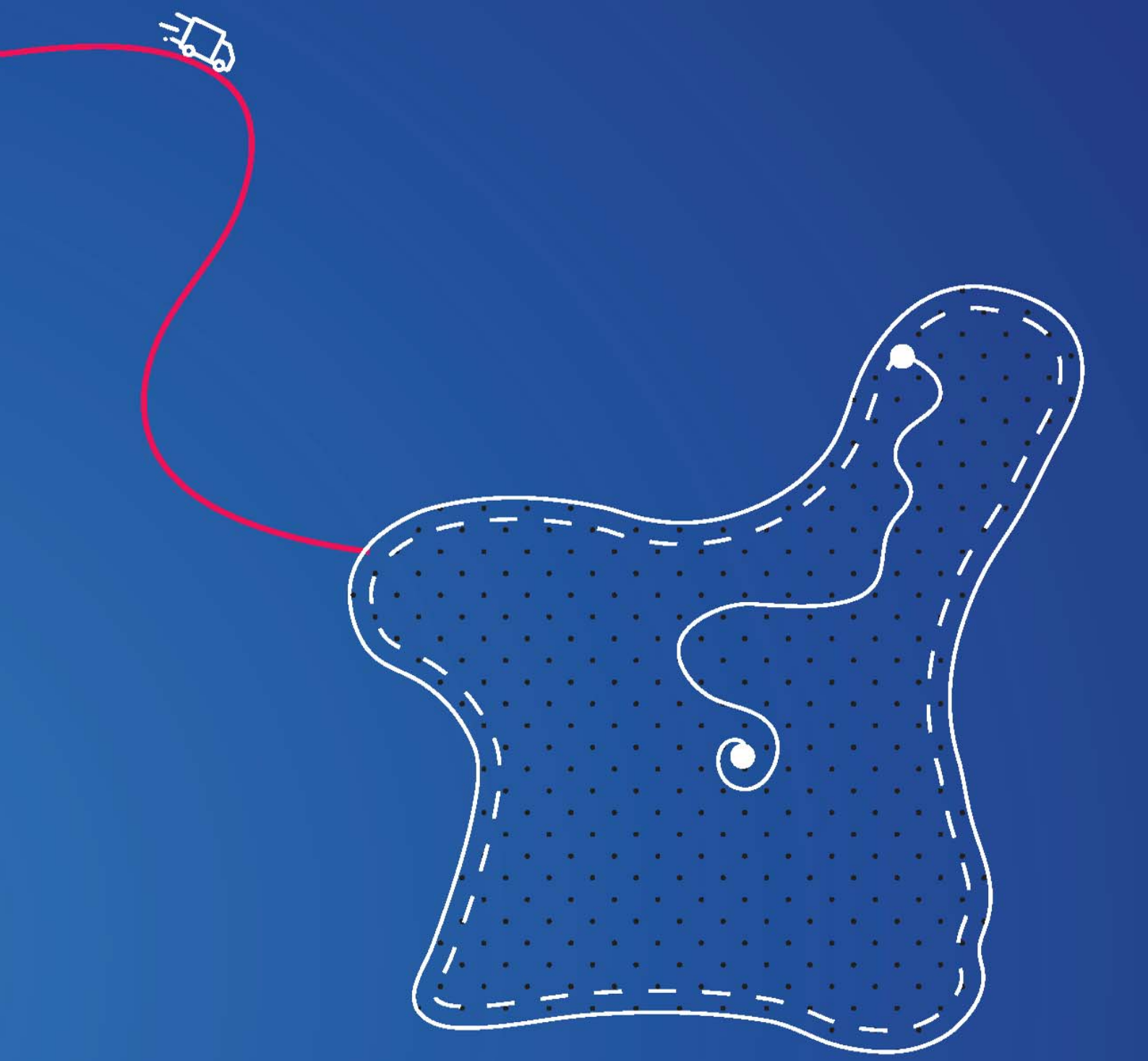


Ruxandra
Bobiti

Sampling-driven stability domains computation and predictive control of constrained nonlinear systems

Sampling–driven stability domains computation and predictive control of constrained nonlinear systems

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de
Technische Universiteit Eindhoven, op gezag van de
rector magnificus, prof.dr.ir. F.P.T. Baaijens, voor een
commissie aangewezen door het College voor
Promoties in het openbaar te verdedigen op
woensdag 25 oktober 2017 om 16:00 uur

door

Ruxandra Valentina Bobiti

geboren te Drobeta–Turnu Severin, Roemenië

Dit proefschrift is goedgekeurd door de promotoren en de samenstelling van de promotiecommissie is als volgt:

voorzitter:	Prof.dr.ir. J.H. Blom
1e promotor:	Prof.dr.ir. P.M.J. Van den Hof
copromotor:	Dr. M. Lazar
leden:	Prof.dr. C. De Persis (RUG)
	Dr. P. Goulart (University of Oxford)
	Prof.dr.ir. N. van de Wouw
	Dr. A. Girard (Laboratoire des Signaux et Systemes - CNRS)
	Dr.ir. T. Keviczky (TUD)

Het onderzoek dat in dit proefschrift wordt beschreven is uitgevoerd in overeenstemming met de TU/e Gedragscode Wetenschapsbeoefening.

Sampling–driven stability domains computation and predictive control of constrained nonlinear systems

Motto:

"*Simplicity is complexity resolved.*"

Constantin Brâncuşi

disc

This dissertation has been completed in fulfillment of the requirements of the Dutch Institute of Systems and Control (DISC). The author has successfully completed the educational program of the DISC Graduate School.

A catalogue record is available from the Eindhoven University of Technology Library.

Sampling–driven stability domains computation and predictive control of constrained nonlinear systems
by Ruxandra Bobiti
Eindhoven: Technische Universiteit Eindhoven, 2017.
Proefschrift.

ISBN: 978-90-386-4353-3
NUR: 992

Cover design: Andrei Chirişi
Reproduction: Gildeprint, Enschede, The Netherlands

To my parents,
Luciana and Angelu.

Contents

Preliminaries **10**
Summary . . . 10
Basic notation and definitions . . . 13
List of abbreviations . . . 16

1 Introduction **17**
1.1 Background and motivation . . . 17
1.1.1 Nonlinearities in real–life systems . . . 17
1.1.2 Safety analysis and control of constrained systems . . . 19
1.1.3 Abstractions and sampling . . . 21
1.2 Research objectives . . . 22
1.2.1 The stability analysis problem for constrained nonlinear systems . . . 22
1.2.2 Predictive control for constrained nonlinear systems . . . 23
1.2.3 Common challenges in stability analysis and predictive control design 24
1.2.4 Sampling methods for stability verification . . . 25
1.2.5 Sampling in predictive control . . . 26
1.2.6 Research question . . . 27
1.3 Outline of the thesis . . . 29
1.4 Summary of publications . . . 30

2 Optimization based stability analysis results for discrete–time systems **33**
2.1 Introduction . . . 33
2.2 System class and finite–step Lyapunov functions . . . 34
2.2.1 Finite–step Lyapunov functions . . . 35
2.3 Computation of LFs on compact sets . . . 37
2.3.1 LF from a FSLF on a compact set . . . 37
2.3.2 A systematic approach for computing regional LFs . . . 39
2.3.3 Computational remarks . . . 41
2.3.4 An iterative algorithm to compute a Lyapunov function and a DOA 42
2.4 Examples . . . 44
2.4.1 Example 1 . . . 44
2.4.2 Example 2 . . . 45
2.5 Conclusions . . . 46

3 Deterministic sampling–driven verification of inequalities on compact sets 47
3.1 Introduction . . . 47
3.2 Sampling with hyper–rectangles . . . 48
3.3 Verification of inequalities prescribed by real–valued functions . . . 49
3.3.1 Problem setup . . . 50
3.3.2 Sampling–driven verification: Fundamental theorem . . . 52
3.3.3 Multi–resolution sampling tools . . . 54
3.3.4 Construction of $\bar{\gamma}$ . . . 55
3.3.5 Prototype sampling–driven verification algorithm . . . 57
3.3.6 Comparison with the method of set inversion via interval analysis . 59
3.4 Sampling–driven verification of finite–step invariance . . . 60
3.4.1 Finite–step invariance verification . . . 60
3.4.2 Numerical example . . . 61
3.5 Conclusions . . . 62

4 Deterministic sampling–driven computation of stability domains 63
4.1 Introduction . . . 63
4.2 Stability analysis tools . . . 65
4.3 Verification of Lyapunov's inequality for discrete–time systems . . . 67
4.4 Verification of Lyapunov's inequality for continuous–time systems . . . 69
4.4.1 Constructing LFs via the discretized system . . . 70
4.4.2 Constructing LFs via polynomial approximation of the solution . . 71
4.5 Sampling–driven DOA estimation . . . 73
4.6 Reflection on the sampling–driven verification framework . . . 75
4.6.1 On automating the process of stability domains computation . . . 76
4.6.2 Comparison with existing methods . . . 77
4.7 Examples . . . 78
4.7.1 Discrete–time 2D continuous dynamics . . . 78
4.7.2 Discrete–time 2D piecewise continuous dynamics . . . 80
4.7.3 Continuous–time 3D continuous dynamics . . . 82
4.7.4 Powertrain Control System . . . 83
4.7.5 Continuous–time systems with polynomial solution approximation . 85
4.8 Conclusions . . . 86

5 On randomized stability analysis for large scale nonlinear systems 89
5.1 Introduction . . . 89
5.2 Preliminaries on randomized certification of property functions . . . 90
5.3 Constructive randomized algorithm for verifying Lyapunov's inequality and finding DOA with a low number of samples . . . 93
5.4 Finding a Lyapunov function candidate . . . 96
5.5 Randomized DOA estimation . . . 97
5.6 Examples . . . 100
5.6.1 2D system . . . 100
5.6.2 Cascade cart—spring–damper systems . . . 102
5.7 Reflection on randomized methods for DOA estimation . . . 104

6 Sampling–driven nonlinear model predictive control **107**
6.1 Introduction . . . 107
6.2 Suboptimal MPC problem formulation . . . 109
6.2.1 Problem formulation . . . 109
6.2.2 Existing NMPC approaches based on sampling . . . 111
6.3 Prototype algorithm . . . 112
6.4 Complexity analysis . . . 115
6.5 Convergence analysis . . . 116
6.5.1 Sufficient conditions for convergence . . . 116
6.5.2 Convergence conditions under uniform deterministic sampling . . . 119
6.5.3 Convergence conditions under random sampling . . . 121
6.6 Input smoothing . . . 122
6.7 Illustrative examples . . . 123
6.7.1 Cart–spring system . . . 123
6.7.2 Buck–Boost power converter . . . 128
6.7.3 Wheeled mobile robot . . . 130
6.8 Conclusion . . . 132

7 Conclusions and recommendations **133**
7.1 Overview of the results . . . 133
7.1.1 Sampling–driven analysis . . . 133
7.1.2 Sampling–driven nonlinear model predictive control . . . 135
7.2 Recommendations and future work . . . 135
7.2.1 Recommendations and extensions . . . 135
7.2.2 Future outlook . . . 137

A Detailed elaborations **139**
A.1 Mean Value Theorem to obtain the Lagrange remainder . . . 139
A.2 Comparison between backward DP, rollout and SDNMPC . . . 140
A.2.1 Backward DP . . . 140
A.2.2 Rollout . . . 141

Bibliography **143**

Acknowledgement **153**

Curriculum Vitae **155**

Summary

Sampling–driven stability domains computation and predictive control of constrained nonlinear systems

The research presented in this thesis considers the computation of stability domains and predictive control for constrained nonlinear systems via a sampling–driven model based approach. This type of systems is particularly relevant for safety critical systems such as the ones in autonomous cars, unmanned aerial vehicles (UAV), smart grids or biomedical systems, which demand reliable stability verification and control tools. Most real–life systems are nonlinear and linear approximations are not able to capture their dynamics. Additionally, in practice, constraints are imposed on states and inputs via actuator limitations, safety regulations and the presence of obstacles. Nonlinear systems exhibit multiple equilibria, stable or unstable, which makes stability analysis and control more challenging. Analysing the stability of these systems is crucial to evaluate the region of the state space from which trajectories converge to specific desirable equilibria and satisfy the constraints. For control design, model predictive control (MPC) enables the generation of safe trajectories which can attain a specific objective for the nonlinear system while ensuring constraint satisfaction.

Most of the stability analysis formulations for nonlinear systems rely on computing a Lyapunov function. An approximation of the stability domain can subsequently be computed as a levelset of the Lyapunov function. The problem of constructing a Lyapunov function for nonlinear systems in general has inherent difficulties. Additionally, verifying its validity on a constrained set typically resorts to computing the global optimum of a nonlinear optimization problem on a compact set. Similarly, computing a control action which satisfies optimality and stability properties according to the predictive control paradigm, requires solving a nonlinear optimization problem. In both cases, the nonlinearity and the constraints mostly generate non–convexity in the optimization program. In turn, non–convexity causes the optimization to stumble in local minima and requirements such as finite termination time and feasibility are affected. A large state space dimension further degrades scalability. Additionally, for constructing domains of stability, a global optimization problem formulation is required, but often impractical. Indeed, in this case, even for a convex optimization problem, the global search for an optimum rejects a candidate Lyapunov function if a single point in the constraint set does not satisfy Lyapunov's inequality.

Therefore, constructive methods, which are able to automatically compute an estimate of the domain of stability of an equilibrium within the constraint set are required.

To circumvent the requirement of solving a global, possibly non–convex optimization problem, in this thesis we adopt a sampling–driven model based approach to the problems of computing stability domains and designing predictive controllers for constrained nonlinear systems. Sampling the state space, or the input space, has been widely adopted by the industrial community, particularly with the rise of supercomputing. Sampling allows for distributed verification, guaranteeing finite termination of algorithms and improves feasibility. Sampled spaces, however, provide merely an abstraction of the state or input space, and new methods have to be developed to guide the sampling process to ensure that formal guarantees for stability verification or control design are still provided for the complete set of states or inputs of interest.

The thesis content is structured in 7 chapters, as detailed next. Chapter 1 provides a domain overview and states the research objective and the research questions addressed by this thesis. This chapter concludes with a summary of contributions and related publications.

Chapter 2 exposes the limitations of existing nonlinear system analysis techniques based on optimization, particularly for invariance and stability analysis. The limitations are linked with: non–convexity, dimensionality, centralized verification which affects scalability and may generate non–feasible verification problems due to the fact that some regions in the search set might not satisfy the property of interest. In the following chapters we show how these limitations can be overcome via sampling–based methods.

First, Chapter 3 considers the verification problem that checks validity of an inequality of the form $F(x) \leq 0 (F(x) < 0)$ for all x in a proper set $\mathcal{S}$ and where $F : \mathbb{R}^n \to \mathbb{R}$ is a piecewise continuous function. The verification is only performed on a finite number of samples in $\mathcal{S}$. Then, the validity of the inequality is extended to an infinite set of initial conditions by exploiting continuity properties. Scalability is ensured by the parallelizability of distributed verification and by multi–resolution sampling.

The inequality $F(x) \leq 0 (F(x) < 0)$ is a unified representation which can describe many properties. Therefore, the framework in Chapter 3 is adapted in Chapter 4 for the verification of Lyapunov's inequality, invariance, computing domains of attraction (DOA) via level set approximation and other properties. Particular emphasis is placed on stability verification, which presents specific challenges for the sampling–based verification around the origin. Additionally, choosing a candidate Lyapunov function is difficult for general nonlinear systems. By using Finite–Step Lyapunov functions (FSLFs) and converse theorems we simplify the construction of a Lyapunov function.

The sampling–based verification methods proposed so far are suitable for nonlinear systems, but for large dimensions of the state space, the corresponding verification problems become computationally demanding. Chapter 5 proposes a randomized approach for verifying the same properties as in Chapter 4, while avoiding the curse of dimensionality, at the cost of a reduction in the validity of the certificate to a probabilistic certificate. The applicability of these randomized techniques is illustrated on power systems.

Non–convex nonlinear constrained optimization problems, with strict timing constraints, also arise in MPC of nonlinear systems (NMPC). Thus, scalable and efficient methods for solving NMPC problems in real–time are highly relevant. Chapter 6 proposes a new sampling–based nonlinear Model Predictive Control (MPC) algorithm, with a bound on

complexity quadratic in the prediction horizon N and linear in the number of samples. The idea of the proposed algorithm is to use the sequence of predicted inputs from the previous time step as a warm start, and to iteratively update this sequence by changing its elements one by one, starting from the last predicted input and ending with the first predicted input. This strategy, which resembles the dynamic programming principle, allows for parallelization up to a certain level and yields a suboptimal nonlinear MPC algorithm with guaranteed recursive feasibility, stability and improved cost function at every iteration, which is suitable for real–time implementation. Conditions for the convergence of the algorithm are also discussed, as well as its applicability on systems inspired from real–life.

Chapter 7 reflects on the methods developed in this thesis, with an overview on open problems and recommendations for future work.

Basic notation and definitions

Sets and set operations

The following standard sets and set operations are considered:

$\mathbb{R}$, $\mathbb{R}_+$, $\mathbb{Z}$, $\mathbb{Z}_+$	The set of real numbers, of nonnegative reals, of integers and of nonnegative integers;
$\Pi_{\geq c}$, $\Pi_{\leq c}$, $\mathbb{R}_\Pi$, $\mathbb{Z}_\Pi$	The sets $\{r \in \Pi \ : \ r \geq c\}$, $\{r \in \Pi : r \leq c\}$, $\mathbb{R} \cap \Pi$, $\mathbb{Z} \cap \Pi$, where $c \in \mathbb{R}$ and $\Pi \subseteq \mathbb{R}$;
$\mathbb{S}^h$	The h–times Cartesian product of $\mathbb{S} \subseteq \mathbb{R}^n$, i.e., $\mathbb{S} \times \ldots \times \mathbb{S}$, $h \in \mathbb{Z}_{\geq 1}$;
$\text{int}(\mathbb{S})$, $\partial\mathbb{S}$, $\overline{\mathbb{S}}$	The interior, boundary and closure of $\mathbb{S}$;
$c\mathbb{S}$	The set $\{cx : x \in \mathbb{S}\}$ for any $c \in \mathbb{R}$;
$\mathbb{S}_1 \oplus \mathbb{S}_2$	The Minkowski addition of $\mathbb{S}_1 \subset \mathbb{R}^n$ and $\mathbb{S}_2 \subset \mathbb{R}^n$, i.e., $\{x + y \ : \ x \in \mathbb{S}_1, \ y \in \mathbb{S}_2\}$;
$\bigoplus_{i=1}^N \mathbb{S}_i$	The Minkowski addition of the sets $\{\mathbb{S}_i\}_{i \in \mathbb{Z}_{[1,N]}}$, where $\mathbb{S}_i \subset \mathbb{R}^n$ for all $i \in \mathbb{Z}_{[1,N]}$;
$2^{\mathbb{X}}$	The set of all subsets of the set $\mathbb{X}$, or the power set of $\mathbb{X}$;

- A *polyhedron* is a set obtained as the intersection of a finite number of half–spaces and a *polytope* is a compact polyhedron;
- A set $\mathcal{S} \subset \mathbb{R}^n$ is called proper if it has non–empty interior, it is compact and $0 \in int(\mathcal{S})$. Given a proper set $\mathcal{S} \subset \mathbb{R}^n$, for any $\mathcal{N}(0) \subset \mathcal{S}$, i.e., a proper neighborhood of 0, the set $\mathcal{A} := \overline{\mathcal{S} \setminus \mathcal{N}(0)} \neq \emptyset$ is called an annulus of $\mathcal{S}$.
- Two sets $\mathcal{S}_1 \subset \mathbb{R}^n$ and $\mathcal{S}_2 \subset \mathbb{R}^n$ are said to have the same cardinality if there exists a bijection from $\mathcal{S}_1$ to $\mathcal{S}_2$. A set $\mathcal{S} \subset \mathbb{R}^n$ is called finite if it is empty or has the same cardinality as the set $\mathbb{Z}_{[1,q]}$ for some $q \in \mathbb{Z}_+$. Otherwise, the set is called infinite. A set $\mathcal{S} \subset \mathbb{R}^n$ is called countably infinite, or countable, if it has the same cardinality as $\mathbb{Z}_+$. A set is called uncountable if it is infinite and not countable.
- We use the notation $j = x_1 : d : x_2$ with $j, x_1, d, x_2 \in \mathbb{R}$ to denote that j takes values of the type $j = x_1 + kd$ with $k \in \mathbb{Z}_{\geq 0}$, in order, from $k = 0$, as long as k satisfies $x_1 + kd \leq x_2$.

Vectors, matrices and norms

The following definitions regarding vectors and matrices are used:

$\lceil x \rceil$	The smallest integer number larger than $x \in \mathbb{R}$;
$\mathbf{1}_n$, $\mathbf{0}_n$, I_n, $0_{n \times m}$	A vector in $\mathbb{R}^n$ with all elements equal to 1, a vector in $\mathbb{R}^n$ with all elements equal to 0, the n–th dimensional identity matrix and a matrix in $\mathbb{R}^{n \times m}$ with all elements equal to 0;
$[x]_i$, $[A]_{i,j}$, $[A]_{:,j}$	The i–th component of $x \in \mathbb{R}^n$, the i,j–th entry of $A \in \mathbb{R}^{n \times m}$ and the j–th column of A, where $(i,j) \in \mathbb{Z}_{[1,n]} \times \mathbb{Z}_{[1,m]}$;
$\|x\|_p$, $\|x\|_\infty$, $\|x\|$	The p–norm, $p \in \mathbb{Z}_{\geq 1}$, i.e., $(\sum_{i=1}^n \|[x]_i\|^p)^{\frac{1}{p}}$, the infinity–norm, i.e. $\max_{i \in \mathbb{Z}_{[1,n]}} \|[x]_i\|$, and an arbitrary norm of the vector x;
$\|\mathbf{x}\|$	The norm of the sequence $\mathbf{x}$ defined as $\sup\{\|x_l\| : l \in \mathbb{Z}_+\}$;

$\|x\|$	The vector of absolute values of the vector $x \in \mathbb{R}^n$, i.e., $\|x\| := [\|x_1\| \dots \|x_n\|]^T$;
$\|A\|_p$	The induced norm of A, i.e., $\max_{\|x\|=1} \frac{\|Ax\|_p}{\|x\|_p}$ for all $p \in \mathbb{Z}_{\geq 1}$;
$Z \succ 0$, $Z \succeq 0$	The symmetric matrix $Z \in \mathbb{R}^{n \times n}$ is positive definite and positive semidefinite;

Basic functions and classes of functions

The following definitions and classes of functions are distinguished:

$\alpha_1 \circ \alpha_2(\cdot)$	The composition of $\alpha_1 : \mathbb{R} \to \mathbb{R}$ with $\alpha_2 : \mathbb{R} \to \mathbb{R}$, i.e., such that $\alpha_1 \circ \alpha_2(r) := \alpha_1(\alpha_2(r))$ for all $r \in \mathbb{R}$;
$\alpha^k(\cdot)$	The k–times composition of α;
id	The identity function, i.e., $id : \mathbb{S} \to \mathbb{S}$ such that for any $x \in \mathbb{S}$, $id(x) = x$;
$f : \mathbb{R}^n \rightrightarrows \mathbb{R}^m$	A set–valued map from $\mathbb{R}^n$ to $\mathbb{R}^m$, i.e., $f(x) \subseteq \mathbb{R}^m$ for all $x \in \mathbb{R}^n$;
$\mathcal{K}$, $\mathcal{K}_\infty$	The class of all functions $\alpha : \mathbb{R}_+ \to \mathbb{R}_+$, that are continuous, strictly increasing and satisfy $\alpha(0) = 0$ and the class of all $\alpha \in \mathcal{K}$ such that $\lim_{r \to \infty} \alpha(r) = \infty$;
$\mathcal{KL}$	The class of all continuous functions $\beta : \mathbb{R}_+ \times \mathbb{Z}_+ \to \mathbb{R}_+$, such that for each fixed $s \in \mathbb{Z}_+$, $\beta(r, s) \in \mathcal{K}$ with respect to r and for each fixed $r \in \mathbb{R}_+$, $\beta(r, s)$ is decreasing with respect to s and $\lim_{s \to \infty} \beta(r, s) = 0$.

- For any two functions $\alpha_1, \alpha_2 : \mathbb{R}_+ \to \mathbb{R}_+$, $\alpha_1 < \alpha_2$ denotes that $\alpha_1(s) < \alpha_2(s)$ for all $s \in \mathbb{R}_+$.
- A map $G : \mathbb{R}^n \to \mathbb{R}^n$ is called $\mathcal{K}$–bounded on $\mathbb{X}$ if there exists a function $\omega \in \mathcal{K}$ such that
$$\|G(x)\| \leq \omega(\|x\|), \quad \forall x \in \mathbb{X}.$$
- Let $\mathcal{S} \subseteq \mathbb{R}^n$. Then, a map $G : \mathcal{S} \to \mathcal{S}$ is called $\mathcal{K}$–continuous in $\mathcal{S}$ if there exists a function $\sigma \in \mathcal{K}$ such that
$$\|G(x) - G(y)\| \leq \sigma(\|x - y\|), \quad \forall (x, y) \in \mathcal{S} \times \mathcal{S}.$$
We call σ the continuity function of the map G. If $\sigma(s) = as$ with $a \in \mathbb{R}_+$, then $\mathcal{K}$–continuity recovers Lipschitz continuity.

List of abbreviations

The following abbreviations are used throughout this thesis:

GAS	globally asymptotically stable
GES	globally exponentially stable
LF	Lyapunov function
FSLF	Finite–step Lyapunov function
FTLF	Finite–time Lyapunov function
DOA	domain of attraction
LMI	linear matrix inequality
BMI	bilinear matrix inequality
MPC	model predictive control
NMPC	nonlinear model predictive control
SDNMPC	sampling–driven nonlinear model predictive control
SBMPC	sampling–based model predictive control
ODE	ordinary differential equation
LP	linear programming
DP	dynamic programming
SDP	semidefinite programming
SOS	sum–of–squares
CPA	continuous piecewise affine

Chapter 1

Introduction

1.1 Background and motivation

Correct (according to specifications) functioning is the most crucial in safety critical systems. These are systems for which failure can have massive impact for human life, or they can cause significant damage of property or the environment (Knight, 2002). Such systems are often encountered in aerospace, automotive, power systems and even more so, in biomedical technology. Developing safety critical systems demands significant advances in both software technology and hardware, and their interplay. Multidisciplinary efforts have to ensure that the complete process, from specification design to architecture, control, and verification is faultless.

In order to design control and verification mechanisms to achieve safety, we need to understand the dynamic behavior of complex systems, which is typically modeled via differential or difference equations. Most real–life systems are nonlinear and linear approximations are not able to capture their dynamics. In what follows we discuss different classes of dynamical systems and typical nonlinearities which describe their behavior.

1.1.1 Nonlinearities in real–life systems

For instance, biological systems, typically recognized as safety critical, include a vast variety of nonlinear dynamics. The most representative are predator–pray type of dynamics, see (Doban, 2016) for an overview. Predator–pray dynamics are found for example in tumor dynamics and they are mathematically represented via polynomial and rational nonlinearities. The ability to represent these systems mathematically has large potential for cancer therapy design. Other dynamics frequently encountered are exponential dynamics in gene–regulatory networks and in the hypothalamic–pituitary–adrenal gland (HPA), which gives insight in hormonal balance. Rational terms are encountered in Hill functions for contraction dynamics formulations for muscle force estimation, of high impact for neuromusculoskeletal modeling and control (Buchanan et al., 2004). Heart related models, which address the crucial problem of treating cardiac disorders such as tachycardia and atrial fibrillation involve both continuous and discrete dynamics and they are represented via hybrid cardiac–cell models (Grosu et al., 2011), see Figure 1.1. Besides nonlinearity, discrete jumps further complicate safety analysis for such models.

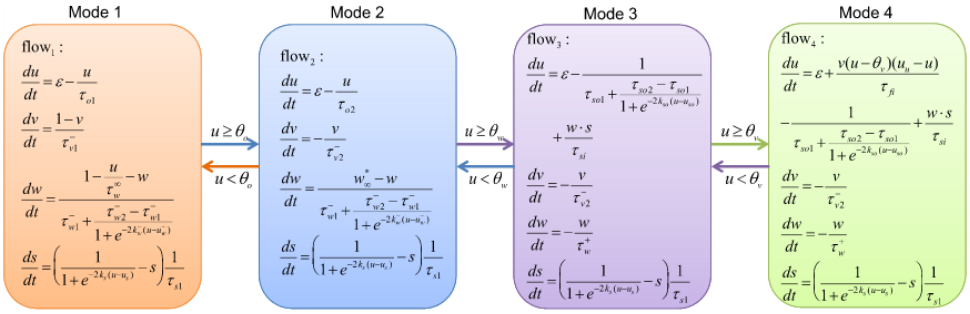


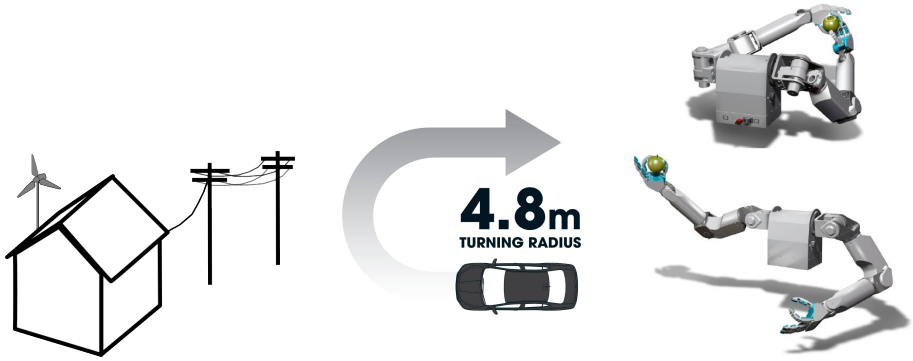
Figure 1.1: Mathematical model for atrial fibrillation (dReal, 2017).

In interconnected power networks it is typical to encounter sinusoidal nonlinearities. This is due to the fact that the power on the connecting power grid lines is directly proportional with the sine of the phase angle difference between the different generating units (Kundur, 1994, pag. 602). Considering the large scale of the power systems models, it is cumbersome to treat the power systems as a nonlinear system. For this reason, when analyzing power systems, a simplifying assumption is made that the phase angle difference is small, in which case the nonlinearity can be eliminated from the model. With the obtained linear model it is more easy to develop safety certificates and design controllers. However, precaution is required when drawing conclusions for the nonlinear system based on the linear approximation, because power systems failure can cause significant damage, at least at an economic level. With the introduction of renewable energy sources which introduce faster fluctuations (Camacho et al., 2011), electric vehicle charging, and deregulation of markets, require more reliable operation with safety guarantees.

We continue the exposition on typical nonlinearities encountered in real–life applications with the electronic power converters. They provide the backbone for many applications nowadays, from power networks to mobile phones, and they are described by nonlinear dynamics including bilinearities (Mohan, 2012). The dynamics are fast and accuracy of the control design is of great importance. While the societal demand for energy and consumption has increased widely, the requirement for smaller, faster, more reliable converters has become more stringent as well.

Trigonometric nonlinearities appear also in mechanical systems which perform rotational movements. Examples vary from pendulum type of objects to robotic manipulators, for which the joints mostly rotate along their axis, see for example (Murray et al., 1994). For such systems, the location of the tip of the robot can be represented through the configuration of the joints, with forward kinematics, and the other way around, via inverse kinematics, which are both nonlinear with trigonometric maps. While these dependencies are static, the dynamics appears when a task has to be effectively executed by the robot. The actuators apply torques at the robot joints, which in turn enable the movement of the robotic arm. The overall mathematical modelling can be performed using, for instance, a Lagrangian analysis of Newton–Euler equations of motion, and it will include the same types of nonlinearities.

In automotive systems, the array of challenges is as diverse as the perspectives with



(a) Power grids (Tremblay, 2017).

(b) Automotive (Mitsubishi, 2017).

(c) Robotic manipulators (Deyle, 2009).

Figure 1.2: Systems with trigonometric nonlinearities.

which we can approach this domain. For instance, autonomous cars have become a matter of high concern in terms of safety. If we abstract a car to the level of its kinematic properties, then we can develop methods to deal with safety when overtaking or performing other special maneuvers. In this case, we encounter the same sinusoidal dynamics as with the robotic systems (Althoff et al., 2007). See Figure 1.2 for a summary on systems which present trigonometric nonlinearities.

Other systems in automotive, which are not safety–critical, are related to control of CO_2 emissions, to comply to environmental regulations. Mathematical models related to powertrain control include square roots in the inlet air mass flow rate, powers of the pressure manifold and rational functions of the air–fuel ratio, see for example (Kolmanovsky and Sun, 2006). At a higher level, platoons of cars can be modeled as a collection of mass–spring–damper systems, which can generate exponential nonlinearities, for instance, when the elasticity constant varies with the position (Raimondo et al., 2009).

When performance demands are increasing, and linear approximations around an operating point are not satisfying for achieving the desired performance, the essential nonlinear dynamics of the systems need to be modeled and taken into account.

1.1.2 Safety analysis and control of constrained systems

Additionally to nonlinearity, in practice, constraints are imposed on states and inputs via actuator limitations, safety regulations, physical limitations and the presence of obstacles. Robotic manipulators, for instance, present kinematic constraints and, additionally, they are designed to operate in cluttered environments, very often in the presence of human operators. Biological systems are constrained by physiological limitations. Powertrain systems aim to maintain the air/fuel ratio within acceptable bounds.

In the case of nonlinear systems, safety analysis and control becomes even more challenging due to multiple equilibria, which can be stable or unstable. Safety analysis is generally approached in control theory by analysing reachability, invariance, or stability of a specific equilibrium. Particularly, investigating the stability of these systems is crucial to

Figure 1.3: X–ray machine in the operating room (Keckler Medical Co., 2017).

evaluate the region of the state space from which trajectories converge to specific desirable equilibria and satisfy the constraints. For instance, in control engineering it is common to tune a simple PID controller which achieves certain specifications. However, it is generally not known if the control system satisfies the constraints, especially away from the operating point for which the PID control has been designed. This is the case for the Powertrain Control System in (Kapinski et al., 2014), which is equipped with a PI controller and it has to maintain the air/fuel (A/F) ratio within 10% of the optimal value.

If the control system does not achieve the constraint specifications, a new control strategy has to be developed to ensure safety and performance of the nonlinear system. Control design enables the generation of safe trajectories which can attain a specific objective for the nonlinear system while ensuring constraint satisfaction. Consider, for example, an interventional X–ray machine (van der Maas, 2016), which guides doctors in performing surgeries on the blood vessels. There are tight requirements for collision avoidance with the medical staff and the objects in the operation room, see Figure 1.3. Moreover, the system must function at the limit of constraints to achieve fast scanning of the human body, for minimal exposure of the patient to the contrast fluid which enables the creation of X–ray images. Safety and guaranteed performance of such systems is very difficult to achieve.

All of the above indicate that the problem of guaranteed safety and control for constrained nonlinear systems is highly relevant. Therefore, in this thesis we aim at designing tools which can formally guarantee the safety and performance satisfaction of existing nonlinear control systems, and design safe controllers for nonlinear systems. In what follows

we will discuss the source of complexity in the verification of safety and the control of constrained nonlinear systems and the typical solution to handle complexity.

1.1.3 Abstractions and sampling

A dynamical system has a state represented by a collection of real numbers, or more generally, by a set of points in an appropriate state space. The set is uncountable, but bounded, due to constraints. Nonlinearity and constraints pose serious limitations, both theoretical and computational, for achieving reliable safety verification and control tools. This is the main source of complexity in these methods, which is due to the fact that such tools require a search for the optimal solution (according to a specific criterion) in a bounded but uncountable, which is generally approached via nonlinear optimization. In turn, many nonlinear optimization problems are either computationally costly, particularly when it is nonconvex, or even unfeasible.

An alternative to providing guarantees directly on an infinite space is to employ an abstraction strategy, to discretize the work space, which is, for instance, the state space for performance analysis, or the input space for control design. An example of such an abstraction is sampling, which is employed in either time, state or input space.

Sampling is based on a "divide and conquer" type of approach. Probably the most commonly known sampling–based method across disciplines is related to FEM modelling, which relies on spatial sampling. To solve a problem, a large problem is divided into smaller, simpler units called finite elements. The equations modeling this finite number of elements are then collected to generate a larger system of equations which models the whole problem.

Looking at complex problems from a computer science perspective, digitization is the main tool which enables computer processing. In control systems for instance, it is common practice to sample in time continuous–time systems to obtain discrete–time systems, which are suitable for numerical computing. In digital signal processing, the Nyquist–Shannon sampling theorem establishes the minimum sampling rate which enables a discrete sequence of samples to capture a continuous–time signal without losing information.

Methods of system abstraction from computer science have been adopted for verification of hybrid systems. For an overview of abstraction techniques for formal verification, the reader is referred to the comprehensive expositions in (Baier et al., 2008) and (Tabuada, 2009). Basically, a state transition system is abstracted to another, ensuring that a bisimulation relation is maintained between the two. In (Soudjani and Abate, 2011), a method is proposed for adaptive gridding to achieve a finite abstraction of a hybrid system. Such an abstraction can be used in formal verification of probabilistic properties by model checkers.

Closer to the control theory domain, a classic example of solving a problem only on a finite number of samples in a set and to provide guarantees for the whole set is encountered in uncertain linear systems, where the uncertainties take values in a convex set $\mathcal{D}$. It is customary to solve robust control problems for uncertain linear systems by solving LMIs only in the vertices of the set $\mathcal{D}$ (de Souza et al., 2000). Guarantees for the whole set $\mathcal{D}$, which is bounded, but contains an infinite number of points, are provided through the convexity of the set $\mathcal{D}$ and the linearity of the system.

Based on the above, we conclude that abstraction techniques, such as sampling, are theoretically sound for addressing complexity in infinite spaces. The successful deployment of sampling–based strategies in solving complex problems in a large variety of domains

suggests that sampling is possibly an enabling technique in safety verification and control design of nonlinear constrained systems as well, with the condition that the validity of the analysis of the sampled space can be extended to the non–sampled initial problem.

1.2 Research objectives

In what follows we introduce a discussion of specific challenges related to safety verification and control for constrained nonlinear systems and possible methods to solve these challenges, with a stronger focus on sampling approaches.

1.2.1 The stability analysis problem for constrained nonlinear systems

Computing solutions of nonlinear dynamical systems solves at least the problem of safety verification. However, other than numerical solutions for a finite number of points in the state space in a specific time interval, it is very difficult to find for nonlinear systems analytical solutions that are valid on an infinite subset of the state space, such as a constraint set. This problem has been recognized a long time ago. The first alternative has been proposed by Lyapunov in 1892 (Lyapunov, 1992). He proposed the so–called Lyapunov functions (LFs), real–valued functions of the state which decrease along the system trajectories. If a system allows such a function, then stability of the equilibrium point can be verified without computing the solution of the nonlinear system.

The stability property is of crucial importance for safety verification of dynamical systems because it characterizes both infinite–time reachability, as well as convergence properties. The largest level set of a LF inside the state constraint set provides a subset of the stability domain, often referred to as Domain of Attraction (DOA) around desired equilibria (Khalil, 2002), (Vidyasagar, 2002).

While choosing a suitable candidate LF is in itself a difficult problem, once such a function is known, depending on the system dynamics, the candidate LF and the set of initial states of interest, one generally needs to solve a nonlinear optimization problem to verify Lyapunov's inequality. For instance, if the candidate LF is $W : \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ and the nonlinear discrete–time system dynamics is defined as $x^+ = G(x)$ with $G : \mathbb{R}^n \to \mathbb{R}^n$ a nonlinear map and $\mathbb{X}$ a state constraint set, then the problem of verifying that W is indeed a LF on $\mathbb{X}$, i.e., $W(G(x)) \leq \rho(W(x))$ for all $x \in \mathbb{X}$, with a given $\rho \in \mathcal{K}$ with $\rho < id$, can be posed by solving the following optimization problem:

$$W^* := \inf_{x \in \mathbb{X}} \{\rho(W(x)) - W(G(x))\}. \tag{1.1}$$

If W^* exists and it is attainable and $W^* \geq 0$, then W is a LF for the given nonlinear system inside the constraint set $\mathbb{X}$.

Most methods for constructing LFs rely on the classical Lyapunov stability theorem (Khalil, 2002), (Vidyasagar, 2002), which requires the existence of a positive definite real–valued function that has a negative definite derivative along the system solutions. The difficulty in constructing a LF for general nonlinear dynamics comes from the uncertainty regarding non–conservative classes of LF candidates and solving the corresponding, possibly non–convex optimization problem. The most successful (by this we mean systematic and generally applicable) approaches so far have been using polynomial candidate LFs and sum–of–squares (SOS) techniques (Chesi, 2011), (Papachristodoulou et al., 2013), (Ma-

jumdar et al., 2014) and continuous piecewise affine (CPA) candidate LFs, simplicial state–space partitions and linear programming, (Björnsson et al., 2015), (Bjornsson et al., 2014), (Hafstein et al., 2014b). The merit of these approaches is that they formulate the problem of verifying Lyapunov's inequality into a convex optimization problem for a fairly general class of nonlinear dynamics. This is however achieved at the price of introducing some conservativeness, e.g., in the SOS formulation different terms of the polynomial expression are concatenated into a higher dimensional vector and also, SOS polynomials are only a subset of the non–negative polynomials, see, e.g., (Aylward et al., 2008). In the CPA formulation conservativeness comes from the fact that the Lyapunov's inequality is made more stringent at simplex vertices.

LFs parameterized via polynomials are used also in (Ratschan and She, 2010). Therein, interval analysis is exploited to compute a polynomial LF and a corresponding DOA, for systems with polynomial vector fields. To these methods we can also add the results based on maximal LFs (Vannelli and Vidyasagar, 1985) that use rational LF candidates, see for example (Doban and Lazar, 2014) and the references therein. This type of LF candidates is less conservative than the polynomial ones, but the computation of maximal rational LFs requires solving a nonlinear non–convex optimization problem to equate coefficients of polynomials.

1.2.2 Predictive control for constrained nonlinear systems

In what concerns control design, a vast literature on nonlinear model predictive control (NMPC) has proven both the theoretical (Grüne and Pannek, 2011), as well as the practical advantage (Magni et al., 2009) of this method in treating optimally the control of multi–variable nonlinear systems subject to constraints on state and inputs. Computing the control law via NMPC requires solving a nonlinear optimization problem to minimize a cost function, which is a function of an input sequence over the control horizon. Additionally, the cost function depends on the initial state and the system dynamics.

Common research interests in predictive control include methods for reducing the complexity of the NMPC algorithms, to make them applicable on devices of low memory (ASIC, FPGA), such as explicit NMPC (ENMPC), see, e.g., (Johansen, 2004). Much work has been done on treating other limiting factors, such as the requirement of NMPC to solve an optimization problem online. This is not well achieved by common optimization tools, which have no specific termination time, especially due to non–convexity which may lead to multiple local–minima. Therefore, real time requirements are not met, which limits the industrial impact of NMPC for systems with fast dynamics. Additionally, if the global optimum is not found, stability is not guaranteed. To alleviate these concerns, the literature has proposed multiple solutions, such as approximate dynamic programming (ADP) (Bertsekas, 2005a), suboptimal MPC (Scokaert et al., 1999), NMPC based on system approximation by neural models (Ławryńczuk, 2009), fast NMPC based on off–line approximations of the control law, as in (Canale et al., 2009), explicit NMPC (ENMPC) (Chakrabarty et al., 2016), the fast NMPC methods discussed in (Alamir, 2006) and (Quirynen et al., 2015), or NMPC based on nonlinear programming (Zavala and Biegler, 2009).

Overall, the NMPC problem has not been solved in its full generality and there is room for improvement in terms of, for example, finite–termination, scalability and stability guarantees.

1.2.3 Common challenges in stability analysis and predictive control design

Both stability analysis and predictive control require solving nonlinear optimization problems. In turn, nonlinearity in the system dynamics, the constraints and obstacles all generate non–convexity in the non–linear optimization problem. For example, for the interventional X–ray machine, or for a car, as discussed earlier in this chapter, the trigonometric functions involved in kinematic dependencies and the obstacles generate non–convexity.

Typical problems related to non–convexity:

- optimization gets stuck in local minima, in which case feasibility or optimality is affected;
- it is not possible to guarantee convergence to a given level of accuracy in a specified time, which is particularly unacceptable for real–time NMPC;
- poor scalability with the state–space dimension.

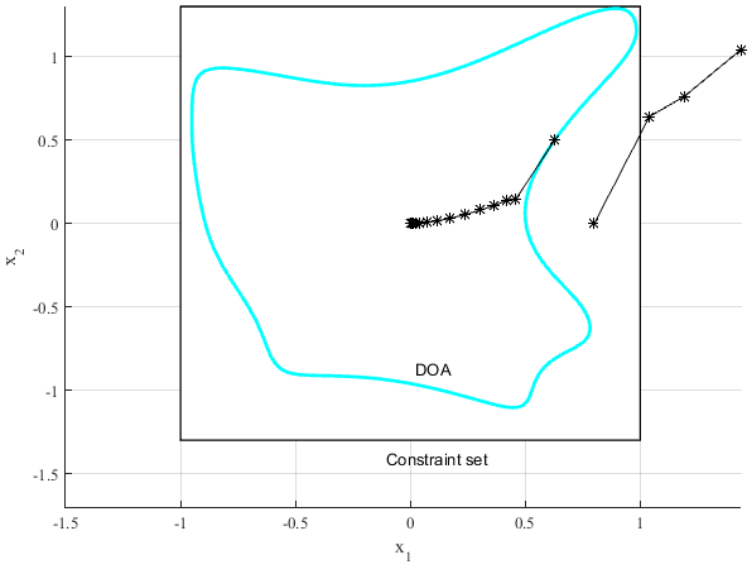


Figure 1.4: Constraint set which contains non–converging states (generating the trajectory in the right) and a DOA approximation (blue).

Additionally, for constructing DOA, a global optimization, even convex, will reject a candidate LF if a single point in the constraint set does not satisfy Lyapunov's inequality. For instance, for a 2D nonlinear system with a candidate LF generating a valid DOA in the constraint set, see Figure 1.4, an optimization program will reject the valid LF. Indeed, notice that the trajectory starting from at least a point in the set does not converge to the

desired equilibrium point 0, i.e., the origin. Therefore, constructive methods, which are able to automatically compute an estimate of the domain of stability of an equilibrium within the constraint set are required.

To address the problems mentioned above related to non–convexity and the lack of constructiveness, in this thesis we adopt a sampling–driven approach to computing DOAs and solving MPC problems for constrained nonlinear systems.

In this thesis, for stability analysis we will use sampling of the set of constraints $\mathbb{X}$. For instance, in (1.1), instead of solving an optimization problem over the complete set $\mathbb{X}$, we select a finite, discrete subset of sample points $\mathbb{X}_s \subset \mathbb{X}$ and we solve the following problem

$$\overline{W} := \min_{x_s \in \mathbb{X}_s} \{\rho(W(x_s)) - W(G(x_s)) - c(x_s)\}, \tag{1.2}$$

where $c : \mathbb{R}^n \rightarrow \mathbb{R}\geq 0$ is a function representing a point–specific conservatism, which will be used to extend the verification over the points $x_s \in \mathbb{X}_s$ to neighborhoods $\mathcal{N}(x_s)$ around these sample points such that the set $\mathbb{X}$ is completely covered by the sets $\mathcal{N}(x_s)$ with $x_s \in \mathbb{X}_s$, i.e.:

$$\mathbb{X} \subseteq \bigcup_{x_s \in \mathbb{X}_s} \mathcal{N}(x_s).$$

The equation (1.2) represents the problem of finding the minimum from among a finite amount of numbers which depend on the sample points $x_s \in \mathbb{X}_s$.

For control, we will sample in the input space $\mathbb{U}$ and similarly to the stability analysis problem, we aim to provide guarantees based solely on a finite number of samples $\mathbb{U}_s \subset \mathbb{U}$.

In what follows we will show how sampling–based methods have already been exploited for stability analysis and MPC for nonlinear systems. Challenges are indicated and the research question of this thesis is formulated.

1.2.4 Sampling methods for stability verification

Sampling–guided approaches have recently been put forward in an attempt to improve upon the classical Lyapunov methods for nonlinear systems. Methods have been developed for simulation–based verification of dynamical systems, see, e.g., (Duggirala et al., 2013) and the references therein. In the light of the discussion on abstractions, simulation–based verification involves abstractions of both the state space, but also in the time space, in an attempt to obtain a set of numerical simulations, or simulation traces of the trajectories of the system over a time period (time sampling), starting from a finite–number of initial states (state sampling). These simulations are used in (Duggirala et al., 2013) to aid in finding the parameters of a candidate LF. In (Fan and Mitra, 2015), discrepancy functions have been used for bounded–time safety verification in a simulation–based framework. Another approach, in (Girard and Pappas, 2006), makes use of bisimulation metrics to infer safety and reachability for metric transition systems by verification on just a finite number of trajectories. Completeness of the result is proven for linear systems.

In terms of deriving LFs from simulation traces, strong contributions have been proposed by (Topcu et al., 2008) and (Kapinski et al., 2014). The work presented in (Topcu et al., 2008) uses information from simulations to obtain local candidate polynomial LFs for dynamical systems with polynomial vector fields. In (Topcu et al., 2008), simulations allow

for converting a set of computationally expensive bilinear matrix inequalities into linear matrix inequalities, which are more tractable. This idea is extended in (Kapinski et al., 2014) by a procedure to improve iteratively the quality of the candidate LF. The procedure relies on a falsification tool in the form of a global optimizer which generates a series of successively improved intermediate LFs. Because of the non–exhaustiveness of the optimization programs, the LF found by the simulation–based iterative technique is validated formally through queries in Satisfiability Modulo Theories (SMT) solvers such as dReal (Gao et al., 2012), z3 (De Moura and Bjørner, 2008), MetiTarski (Akbarpour and Paulson, 2010).

In summary, simulations offer the advantage of gaining exact knowledge about complex systems at low computational expense. However, they provide no formal guarantees for the behavior of the system with infinitely many initial conditions, since it is impossible to simulate all system trajectories. As a result, to validate a local LF or to verify invariance of a compact set most of the existing simulation–guided methods still require solving optimization problems and still suffer from limitations of numerical solvers, scalability issues and conservativity of the chosen LF or bisimulation metric.

An alternative to both classical and simulation–guided Lyapunov methods has been proposed in (Kapinski and Deshmukh, 2013). Therein, a method for checking forward invariance of a given set has been devised based solely on verifying forward invariance of a finite number of points in the set, obtained by δ–sampling (Kapinski and Deshmukh, 2013). The δ–sampling theorem proposed therein holds for Lipschitz–continuous dynamics. The main advantage of this approach is that the need for using an optimization solver is removed. The main effort required is to compute a δ–sampling discretization which can be handled by discrete computational geometry methods, for example lattice based methods, see, e.g., (Lenstra et al., 1982). Once the δ–sampling discretization is generated, then the remaining operations can be parallelized to any degree. However, the assumption of Lipschitz–continuity is a limitation. Also, the same procedure can not be directly applied to verify Lyapunov conditions. Resorting to a Lipschitz–continuous LF is somewhat limiting, and the bound $\gamma > 0$ of the algorithm (see (Kapinski and Deshmukh, 2013)) can not be satisfied in the vicinity of the equilibrium, given that the LF evaluated at the equilibrium is equal to zero. A solution to this issue related to sampling–based stability verification is proposed in this thesis.

To summarize, on finite–time reachability of nonlinear systems there exists a vast literature, mostly based on sampling/simulation. Infinite–time reachability for general nonlinear systems, critical for verifying safety of a system, has been treated less, as well as verifying Lyapunov's inequality. This affects also the applicability of Lyapunov methods to complex systems. Additionally, in most cases, the sampling/simulation based strategy is mainly used to discover a good candidate LF, while for verification of its validity, an optimization problem is still required.

1.2.5 Sampling in predictive control

NMPC could also benefit from sampling to obtain guarantees for finite–time termination of the optimization problem. It is possible to draw samples from either the state or input space, to design computationally feasible NMPC methods. See for example, the randomized approach in (Piovesan and Tanner, 2009), which proposes a randomized approach to sampling the space of predicted input sequences. Input and state space sampling–methods for solving

NMPC via ADP have also been proposed, see (Bertsekas, 2005a), though they inherit the dimensionality issues of DP (Lee and Lee, 2004). Another relevant sampling–based strategy, the so–called sampling–based MPC (SBMPC), has been proposed in (Dunlap et al., 2010). The method therein is applicable to nonlinear systems in general, though, its performance is dependent on a user–specified heuristic.

A common problem of sampling–guided methods for NMPC is the sampling strategy. For example, with each input sample, a tree is expanded. After the tree is built, the path of least cost in the tree is selected from the initial state to the desired state. If the sampling is performed over the input space, and each sample is connected to all the samples in the input space for the next time step in the control horizon, then the tree growth is exponential with the horizon. Alternatively, as in randomized MPC (Piovesan and Tanner, 2009), sampling randomly in the input space, of dimension m, augmented to the horizon of dimension N requires a large number of samples, in an mN dimensional space, to achieve a significant accuracy.

Though many methods for NMPC based on sampling exist, they generally suffer from:

- "Curse of dimensionality";
- Recursive feasibility and stability issues;
- Requirement for user–defined heuristics.

In parallel with the progress of academia on sampling–based verification and control, there is increasing pressure by industry to increase the size of the state space dimension and dynamics complexity and at the same time still provide guarantees for stability and various performance measures. In this regard, see, for example, the HSCC benchmarks related to powertrain systems (Jin et al., 2014), power converters (Nguyen and Johnson, 2014), platoons of vehicles (Makhlouf and Kowalewski, 2014) but also biomedical, such as the anaestesia delivery model in (Gan et al., 2014) and control. See for example also the European Control Conference 2015 challenge from Toyota (Watanabe and Ohata, 2014) for control of an engine based on a complex Simulink model.

1.2.6 Research question

This thesis aims to contribute to stability analysis and fast NMPC for constrained nonlinear systems. As motivated in the previous sections, sampling has potential to tackle nonlinear systems by reducing a complex optimization problem to distributed verification in a finite number of samples. To develop a sampling–driven analysis and design methodology, in this thesis, the following key research objective is investigated:

Can formal guarantees be attained for (complex) nonlinear systems in terms of stability and DOA estimation, and real–time feasibility and stability of NMPC, using a sampling–driven approach?

In the question formulated above we can distinguish two different topics. The first one is an analysis problem, i.e., stability and DOA estimation. The second one is the control problem of designing real–time feasible and stable NMPC. Both topics address dynamical

systems with general nonlinearities, such as the ones mentioned in Section 1.1.1. Therefore, at the heart of both of these questions lie non–convex, possibly intractable or non–feasible (in the case of DOA estimation, see the example in Figure 1.4) optimization problems. Our goal is to circumvent solving such problems by exploring the state and/or input space using a sampling–driven approach. The key challenge is to design the verification criterion for the sampling points such that formal guarantees are obtained for the complete set of initial conditions of interest.

The research question we posed addresses two major topics in *analysis* and *control* of constrained nonlinear systems: stability domains computation and predictive control, with different specific challenges.

For *analysis*, we will answer step–by–step the following sub–questions. Firstly, it is important to set the stability analysis problem with optimization problems, to see what is the extent to which stability analysis can be approached via optimization and to answer the following question.

Q_1 : What are common limitations in solving analysis problems for constrained nonlinear systems?

Then, we can formulate the sampling–based verification problem by answering the generic question:

Q_2 : Can we systematically verify generic properties of the type $F(x) < 0$ for all x in a set $\mathcal{S}$ or find the subset of $\mathcal{S}$ where this property holds based solely on verifying $F(x) < 0$ for samples x in a finite subset of $\mathcal{S}$?

Property functions of the type $F(x) < 0$ are a common representation for many properties. To analyze stability and other relevant safety properties it is necessary to address the question:

Q_3 : How to verify specific properties for nonlinear systems with this sampling–driven framework? Particular emphasis on: verifying stability and computing domains of attraction for discrete–time and continuous–time nonlinear systems, invariance and safety verification.

It is now that we can observe the extent to which the sampling–based verification methodology developed by this thesis addresses the limitations, observed by answering Q_1, in terms of stability analysis for nonlinear systems, of potentially large scale. Thus, we conclude the sampling–driven verification section of this thesis by answering the following questions:

Q_4 : What are the limitations with respect to dimensionality of the sampling–driven analysis proposed so far and how to overcome scalability issues?

Q_5 : Is resorting to randomized sampling and probabilistic certificates of validity for safety critical systems wise?

For *control*, we choose to answer the following sub–questions, which should, at least partly, address the limitations of online optimization–based NMPC via sampling of the input space:

Q_6 : How to sample to achieve suboptimal, feasible in real–time, sampling–driven NMPC (SDNMPC)?

Q_7 : Under what conditions can we guarantee stability for SDNMPC?

Q_8 : Can convergence to the optimal input sequence be guaranteed with SDNMPC?

Q_9 : How to achieve smooth inputs to avoid unsafe actuator operation?

1.3 Outline of the thesis

The above research questions are answered, partially or fully, within this thesis, as follows.

The remainder of this thesis contains six chapters of which the first five address the research questions Q_1–Q_9, as detailed next, and the last chapter summarizes the findings of this thesis and presents recommendations for future research.

Chapter 2 provides an answer to Q_1. Therein, the limitations of existing nonlinear system analysis techniques based on optimization, particularly for invariance and stability analysis, are exposed. For stability analysis, a converse result based on finite–step Lyapunov functions (FSLFs) is used to construct candidate LFs. The approach is systematic for general constrained nonlinear systems and it resorts to solving several nonlinear optimization problems. If the dynamics of the system is Lipschitz–continuous, then this method applies to general nonlinear systems. The limitations of such problems are linked with non–convexity, centralized verification which affects scalability and may generate non–feasible verification problems due to the fact that some regions in the search set might not satisfy the property of interest. The next chapters will show how these limitations can be overcome via sampling–driven methods.

Chapter 3 addresses question Q_2 by considering the verification problem that checks validity of an inequality of the form $F(x) \leq 0 (F(x) < 0)$ for all x in a proper set $\mathcal{S}$ and where $F : \mathbb{R}^n \to \mathbb{R}$ is a piecewise continuous function. The verification is only performed on a finite number of samples in $\mathcal{S}$. Then, the validity of the inequality is extended to an infinite set of initial conditions by exploiting continuity properties. The operations to be performed for each sampling point in the state–space can be carried out in parallel, which improves scalability. The procedure returns a subset $\mathcal{A}$ of points in $\mathcal{S}$ which satisfy the given property, thus possible non–feasibility of the inequality on the complete set $\mathcal{S}$ is avoided. Already addressing Q_3 we also show in this chapter that $F(x) \leq 0 (F(x) < 0)$ is a generic inequality which may represent different analysis problems, of which (finite–step) invariance and safety are straightforward, as we illustrate through an example.

To further answer question Q_3, the framework in Chapter 3 is adapted in Chapter 4 for the verification of Lyapunov's inequality and computing DOAs via level set approximation. Stability verification presents specific challenges for the sampling–driven verification around the origin. Therefore, in this chapter we introduce a result which offers a solution to the problem of the singularity of the LF at the origin. Additionally to the sampling–driven verification, building LFs from FSLFs for discrete–time systems as in Chapter 2 and finite–time LFs (FTLFs) for continuous–time systems further simplifies the construction of a LF. For continuous–time systems we have proposed two different alternatives. A first possibility is to compute a LF for the discretized system and then verify its validity for the original continuous–time system via the sampling–driven result in Chapter 2. Alternatively, we can

construct approximations of a FTLF via polynomial approximations of the continuous–time dynamics. Then again, from the converse theorem, this FTLF can be exploited to compute a LF, which is verified via the sampling–driven result. Chapter 4 answers also Q_4 by the observation that the sampling–based verification methods proposed so far are suitable for nonlinear systems, but for large dimensions of the state space, the corresponding verification problems become computationally demanding.

Chapter 5 addresses Q_5 by proposing a randomized approach for verifying the same properties as in Chapter 4, while avoiding the curse of dimensionality, at the cost of a reduction in the validity of the certificate to a probabilistic certificate. In Chapter 5 we illustrate also an iterative algorithm of maximizing the DOA estimation inside the constraint set, to the limit of the available tools (FSLFs, probabilistic certificates). The applicability of these randomized techniques is illustrated on examples of increasing state space dimension.

For control, Q_6–Q_9 are answered in Chapter 6, which proposes a new SDNMPC algorithm, with a bound on complexity quadratic in the prediction horizon N and linear in the number of samples. Answering Q_6, the idea of the proposed algorithm is to use the sequence of predicted inputs from the previous time step as a warm start, and to iteratively update this sequence by changing its elements one by one, starting from the last predicted input and ending with the first predicted input. This strategy, which resembles the dynamic programming principle and the rollout MPC, allows for parallelization up to a certain level and yields a suboptimal NMPC algorithm with guaranteed recursive feasibility, stability (answering Q_7) and improved cost function at every iteration, which is suitable for real–time implementation. To answer Q_8, conditions for the convergence of the algorithm are also discussed, as well as similarities and distinction from the rollout algorithms and its applicability on systems inspired from real–life. Additionally, to partly answer Q_9 we propose a method for smoothening of the sampled inputs by imposing an additional constraint which limits the fluctuation in time of the input to be applied to the system.

Chapter 7 reflects on the methods developed in this thesis, and formulates recommendations for future work.

1.4 Summary of publications

The chapters of this thesis are based on 8 scientific publications in conference proceedings and journals. In this section we indicate, chapter by chapter, where the results appeared originally. The reader should be aware that the order of the content of these chapters does not match completely that of the original papers, and therefore, the content of a paper might be split among different chapters. Furthermore, the chapters are generally self–contained, except for Chapter 4, which builds upon the results formulated in Chapter 3.

Chapter 2 contains results that were presented in:

- R.V. Bobiti and M. Lazar, On the computation of Lyapunov functions for discrete–time nonlinear systems. In the proceedings of the 18th IEEE International Conference on System Theory, Control and Computing (ICSTCC), Sinaia, Romania, 2014.

Chapter 3 contains results that were presented in:

- R.V. Bobiti and M. Lazar, Automated sampling–based stability verification and DOA estimation for nonlinear systems. Submitted for publication to a journal, Available: http://arxiv.org/abs/1609.00302, 2017.

- R.V. Bobiti and M. Lazar, A delta–sampling verification theorem for discrete–time, possibly discontinuous systems, in the proceedings of the 18th International Conference on Hybrid Systems: Computation and Control (HSCC), Seattle, Washington, USA, 2015.
- R.V. Bobiti and M. Lazar, A Sampling Approach to finding Lyapunov functions for nonlinear discrete–time systems. In the proceedings of the 15th European Control Conference (ECC), Aalborg, Denmark, 2016.

Chapter 4 contains results that were presented in:

- R.V. Bobiti and M. Lazar, A Sampling Approach to finding Lyapunov functions for nonlinear discrete–time systems. In the proceedings of the 15th European Control Conference (ECC), Aalborg, Denmark, 2016.
- R.V. Bobiti and M. Lazar, A delta–sampling verification theorem for discrete–time, possibly discontinuous systems, in the proceedings of the 18th International Conference on Hybrid Systems: Computation and Control (HSCC), Seattle, Washington, USA, 2015.
- R.V. Bobiti and M. Lazar, A sampling approach to constructing Lyapunov functions for nonlinear continuous–time systems. In the proceedings of the 55th IEEE Conference on Decision and Control (CDC), Las Vegas, USA, 2016.
- R.V. Bobiti and M. Lazar, Automated sampling–based stability verification and DOA estimation for nonlinear systems. Submitted for publication to a journal, Available: http://arxiv.org/abs/1609.00302, 2017.

Chapter 5 contains results that were presented in:

- R.V. Bobiti and M. Lazar, A sampling approach to constructing Lyapunov functions for nonlinear continuous–time systems. In the proceedings of the 55th IEEE Conference on Decision and Control (CDC), Las Vegas, USA, 2016.
- R.V. Bobiti and M. Lazar, A randomized approach to constructing domains of attraction for constrained nonlinear systems. In preparation for journal submission, 2017.

Chapter 6 contains results that were presented in:

- R.V. Bobiti and M. Lazar, Towards Parallelizable Sampling–based Nonlinear Model Predictive Control. In the proceedings of the 20th World Congress of the International Federation of Automatic Control, Toulouse, France, 2017.
- R.V. Bobiti and M. Lazar, Sampling–driven Nonlinear Model Predictive Control with convergence guarantees and smooth inputs. In preparation for journal submission, 2017.

Where appropriate, a reference to one or more of these articles has been included in this thesis for further reading.

Chapter 2

Optimization based stability analysis results for discrete–time systems

In this chapter we compute stability domains for discrete–time constrained nonlinear systems. Due to constraints, it is necessary to build our analysis tools for states taking values in bounded sets, which can be simply represented through compact sets. More specifically, we consider an optimization–based result for constructing and verifying validity of a Lyapunov function for constrained nonlinear discrete–time systems. The proposed solution is systematic and consists of two steps. First, a non–monotone Lyapunov function, called finite–step Lyapunov function, is computed by solving a finite dimensional nonlinear optimization problem. Then, a converse theorem is employed, which gives an explicit construction of a Lyapunov function from a finite–step Lyapunov function. This procedure produces additionally an invariant set, a subset of the domain of attraction of the origin through a nonlinear optimization program. An additional iterative algorithm illustrates a procedure of computing a domain of attraction within a set where there are states which converge to other equilibria. Examples illustrate the developed procedures and give insight into the problem of the underlying optimization problems complexity.

2.1 Introduction

Stability analysis of nonlinear systems is an inherently difficult problem which is usually addressed by constructing Lyapunov functions (LFs), see, e.g., (Khalil, 2002) and (Vidyasagar, 2002). However, computing a LF for general nonlinear systems is rather difficult, as one has to deal with several hard problems: how to choose a non–conservative LF candidate, how to parameterize a chosen LF and, lastly, how to translate the computation of the LF into a tractable optimization problem.

Several methods for dealing with the problems posed by stability analysis of nonlinear systems and computation of LFs have been developed. Note the approaches to parameterize LFs through linear programming (LP) procedures, see, e.g., (Johansen, 2000), (Julian et al., 1999), (Giesl and Hafstein, 2014), (Hafstein et al., 2014a) and the references therein. Another approach aiming at benefiting from the computational advantages of LP is the simulation based method in (Kapinski et al., 2014), which relies on concrete executions of the system to formulate a candidate LF as the solution to an LP. Similarly, in (Topcu

et al., 2008), a simulation based approach is employed to obtain candidate LFs parameterized as finite–dimensional polynomials. Therein, simulations are processed for converting a set of computationally expensive bilinear matrix inequalities into linear matrix inequalities, which are tractable. Without aiming at an extensive literature overview, we mention also the analytic methods in (Matallana et al., 2010), (Vannelli and Vidyasagar, 1985), (Tan and Packard, 2008), the methods for computing reachable sets in (Mitchell et al., 2005), the methods to compute optimal quadratic LFs in (Michel et al., 1982) and (Panikhom and Sujitjorn, 2012). Of these contributions, the discrete–time case is considered only in (Kapinski et al., 2014) and (Hafstein et al., 2014a).

However, most of the above LF methods are based on parameterizations of LFs with a given structure, which might be conservative. To avoid searching for a special class of LFs, this chapter considers finding a finite–step Lyapunov function (FSLF) instead. The FSLF, see, e.g., (Gielen and Lazar, 2012), (Lazar et al., 2013a), (Bobiti et al., 2013), is a relaxed form of a LF, i.e., a non–monotonous LF, for which the decrease of the LF is required in a finite number of steps rather than at each time step. The advantage of FSLFs is that, under the assumption of exponential stability, any candidate LF can be turned into a FSLF, for some finite step value, see (Bobiti et al., 2013). This removes the need for searching for a special candidate LF. This advantage was already exploited in (Bobiti et al., 2013) and (Lazar et al., 2013a) for developing scalable and non–conservative stability tests for linear and switched linear discrete–time systems. Moreover, recently, in (Geiselhart et al., 2014) an alternative converse theorem has been introduced that gives an explicit construction of a standard LF from a FSLF.

The results of (Geiselhart et al., 2014) consider a global setting, i.e., the properties hold for all states in $\mathbb{R}^n$, and therefore, they require an analytical solution, which depends on the vector field of the particular system. In this thesis, we restrict the search space to a compact set, which allows for a formulation of the converse result in (Geiselhart et al., 2014) as a nonlinear optimization problem. In this chapter, a systematic methodology for constructing LFs for nonlinear discrete–time systems is developed. First, it is shown that the construction of FSLFs can be formulated as a well posed optimization problem for nonlinear systems, under the assumption of Lipschitz continuity of the dynamics. Then, a standard LF is obtained by directly applying the converse theorem in (Geiselhart et al., 2014) for FSLFs on a compact set. As a by–product, the developed procedure also produces an invariant set, which is a subset of the domain of attraction of the origin.

2.2 System class and finite–step Lyapunov functions

Consider the discrete–time autonomous nonlinear system

$$x_{k+1} = G(x_k), \quad k \in \mathbb{Z}_+, \tag{2.1}$$

where $x_k \in \mathbb{X}$ is the state, $\mathbb{X}$ is a compact set with $0 \in \operatorname{int}(\mathbb{X})$, and $G : \mathbb{R}^n \to \mathbb{R}^n$ is a nonlinear function. A point $x^* \in \mathbb{X}$ is an equilibrium point of system (2.1) if $G(x^*) = x^*$. We assume $G(0) = 0$.

For any $i \in \mathbb{Z}_{\geq 1}$, with an abuse of notation[1], let $\overline{G}^i : 2^{\mathbb{R}^n} \to 2^{\mathbb{R}^n}$ be a set valued map

[1]Here, $\overline{G}$ is used to define a set valued map, while in general, we use $\overline{\mathcal{S}}$ to define the closure of the set $\mathcal{S}$. The confusion can be eliminated by noticing that the new notation refers to a map, while the notation $\overline{\mathcal{S}}$ refers to sets.

such that

$$\overline{G}^i(\mathbb{X}) := \{G^i(x) | x \in \mathbb{X}\},$$

for any set $\mathbb{X} \subseteq \mathbb{R}^n$. Recall, from the basic notations and definitions on page 14, that $2^{\mathbb{R}^n}$ denotes the set of subsets of the set $\mathbb{R}^n$. By convention, $\overline{G}^0(\mathbb{X}) := \mathbb{X}$. For a fixed scalar $k \in \mathbb{Z}_{\geq 1}$, a set $\mathbb{V}$ and a map G denote

$$R_k^{\mathbb{V}} := \bigcup_{i=0}^{k-1} \overline{G}^i(\mathbb{V}),$$

namely, the union of all the k–step trajectories originating from $\mathbb{V}$.

Definition 2.2.1 The system (2.1) is called $\mathcal{KL}$–stable if there exists a $\mathcal{KL}$ function $\beta : \mathbb{R}_+ \times \mathbb{R}_+ \to \mathbb{R}_+$ such that $\|x_k\| \leq \beta(\|x_0\|, k)$ for all $(x_0, k) \in \mathbb{X} \times \mathbb{Z}_+$. □

$\mathcal{KL}$–stability implies $\mathcal{K}$–boundedness, with $k = 1$ and $\omega(\|x\|) = \beta(\|x\|, 1) \in \mathcal{K}$ for all $x \in \mathbb{X}$.

Definition 2.2.2 For the discrete–time system (2.1), the domain of attraction (DOA) of the origin in a compact set $\mathbb{X}$ is the set

$$\mathbb{W} = \{x_0 \in \mathbb{X} : \lim_{k \to \infty} G^k(x_0) = 0\}.$$

Notice in Definition 2.2.2 that the concept of DOA requires an analytic solution of system (2.1) to be able to evaluate the limit of the iterated map G^k for $k \to \infty$. Assume that the equilibrium is asymptotically stable in the Lyapunov sense. Then, to avoid the requirement of computing the map iterates G^k for a nonlinear system, it is customary to approximate the DOA using sublevel sets of a LF, as described in the following subsection.

2.2.1 Finite–step Lyapunov functions

To avoid typical difficulties related to conservative LF candidates, in this thesis we use the relaxed notion of a FSLF to construct a true LF. Before this, we introduce the instrumental notions of a contractive set, an M–step invariant set, an M–step contractive set (Lazar et al., 2013a), the concept of a FSLF[2] and DOA as a levelset of a LF.

Definition 2.2.3 Let $M \in \mathbb{Z}_{\geq 1}$. The set $\mathbb{X}$ is called M–step λ–contractive with respect to the map G of system (2.1) if there exists a $\lambda \in \mathbb{R}_{[0,1)}$ such that for any $x \in \mathbb{X}$, it holds that $G^M(x) \in \lambda\mathbb{X}$. □

If $M = 1$ in Definition 2.2.3, then the set $\mathbb{X}$ is called λ–contractive.

Definition 2.2.4 Let $M \in \mathbb{Z}_{\geq 1}$. The set $\mathbb{X}$ is called M–step invariant with respect to the map G of system (2.1), if for all $x \in \mathbb{X}$, it holds that $G^M(x) \in \mathbb{X}$. □

[2]The FSLF is the same as the finite–time Lyapunov function defined in (Lazar et al., 2013a), (Bobiti and Lazar, 2014a). Following (Geiselhart et al., 2014), the concept has been redefined to avoid the confusion with the notion of finite–time stability, see, e.g., (Bhat and Bernstein, 2000).

A similar definition is proposed in (Doban and Lazar, 2016a, Definition 1.7) for d–invariance of sets with respect to continuous–time systems. For $M = 1$, the M–step invariance reduces to the classical notion of positive invariance. Additionally, if the set $\mathbb{X}$ is M–step invariant, then a standard invariant set can be constructed as in the following result.

Lemma 2.2.5 *If $\mathbb{X}$ is M–step invariant, then*

$$R_M^{\mathbb{X}} = R_\infty^{\mathbb{X}} \tag{2.2}$$

is a standard invariant set with respect to the map G, which contains all the trajectories of system (2.1) starting in $\mathbb{X}$. □

Proof: If $\mathbb{X}$ is M–step invariant, then $\overline{G}^M(\mathbb{X}) \subseteq \mathbb{X}$. By the fact that if the sets A and B satisfy $A \subseteq B$, then $\overline{G}(A) \subseteq \overline{G}(B)$, it follows that

$$\overline{G}^{kM}(\mathbb{X}) \subseteq \overline{G}^{(k-1)M}(\mathbb{X}) \subseteq \ldots \subseteq \mathbb{X},$$

for all $k \in \mathbb{Z}_{\geq 1}$. Therefore,

$$\overline{G}^j(\overline{G}^{kM}(\mathbb{X})) \subseteq \overline{G}^j(\mathbb{X}),$$

for all $j \in \mathbb{Z}_{[0,M-1]}$. Since

$$R_\infty^{\mathbb{X}} = \left(\bigcup_{i \in \mathbb{Z}_{\geq M}} \overline{G}^i(\mathbb{X}) \right) \bigcup R_M^{\mathbb{X}},$$

and $\bigcup_{i \in \mathbb{Z}_{\geq M}} \overline{G}^i(\mathbb{X}) \subseteq R_M^{\mathbb{X}}$, then $R_\infty^{\mathbb{X}} = R_M^{\mathbb{X}}$. By the same reasoning, $\overline{G}(R_M^{\mathbb{X}}) \subseteq R_M^{\mathbb{X}}$, and therefore the set $R_M^{\mathbb{X}}$ is an invariant set. ■

We are now ready to introduce the concept of a FSLF via the following result.

Proposition 2.2.6 *Let $\mathbb{X}$ be a compact set. Let $\alpha_1, \alpha_2 \in \mathcal{K}_\infty$. Suppose that the map G corresponding to the dynamics (2.1) is $\mathcal{K}$–bounded on $\mathbb{X}$, $\mathbb{X}$ is M–step invariant with respect to the map G, there exists a function $V : \mathbb{R}^n \to \mathbb{R}_+$ such that*

$$\alpha_1(\|x\|) \leq V(x) \leq \alpha_2(\|x\|), \quad \forall x \in R_M^{\mathbb{X}}, \tag{2.3a}$$

and there exists an $M \in \mathbb{Z}_{\geq 1}$ and a corresponding $\rho \in \mathcal{K}$ with $\rho < id$ such that

$$V(G^M(x)) \leq \rho(V(x)), \quad \forall x \in \mathbb{X}. \tag{2.3b}$$

Then, system (2.1) is $\mathcal{KL}$-stable in $\mathbb{X}$. □

The real valued function V which satisfies the conditions of Proposition 2.2.6 is called a finite–step Lyapunov function (FSLF) on $\mathbb{X}$. For $M = 1$, Proposition 2.2.6 recovers the classic Lyapunov theorem restricted to compact sets, see (Lazar, 2006), and V becomes

a true Lyapunov function (LF) in the set $\mathbb{X}$. Due to Remark **??**, the assumption of $\mathcal{K}$–boundedness of the map G in Proposition 2.2.6 is not restrictive. Proposition 2.2.6 is the equivalent of the global Theorem 7 from (Geiselhart et al., 2014) on a compact set $\mathbb{X}$. In Theorem IV.5 from (Lazar et al., 2013a), the same theorem on compact sets considers $\rho = \rho_0 id$, with $\rho_0 \in \mathbb{R}_{[0,1)}$. The proof of Proposition 2.2.6 is similar to the proof of (Lazar et al., 2013a, Theorem IV.5).

The following result, the equivalent of (Doban, 2016, Theorem 2.1) for discrete–time systems, is instrumental for the procedure of computing an estimation of the DOA of the origin for system (2.1):

Theorem 2.2.7 *Let $V : \mathbb{R}^n \to \mathbb{R}_+$ be a LF for system* (2.1) *in the set $\mathbb{X}$, and let $L^* \in \mathbb{R}_+$ be the largest value such that the level set $V(x) = L^*$ is contained in $\mathbb{X}$. Then, the set*

$$\mathbb{W} = \{x \in \mathbb{X} : V(x) \leq L^*\} \tag{2.4}$$

is an invariant subset of the DOA of the origin of (2.1). □

2.3 Computation of LFs on compact sets

In general the computation of a LF for discrete–time nonlinear systems is a hard problem. The existing approaches rely on certain parameterizations of LFs. If a system does not admit a LF of the given structure, then most of the procedures halt without providing any path forward. To avoid searching for a special, non–conservative class of LFs, we aim to exploit FSLFs instead.

In (Geiselhart et al., 2014, Theorem 20), it was proven that a converse global Lyapunov theorem can be formulated in terms of FSLF. However, for nonlinear systems which present multiple equilibrium points a global LF does not exist. Therefore, in this section we restrict (Geiselhart et al., 2014, Theorem 20) to finding a FSLF which is valid on a compact set.

2.3.1 LF from a FSLF on a compact set

The converse theorem in (Geiselhart et al., 2014) can be reformulated on compact sets as follows:

Theorem 2.3.1 *Let $\mathbb{X}$ be an M–step invariant set with respect to the map G, and let V be a FSLF on $\mathbb{X}$ for system* (2.1) *with $M \in \mathbb{Z}_{\geq 1}$ satisfying* (2.3b). *Let $\mathbb{W} \subseteq \mathbb{X}$ be any invariant set with $0 \in int(\mathbb{W})$. Then the function $W : \mathbb{R}^n \to \mathbb{R}_+$ defined as*

$$W(x) = \sum_{j=0}^{M-1} V(G^j(x)) \tag{2.5}$$

is a LF for system (2.1) *on $\mathbb{W}$.* □

Proof: The proof is similar to the proof of Theorem 20 in (Geiselhart et al., 2014), with $x \in \mathbb{W}$ instead of $x \in \mathbb{R}^n$.

First, the lower bound on W is computed. From (2.3a) it follows that

$$\alpha_1(\|x\|) \leq V(x) \leq W(x), \ \forall x \in R_M^{\mathbb{X}}, \tag{2.6}$$

and therefore, by the fact that $\mathbb{W} \subseteq R_M^{\mathbb{X}}$, if we denote $\tilde{\alpha}_1 = \alpha_1 \in \mathcal{K}_\infty$, then

$$\tilde{\alpha}_1(\|x\|) \leq W(x),\ \forall x \in \mathbb{W}. \tag{2.7}$$

The upper bound on W is computed through a more involved procedure, as follows. According to (2.3a), for all $k \in \mathbb{Z}_{\geq 0}$ we can express

$$\alpha_1(\|G^k(x)\|) \leq V(G^k(x)),\ \forall x \in \mathbb{W}. \tag{2.8}$$

From (2.8) it follows that

$$\begin{aligned} \|G^k(x)\| &\leq \alpha_1^{-1}\left(V(G^k(x))\right) \\ &\leq \max_{j \in \mathbb{Z}_{\geq 0}} \alpha_1^{-1}\left(V(G^j(x))\right),\ \forall x \in \mathbb{W}. \end{aligned} \tag{2.9}$$

Consider $j = pM + i$ for all $j \in \mathbb{Z}_{\geq 0}$, where $i \in \mathbb{Z}_{[0,M-1]}$ and $p \in \mathbb{Z}_{\geq 0}$. Also, if $x \in \mathbb{W}$, then $R_\infty^{\mathbb{W}} \subseteq \mathbb{X}$, and therefore $G^j(x) \in \mathbb{X}$ for all $j \in \mathbb{Z}_{\geq 0}$. Then, the inequalities in (2.3b) and (2.3a) can be iteratively exploited such that

$$\begin{aligned} V(G^{pM+i}(x)) &\leq \rho(V(G^{(p-1)M+i}(x))) \\ &\leq \ldots \leq \rho^p(V(G^i(x))), \\ &\leq \ldots \leq \rho^p \circ \alpha_2(\|G^i(x)\|),\ \forall x \in \mathbb{W}. \end{aligned} \tag{2.10}$$

Therefore, from (2.9) and (2.10) and the fact that $\rho^p < id$ it follows

$$\|G^k(x)\| \leq \max_{i \in \mathbb{Z}_{[0,M-1]}} \alpha_1^{-1} \circ \alpha_2(\|G^i(x)\|),\ \forall x \in \mathbb{W}. \tag{2.11}$$

Because of the existence of a FSLF on $\mathbb{X}$, then system (2.1) is $\mathcal{KL}$–stable in $\mathbb{X}$, and therefore, $\mathcal{K}$–bounded. Thus, there exists a function $\omega \in \mathcal{K}$ such that $\|G(x)\| \leq \omega(\|x\|)$, for all $x \in \mathbb{X}$. From (2.3a) and (2.11) it follows that

$$\begin{aligned} \|G^k(x)\| &\leq \max_{i \in \mathbb{Z}_{[0,M-1]}} \alpha_1^{-1} \circ \alpha_2 \circ \omega^i(\|x\|) \\ &=: \sigma(\|x\|),\ \forall x \in \mathbb{W}, \forall k \in \mathbb{Z}_+, \end{aligned} \tag{2.12}$$

where $\sigma = \max_{i \in \mathbb{Z}_{[0,M-1]}} \alpha_1^{-1} \circ \alpha_2 \circ \omega^i \in \mathcal{K}_\infty$.

Then, by the same steps as in the proof of Theorem 20 in (Geiselhart et al., 2014), it can be shown that

$$W(x) \leq \tilde{\alpha}_2(\|x\|),\ \forall x \in \mathbb{W},$$

with $\tilde{\alpha}_2 = M\alpha_2 \circ \sigma \in \mathcal{K}_\infty$, and

$$W(G(x)) \leq \tilde{\rho}(W(x)),\ \forall x \in \mathbb{W},$$

where $\tilde{\rho} = (id - (id - \rho) \circ \tilde{\alpha}_1 \circ \tilde{\alpha}_2^{-1})$, and $0 \leq \tilde{\rho} < id$. This leads to the conclusion that W is indeed a LF on $\mathbb{W}$ with respect to the map G. ■

Remark 2.3.2 An alternative construction of a LF from a FSLF, using a $\max$ function instead of summation, is given by equation (10) in (Geiselhart et al., 2014). Other definitions of a FSLF are possible, when, for instance, for all $x \in \mathbb{X}$, there exists $j \in \mathbb{Z}_{[1,M]}$ such that $V(G^j(x)) \leq \rho^j(V(x))$. Then, a LF is given by $W(x) = \min_{j=0}^{M-1} \rho^{-j}(V(G^j(x)))$. The value of j depends on x in this case. Notice that any FSLF, as defined in (2.3b), satisfies the condition above. However, for simplicity in the sampling–driven verification algorithms we propose in the subsequent chapters, we will only refer in this thesis to the definition of a FSLF proposed via Proposition 2.2.6 with the LF construction from (2.5).

A systematic approach to compute a LF W on $\mathbb{W}$ with respect to the map G is described in the following subsection.

2.3.2 A systematic approach for computing regional LFs

Observe that Proposition 2.2.6 provides a set of conditions which can be used to verify stability of the origin of system (2.1) on a compact, M–step invariant set $\mathbb{X}$. Afterwards, Theorem 2.3.1 provides a LF on $\mathbb{W} \subseteq \mathbb{X}$.

To this end, the set of conditions for finding a LF W, given a set $\mathbb{X}$, an M and any function V that satisfies (2.3a), can be reduced to the following steps:

i) check that a given proper set $\mathbb{X}$ is M–step invariant;

ii) check that a given V satisfies (2.3b);

iii) compute the LF W as in (2.5) and an invariant set $\mathbb{W}$, such that $0 \in int(\mathbb{W})$.

Let us first approach step i). In what follows, for computational reasons, we focus on sets $\mathbb{X}$ that are compact and convex, with $0 \in \mathrm{int}(\mathbb{X})$.

Then, consider the Minkowski (gauge) function of the set $\mathbb{X}$ at the point $\xi \in \mathbb{R}^n$: $\mathrm{gauge}(\mathbb{X}, \xi) := \inf_\mu\{\mu \in \mathbb{R}_{\geq 0} : \xi \in \mu\mathbb{X}\}$. Fix $M \in \mathbb{Z}_{\geq 1}$, and let the cost function $F_M : \mathbb{X} \to \mathbb{R}$ be defined as:

$$F_M(x) := -\mathrm{gauge}(\mathbb{X}, G^M(x)) + 1, \tag{2.13}$$

and consider the following minimization problem:

$$\inf_{x \in \mathbb{X}} F_M(x). \tag{2.14}$$

The cost function (2.13) is defined by using the Minkowski function due to the fact that the M–step invariant set condition $G^M(x) \in \mathbb{X}$, for any $x \in \mathbb{X}$, can be written equivalently as $\mathrm{gauge}(\mathbb{X}, G^M(x)) \leq 1$, and therefore $-\mathrm{gauge}(\mathbb{X}, G^M(x)) + 1 \geq 0$. This yields the following result.

Proposition 2.3.3 *Suppose the global optimum in* (2.14) *exists and it is attainable, and let* F_M^* *denote the corresponding value function. If it holds that* $F_M^* \geq 0$*, then the set* $\mathbb{X}$ *is* M*–step invariant for system* (2.1). □

Next, step ii) requires a similar optimization program, as indicated in what follows. Fix $M \in \mathbb{Z}_{\geq 1}$ and $\rho = \rho_0 id$ with $\rho_0 \in \mathbb{R}_{[0,1)}$, let $\mathbb{X}$ be a compact set with the origin in its interior, which is M–step invariant with respect to the map G, and let $V : \mathbb{R}^n \to \mathbb{R}_+$ be a function which satisfies (2.3a). Let the cost function $F : \mathbb{X} \to \mathbb{R}$ be defined as:

$$F(x) := \rho_0 V(x) - V(G^M(x)), \tag{2.15}$$

and consider the following minimization problem:

$$\inf_{x \in \mathbb{X}} F(x). \tag{2.16}$$

The following proposition is a direct consequence of Proposition 2.2.6.

Proposition 2.3.4 *Suppose the global optimum in* (2.16) *exists and it is attainable, and let* F^* *denote the corresponding value function. If it holds that* $F^* \geq 0$*, then* V *is a FSLF for system* (2.1) *on the* M*–step invariant set* $\mathbb{X}$*.* □

Step iii), namely, computing a true LF on $\mathbb{X}$ for (2.1), reduces to applying (2.5) to compute W, once M and V are obtained.

It is also of interest to find the domain $\mathbb{W}$ on which W is a valid LF. Observe that any invariant set is a candidate $\mathbb{W}$. However, constructing such a set is difficult as it is analogous to constructing a LF. Nevertheless, since the LF W is known already, the knowledge of W can be further exploited to find a contractive set $\mathbb{W}$, e.g., the largest level set of W which is contained in $\mathbb{X}$, which is also a subset of the DOA of the origin.

In order to obtain the largest sublevel set of W included in $\mathbb{X}$ consider the following optimization problem:

$$\begin{aligned} L^* = & \min_{x,L} L \\ & \text{s.t. } \operatorname{gauge}(\mathbb{X}, x) = 1, \\ & W(x) = L. \end{aligned} \tag{2.17}$$

The solution to the optimization problem (2.17) provides the contractive set

$$\mathbb{W} := \{x \in \mathbb{R}^n : W(x) \leq L^*\} \subseteq \mathbb{X}, \tag{2.18}$$

which is also invariant by definition of a contractive set.

Remark 2.3.5 Let $O_\infty \subseteq \mathbb{X}$ denote the maximal invariant set in $\mathbb{X}$ for system (2.1). Thus, $\mathbb{W}$ of (2.18) satisfies $\mathbb{W} \subseteq O_\infty$, by the maximality of O_∞. An enlargement procedure for the computed set $\mathbb{W}$ is illustrated in Algorithm 2 in Section 2.3.4. □

Remark 2.3.6 For a system (2.1) which is $\mathcal{KL}$–stable in the set $\mathbb{X}$, any set $\mathbb{P} \subseteq \mathbb{X}$ with $0 \in int(\mathbb{P})$ is M–step λ–contractive, and hence also M–step invariant and yields an invariant set $\mathbb{W} := R_\infty^{\mathbb{P}} = R_M^{\mathbb{P}}$, by Lemma 2.2.5. □

Algorithm 1 Computation of a LF W and a contractive set $\mathbb{W}$ for system (2.1).

Input: $\mathbb{X}$ with $0 \in int(\mathbb{X})$, V which satisfies (2.3a), $\rho \in \mathbb{R}_{[0,1)}$, $M = 1$, M_{max};
Output: $W, \mathbb{W}$

1: (2.14) $\rightarrow F_M^*$;
2: (2.16) $\rightarrow F^*$;
3: **if** ($F_M^* < 0$ or $F^* < 0$) and $M < M_{max}$ **then**
4: $M \leftarrow M + 1$;
5: go to 1.
6: **else if** $M \geq M_{max}$ **then**
7: Choose another FSLF V;
8: $M \leftarrow 1$;
9: go to 1.
10: (2.5) $\rightarrow W$;
11: Solve (2.17) and compute (2.18) $\rightarrow \mathbb{W}$;

Algorithm 1 summarises the steps to compute a LF W and a contractive set $\mathbb{W}$ for system (2.1). Problems (2.14) and (2.16) are verified for the same fixed M. If for a given M either (2.14) or (2.16) are not feasible, M is increased until both (2.14) and (2.16) provide a positive global optimum. If M reaches an expected maximum, namely M_{max}, then either $\mathbb{X}$ is not M–step invariant and then $\mathbb{X}$ must be chosen differently, or the convergence rate of system (2.1) to the equilibrium point is too slow, or the system is unstable. However, one can choose another $\mathbb{X}$ and repeat the procedure.

Remark 2.3.7 The computational complexity of Algorithm 1 depends on steps 1, 2 and 11, corresponding to the optimization problems (2.14), (2.16) and (2.17) respectively, repeated iteratively for each value of M. Also, if one chooses an automatic procedure for step 7, then Algorithm 1 is fully automatic and tractable to the extent to which (2.14), (2.16) and (2.17) are tractable. Additional computational details are presented in Section 2.3.3. □

2.3.3 Computational remarks

The steps i)-iii) are based on solving the nonlinear optimization problems specified in (2.14), (2.16) and (2.17). In order to succeed with this three steps approach, the three optimization problems are required to be well–posed, i.e., for each problem, a global optimum should exist and it should be attainable.

In order to cope with the nonlinear optimization problems, the flexibility in choosing any candidate FSLF can be exploited. For example, let $\mathbb{X}$ be a convex polytope with $0 \in int(\mathbb{X})$, which can be written as:

$$\mathbb{X} = \{x \in \mathbb{R}^n : Hx \leq \mathbf{1}_m\}, \tag{2.19}$$

where $H \in \mathbb{R}^{m \times n}$ with $m \geq n + 1$. Then, the constraint $x \in \mathbb{X}$ becomes the linear constraint $Hx \leq \mathbf{1}_m$. Therefore, the optimization problems in i) and ii) have linear constraints.

Considering that $\text{gauge}(\mathbb{X}, x) = \max_{i\in\mathbb{Z}_{[1,m]}}[Hx]_i$ if $rank(H) = n$ (Blanchini, 1999), then the cost function F_M in i) can be redefined as:

$$F_M(x) := 1 - \max_{i\in\mathbb{Z}_{[1,m]}} \left[HG^M(x)\right]_i . \tag{2.20}$$

In what concerns (2.16), consider the FSLF V as the Minkowski function of $\mathbb{X}$. Therefore, the cost function F in ii) is redefined as:

$$F(x) := \rho_0 \max_{i\in\mathbb{Z}_{[1,m]}} [Hx]_i - \max_{i\in\mathbb{Z}_{[1,m]}} \left[HG^M(x)\right]_i . \tag{2.21}$$

The cost functions in i) and ii), as revealed by (2.20) and (2.21) are nonlinear because of the term $G^M(x)$. However, if the cost function is Lipschitz continuous and the constraints are convex, then the global minimum on $\mathbb{X}$ is attainable and it can be computed, through, for example, the Schubert–Mladineo algorithm, described on page 67 in (Astolfi, 2006). Therefore, under the assumption of Lipschitz continuity of the system dynamics (2.1), the optimization problems in i) and ii) are guaranteed to provide a globally optimal solution.

In what concerns iii), (2.17) is an optimization problem with linear cost function and nonlinear equality constraints which is well–posed.

2.3.4 An iterative algorithm to compute a Lyapunov function and a DOA

For the case when the set $\mathbb{X}$ proposed as an input to Algorithm 1 is not M–step invariant for any $M \in \mathbb{Z}_+$, we develop an iterative approach to computing a regional LF and implicitly a subset of the DOA. The approach is summarized in the following algorithm.

Algorithm 2 starts with the initial constraint set $\mathbb{E}$ by selecting $\mathbb{X} = \mathbb{E}$, and it applies iterative refinements to $\mathbb{X}$ until $\mathbb{X}$ becomes an M–step invariant set which admits an M–step LF V. A LF W on the refined set $\mathbb{X}$ is then computed via (2.5). We now consider W as a new FTLF on the initial constraint set $\mathbb{E}$ and "inflate" the level set of the LF W until the level set, of value c^*, intersects the boundary of the constraint set $\mathbb{E}$. This operation corresponds to steps 15–16 in Algorithm 2.

Next, we iterate on the new FTLF candidate until a valid M is found. If no M was found up to an bound M_{max}, then the level set of the function W is reduced according to step 23, to a value between 1 and c^*. Then, we re–initialize M with $M = 1$ and we repeat the steps 16–25 until a valid FSLF is found, on an M–step invariant set refinement $\mathbb{X}$. Then, a LF and a DOA can be computed as in step 27. The process repeats until the change in the volume of $\mathbb{W}$, quantified via the scalar ε, is not significant anymore, or until a predefined bound $\bar{k}$ on the number of algorithm iterations is reached.

This algorithm might prove itself useful under the same conditions of Lipschitz–continuity of the dynamics in (2.1) and convexity of constraints, as it will be later illustrated through an example. However, it is in general not obvious how a refinement of $\mathbb{X}$ can be performed. Additionally, the computation of volumes as in step 28 is not trivial for sets $\mathbb{W}$ computed via FSLFs, because the set $\mathbb{W}$ is generally not convex. Also, there is no guarantee of convergence for Algorithm 2. These shortcomings will be approached in the next chapters of this thesis via a sampling–driven strategy for LF and DOA computation.

Remark 2.3.8 In terms of computational complexity, the same observations hold for Algorithm 2 as for Algorithm 1: the optimization problems (2.14), (2.16) and (2.17) dictate the

Algorithm 2 Iterative computation of a LF W and a contractive set $\mathbb{W}$ for system (2.1).

Input: $\mathbb{E}$ with $0 \in int(\mathbb{E})$, $\rho \in \mathbb{R}_{[0,1)}$, $M = 1$, M_{max}, $\varepsilon > 0$, $\bar{k} \in \mathbb{R}_{\geq 1}$
Output: $W, \mathbb{W}$

1: $\mathbb{X} \leftarrow \mathbb{E}$, $k \leftarrow 0$
2: $V(x) \leftarrow \text{gauge}(\mathbb{X}, x)$
3: (2.14) $\rightarrow F_M^*$;
4: (2.16) $\rightarrow F^*$;
5: **if** ($F_M^* < 0$ or $F^* < 0$) and $M < M_{max}$ **then**
6: $\quad M \leftarrow M + 1$;
7: $\quad$ go to 3.
8: **else if** $M \geq M_{max}$ **then**
9: $\quad$ Refine $\mathbb{X}$ such that $0 \in int(\mathbb{X})$;
10: $\quad M = 1$;
11: $\quad$ go to 2.
12: (2.5) $\rightarrow W$;
13: Solve (2.17) $\rightarrow L^*$ and compute (2.18) $\rightarrow \mathbb{W} \in \mathbb{X}$;
14: $W_{new} \leftarrow \frac{1}{L^*} W(x)$, $\mathbb{W} = \{x \in \mathbb{R}^n : W_{new}(x) \leq 1\}$, $\mathbb{W}_{new} \leftarrow \mathbb{W}$;
15: $c^* = \max_{x,c} c$, s.t. $\{x \in \mathbb{E}, c \geq 1, W_{new}(x) \leq c\}$
16: $V(x) \leftarrow \frac{1}{c^*} W_{new}(x)$; $M \leftarrow 1$, $\mathbb{X} \leftarrow c^* \mathbb{W}$;
17: (2.14) $\rightarrow F_M^*$;
18: (2.16) $\rightarrow F^*$;
19: **if** ($F_M^* < 0$ or $F^* < 0$) and $M < M_{max}$ **then**
20: $\quad M \leftarrow M + 1$;
21: $\quad$ go to 17.
22: **else if** $M \geq M_{max}$ **then**
23: $\quad c^* \leftarrow c^* - \frac{c^* - 1}{2}$;
24: $\quad M \leftarrow 1$;
25: $\quad$ go to 16.
26: (2.5) $\rightarrow W$;
27: Solve (2.17) $\rightarrow L^*$ and compute (2.18) $\rightarrow \mathbb{W} \in \mathbb{X}$;
28: **if** ($|volume(\mathbb{W}) - volume(\mathbb{W}_{new})| < \varepsilon$ or $k = \bar{k}$) **then**
29: $\quad$ Return $W, \mathbb{W}$
30: **else**
31: $\quad k \leftarrow k + 1$;
32: $\quad$ Go to 15.

tractability of these algorithms. Also, Algorithm 2 is automatic to the extent to which step 9 is automatic. □

Remark 2.3.9 Comparing Algorithm 1 and Algorithm 2 to other optimization based method, the LP–simplicial method in (Giesl and Hafstein, 2014), notice the following. Lipschitz–continuity is required also in (Giesl and Hafstein, 2014), to bound the error in the simplices of the state space partitioning. Also, a simplicial partition is difficult to perform beyond 3D. However, once such a partition exists, the problem of finding a LF resumes to solving an LP. The approach presented in Algorithm 1 and Algorithm 2 is more scalable with the state space dimension, because, under Lipschitz–continuity of the system dynamics, even standard nonlinear optimization problems in Matlab can be solved for dimensions larger than four. □

2.4 Examples

2.4.1 Example 1

To illustrate Algorithm 1, consider the nonlinear discrete–time system equation:

$$x^+ := G(x), \tag{2.22}$$

where x^+ stands for the successor of x, $x \in \mathbb{X} \subset \mathbb{R}^3$, $\mathbb{X}$ is a polytopic set defined as in (2.19) and

$$G(x) := \begin{bmatrix} x_1 - 0.5x_2 \\ \sin x_1 \\ x_2 - 0.5x_3 \end{bmatrix}.$$

Observe that the dynamics G is Lipschitz continuous with a Lipschitz constant $a = 3$. Therefore, also $F_M(x)$ in (2.20) is Lipschitz continuous with a Lipschitz constant $L_i = \|H\|_\infty a^M$ and $F(x)$ in (2.21) is Lipschitz continuous with a Lipschitz constant $L_{ii} = \rho_0\|H\|_\infty + \|H\|_\infty a^M$. Given the Lipschitz continuity of the cost functions, according to Section 2.3.C, it follows that the global optimum for (2.14) and (2.16) is achievable and it can be computed with the Schubert–Mladineo algorithm, for example.

Let us apply Algorithm 1, with $\mathbb{X}$ represented by the following set

$$\mathbb{X} = \{x \in \mathbb{R}^n : \|x\|_\infty \leq 2\}.$$

V is constructed as the Minkowski function of $\mathbb{X}$ and $\rho = 0.9id$.

Solving (2.14) yields $F_M^* = 0.2606 > 0$ and solving (2.16) yields $F^* = 0.1606 > 0$ for $M = 6$ with the Schubert–Mladineo algorithm in 1.1297 seconds on a Windows PC with processor Intel Core i7–3770 CPU 3.40 GHz. The function `fmincon` from Matlab provides $M = 6$ with $F_M^* = 0.2606$ and $F^* = 0.1606$ in 2.1102 seconds.

Thus, according to Proposition 2.3.3, $\mathbb{X}$ is an M–step invariant set with respect to the map G, and V is a FSLF for the system (2.22) on $\mathbb{X}$. Finally, applying (2.17), one obtains the LF W and the contractive set $\mathbb{W}$ illustrated in Figure 2.1, with $L^* = 1.5918$. From one vertex of $\mathbb{X}$, an initial condition for a simulation trajectory was chosen. Observe that the trajectory escapes the M–step invariant set $\mathbb{X}$ for a while, and it returns in $\mathbb{X}$ before M

steps. Note that when the trajectory enters the set $\mathbb{W}$, it does not leave $\mathbb{W}$. Moreover, the trajectory is guaranteed to converge to the origin, by the contractivity of $\mathbb{W}$.

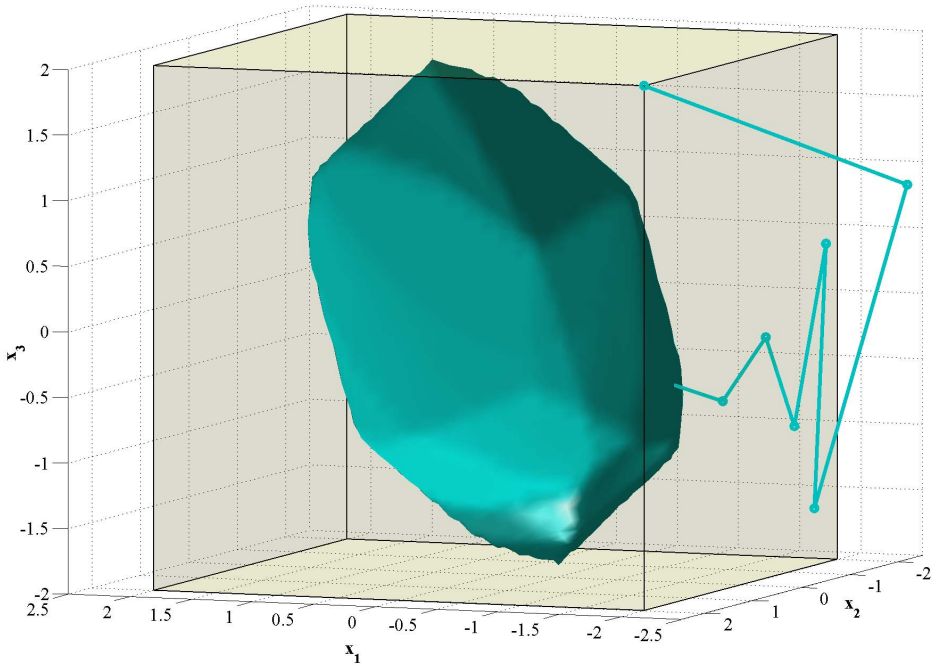


Figure 2.1: Contractive set $\mathbb{W}$ (blue) in $\mathbb{X}$ (yellow).

2.4.2 Example 2

To illustrate Algorithm 2, consider the nonlinear discrete–time system:

$$x^+ := G(x) = \begin{bmatrix} x_1 - 0.5x_2 \\ \sin(x_1) \end{bmatrix}, \tag{2.23}$$

where $x \in \mathbb{E} \subset \mathbb{R}^2$, and $\mathbb{E}$ is a rectangular set, represented with gray in Figure 2.2. The set $\mathbb{E}$ is not M–step invariant with respect to the dynamics (2.23), due to the fact that there exist points in the set $\mathbb{E}$ from which trajectories converge to another equilibrium point, see for example the point $x_0 = [-3.5 \ -0.25]^T$.

By applying two iterations from the Algorithm 2 with $\rho = 0.9$ and $M_{max} = 10$ we obtain the following results. When $\mathbb{X} = \mathbb{E}$ we obtain $F_M^* < 0$ for all $M \leq 10$. Therefore, we refine $\mathbb{X}$ to the value $\frac{1}{2}\mathbb{E}$, illustrated with yellow in Figure 2.2, which is an M–step invariant set with $M = 5$. Also V is a FSLF with $M = 5$. From step 14 we obtain $L^* = 2.0137$ and the set $\mathbb{W}_{new}$ illustrated in red. We expand this set according to step 15 in

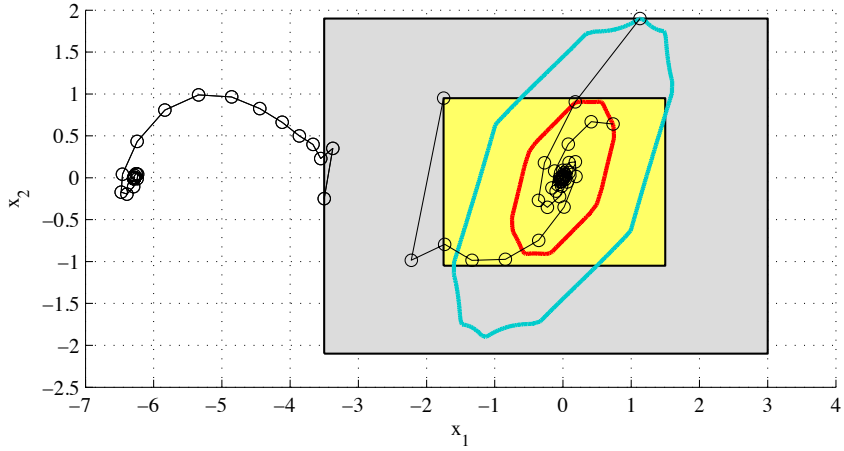


Figure 2.2: Contractive set $\mathbb{W}$ (blue boundary) in $\mathbb{E}$ (gray).

Algorithm 2 and we obtain $c^* = 1.8842$. By applying steps 17–25 we obtain that the new FSLF V computed as in step 16 is a true LF on the new set $\mathbb{W} \subset \mathbb{E}$ illustrated in blue. The set $\mathbb{W}$ is also a contractive set.

2.5 Conclusions

This chapter has developed a systematic approach for computing a LF on a compact set for nonlinear discrete–time systems. The developed solution consists of two steps: first, a FSLF is computed by solving a well posed nonlinear optimization problem. Second, a converse theorem is applied to obtain an explicit LF from a FSLF. The approach can be applied to any Lipschitz continuous nonlinear dynamics with convex constraints. Caution is needed though for non–Lipschitz dynamics.

If the system dynamics (2.1) is not Lipschitz continuous or the constraints are not convex, then the optimization problems in i) and ii) may still be used, though without formal guarantees of achieving a global optimum.

In the remainder of this thesis we undertake a similar approach to computing a subset of the DOA of the origin, due to the advantage that we can construct a LF from a "freely chosen" FSLF. However, to avoid the limitations related to non–Lipschitz dynamics, non–convex optimization problems, feasibility and scalability, we replace the optimization problem with a constructive, sampling–driven approach. For this reason, in the next chapter we develop a general sampling–driven result which will prove useful for verification of several properties, among which also stability and computation of stability domains for constrained nonlinear systems.

Chapter 3

Deterministic sampling–driven verification of inequalities on compact sets

As discussed in Chapter 1, safety verification problems for dynamical systems, such as stability analysis, are generally posed via a nonlinear, possibly non–convex optimization problem. To avoid issues related to non–convexity and non–feasibility of such problems, in this chapter we develop a new deterministic sampling–driven method for verification of generic properties expressed by an inequality of the form $F(x) < 0$ (or $F(x) \leq 0$) with x taking values in a compact set $\mathcal{S}$. The proposed approach firstly distributes the verification of the property on a finite sampling of a bounded set of states of interest. Secondly, it extends the validity of the property to an infinite, bounded set of states by automatically exploiting local continuity properties. Efficient state–space exploration is achieved using multi–resolution sampling and hyper–rectangles as basic sampling blocks. The operations that need to be performed for each sampling point in the state–space can be carried out in parallel, which improves scalability. The procedure returns a subset $\mathcal{A}$ of points in $\mathcal{S}$ which satisfy the given property.

3.1 Introduction

Sampling–based methods for verifying properties of dynamical systems are motivated in real–life applications by model complexity, the curse of dimensionality and feasibility concerns. The main advantage of deterministic or randomized sampling–based methods comes from the possibility to automate and parallelize the verification process, which otherwise would require solving a complex optimization problem. Typically, sampling approaches have been used for finite–time reachability analysis of continuous–time systems, see, e.g., (Dang, 2000), (Girard, 2005), (Althoff et al., 2008b), (Althoff et al., 2008a). See also (Fan and Mitra, 2015), where discrepancy functions were used for bounded–time safety verification in a simulation–based framework. Regarding infinite–time reachability problems, such as safety verification via invariant sets, fewer sampling–based methods exist, as for example (Kapinski and Deshmukh, 2013). A method similar to sampling, namely, cell–mapping (van der Spek, 1994), (Castillo and Zufiria, 2012), uses a partitioning of the state space in cells, to discover complex attractors. However, for formal guarantees it relies on optimization or non–deterministic tools.

In this chapter we address the problem of automatically obtaining formal guarantees for satisfiability of a property on a compact set. This problem is addressed using a generic inequality verification theorem that exploits local continuity to extend validity in a finite number of sampling points to an infinite, bounded set of points. The resulting automated sampling–driven verification method provides certain attractive features. The verification for the points in the finite set of samples can be independently performed and therefore the methodology is spatially decentralized. This feature makes this approach applicable to sets which do not fully satisfy the desired property and would deem a standard, global optimization problem unfeasible, while it also improves scalability.

The sampling–driven verification framework can deal with inequalities of the type $F(x) \leq (<)0$, for all $x \in \mathcal{S}$, where $F : \mathbb{R}^n \to \mathbb{R}$ may be piecewise continuous and $S \subset \mathbb{R}^n$ is a compact set. Efficient state–space sampling is performed via multi–resolution techniques and using hyper–rectangle sampling units. The proposed method can be terminated at any iteration; at every iteration an updated subset of the set of states of interest where the inequality has been verified is returned.

The proposed sampling–driven methodology will be used in this thesis mainly for verification of Lyapunov's inequality (Khalil, 2002) or for computation of domains of attraction. However, in this chapter we show that, due to the generic representation through $F(x)$, other relevant properties can be verified as well. Specifically, we illustrate on a simple example the verification of finite–step invariance of a set $\mathcal{S}$ with guarantees that the trajectories starting from $\mathcal{S}$ will never exceed a safe set $\mathbb{E}$. The verification of Lyapunov's inequality is postponed to the next chapter.

3.2 Sampling with hyper–rectangles

Before formulating the sampling–driven verification algorithm, let us specify the basic sampling element. A hyper–rectangle will be used throughout the thesis to describe the region around a sampling point $x_s \in \mathcal{S}$ to which we extend the verification of a property which is performed only in the sampling point x_s.

Definition 3.2.1 Given $x_s \in \mathbb{R}^n$ and $\delta_{x_s} \in \mathbb{R}^{2n}$, define $\mathcal{B}_{\delta_{x_s}}(x_s)$ as:

$$\mathcal{B}_{\delta_{x_s}}(x_s) := \{\xi \in \mathbb{R}^n : \max_{i \in \mathbb{Z}_{[1,2n]}} [P_{x_s}]_{i:}(\xi - x_s) \leq 1\},$$

where the matrix $P_{x_s} \in \mathbb{R}^{2n \times n}$ is defined as follows:

$$P_{x_s} := diag\left(\left[\begin{array}{c} \frac{1}{\delta_{x_s}(2j-1)} \\ \frac{1}{-\delta_{x_s}(2j)} \end{array} \right]_{j \in \mathbb{Z}_{[1,n]}} \right),$$

which stands for the block diagonal matrix P_{x_s} having vectors of the type

$$\left[\begin{array}{cc} \frac{1}{\delta_{x_s}(2j-1)} & \frac{1}{-\delta_{x_s}(2j)} \end{array} \right]^T, \quad j \in \mathbb{Z}_{[1,n]},$$

on the diagonal. The set $\mathcal{B}_{\delta_{x_s}}(x_s)$ is called a hyper–rectangle. □

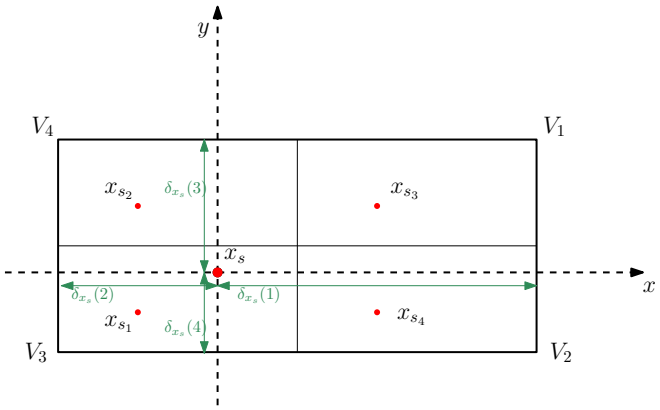


Figure 3.1: 2D hyper–rectangle refinement illustration: the hyper–rectangle $\mathcal{B}_{\delta_{x_s}}(x_s)$ is refined by 4 hyper–rectangles, i.e., $\mathcal{B}_{\frac{\delta_{x_s}}{2}}(x_{s_i})$, for all $i \in \mathbb{Z}_{[1,4]}$.

The hyper–rectangle $\mathcal{B}_{\delta_{x_s}}(x_s)$ in Definition 3.2.1 is defined via a hyper–plane representation. For an alternative definition of a hyper–rectangle, consider $V_j \in \mathbb{R}^n$ with $j \in \mathbb{Z}_{[1,2^n]}$, i.e., the vertices of the hyper–rectangle, and define a vector $\delta_{x_s} \in \mathbb{R}^{2n}$ as follows:

$$\delta_{x_s}(2i-1) := \max_{j \in \mathbb{Z}_{[1,2^n]}} \{V_j(i) - x_s(i)\},$$

$$\delta_{x_s}(2i) := \min_{j \in \mathbb{Z}_{[1,2^n]}} \{V_j(i) - x_s(i)\},$$

for all $i \in \mathbb{Z}_{[1,n]}$. Note that in the definition of $\delta_{x_s}(2i-1)$ and $\delta_{x_s}(2i)$, $V_j(i) - x_s(i)$ are real values, defining distance projections on the axes, represented by the index i. The hyper–rectangle $\mathcal{B}_{\delta_{x_s}}(x_s)$ is illustrated in Figure 3.1 for $n = 2$.

Notice that $\mathcal{B}_{\delta_{x_s}}(x_s)$ is represented through a gauge function inequality. If the hyper–rectangle is a hyper–cube, then δ_{x_s} can be reduced to a scalar, and the sampling unit is represented through a norm inequality:

$$\mathcal{B}_{\delta_{x_s}}(x) := \{\xi \in \mathbb{R}^n : \|\xi - x_s\|_\infty \leq \delta_{x_s}\},$$

i.e., a symmetric gauge function inequality. Let $\mathcal{B}_{\delta_{x_s}} := \mathcal{B}_{\delta_{x_s}}(0)$.

3.3 Verification of inequalities prescribed by real–valued functions

Property functions $F(\cdot)$, as generated by Lyapunov inequalities involving piecewise continuous dynamics or candidate LFs, are piecewise continuous. Therefore, this section considers the verification problem that checks validity of an inequality of the form $F(x) \leq 0 (F(x) < 0)$ for all x in a proper set $\mathcal{S}$ and where $F : \mathbb{R}^n \to \mathbb{R}$ is a piecewise continuous function.

3.3.1 Problem setup

To accommodate for piecewise continuous property functions $F(\cdot)$, as generated by Lyapunov inequalities involving piecewise continuous dynamics or candidate Lyapunov functions, let the sets $\mathcal{S}_i$ with $i \in \mathcal{I} := \{1, \ldots, N\}$ for some $N \in \mathbb{N}$ satisfy $\cup_{i\in\mathcal{I}}\mathcal{S}_i = \mathcal{S}$ and $\mathcal{S}_i \cap \mathcal{S}_j = \emptyset$ for all $i, j \in \mathcal{I}$. The sets $\mathcal{S}_i$ define a partition of the compact set $\mathcal{S}$.

To formally define the sampling–driven verification problem, consider a set $\mathcal{S}_s$ of a finite number of sample points in a compact set $\mathcal{S} \subset \mathbb{R}^n$. Let us define the sets

$$\Delta_{\mathcal{S}}^{(x,\delta)} := \{(x_s, \delta_{x_s}) \in \mathcal{S}_s \times \mathbb{R}^{2n} : \mathcal{S} \subseteq \cup_{x_s\in\mathcal{S}_s}\mathcal{B}_{\delta_{x_s}}(x_s)\}$$

$$\Delta_{\mathcal{S}}^{\delta} := \{\delta_{x_s} : (x_s, \delta_{x_s}) \in \Delta_{\mathcal{S}}^{(x,\delta)}\}.$$

Then, the set $\mathcal{S}_s$ is called a $\Delta_{\mathcal{S}}^{\delta}$–sampling of $\mathcal{S}$. In this case, observe that for all $x \in \mathcal{S}$, there exists at least one pair $(x_s, \delta_{x_s}) \in \Delta_{\mathcal{S}}^{(x,\delta)}$ s.t. $\|x - x_s\| \leq \max|\delta_{x_s}|$. Given a $\Delta_{\mathcal{S}}^{\delta}$–sampling $\mathcal{S}_s$ of $\mathcal{S}$, define

$$\mathcal{S}_s^i := \{x_s : \mathcal{B}_{\delta_{x_s}}(x_s) \cap \mathcal{S}_i \neq \varnothing\},$$

$$\Delta_{\mathcal{S},i}^{\delta} := \{\delta_{x_s} : (x_s, \delta_{x_s}) \in \Delta_{\mathcal{S}}^{(x,\delta)}, x_s \in \mathcal{S}_s^i\},$$

$$I_{x_s} := \{i \in \mathcal{I} : \mathcal{B}_{\delta_{x_s}}(x_s) \cap \mathcal{S}_i \neq \varnothing\},$$

$$\delta_i := \max_{\delta_{x_s}\in\Delta_{\mathcal{S},i}^{\delta}} |\delta_{x_s}|.$$

Recall that $|\delta_{x_s}|$ denotes a vector containing the absolute values of the elements of δ_{x_s}. Let $F_i : \mathcal{S}_i \oplus \mathcal{B}_{\delta_i} \to \mathbb{R}$ be real valued continuous functions for all $i \in \mathcal{I}$. The function $F(x)$ is defined as $F(x) = F_i(x)$ if $x \in \mathcal{S}_i$. Note that $\Delta_{\mathcal{S}}^{\delta} = \cup_{i\in\mathcal{I}}\Delta_{\mathcal{S},i}^{\delta}$.

The following assumption is instrumental in what follows.

Assumption 3.3.1 For all $x_s \in \mathcal{S}_s$, and all corresponding $i \in I_{x_s}$, there exist $a_{x_s}^i, b_{x_s}^i \in \mathbb{R}$ such that:

$$|F_i(x) - F_i(x_s)| \leq a_{x_s}^i\|x - x_s\| + b_{x_s}^i, \quad \forall x \in \mathcal{B}_{\delta_{x_s}}(x_s). \tag{3.1}$$

If we write $a_{x_s} := \max_{i\in I_{x_s}} a_{x_s}^i$ and $b_{x_s} := \max_{i\in I_{x_s}} b_{x_s}^i$, then (3.1) implies

$$|F_i(x) - F_i(x_s)| \leq a_{x_s}\|x - x_s\| + b_{x_s}, \tag{3.2}$$

for all $i \in I_{x_s}$ and $x \in \mathcal{B}_{\delta_{x_s}}(x_s)$.

Assumption 3.3.1 allows that F is discontinuous on $\mathcal{S}$. For example, F could be constructed by switching among the different continuous F_i functions within the partition of the set $\mathcal{S}$, see Example 1. Next, define the set–valued regularization map $\overline{F} : \mathcal{S} \rightrightarrows \mathbb{R}$ as

$$\overline{F}(x) := \bigcap_{\rho>0} \overline{\bigcup_{\epsilon\in\mathcal{B}_\rho} F(x+\epsilon)}.$$

For all $x \in \mathcal{S}$ define the index set:

$$I(x) := \{i \in \mathcal{I} : F_i(x) \in \overline{F}(x)\}.$$

Furthermore, define the real–valued function $\varepsilon : \mathbb{R}^n \to \mathbb{R}_+$,

$$\varepsilon(x) := \max\{|F_i(x) - F_j(x)| \ : \ (i,j) \in I(x) \times I(x)\}.$$

Observe that ε yields the maximum absolute jump that can occur in the function F at a point x, due to discontinuity. If F is continuous at x, then consequently $\varepsilon(x) = 0$. As such, if F is continuous on $\mathcal{S}$, then $\varepsilon(x) = 0$ for all $x \in \mathcal{S}$.

Example 1 To illustrate the notions introduced so far, consider the system (Luk, 2015, Example 4):

$$x^+ := G(x) = \begin{cases} G_1(x) & \text{if } x \in S_1 \\ G_2(x) & \text{if } x \in S_2, \end{cases}$$

where

$$G_1(x) = \begin{bmatrix} 0.5x_1 & -0.8x_2 - x_1^2 \end{bmatrix}^T,$$
$$G_2(x) = \begin{bmatrix} 0.5x_1 + x_1x_2 & -0.8x_2 \end{bmatrix}^T,$$
$$\mathcal{S} = \{x \in \mathbb{R}^2 : \|x\|_\infty \leq 1.5\}$$
$$S_1 := \{x \in \mathbb{R}^2 : x_2 \geq 0\} \cap \mathcal{S}, \ S_2 := \{x \in \mathbb{R}^2 : x_2 < 0\} \cap \mathcal{S}.$$

The system dynamics map G is discontinuous at points of the form $x = (x_1, 0)^T$. Hence, consider the sampling point $x_s = \begin{bmatrix} 1 & 0 \end{bmatrix}^T$. We want to compute $\varepsilon(x_s)$ for the function $F : \mathbb{R}^2 \to \mathbb{R}$ defined by $F(x) = V(G^3(x)) - \rho V(x)$, where $V : \mathbb{R}^2 \to \mathbb{R}_+$ is defined by $V(x) = x^T x$ and $\rho \in \mathbb{R}_{[0,1]}$. Notice that if $F(x) \leq 0$ for a specific $x \in \mathbb{R}^n$, then (2.3b) holds for that x, with $M = 3$. Furthermore, $I(x_s) = \{1, 2\}$.

To compute $F_1(x_s)$ evaluate $G_1(x_s) = \begin{bmatrix} 0.5 & -1 \end{bmatrix}^T$. Since the second element in this vector is less than 0, then $G^2(x_s) = G_2(G_1(x_s)) = \begin{bmatrix} -0.25 & 0.8 \end{bmatrix}^T$, and similarly, $G^3(x_s) = G_1(G_2(G_1(x_s))) = \begin{bmatrix} -0.125 & -0.7025 \end{bmatrix}^T$. Therefore,

$$F_1(x_s) = V(G_1(G_2(G_1(x_s)))) - \rho V(x_s) = 0.5091 - \rho.$$

Similarly, for computing $F_2(x_s)$ we evaluate

$$G_2(x_s) = \begin{bmatrix} 0.5 & 0 \end{bmatrix}^T, G_1(G_2(x_s)) = \begin{bmatrix} 0.25 & -0.25 \end{bmatrix}^T$$

and

$$G^3(x_s) = G_2(G_1(G_2(x_s))) = \begin{bmatrix} 0.0625 & 0.2 \end{bmatrix}^T,$$

which gives $F_2(x_s) = V(G_2(G_1(G_2(x_s)))) - \rho V(x_s) = 0.0439 - \rho$. Therefore, $\varepsilon(x_s) = |F_1(x_s) - F_2(x_s)| = 0.4652$, which illustrates the discontinuity of $F(x)$. □

Remark 3.3.2 If the value of $\varepsilon(\cdot)$ for some sampling point is too large, it is possible to avoid choosing sampling points on the boundaries of regions $\mathcal{S}_i$, which can be covered by sets $\mathcal{B}_{\delta_{x_s}}(x_s)$ for points x_s close to the boundary. □

In this chapter we aim to provide a solution to the following problem. For brevity of notation, the indices of $\Delta_{\mathcal{S}}^{\delta}$ are dropped. The notation carries on in the following results.

Problem 3.3.3 Given a function $F : \mathbb{R}^n \to \mathbb{R}$ which satisfies Assumption 3.3.1 and a proper set $\mathcal{S}$, construct *(i)* a Δ–sampling $\mathcal{S}_s$ of $\mathcal{S}$ and *(ii)* a function $\bar{\gamma} : \mathbb{R}_+ \times \mathcal{S}_s \to \mathbb{R}_+$ such that:

$$F(x_s) \leq (<) - \bar{\gamma}(\max|\delta_{x_s}|, x_s), \quad \forall x_s \in \mathcal{S}_s$$

implies $F(x) \leq (<) 0$ for all $x \in \mathcal{A} \subseteq \mathcal{S}$, where the set $\mathcal{A}$ should be as large as possible. □

3.3.2 Sampling–driven verification: Fundamental theorem

In what follows we develop a multi–resolution decentralized approach towards solving Problem 3.3.3 that encompasses automatic methods for generating the sampling points x_s and the function $\bar{\gamma}(\cdot,\cdot)$, while yielding the set $\mathcal{A}$ as a union of hyper–rectangles $\mathcal{B}_{\delta_{x_s}}(x_s)$ corresponding to all "true" sampling points x_s (i.e., points where the inequality holds true).

The main sampling–driven verification theorem is stated next.

Theorem 3.3.4 *Suppose Assumption 3.3.1 holds. Let $\mathcal{S}_s$ be a Δ–sampling of the set $\mathcal{S}$ and let $F : \mathcal{S} \to \mathbb{R}$ and the associated functions $F_i : \mathcal{S}_i \oplus \mathcal{B}_{\delta_i} \to \mathbb{R}$ be given. Let $\bar{\gamma} : \mathbb{R}_+ \times \mathcal{S}_s \to \mathbb{R}_+$ be defined as $\bar{\gamma}(\xi, x_s) := a_{x_s}\xi + b_{x_s} + \varepsilon(x_s)$. Let $\mathcal{A}_s \subseteq \mathcal{S}_s$ be such that for all $x_s \in \mathcal{A}_s$ it holds that*

$$\begin{aligned} F(x_s) &\leq -\bar{\gamma}(\max|\delta_{x_s}|, x_s) \\ \textit{(resp. } F(x_s) &< -\bar{\gamma}(\max|\delta_{x_s}|, x_s) \textit{).} \end{aligned} \tag{3.3}$$

Then $F(x) \leq 0$ (resp. $F(x) < 0$) holds for all $x \in \mathcal{A} := \cup_{x_s \in \mathcal{A}_s} \mathcal{B}_{\delta_{x_s}}(x_s) \subseteq \mathcal{S}$. □

Proof: By hypothesis $F(x_s) \leq -\bar{\gamma}(\max|\delta_{x_s}|, x_s)$ (resp. $F(x_s) < -\bar{\gamma}(\max|\delta_{x_s}|, x_s)$) holds for all $x_s \in \mathcal{A}_s$. Assume that there exists a point $x \in \mathcal{A}$ such that $F(x) > 0$ (resp. $F(x) \geq 0$). Take any point $x_s \in \mathcal{A}_s \subseteq \mathcal{S}_s$ such that $\|x - x_s\| \leq \max|\delta_{x_s}|$. Observe that such a point always exists, by the definition of a Δ–sampling of a set. By Assumption 3.3.1 it follows that (3.2) holds for all $i \in I_{x_s}$.

Furthermore, let $i \in \mathcal{I}$ be such that $F(x) = F_i(x)$ and let $j \in \mathcal{I}$ be such that $F(x_s) = F_j(x_s)$.

Then, by the triangle inequality and (3.2) it follows that

$$\begin{aligned} |F(x) - F(x_s)| &= |F_i(x) - F_j(x_s)| \\ &= |F_i(x) - F_i(x_s) + F_i(x_s) - F_j(x_s)| \\ &\leq |F_i(x) - F_i(x_s)| + |F_i(x_s) - F_j(x_s)| \\ &\leq a_{x_s}\|x - x_s\| + b_{x_s} + \varepsilon(x_s) \\ &= \bar{\gamma}(\|x - x_s\|, x_s) \\ &\leq a_{x_s}\max|\delta_{x_s}| + b_{x_s} + \varepsilon(x_s) \\ &= \bar{\gamma}(\max|\delta_{x_s}|, x_s). \end{aligned} \tag{3.4}$$

Since $F(x) > 0$ (resp. $F(x) \geq 0$) for some $x \in \mathcal{A}$, then

$$-F(x) < 0 \text{ (resp. } -F(x) \leq 0\text{).} \tag{3.5}$$

Also, for any $x_s \in \mathcal{A}_s$ such that $\|x - x_s\| \leq \max|\delta_{x_s}|$ we have

$$\begin{aligned} F(x_s) &\leq -\bar{\gamma}(\max|\delta_{x_s}|, x_s) \\ \text{(resp. } F(x_s) &< -\bar{\gamma}(\max|\delta_{x_s}|, x_s)\text{).} \end{aligned} \tag{3.6}$$

By summing up (3.5) and (3.6) we obtain:

$$F(x_s) - F(x) < -\bar{\gamma}(\max|\delta_{x_s}|, x_s) < 0, \tag{3.7}$$

and therefore

$$|F(x) - F(x_s)| > \bar{\gamma}(\max|\delta_{x_s}|, x_s). \tag{3.8}$$

By inspecting (3.4) and (3.8) we observe that a contradiction was reached. Hence, the statement is proven. ■

Similarly with verifying $F(x) \leq 0$ (resp. $F(x) < 0$) we can also verify $F(x) \geq 0$ (resp. $F(x) > 0$), as in the following corollary. The proof is similar to the proof of Theorem 3.3.4.

Corollary 3.3.5 *Suppose Assumption 3.3.1 holds. Let $\mathcal{S}_s$ be a Δ–sampling of the set $\mathcal{S}$ and let $F : \mathcal{S} \to \mathbb{R}$ and the associated functions $F_i : \mathcal{S}_i \oplus \mathcal{B}_{\delta_i} \to \mathbb{R}$ be given. Let $\bar{\gamma} : \mathbb{R}_+ \times \mathcal{S}_s \to \mathbb{R}_+$ be defined as $\bar{\gamma}(\xi, x_s) := a_{x_s}\xi + b_{x_s} + \varepsilon(x_s)$. Let $\mathcal{A}_s \subseteq \mathcal{S}_s$ be such that for all $x_s \in \mathcal{A}_s$ it holds that*

$$\begin{aligned} F(x_s) &\geq \bar{\gamma}(\max|\delta_{x_s}|, x_s) \\ \textit{(resp. } F(x_s) &> \bar{\gamma}(\max|\delta_{x_s}|, x_s)\textit{).} \end{aligned} \tag{3.9}$$

Then $F(x) \geq 0$ (resp. $F(x) > 0$) holds for all $x \in \mathcal{A} := \cup_{x_s \in \mathcal{A}_s} \mathcal{B}_{\delta_{x_s}}(x_s) \subseteq \mathcal{S}$. □

Remark 3.3.6 The result of Theorem 3.3.4 enables decentralized verification by allowing $\bar{\gamma}$ to have different coefficients a_{x_s} and b_{x_s} for each sampling point x_s and by not using any central variable, i.e., any common variable for all $x_s \in S_s$. □

Remark 3.3.7 In the case that F is continuous on S, then $\varepsilon(x_s) = 0$ for all $x_s \in \mathcal{S}_s$, and the result in (Bobiti and Lazar, 2016, Theorem III.3) is recovered with $\bar{\gamma}(\max|\delta_{x_s}|, x_s) = a_{x_s}\max|\delta_{x_s}| + b_{x_s}$, for the particular case when the sets $\mathcal{B}_{\delta_{x_s}}(x_s)$ are hyper–cubes. □

In what follows we will develop a prototype algorithm for verifying the conditions (3.3) of Theorem 3.3.4. To this end, two ingredients are needed: *(i)* a method for obtaining the sampling $\mathcal{S}_s$ of $\mathcal{S}$, which is proposed in Section 3.3.3, and *(ii)* a procedure for computing the function $\bar{\gamma}$, which is presented in Section 3.3.4.

3.3.3 Multi–resolution sampling tools

The process of constructing the set $\mathcal{A}$ in Theorem 3.3.4 is based on multi–resolution sampling of the set $\mathcal{S}$. To define the concept of multi–resolution, let us define the N–refinement of a hyper–rectangle $\mathcal{B}_{\delta_{x_s}}(x_s)$ as follows.

Definition 3.3.8 Let $x_s \in \mathbb{R}^n$, $\delta_{x_s} \in \mathbb{R}^{2n}$ be arbitrarily chosen. An N–refinement of $\mathcal{B}_{\delta_{x_s}}(x_s)$, with $N \in \mathbb{Z}_{>1}$ is a finite set $\mathcal{S}_{sx_s} \subset \mathcal{B}_{\delta_{x_s}}(x_s)$ s.t. for all $x \in \mathcal{B}_{\delta_{x_s}}(x_s)$, there exists at least one vector $x' \in \mathcal{S}_{sx_s}$ s.t. $\max_{i \in \mathbb{Z}_{[1,2n]}} [P_{x_s}]_{i:}(x - x') \leq \frac{1}{N}$. □

Notice that $\mathcal{B}_{\delta_{x_s}}(x_s) \subseteq \cup_{x' \in \mathcal{S}_{sx_s}} \mathcal{B}_{\frac{\delta_{x_s}}{N}}(x')$ and that Definition 3.3.8 does not require $\mathcal{S}_{sx_s}$ to be minimal. A 2–refinement will be used throughout this thesis, because it enables sampling of $\mathcal{B}_{\delta_{x_s}}(x_s)$ with a minimum number of sampling points x', without overlay. The 2–refinement will be simply referred to as a *refinement*. The refinement allows for multi–resolution sampling of the state–space. See Figure 3.2 for an exemplification of the concept of set sampling and set refinement. In Figure 3.2, the green set is $\mathcal{S}$, the red points are $\mathcal{S}_s$. Therein, we apply a 2–refinement to the set $\mathcal{B}_{\delta_{x_s}}(x_s)$ where $x_s \in \mathcal{S}_s$ and $\delta_{x_s} \in \mathbb{R}^4$. The values $\delta_{x_s}(i)$ with $i \in \mathbb{Z}_{[1,4]}$ represent the elements of the vector δ_{x_s}, and x_{s_i} with $i \in \mathbb{Z}_{[1,4]}$ represent the 2–refinement of the set $\mathcal{B}_{\delta_{x_s}}(x_s)$.

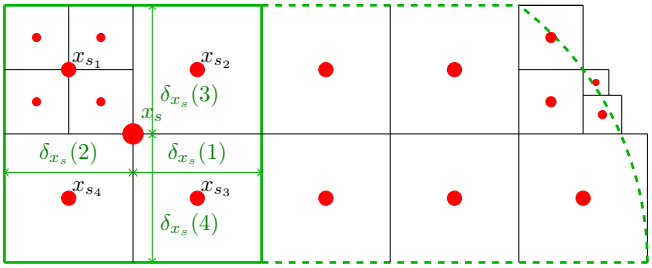


Figure 3.2: Sampling of $\mathcal{S}$ (green) with $\mathcal{S}_s$ (red) and a 2–refinement for the set $\mathcal{B}_{\delta_{x_s}}(x_s)$.

In (Bobiti and Lazar, 2015) and (Bobiti and Lazar, 2016), a hyper–cube was proposed as a sampling unit. This is useful when the dimension of the search set $\mathcal{S}$ is similar on different axes. However, system state constraints generally present different bounds on different axes. In this case, a non–uniform sampling based on hyper–rectangles could reduce significantly the number of sampling points.

If we want to sample a hyper–rectangle $\mathcal{P}$ in n dimensions and $V_1, \ldots, V_{2^n}$ are the vertices of $\mathcal{P}$, then by refinement of $\mathcal{P}$ on $n_r < n$ dimensions, 2^{n_r} sampling points are obtained, and they are

$$\left\{ \frac{x_s + V_1}{2}, \ldots, \frac{x_s + V_{2^{n_r}}}{2} \right\},$$

with their corresponding intervals. A similar multi–resolution strategy based on hyper–rectangles was used in (Ratschan and She, 2010). Therein, the problem of estimating DOA for continuous–time polynomial systems was tackled using polynomial LFs and interval analysis, with quantified constraint solving instead of samples.

The sampling strategy presented so far will be illustrated through examples in the next chapter. Notice that the sampling–driven verification result of Theorem 3.3.4 is independent of the sampling strategy.

3.3.4 Construction of $\bar{\gamma}$

This subsection provides a constructive method to compute the coefficients used to define the function $\bar{\gamma}$. For each $x_s \in \mathcal{S}_s$, given the index set I_{x_s}, one needs to compute $a^i_{x_s}$ and $b^i_{x_s}$ for all $i \in I_{x_s}$, such that (3.1) holds. In turn, this will yield that (3.3) holds with $\bar{\gamma}(\max|\delta_{x_s}|, x_s) := a_{x_s} \max|\delta_{x_s}| + b_{x_s} + \varepsilon(x_s)$, where $a_{x_s} := \max_{i\in I_{x_s}} a^i_{x_s}$ and $b_{x_s} := \max_{i\in I_{x_s}} b^i_{x_s}$.

To this end, the following assumption is instrumental.

Assumption 3.3.9 F_i is at least two times differentiable on $\mathcal{B}_{\delta_{x_s}}(x_s)$ for each $x_s \in \mathcal{S}^i_s$. □

Notice that $\mathcal{B}_{\delta_{x_s}}(x_s)$ is a convex set for all $x_s \in \mathcal{S}_s$. Assumption 3.3.9 allows for Taylor series expansion and application of the Mean Value theorem in the following manner. Denote:

$$T(x, x_s, m) := \sum_{v=0}^{m} \frac{([(x - x_s)\nabla]^v F_i)(x_s)}{v!},$$

for all $x \in \mathcal{B}_{\delta_{x_s}}(x_s)$, where, e.g., ∇F_i stands for the Jacobian and $\nabla^2 F_i$ is the Hessian of the real valued function F_i. Notice that if x (and consequently x_s) is univariate, i.e., if $x \in \mathbb{R}$, then $T(x, x_s, m)$ can be written as $\sum_{v=0}^{m} \frac{F_i^{(v)}(x_s)}{v!}(x - x_s)^v$.

Also,

$$F_i(x) = T(x, x, p) = T(x, x_s, \infty), \quad \forall p \geq 0 \tag{3.10}$$

is the Taylor series expansion of F_i around the sampling point x_s, in the set $\mathcal{B}_{\delta_{x_s}}(x_s) \subseteq \mathcal{S} \oplus \mathcal{B}_{\delta_{x_s}}$. Then (3.10) can be rewritten

$$\begin{aligned} F_i(x) &= T(x, x_s, \infty) \\ &= T(x, x_s, \infty) + T(x, x_s, m) - T(x, x_s, m) \\ &= \underbrace{T(x, x_s, m)}_{\text{m–th order Taylor expansion}} + \underbrace{T(x, x_s, \infty) - T(x, x_s, m)}_{\text{remainder}}, \end{aligned} \tag{3.11}$$

which is the m–th order Taylor series expansion of F_i, with remainder. We refer the reader to Appendix A.1 for further processing of (3.11) to an equality which replaces the infinite number of terms in the remainder with a Lagrange remainder.

The infinite Taylor series can be over–approximated by a first order Taylor expansion and the Lagrange remainder:

$$F_i(x) = F_i(x_s) + \nabla F_i(x_s)(x - x_s) + L_i(x, x_s, \xi) \tag{3.12}$$

where

$$L_i(x, x_s, \xi) := \frac{1}{2}(x - x_s)^T \nabla^2 F_i(x_s + \xi(x - x_s))(x - x_s)$$

and $\xi \in (0,1)$. For any $x_s \in \mathcal{S}_s$ there exists $b_{x_s} \in \mathbb{R}_+$ such that

$$|L_i(x, x_s, \xi)| \leq b_{x_s}, \quad \forall x \in \mathcal{B}_{\delta_{x_s}}(x_s), \forall \xi \in (0,1). \tag{3.13}$$

It is possible to compute such bounds for a convex set $\mathcal{B}_{\delta_{x_s}}(x_s)$, as follows:

Proposition 3.3.10 (Althoff et al., 2008a) *The bounds on the absolute values of the Lagrange remainder in* (3.13) *for an $x_s \in \mathcal{S}_s$, can be computed as follows:*

$$b^i_{x_s} = \frac{1}{2}\tau^T_{x_s} \max_{x \in \mathcal{B}_{\delta_{x_s}}(x_s), \xi \in (0,1)} (|\nabla^2 F_i(x_s + \xi(x - x_s))|)\tau_{x_s}, \tag{3.14}$$

where $\tau_{x_s} \in \mathbb{R}^n$ and $\tau_{x_s}(i) = \max\{|\delta_{x_s}(2i-1)|, |\delta_{x_s}(2i)|\}$. □

The proof is similar to the proof in (Althoff et al., 2008a), but the zonotopes therein reduce here to hyper–rectangles.

The term $\max_{x \in \mathcal{B}_{\delta_{x_s}}(x_s), \xi \in (0,1)} (|\nabla^2 F_i(x_s + \xi(x - x_s))|)$ in (3.14) can be computed via interval analysis, see (Jaulin et al., 2001) or (Moore et al., 2009). In Matlab, efficient interval analysis can be performed via INTLAB (Rump, 1999).

By (3.12), (3.13) and the triangle inequality we see that

$$\begin{aligned} |F_i(x) - F_i(x_s)| =& |F_i(x_s) + \nabla F_i(x_s)(x - x_s) + L_i(x, x_s, \xi) - F_i(x_s)| \\ \leq& |\nabla F_i(x_s)(x - x_s)| + |L_i(x, x_s, \xi)| \\ \leq& \|\nabla F_i(x_s)\| \|x - x_s\| + b^i_{x_s}, \end{aligned} \tag{3.15}$$

for all $x \in \mathcal{B}_{\delta_{x_s}}(x_s)$. Denoting $a^i_{x_s} := \|\nabla F_i(x_s)\|$, (3.15) becomes:

$$|F_i(x) - F_i(x_s)| \leq a^i_{x_s}\|x - x_s\| + b^i_{x_s}, \quad \forall x \in \mathcal{B}_{\delta_{x_s}}(x_s). \tag{3.16}$$

Therefore, Assumption 3.3.9 implies (3.16), which is identical to (3.1), which means that Assumption 3.3.9 implies Assumption 3.3.1. This means that Assumption 3.3.9 is sufficient to guarantee the satisfaction of Assumption 3.3.1 for Theorem 3.3.4.

To decrease the conservatism of inequality (3.16), we can reduce the size of the bound b_{x_s} as follows. For example, we can modify (3.15) in the following manner:

$$\begin{aligned} |F_i(x) - F_i(x_s)| =& |\nabla F_i(x_s)(x - x_s) + L_i(x, x_s, \xi)| \\ \leq& a^i_{x_s}\|x - x_s\| + b^i_{x_s}, \end{aligned} \tag{3.17}$$

where $b^i_{x_s} = 0$ and $a^i_{x_s} := \|\nabla F_i(x_s) + \frac{1}{2}(x - x_s)^T \nabla^2 F_i(x_s + \xi(x - x_s))\|$ can be computed via interval analysis. In this way, the triangle inequality is not used in (3.15) and the bound may become less conservative. With this approach, (3.17) may replace (3.16).

We are now ready to state a prototype algorithm for verifying the conditions (3.3).

3.3.5 Prototype sampling–driven verification algorithm

Algorithm 3 reports all the operations necessary for verifying that $F(x) \leq (<)0$ on $\mathcal{A} \subset \mathcal{S}$, via Theorem 3.3.4. As detailed therein, the verification starts from the complete set $\mathcal{S}$ and gradually the set $\mathcal{A}$ is constructed by the subsets $\mathcal{B}_{\delta_{x_s}}(x_s)$ which do satisfy $F(x_s) \leq (<)-\bar{\gamma}(\max|\delta_{x_s}|, x_s)$. Note that Algorithm 3 illustrates a multi–resolution sampling approach to Theorem 3.3.4.

Proposition 3.3.11 *(Convergence of Algorithm 3) If $F(x) \leq (<)0$ for all $x \in \mathcal{S}$, and if F is continuous and two times differentiable on $\mathcal{S}$, then Algorithm 3 generates a set $\mathcal{A}(\delta_{min}) \to \mathcal{S}$ if $\delta_{min} \to 0$.* □

Proof: If F is continuous on $\mathcal{S}$, then $\varepsilon(x) = 0$ for all $x \in \mathcal{S}$. Furthermore, if F is two times differentiable on $\mathcal{S}$, then, by Section 3.3.4 and step 5 of Algorithm 3 we obtain $\bar{\gamma}(\max|\delta_{x_s}|, x_s) = a_{x_s} \max|\delta_{x_s}| + b_{x_s}$ for all $x_s \in \mathcal{S}_s$.

If $\delta_{min} \to 0$, then, for all x_s for which the verification at step 21 in Algorithm 3 is performed, but not validated, it means that $\delta_{x_s} \leq \delta_{min} \to 0$, and thus $a_{x_s} \max|\delta_{x_s}| \to 0$. Also, by (3.14), $b_{x_s} \to 0$. Therefore, $\bar{\gamma}(\max|\delta_{x_s}|, x_s) \to 0$. When verifying step 6 for such an x_s in Algorithm 3, we obtain $F(x_s) \leq (<) - \bar{\gamma}(\max|\delta_{x_s}|, x_s) \to 0$, which holds asymptotically by the assumption that $F(x) \leq (<)0$ for all $x \in \mathcal{S}$. Thus, steps 12–17 are applied, which shows that $\mathcal{A}(\delta_{min}) \to \mathcal{S}$ for $\delta_{min} \to 0$. Indeed, this is the case as inequality (3.3) is satisfied for all the $x_s \in \mathcal{S}_s$ with $\delta_{x_s} > \delta_{min}$. ■

Remark 3.3.12 (Numerical complexity analysis) The main scalability challenges in applying Algorithm 3 come from the number of samples, which is an exponential function of the state dimension, and the level of multi–resolution. It is therefore beneficial to exploit the decentralized feature of this algorithm in each sampling point via parallelization. To assess the computational load of Algorithm 3, let us assume that the computational cost of steps 4–5 for one sampling point is $c \in \mathbb{R}_+$. Also, assume that there exists a number $p \in \mathbb{Z}_+$ of parallel threads and the level of multi–resolution that we employ is $m \in \mathbb{Z}_+$. Also, denote by $w_i \in \mathbb{Z}_+$ the number of samples that were not verified at the previous multi–resolution step, where $i \in \mathbb{Z}_{[1,m]}$. Notice that w_1 is the initial number of samples, which is the number of elements in $\mathcal{S}_s$, at step 2 of Algorithm 3. Considering also that by multi–resolution of one hyper–rectangle we obtain 2^{n_r} new samples, then, the computational complexity of the `for` loop at steps 3–17 in Algorithm 3 is of the order $C = c\left(\lceil w_1/p \rceil + \lceil w_2 2^{n_r}/p \rceil + \ldots + \lceil w_m 2^{n_r}/p \rceil\right)$. Notice that, if the number of threads is unlimited, i.e., $p \to \infty$, then $C = c * m$, because at every level of multi–resolution, the number of threads in use is the same as the number of sampling points which we verify. □

Remark 3.3.13 If F is continuous at 0, $\mathcal{S}$ satisfies $0 \in int(\mathcal{S})$ and $F(0) = 0$, the inequality $F(x_s) \leq -\bar{\gamma}(\max|\delta_{x_s}|, x_s)$ can not be satisfied for $x_s = 0$. Hence, the closer x_s will be to zero, the more conservative the condition becomes. For this reason, the set $\mathcal{A}$, computed via Algorithm 3 will be an annulus in such a case, as it will be shown in the next chapter, dedicated to verifying Lyapunov's inequality. □

Algorithm 3 Construct $\mathcal{A} \subseteq \mathcal{S}$ such that $F(x) < 0$ on $\mathcal{A}$.

Input: $\mathcal{S}$, F, δ_{min}
Output: $\mathcal{A}$, ($F(x) < 0$ on $\mathcal{A}$)

1: $wrong \leftarrow []; r \leftarrow 0; \mathcal{A} = \varnothing; \mathcal{A}_s = \varnothing; good \leftarrow []; p \leftarrow 0$
2: Generate a Δ–sampling $\mathcal{S}_s$ of $\mathcal{S}$
3: **for all** $x_s \in \mathcal{S}_s$ **do**
4: Compute $a_{x_s}, b_{x_s}, \varepsilon(x_s)$
5: $\bar{\gamma}(\max|\delta_{x_s}|, x_s) = a_{x_s} \max|\delta_{x_s}| + b_{x_s} + \varepsilon(x_s)$
6: **if** $F(x_s) > -\bar{\gamma}(\max|\delta_{x_s}|, x_s)$ **then**
7: $r \leftarrow r + 1$
8: $wrong(r).del \leftarrow \delta_{x_s}$
9: $wrong(r).spoint \leftarrow x_s$
10: $wrong(r).tau \leftarrow \tau_{x_s}$
11: **else**
12: $p \leftarrow p + 1$
13: $good(p).del \leftarrow \delta_{x_s}$
14: $good(p).spoint \leftarrow x_s$
15: $good(p).tau \leftarrow \tau_{x_s}$
16: $\mathcal{A}_s \leftarrow \mathcal{A}_s \cup \{x_s\}$
17: $\mathcal{A} \leftarrow \mathcal{A} \cup \mathcal{B}_{\delta_{x_s}}(x_s)$
18: $k \leftarrow 1$
19: **if** $r > 0$ **then**
20: **while** 1 **do**
21: **if** $\max\{wrong(k).del\} > \delta_{min}$ **then**
22: Generate set $\mathcal{B}^s_{\delta_{x_s}}(wrong(k).spoint)$ of samples by multi–resolution on $\mathcal{B}_{\delta_{x_s}}(wrong(k).spoint)$
23: **for all** $x_s \in \mathcal{B}^s_{\delta_{x_s}}(wrong(k).spoint)$ **do**
24: $\delta_{x_s} = wrong(k).del/2$
25: Apply steps 4–16
26: **if** $k == r$ **then**
27: break
28: $k \leftarrow k + 1$

The operations performed in Algorithm 3 can be done in an automated fashion. This holds true assuming that the set $\mathcal{S}$, the real–valued function F and the minimum sampling density δ_{min} are provided. The set $\mathcal{S}$ is typically chosen as the constraint set, and δ_{min} is also chosen as the result of a trade–off process which aims to balance computational complexity and an acceptable accuracy. When these choices have been set, then Algorithm 3 is automatic in the sense that the constants a_{x_s}, b_{x_s}, $\varepsilon(x_s)$ and finally the set $\mathcal{A}$ are constructed without additional intervention. The level to which Algorithm 3 is automatic is particularly of importance for stability domains computation for nonlinear systems, where the choice of $F(x)$ is, in general, not a trivial decision. For this reason, we will treat this topic separately, in Chapter 4.

3.3.6 Comparison with the method of set inversion via interval analysis

The problem of verifying an inequality of the type $F(x) < 0$ for all $x \in \mathcal{S}$ can be set in the context of a set inversion problem as well, see (Jaulin and Walter, 1993).

There are many elements in common between the two methods. The correspondents of the parameters f, $\mathbb{Y}$, K_{in}, ε_r, $[x](0)$ from (Jaulin and Walter, 1993) are, in this thesis, F, $(-\infty, 0)$, $\mathcal{A}$, δ_{min} and respectively $\mathcal{S}$. In (Jaulin and Walter, 1993), the authors start with an initial set $[x](0)$, in which, via many iterations, using pavings, the set $\mathbb{K}_{in}$ is constructed, which is a lower estimate of the set $\mathbb{X}$ for which $f(\mathbb{X}) = \mathbb{Y}$. This procedure is similar in this thesis, where multi–resolution is used to construct the set $\mathcal{A}$. Basically, any method of splitting boxes can be used in both the procedure from (Jaulin and Walter, 1993) and this thesis, as well as many approaches within the interval analysis community.

The approach to verification of $F(x) < 0$ ($f([x](k))$ in (Jaulin and Walter, 1993)) differs between this thesis and the paper of (Jaulin and Walter, 1993). Therein, an inclusion function is used, $\mathbb{F}$ and then interval analysis is used to directly compute $\mathbb{F}([x](k))$ and verify that $\mathbb{F}([x](k)) \subset \mathbb{Y}$. In this thesis we replace the verification of $F(x) < 0$ on a set by verifying the conservative inequality $F(x_s) \leq -\overline{\gamma}(\max|\delta_{x_s}|, x_s)$ in only one sample point x_s. Then, we conclude $F(x) < 0$ for all $x \in \mathcal{B}_{\delta_{x_s}}(x_s)$. We conclude this by making use of $\overline{\gamma}(\max|\delta_{x_s}|, x_s)$, which is computed via interval analysis. Interval analysis is used differently in the two methods. In (Jaulin and Walter, 1993), it is used directly to evaluate the function f over an interval. In this thesis, it is used to evaluate the continuity constant $\overline{\gamma}(\max|\delta_{x_s}|, x_s)$ over an interval, after which the inequality $F(x) < 0$ is only verified in a sample point.

In this context, we identify the following differences between the two approaches. In (Jaulin and Walter, 1993), $\mathbb{Y}$ is required to be compact. The corresponding set $(-\infty, 0)$ in this thesis is not compact, because it is unbounded. The highest difference lays in the fact that we use samples in the verification. This allows the computation, within the function $\overline{\gamma}$, of a constant $\varepsilon(x_s)$ which is connected with the discontinuity of F. In this way we allow for F to be piecewise continuous, and not only continuous, as required in (Jaulin and Walter, 1993), which imposes continuity of f to guarantee existence of an inclusion function $\mathbb{F}$ (see (Jaulin and Walter, 1993, Remark 2, page 1056)). Our method, instead, by using samples, can detect precisely if the current verification is performed on a point of discontinuity, allowing the extension of interval analysis based verification to hybrid systems. However, for continuous functions F, the verification of F directly on an interval, as in (Jaulin and Walter, 1993) may be computationally less costly than the procedure we

follow in this thesis, where both constants a_{x_s} and b_{x_s} have to be computed via interval analysis. To evaluate these constants, we require computation of hessians and jacobians, which involve additional computational overhead.

3.4 Sampling–driven verification of finite–step invariance

The main property considered for verification using the sampling–driven procedure developed in this thesis is stability of an equilibrium point for a dynamical system. However, as we will illustrate in this section, the generic inequality $F(x) \leq (<)0$ is useful for verifying also other relevant properties, such as safety and finite–step invariance.

3.4.1 Finite–step invariance verification

The safety verification problem is approached in this subsection by finite–step invariance, which offers the advantage of choosing any set as a candidate finite–step invariant set, under the conditions mentioned in Remark 2.3.6.

Let us formulate the problem of verifying finite–step invariance of a compact set $\mathcal{S}$ such that none of the trajectories starting from $\mathcal{S}$ will ever exceed the safe set $\mathbb{E}$ for which $\mathcal{S} \subseteq \mathbb{E}$. Let us define

$$\mathcal{S} := \{x \in \mathbb{R}^n \ : \ V(x) \leq L\},$$

and

$$\mathbb{E} := \{x \in \mathbb{R}^n \ : \ V_E(x) \leq L_E\},$$

where $(L, L_E) \in (\mathbb{R}_{>0})^2$ and $V : \mathbb{R}^n \to \mathbb{R}$, $V_E : \mathbb{R}^n \to \mathbb{R}$ are nonlinear functions, such that the sets $\mathcal{S}$ and $\mathbb{E}$ have non–empty interior and they are connected.

The safety of the system with respect to the safe set $\mathbb{E}$ is verified through the following functions

$$F_i(x) = \frac{1}{L_E} V_E(G^i(x)) - 1,$$

where $i \in \mathbb{Z}_{[1,M-1]}$. For safety, the inequality

$$F_i(x) \leq 0$$

must hold for all $x \in \mathcal{S}$ and all $i \in \mathbb{Z}_{[1,M-1]}$.

The M–step invariance property can be formulated through the following function:

$$F_M(x) = \frac{1}{L} V(G^M(x)) - 1,$$

for which it is required that

$$F_M(x) \leq 0, \quad \forall x \in \mathcal{S}.$$

Observe that verifying safety reduces to verifying M inequalities, due to Lemma 2.2.5. Moreover, by Lemma 2.2.5 it is established that the M–step invariant set, together with the trajectories starting inside it, compose an invariant set.

Given the fact that we can split the problem of verifying safety into M separate problems, it means that we can verify M independent problems as in (3.3), for each of the functions F_i, where $i \in \mathbb{Z}_{[1,M]}$, on the set $\mathcal{S}$. If (3.3) holds for each of the M functions F_i, then the set $\mathbb{E}$ is safe for all the trajectories starting in the set $\mathcal{S}$.

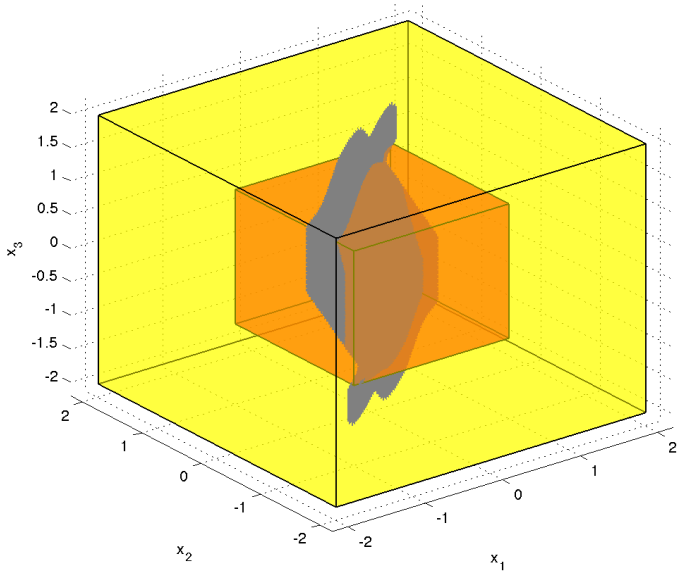


Figure 3.3: M–step invariant set verification for the nonlinear system (2.22).

3.4.2 Numerical example

To illustrate the problem of verifying safety and finite–step invariance as discussed above, let us consider again the dynamics from Example 1 in Chapter 2. Let the safe set be

$$\mathbb{E} := \{x \in \mathbb{R}^3 \ : \ \|x\|_\infty \leq 2\},$$

and choose a set, e.g.,

$$\mathcal{S} := \{x \in \mathbb{R}^3 \ : \ \|x\|_\infty \leq 1\},$$

for which we want to verify that it is an M–step invariant set for the nonlinear system (2.22), with safe set $\mathbb{E}$. For simplicity, let us perform sampling via hyper–cubes, with $\delta = 0.05$ as the discretization constant which gives $\delta_{x_s} = \delta \mathbf{1}_6$ for all $x_s \in \mathcal{S}_s$. This generates the Δ–sampling of the set $\mathcal{S}$, with a sample set $\mathcal{S}_s$. Also, select $\delta_{min} = \delta$. This means that in this example we will not resort to the mechanism of multi–resolution sampling, which we will extensively use in the next chapter.

By applying Algorithm 3 for each of the functions F_i, for various values of $M \in \mathbb{Z}_{\geq 1}$, we obtain that, for $M = 5$, all the samples in the Δ–sampling of the set $\mathcal{S}$ satisfy (3.3) for all functions F_i with $i \in \mathbb{Z}_{[1,5]}$. Therefore, the set $\mathcal{S}$ is 5–step invariant for the nonlinear system (2.22), with safe set $\mathbb{E}$.

In this example, it is not necessary to construct $\overline{\gamma}$ as in Section 3.3.4, because it can be immediately verified that a_{x_s} is the Lipschitz constant a_{F_i} of F_i. This constant can be computed, see (Kellett, 2014) for the properties of $\mathcal{K}$–functions, for the functions F_i for all

$i \in \mathbb{Z}_{[1,M]}$ and we obtain $a_{F_1} = 1.5$, $a_{F_2} = 2$, $a_{F_3} = 2.625$, $a_{F_4} = 3.5625$, $a_{F_5} = 11.5$. Then, notice that $b_{x_s} = 0$ and $\varepsilon(x_s) = 0$, because of the continuity of the dynamics (2.22) for all $x_s \in \mathcal{S}_s$.

In Figure 3.3 we illustrate by yellow the safe set $\mathbb{E}$, by red the M–step invariant set $\mathcal{S}$ and in blue the set

$$T := \bigcup_{x_s \in \mathcal{S}_s, i \in \mathbb{Z}_{[1,M]}} G^i(x_s).$$

The set $\mathcal{S} \cup T$ is a subset of the invariant set $R_M^{\mathcal{S}}$, see Lemma 2.2.5.

3.5 Conclusions

We have developed a sampling–driven algorithm for verification of generic properties of the type $F(x) \leq (<)0$ on compact sets $\mathcal{S}$. This property, originally inspired by Lyapunov inequalities, has the potential of representing a large variety of properties of interest when analyzing safety for constrained nonlinear systems. Indeed, as shown in this chapter, the property function $F(x)$ can be used, for instance, in the verification of finite–step invariance, or safety of system trajectories with respect to a safe set $\mathbb{E}$. Typical solutions for verifying such properties require solving optimization problems, which suffer from non–convexity, non–feasibility, scalability and numerical solvers issues. The algorithm presented in this chapter is based on a sampling–guided verification theorem that extends a previous result for Lipschitz continuous dynamics to general discrete–time, possibly discontinuous dynamics. This opens up the application of sampling–driven verification to hybrid systems. Because of typical issues in finding a LF, but also particular challenges related to sampling–driven verification of Lyapunov's inequality around the origin, we dedicate the next chapter to applying the algorithm developed so far to verify Lyapunov's inequality for both discrete–time as well as continuous–time nonlinear systems, but also for computing DOAs.

Chapter 4

Deterministic sampling–driven computation of stability domains

This chapter develops a solution for stability domains computation of piecewise continuous nonlinear systems via Lyapunov functions. Depending on the nonlinear system dynamics, the candidate Lyapunov function and the set of states of interest, verifying stability requires solving complex, possibly non–convex or infeasible optimization problems. To avoid these issues, in this chapter we adopt a sampling–driven approach, as proposed in Chapter 3. Moreover, to achieve a constructive sampling–driven stability verification, a Lyapunov function is built via finite–step Lyapunov functions for discrete–time systems, as discussed in Chapter 2, and finite–time Lyapunov functions for continuous–time systems. Besides adapting the sampling–driven verification for property functions which represent the relaxed Lyapunov functions, this chapter addresses the problem of verifying Lyapunov's inequality at the origin via a theorem which connects the annulus computed via Algorithm 3 with the neighborhood of the origin in the verification of stability. Verification of Lyapunov's inequality is presented for both discrete–time systems and continuous–time systems, which present different challenges for verification. Additionally, a sampling–driven method for estimating the domain of attraction (DOA) via level sets of a validated Lyapunov function is also presented. The proposed methodology is illustrated for various benchmark examples from the literature.

4.1 Introduction

A LF is typically constructed for stability verification of constrained nonlinear systems, and its largest viable level set inside the set of constraints is computed to estimate the domain of attraction (DOA) of an equilibrium of interest (Khalil, 2002), (Vidyasagar, 2002). Most Lyapunov methods rely on the following common approach: verify the decrease condition for a candidate LF and a candidate subset of $\mathbb{R}^n$, which can be a bounded or unbounded set, an infinite or finite set of states (e.g., generated by simulations (Kapinski et al., 2014) or state–space sampling (van der Spek, 1994), (Kapinski and Deshmukh, 2013)). For linear, switched–linear or polynomial systems, the decrease condition for a candidate LF can be posed via semidefinite programming (SDP). In the case of nonlinear systems with CPA candidate LFs, simplicial state–space partitions and linear programming are used to verify

the decrease condition of the LF candidate. However, for general nonlinear systems, such standard Lyapunov methods require attaining the global optimum of complex, possibly non–convex optimization problems or they may result in infeasible problems. The corresponding optimization problems do not scale well with the state–space dimension and in the non–convex case, attaining a global optimum for a large set of states cannot be guaranteed.

Sampling–based approaches have been used in stability verification mostly to overcome the general conservativeness of candidate LFs. As such, in (Topcu et al., 2008) and (Kapinski et al., 2014), sampling–based approaches have been combined with optimization–based approaches for constructing LFs. In (Topcu et al., 2008), samples in the state space are used to generate simulation traces and computing local candidate polynomial LFs for polynomial dynamics. Therein, sampling and simulations enable the conversion of a set of computationally expensive bilinear matrix inequalities into linear matrix inequalities. This idea has been extended in (Kapinski et al., 2014) by a procedure to improve iteratively the quality of the candidate polynomial LF. The procedure therein relies on a falsification tool in the form of a global optimizer which generates a series of successively improved intermediate polynomial LFs. Additionally, the resulting polynomial LF found by the simulation–based iterative technique is validated formally through queries in Satisfiability Modulo Theories (SMT) solvers such as dReal (Gao et al., 2012), Z3 (De Moura and Bjørner, 2008), Meti-Tarski (Akbarpour and Paulson, 2010).

Alternatively, in (Berkenkamp et al., 2016), a subset of the DOA of uncertain systems is estimated. To achieve this, the candidate LF is fixed and the DOA is estimated based on arbitrary experiments and learning strategies. The DOA holds with an accuracy determined by a probabilistic certificate. Therein, a constructive method for choosing the candidate LF is not specified. In (Najafi et al., 2016), a fast method for estimating the DOA was proposed based on random samples in the state space, with no formal guarantees.

In this chapter, the problem of automatically obtaining formal stability guarantees via LFs is addressed for general nonlinear systems using a sampling–driven approach and FSLFs for discrete–time systems and finite–time Lyapunov functions (FTLFs) (Doban and Lazar, 2016a) for continuous–time systems respectively. FSLFs and FTLFs are used in a converse result, to construct candidate LFs and the Lyapunov's inequality is verified on a set $\mathcal{S}$ via Algorithm 3. Two patching theorems allow binding of the validated LF with a local LF. This result is required to deal with singularity at the equilibrium point. Additionally, a sampling–based method for estimating the largest viable level set of the validated LF is developed to generate DOA estimates. This method exploits the samples which have already been explored by Algorithm 3 to compute an under–approximation of the maximum level set of the LF inside the set of points which do satisfy Lyapunov's inequality.

Particularly, for continuous–time systems, finding a true FTLF requires computing the solution of a nonlinear differential equation. To avoid this, we present two approaches. The first approach is to discretize the system, compute a FSLF and a LF for the discretized system, and then verify the candidate LF's validity on the original continuous–time system via the sampling–guided method in Algorithm 3 in Chapter 3. This approach relies on the assumption that a discretized system with a sufficiently small sampling time is a satisfactory abstraction of the original continuous–time system. Alternatively, a candidate LF can be obtained by the integration of a FTLF over a finite time interval. The drawback of this approach is that it requires knowledge of the system solution for a finite time interval. To

solve this issue, we generate simulation traces by numerical integration for initial conditions inside a ball around each sample point and we fit a polynomial function to approximate the true solution. This yields an analytic formula for the LF candidate which is defined differently for each sample point. Then, we again extend the validity of the LF computed for each sample point as in Algorithm 3.

Algorithm 3 enables usage of the same prototype verification method for verifying Lyapunov inequalities for general piecewise continuous dynamics for both continuous–time and discrete–time systems. Solving a posteriori a global optimization problem for validating the resulting function for constrained nonlinear systems is not required by the proposed method, compared with other simulation–based approaches, such as (Kapinski et al., 2014).

4.2 Stability analysis tools

Let us first recall the basic ingredients for stability analysis. Consider the autonomous nonlinear system in discrete–time

$$x_{k+1} = G(x_k), \quad k \in \mathbb{Z}_+, \tag{4.1}$$

and in continuous–time

$$\dot{x}(t) = G_c(x(t)), \quad t \in \mathbb{R}_+ \tag{4.2}$$

where $x_k \in \mathcal{S}$ (resp. $x(t) \in \mathcal{S}$) is the state, $\mathcal{S}$ is a proper set denoting a subset of the set of constraints and $G : \mathbb{R}^n \to \mathbb{R}^n$, $G_c : \mathbb{R}^n \to \mathbb{R}^n$ are piecewise continuous nonlinear functions. Additionally, G_c is locally Lipschitz. A point $x^* \in \mathcal{S}$ is an equilibrium point of system (4.1) if $G(x^*) = x^*$, and of system (4.2) if $G_c(x^*) = 0$. We assume $G(0) = 0$ and $G_c(0) = 0$. This assumption is not limiting the applicability of the methods developed in this thesis, because any other equilibrium point can be translated in 0 via a simple set of operations. Additionally, other equilibria are discarded when the stability and DOA of only one equilibrium point is studied. Denote the solution of (4.2) with initial state $x(0)$ at time $t = 0$ by $x(t)$ for any $t \in \mathbb{R}_{\geq 0}$. Assume that $x(t)$ exists and it is unique for all $t \in \mathbb{R}_{\geq 0}$. For system (4.1), define the one–step reachable set from $\mathcal{S}$ as $Reach(\mathcal{S}) := \cup_{\xi \in \mathcal{S}} G(\xi)$.

Definition 4.2.1 The system (4.1) (resp. (4.2)) is called $\mathcal{KL}$–stable in $\mathcal{S}$ if there exists a $\mathcal{KL}$ function $\beta : \mathbb{R}_+ \times \mathbb{R}_+ \to \mathbb{R}_+$ such that $\|x_{k+1}\| \leq \beta(\|x_0\|, k)$ for all $(x_0, k) \in \mathcal{S} \times \mathbb{Z}_+$ (resp. $\|x(t)\| \leq \beta(\|x(0)\|, t)$ for all $(x(0), t) \in \mathcal{S} \times \mathbb{R}_+$).

Note that the definition of $\mathcal{KL}$–stability from Definition 4.2.1 is similar to the notion of global asymptotic stability (GAS) in (Khalil, 2002). However, for discrete–time systems, there are GAS systems, when the map G in (4.1) is discontinuous, which are not $\mathcal{KL}$–stable. Such an example is detailed in (Mayne and Rawlings, 2009, Appendix B, Example 2, page 7).

The standard regional Lyapunov theorems for both continuous– as well as discrete–time systems are recalled in the next result.

Proposition 4.2.2 *Let $\mathbb{W}$ be a proper invariant set with respect to the dynamics* (4.1) *(resp.* (4.2)*). Let $\alpha_1, \alpha_2 \in \mathcal{K}_\infty$. Suppose that there exists a function $W : \mathbb{R}^n \to \mathbb{R}_+$ such that*

$$\alpha_1(\|x\|) \leq W(x) \leq \alpha_2(\|x\|), \quad \forall x \in \mathbb{R}^n, \tag{4.3a}$$

and that for some $\rho \in \mathcal{K}$ with $\rho < id$ it holds

$$W(G(x)) \leq \rho(W(x)), \quad \forall x \in \mathbb{W}, \tag{4.3b}$$

for system (4.1) *(resp. it holds*

$$\dot{W}(x) < 0, \quad \forall x \in \mathbb{W}\backslash\{0\}, \tag{4.3c}$$

for system (4.2)*). Then, W is a Lyapunov function on $\mathbb{W}$ and system* (4.1) *(resp. system* (4.2)*) is $\mathcal{KL}$-stable in $\mathbb{W}$.*

Let $V : \mathbb{R}^n \to \mathbb{R}_+$ be defined as

$$V(x) := \eta(\|x\|) \tag{4.4}$$

for some ("arbitrary") $\eta \in \mathcal{K}_\infty$ and norm $\|\cdot\|$, see (Doban and Lazar, 2016b, Theorem 2.4). If there exists an $M \in \mathbb{Z}_{\geq 1}$ and a corresponding $\rho \in \mathcal{K}$ with $\rho < id$ such that

$$V(G^M(x)) \leq \rho(V(x)), \quad \forall x \in \mathbb{W}, \tag{4.5}$$

then V is a FSLF for system (4.1). Then, by Theorem 2.3.1, the function $W : \mathbb{R}^n \to \mathbb{R}_+$ which satisfies

$$W(x) = \sum_{j=0}^{M-1} V(G^j(x)), \tag{4.6}$$

is a true LF for the discrete–time system (4.1).

Similarly, for continuous–time systems, if there exists a $d \in \mathbb{R}_{>0}$ such that for any $t \geq 0$, if $x(t) \in \mathbb{W}$ and

$$V(x(t+d)) - V(x(t)) < 0, \tag{4.7}$$

then V is a finite–time Lyapunov function (FTLF) for system (4.2) and the function $W : \mathbb{R}^n \to \mathbb{R}_+$ which satisfies

$$W(x(t)) = \int_t^{t+d} V(x(\tau))d\tau \tag{4.8}$$

is a true LF for the continuous–time system (4.2), see (Doban and Lazar, 2016a, Lemma 2.3, Theorem 2.1).

The above–relation between V and W will be instrumental in applying sampling–driven verification results to construct regional LFs. Recall that Algorithm 3 in Chapter 3 constructs a set $\mathcal{A} \subseteq \mathcal{S}$ such that an inequality of the type $F(x) < (\leq) 0$ holds for all $x \in \mathcal{A}$.

Next, we will show how Algorithm 3 can be utilized to verify Lyapunov's inequality for discrete–time and continuous–time dynamical systems, respectively, and for computing an estimate of the DOA as a viable sublevel set of a validated LF. Thus, we will have to show how inequalities of the type (4.3b), (4.3c), (4.5), or (4.7) can be posed via a property function $F(x) < (\leq) 0$. For instance, for (4.3b), $F(x) = W(G(x)) - \rho(W(x))$. For constructing DOA, another algorithm is developed, which still exploits the samples generated via Algorithm 3.

4.3 Verification of Lyapunov's inequality for discrete–time systems

In general, verification of Lyapunov's inequality (4.3b) hinges upon choosing the right LF candidate W. However, as W is difficult to guess for general nonlinear dynamics, we pursue verification of inequality (4.5) instead, which merely requires an *arbitrary* function V (i.e., $\eta \in \mathcal{K}_\infty$) and a suitable value of M. Given a compact set of interest $\mathcal{S}$ with $0 \in \operatorname{int}(\mathcal{S})$, Lyapunov's inequality can be readily verified by Algorithm 3 by setting the property function F as follows:

$$F(x) := V(G^M(x)) - \rho(V(x)), \quad \forall x \in \mathcal{S}, \tag{4.9}$$

with $\rho \in \mathcal{K}$ which satisfies $\rho < id$. Algorithm 3 will produce a subset $\mathcal{A} \subset \mathcal{S}$ where the inequality holds. To enlarge the set $\mathcal{A}$ of verified points, one can either reduce the sampling resolution δ_{min} or increase the value of M iteratively up to a prescribed upper bound M_{max}. Indeed, as shown in (Gielen and Lazar, 2015), under the assumption of exponential stability of the equilibrium, there exists a critical bound M^* such that inequality (4.5) is satisfied for all $M \geq M^*$, regardless of the candidate function V.

In turn, Lyapunov's inequality (4.3b) is then implicitly verified for all $x \in \mathcal{A}$ for the function $W : \mathbb{R}^n \to \mathbb{R}_+$ in (4.6). However, since both $W(0) = 0$ and $V(0) = 0$, we have that for the property function corresponding to Lyapunov's inequality it holds that $F(0) = 0$ and the limitation indicated in Remark 3.3.13 holds true.

In order to conclude $\mathcal{KL}$–stability for initial conditions that lie in the neighborhood around the origin where the inequality was not verified, we will make use of a set $\mathcal{L}$, which is the level set of a true local LF V_L. The methodology to compute such a local LF can rely on linearization of the dynamics in (4.1), for instance. Indeed, if the origin is a stable equilibrium, then, the LF found for the linear system is also a LF for the nonlinear system in a neighborhood $\mathcal{N}_1(0)$, see (Khalil, 2002, Theorem 4.7, pag. 139). The set $\mathcal{L}$ can then be chosen as the largest level set of the LF V_L inside $\mathcal{N}_1(0)$, see Figure 4.1.

Naturally, a linearization of the nonlinear dynamics can be used to screen for potential candidates for the FSLF candidate V as well. However, most of the time, the LF computed for the linearized system holds for the nonlinear system only locally, on a very small set around the origin. An "educated" guess for a candidate FSLF is recommended, for instance, via linearization, or the approach in (Kapinski et al., 2014), to obtain a small step M. However, theoretically, any function V defined as in (4.4) is a valid candidate. For this reason, we select an *arbitrary* candidate V, which, via the M iterations, will eventually reveal a true FSLF. In what follows, we will establish sufficient conditions such that stability can be guaranteed on $\mathcal{A} \cup \mathcal{L}$ with the ingredients we have worked out so far. To this end, the following fact is instrumental.

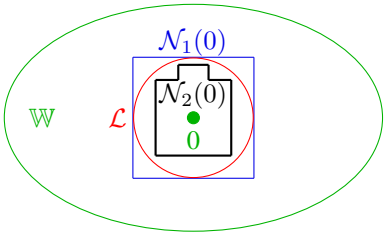


Figure 4.1: Set inclusions for Theorem 4.3.1.

Fact 4.1 Let W be a candidate Lyapunov function satisfying (4.3a). Moreover, $\mathbb{W} := \{x \in \mathbb{R}^n : W(x) \leq L\}$ is a level set of W with $L \in \mathbb{R}_{>0}$ and $\mathcal{L} \subseteq \mathbb{W}$ is a compact invariant set for system (4.1), with $0 \in int(\mathcal{L})$. If $W(G(x)) - \rho(W(x)) < 0$ holds for all $x \in \overline{\mathbb{W} \setminus \mathcal{L}}$ with $\rho < id$, then $\mathbb{W}$ is an invariant set. □

Proof: The set $\mathbb{W}$ is invariant if and only if for all $x \in \mathbb{W}$ if holds that $G(x) \in \mathbb{W}$. If $x \in \mathcal{L}$, then $G(x) \in \mathcal{L} \subseteq \mathbb{W}$ by the invariance of $\mathcal{L}$. Otherwise, if $x \in \overline{\mathbb{W} \setminus \mathcal{L}}$, then $W(G(x)) < \rho\,(W(x)) \leq \rho(L) < L$, and thus, $G(x) \in int(\mathbb{W}) \subset \mathbb{W}$, which completes the proof. ■

If the Lyapunov inequality holds on $\mathcal{S}$ and $\mathcal{S}$ is invariant, then the system (4.1) is $\mathcal{KL}$–stable in $\mathcal{S}$. However, as pointed above, Lyapunov's inequality can not be verified via the sampling–driven method at $x_s = 0$. Therefore, we can at most verify the Lyapunov inequality via sampling on an annulus $\mathcal{A}$. Still, with the aid of the next patching theorem, we can establish $\mathcal{KL}$–stability in a subset of $\mathcal{S}$ that includes the origin it its interior and the set of verified points $\mathcal{A}$ as well.

Theorem 4.3.1 *Let W be a candidate Lyapunov function satisfying* (4.3a)*, with $\mathbb{W} \in \mathbb{R}^n$ a sublevel set of W. Consider a proper set $\mathcal{N}_2(0) \subseteq \mathbb{W}$. Denote by $\mathcal{A}_{\mathbb{W}} := \overline{\mathbb{W} \setminus \mathcal{N}_2(0)}$ the annulus of the compact set $\mathbb{W}$ with respect to the set $\mathcal{N}_2(0)$ and suppose that $W(G(x)) - \rho(W(x)) \leq 0$ holds for all $x \in \mathcal{A}_{\mathbb{W}}$, where G is the nonlinear map of system* (4.1) *and $\rho \in \mathcal{K}$ and $\rho < id$. Assume that there exists a compact set $\mathcal{L}$ (see Figure 4.1) with $\mathcal{N}_2(0) \subseteq \mathcal{L} \subseteq \mathbb{W}$ which is invariant with respect to the nonlinear system* (4.1) *and admits a Lyapunov function $V_L : \mathbb{R}^n \to \mathbb{R}_+$ on $\mathcal{L}$. Then, system* (4.1) *is $\mathcal{KL}$–stable on $\mathbb{W}$.* □

Proof: Given an arbitrary initial condition $x_0 \in \mathbb{W}$, the following situations can be encountered:

1. If $x_0 \in \mathcal{L}$, then Proposition 4.2.2 may be applied for system (4.1) with $W(x) := V_L(x)$, and therefore system (4.1) is $\mathcal{KL}$–stable on $\mathcal{L}$.

2. If $x_0 \in \mathbb{W} \setminus \mathcal{L} \subseteq \mathcal{A}_{\mathbb{W}}$, and since by Fact 4.1 $\mathbb{W}$ is an invariant set, then $G(x_0) \in \mathbb{W}$, which allows for two situations:

(a) If $G(x_0) \in \mathcal{L}$, then the reasoning used in case 1. can be applied again with $\overline{x}_0 := G(x_0)$.

(b) If $G(x_0) \in \mathbb{W} \setminus \mathcal{L} \subseteq \mathcal{A}_{\mathbb{W}}$, suppose $\nexists i \in \mathbb{Z}_{>0}$ such that $G^i(x_0) \in \mathcal{L}$. Then, by the invariance of the set $\mathbb{W}$ it follows that $G^i(x_0) \in \mathbb{W} \setminus \mathcal{L} \subseteq \mathcal{A}_{\mathbb{W}}, \forall i \in \mathbb{Z}_{>0}$. Therefore, the inequality $W(G(x)) - \rho(W(x)) \leq 0$ can be iterated i–times to obtain the following:

$$0 \leq W(G^i(x_0)) \leq \rho^i(W(x_0)), \forall i \in \mathbb{Z}_{>0}.$$

Because $\rho < id$ and $\rho(0) = 0$, then $\lim_{i\to\infty} \rho^i(W(x_0)) = 0$, and therefore

$$\lim_{i\to\infty} W(G^i(x_0)) = 0. \tag{4.10}$$

Moreover, by (4.3a) we know that

$$0 \leq \alpha_1(\|G^i(x_0)\|) \leq W(G^i(x_0)). \tag{4.11}$$

By (4.10) and (4.11) the following limit holds:

$$\lim_{i\to\infty} \alpha_1(\|G^i(x_0)\|) = 0, \tag{4.12}$$

which, by the definition of $\mathcal{K}$–functions, yields:

$$\lim_{i\to\infty} \|G^i(x_0)\| = 0. \tag{4.13}$$

However, because $G^i(x_0) \notin \mathcal{L}, \forall i \in \mathbb{Z}_{>0}$, then

$$\|G^i(x_0)\| > r_B > 0, \forall i \in \mathbb{Z}_{>0}, \tag{4.14}$$

and therefore (4.14) contradicts (4.13). This means that $\exists i \in \mathbb{Z}_{>0}$ such that $G^i(x_0) \in \mathcal{L}$. Thus, the reasoning in case 1. can be applied with $\overline{x}_0 := G^i(x_0)$.

The above cases cover all the possible trajectory situations starting from the set $\mathbb{W}$, and therefore prove $\mathcal{KL}$–stability of system (4.1) on $\mathbb{W}$. ■

In the case when even a local LF V_L can not be found, the safety of the trajectories starting in $\mathbb{W}$ can still be guaranteed if

$$Reach(\mathcal{N}_2(0)) \subseteq \mathcal{A} \cup \mathcal{N}_2(0) \subseteq \mathbb{W}.$$

Then $\mathbb{W}$ is guaranteed to be an invariant set, which provides a safety certificate for trajectories starting in $\mathbb{W}$.

4.4 Verification of Lyapunov's inequality for continuous–time systems

Similarly to the discrete–time case, verifying Lyapunov's inequality for continuous–time systems (4.3c) hinges on finding a suitable candidate LF W. To construct the LF W, we develop two different approaches, as follows.

4.4.1 Constructing LFs via the discretized system

It is common practice to apply time discretization in order to provide analysis and control techniques for continuous–time systems. However, for safety critical systems it is necessary to validate the obtained results for the original continuous–time system. Similarly, in this section we take an indirect approach to find W, i.e., we rely on the candidate function W obtained as in Section 4.3 for a discretized version of the continuous–time dynamics. Then, Algorithm 3 in Chapter 3 can be readily applied with the property function $F(x)$ defined as follows:

$$F(x) := \dot{W}(x), \quad \forall x \in \mathcal{S}. \tag{4.15}$$

Also in the continuous–time setting it holds that $F(0) = 0$, and therefore the inequality $F(x_s) \leq -\bar{\gamma}(\max|\delta_{x_s}|, x_s)$ can not be satisfied for $x_s = 0$. Thus, the set of verified points $\mathcal{A}_c$ produced by Algorithm 3 will be an annulus as well. To cover a neighborhood of the origin $\mathcal{N}_2(0)$ of points that are not validated, we need to contain it within a set $\mathcal{L}_c$, which is the level set of a true local LF V_L. The set $\mathcal{L}_c$ can then be taken as the smallest sublevel set of the LF V_L which covers the hole of the annulus $\mathcal{A}_c$. To connect $\mathcal{A}_c$ with $\mathcal{L}_c$, we formulate the following result.

Theorem 4.4.1 *Let W be a candidate Lyapunov function satisfying* (4.3a)*, with $\mathbb{W}_c := \{x \in \mathbb{R}^n : W(x) \leq L_c\} \subset \mathbb{R}^n$ a sublevel set of W. Consider a proper set $\mathcal{N}_2(0) \subseteq \mathbb{W}$. Denote by $\mathcal{A}_{\mathbb{W}_c} := \overline{\mathbb{W}_c \setminus \mathcal{N}_2(0)}$ the annulus of the compact set $\mathbb{W}_c$ with respect to the set $\mathcal{N}_2(0)$ and suppose that $\dot{W}(x) < 0$ for all $x \in \mathcal{A}_{\mathbb{W}_c}$. Assume that there exists a compact set $\mathcal{L}_c$ with $\mathcal{N}_2(0) \subseteq \mathcal{L}_c \subseteq \mathbb{W}_c$, which is invariant for system* (4.2)*, and admits a Lyapunov function $V_L : \mathbb{R}^n \to \mathbb{R}_+$. Then, system* (4.2) *is $\mathcal{KL}$–stable on $\mathbb{W}_c$.* □

Proof: Notice that, according to (Blanchini and Miani, 2007), $\mathbb{W}_c$ is a practical set, and therefore, according to Nagumo's theorem, the set $\mathbb{W}_c$ is positively invariant with respect to (4.2) if and only if

$$\nabla(W(x) - L)^T G_c(x) = \nabla W(x)^T G_c(x) \leq 0$$

for all $x \in \partial\mathbb{W}_c$. Recall that $\partial\mathbb{W}_c$ denotes the boundary of the set $\mathbb{W}_c$. Since $\partial\mathbb{W}_c \subseteq \partial\mathcal{A}_{\mathbb{W}_c}$, and since according to the hypothesis it holds that

$$\dot{W}(x) = \nabla W(x)^T G_c(x) < 0, \quad \forall x \in \mathcal{A}_{\mathbb{W}_c},$$

then it follows that

$$\nabla W(x)^T G_c(x) \leq 0, \quad \forall x \in \partial\mathbb{W}_c.$$

Therefore, the fact that $\dot{W}(x) < 0$ for all $x \in \mathcal{A}_{\mathbb{W}_c}$ implies that $\mathbb{W}_c$ is positively invariant for (4.2). Given an arbitrary initial condition $x_0 \in \mathbb{W}_c$, the following situations can be encountered:

1. If $x_0 \in \mathcal{L}_c$, then the problem is solved, because system (4.2) is $\mathcal{KL}$–stable on $\mathcal{L}_c$.
2. If $x_0 \in \mathbb{W}_c \setminus \mathcal{L}_c \subseteq \mathcal{A}_{\mathbb{W}_c}$, then, due to the invariance of the set $\mathbb{W}_c$ it means that $x(t) \in \mathbb{W}_c$, for any $t \in \mathbb{R}_+$, which allows for two situations:

(a) If $x(t) \in \mathcal{L}_c$, then the reasoning used in case 1. can be applied again with $\overline{x}_0 := x(t)$.

(b) Assume $x(t) \in \mathbb{W}_c \setminus \mathcal{L}_c \subseteq \mathcal{A}_{\mathbb{W}_c}$ for all $t > 0$. By the continuity of W on the compact set $\mathcal{A}_{\mathbb{W}_c}$ the following maximization problem provides a bounded result:

$$-\beta := \max_{x \in \mathcal{A}_{\mathbb{W}_c}} \dot{W}(x) < 0.$$

Therefore, for all $x(t)$ starting from an initial condition $x(0) \in \mathbb{W}_c \setminus \mathcal{L}_c \subseteq \mathcal{A}_{\mathbb{W}_c}$ we can write

$$W(x(t)) = W(x(0)) + \int_0^t \dot{W}(x(\tau))d\tau \leq W(x(0)) - \beta t.$$

Notice that for all $t > \frac{W(x(0))}{\beta}$ it follows that

$$W(x(t)) \leq W(x(0)) - \beta t < 0,$$

which contradicts the positive definiteness of W. Therefore, for all

$$x_0 \in \mathbb{W}_c \setminus \mathcal{L}_c \subseteq \mathcal{A}_{\mathbb{W}_c}$$

there exists $t \in \mathbb{R}_{>0}$ such that $x(t) \in \mathcal{L}_c$, where the same reasoning as in 2.(a) can be applied.

The above cases prove $\mathcal{KL}$ stability of system (4.2) on the set $\mathbb{W}_c$. ■

4.4.2 Constructing LFs via polynomial approximation of the solution

The problem of verifying $\mathcal{KL}$–stability of system (4.2) on a compact set $\mathcal{S}$, can be posed directly via FTLFs (4.7). Then, the property function to be verified is:

$$F(x(t)) = V(x(t+d)) - V(x(t)) < 0, \quad \forall x(t) \in \mathcal{S}. \tag{4.16}$$

We will use the fact that $F(x(t))$ can be reduced to $F(x(0))$ if $\mathcal{S}$ is invariant. Then, the problem of verifying that (4.16) holds for all $x(0) \in \mathcal{S}$ can be dealt with via Algorithm 3 in Chapter 3.

The main steps of finding a LF for system (4.2) are the following:

1. Verify (4.16) for an "arbitrary" suitable V, d and set $\mathcal{S}$.

2. Derive an expression of $W(x(0))$ at $x(0) = x_s$, via (4.8).

It is known (Doban and Lazar, 2016a) that any function V which satisfies (4.3a) may be selected as a valid candidate FTLF, with a corresponding d. Therefore, to construct a LF $W(x)$, it suffices to find a d such that (4.16) holds, and then the LF $W(x)$ can be constructed via (4.8). However, notice that an analytical solution of system (4.2) is required to be able to compute $x(t+d)$ for a given $x(t)$, $t \geq 0$. For continuous–time nonlinear systems, such a solution is generally not attainable.

Therefore, in this section, an approximation of the trajectory $x(t+d)$ by a polynomial function is constructed. Since in the verification of (4.16) the true solution $x(t+d)$ is approximated by an analytical function, even if (4.16) holds for all $x(t) \in \mathcal{S}$, after we construct $W(x(t))$ as in (4.8), the certification that $W(x(t))$ is a LF is still required.

The next items elaborate on steps 1. and 2. mentioned above:

1. Select a pool of samples $\mathcal{S}_s \subset \mathcal{S}$.

2. Generate an approximation of the true solution $x(t)$ around a point x_s by an analytical polynomial function $f(t, x, x_s)$:

2.1. For all $x_s \in \mathcal{S}_s$, simulate (4.2) for $t = 0 : T_s : T$ starting from a number of random points $x(0)$, uniformly distributed in $\mathcal{N}(x_s)$, where $\mathcal{N}(x_s)$ is a subset of $\mathcal{S}$ that contains the point x_s.

2.2. Fit a polynomial on the generated trajectories, such that

$$x(t) \approx f(t, x, x_s) = \begin{bmatrix} f_1(t, x, x_s) \\ f_2(t, x, x_s) \\ \cdots \\ f_n(t, x, x_s) \end{bmatrix},$$

where $f_i(t, x, x_s)$ are n polynomials of degree m. Notice that the variables of the polynomials are x and t, and x_s is the sampling point around which the polynomial approximations were generated.

3. Fix a candidate FTLF V as in (4.4). Find a $d \in \mathbb{R}_+$ such that

$$V(f(d, x_s, x_s)) - V(x_s) < 0, \quad \forall x_s \in \mathcal{S}_s.$$

4. Construct a candidate LF:

$$W(x) = \int_0^d V(f(\tau, x, x_s))d\tau, \quad \forall x \in \mathcal{N}(x_s). \tag{4.17}$$

Let us illustrate the construction of a LF as in (4.17) in the case when the FTLF is a quadratic function $V(x) = x^T P x$, where $P \in \mathbb{R}^{n \times n}$ and $P \succ 0$. Assume $f_i(t, x, x_s)$, for all $i \in \mathbb{Z}_{[1,n]}$ is an m–th order polynomial, which has $n_r \in \mathbb{Z}_+$ coefficients, placed in a column vector $c_i \in \mathbb{R}^{n_r}$. A matrix $C \in \mathbb{R}^{n_r \times n}$ is defined as $C = [\ c_1 \quad c_2 \quad \dots \quad c_n\ ]$, and it represents the matrix of all the coefficients for the polynomials $f_i(t, x, x_s)$, for all $i \in \mathbb{Z}_{[1,n]}$. For each polynomial $f_i(t, x, x_s)$, for all $i \in \mathbb{Z}_{[1,n]}$, build also a matrix containing on each line the powers with which each variable $(t, x_1, \dots, x_n)$ appears in each of the n_r terms (a similar reasoning is used within the `polyfitn` function in Matlab). Denote such a matrix $T_i \in \mathbb{Z}^{n_r \times (n+1)}$. A matrix $T \in \mathbb{R}^{n_r \times (n(n+1))}$ is defined as $T = [\ T_1 \quad T_2 \quad \dots \quad T_n\ ]$.

In this context, each polynomial can be written as

$$f_i(t, x, x_s) = \sum_{j=1}^{n_r} C(j, i) t^{T(j,4(i-1)+1)} \left(\prod_{p=1}^{n} x_p^{T(j,4(i-1)+p+1)} \right). \tag{4.18}$$

The LF is $W(x) = \int_0^d f(\tau, x, x_s)^T P f(\tau, x, x_s) d\tau$, which, after the corresponding substitutions becomes

$$W(x) = \sum_{k=1}^{n}\sum_{q=1}^{n} P(k,q) \sum_{i=1}^{n_r}\sum_{j=1}^{n_r} C(i,k)C(j,q) \frac{d^{T(i,(n+1)(k-1)+1)+T(j,(n+1)(q-1)+1)+1}}{T(i,(n+1)(k-1)+1)+T(j,(n+1)(q-1)+1)+1} \prod_{p=1}^{n} x_p^{T(i,(n+1)(k-1)+p+1)+T(j,(n+1)(q-1)+p+1)}. \tag{4.19}$$

Remark 4.4.2 To obtain a high accuracy fit for the function f, there are two degrees of freedom in the number of samples in $\mathcal{S}_s$ and the degree of the polynomial f. A high number of samples increases the accuracy of f. However, if the number of samples is too large, we might unnecessarily increase the computational load for an insignificant increase in accuracy. The same observation holds for increasing the degree m of the polynomials. Moreover, the fitting error does not necessarily disrupt the validity of the Lyapunov function W, which is certified via Algorithm 3. □

Then, the validity of the LF (4.19) for system (4.2) can be verified by use of Theorem 4.4.1 and Algorithm 3 with the property function (4.15).

4.5 Sampling–driven DOA estimation

Previous subsections have shown how to construct and validate LFs W for both discrete– and continuous–time systems. The set $\mathcal{A} \cup \mathcal{L}$ computed as in the previous sections is the set on which Lyapunov's inequality holds, and not necessarily a subset of the DOA in the constraint set $\mathcal{S}$. To obtain the proper invariant set $\mathbb{W}$, required in Proposition 4.2.2, which is a subset of the DOA, we need to compute the largest level set of the LF W inside $\mathcal{A} \cup \mathcal{L}$. Computing the largest level set

$$\mathbb{W}^* := \{x \in \mathbb{R}^n : W(x) \leq L^*\} \subseteq (\mathcal{A} \cup \mathcal{L})$$

via nonlinear optimization might suffer from non–convexity issues. This is due to the fact that the set $\mathcal{A} \cup \mathcal{L}$ constructed via Algorithm 3 might be non–convex, because $\mathcal{A}$ is the union of a finite number of hyper–rectangles $\mathcal{B}_{\delta_{x_s}}(x_s)$.

In this section we propose a sampling–driven approach to compute an estimate $\mathbb{W}$ of the DOA by estimating the largest level set of the LF W included in $\mathcal{A} \cup \mathcal{L}$ (for the continuous–time system, included in $\mathcal{A}_c \cup \mathcal{L}_c$). On such a set $\mathbb{W}$, $\mathcal{KL}$–stability is guaranteed and therefore, $\mathbb{W}$ is an invariant subset and an approximation of the DOA of the origin. Note that the computation of $\mathbb{W}$ does not involve the system dynamics, hence the same procedure applies to a LF verified for a discrete–time dynamics or a continuous–time dynamics.

We propose computing an estimation of L^* by a value $\overline{L}$, via sampling. This method assumes that the following necessary condition is satisfied:

$$\exists L > 0 : \mathcal{L} \subseteq \{x \in \mathbb{R}^n : W(x) \leq L\} \subseteq \mathcal{A} \cup \mathcal{L}.$$

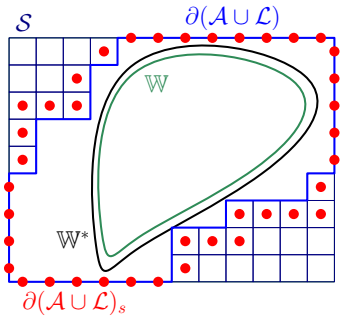


Figure 4.2: Illustration of instrumental sets for Algorithm 4.

The method firstly builds an over–approximation of the boundary of $\mathcal{A}\cup\mathcal{L}$, i.e., $\partial(\mathcal{A}\cup\mathcal{L})$, via hyper–rectangles and iterative refinements. Secondly, it builds an under–approximate of the largest level set of W that does not exceed the boundary of $\mathcal{A}\cup\mathcal{L}$. In Figure 4.2, the set $\partial(\mathcal{A}\cup\mathcal{L})$ is depicted with bold blue. The red dots represent the set of samples in the Δ–sampling of $\partial(\mathcal{A}\cup\mathcal{L})$, i.e., $\partial(\mathcal{A}\cup\mathcal{L})_s$. The largest level set of W not exceeding $\partial(\mathcal{A}\cup\mathcal{L})$, i.e., $\mathbb{W}^*$ and the approximation $\mathbb{W}$ of $\mathbb{W}^*$, as computed via Algorithm 4 have black, respectively green boundary. In Algorithm 4, the inputs are the following. The vector $good$, as computed via Algorithm 3, which contains the set of samples that satisfy (3.3). A vector $extrabound$ containing all points that do not satisfy (3.3) in Algorithm 3 at the end of the multi–resolution process and do not belong to the set $\mathcal{L}$. Formally, this includes all the points $wrong(i)$, for all $i = 1 : k$ such that $wrong(i).del \leq \delta_{min}$, $wrong(i).spoint \notin \mathcal{L}$, and the points generated via a Δ–sampling $\partial\mathcal{S}_s$ of the set $\partial\mathcal{S}$, where Δ has values which are small enough to obtain a fine sampling of the boundary of $\mathcal{S}$. Notice (as in Figure 4.2) that the hyper–rectangles around the points in $\partial\mathcal{S}_s$ are $n-1$ dimensional hyper–rectangles, because the hyper–plane on which the current sampling point lies is no longer considered.

In Algorithm 4, the steps 1–6 are concerned with computing the set of samples which, together with their corresponding hyper–rectangles, over–approximate the set $\partial(\mathcal{A}\cup\mathcal{L})$. In steps 7–10, the estimation of L^* by a value $\overline{L}$ is performed. For each sample $x_s \in \partial(\mathcal{A}\cup\mathcal{L})_s$, the minimum level set of W intersecting $\mathcal{B}_{\delta_{x_s}}(x_s)$ is $\mathbb{W}^*_{x_s} := \{x \in \mathbb{R}^n : W(x) \leq L^*_{x_s}\}$, where:

$$\begin{aligned} L^*_{x_s} = &\min_{x,c} c \\ &\text{s.t. } x \in \mathcal{B}_{\delta_{x_s}}(x_s), \\ &W(x) = c. \end{aligned} \tag{4.20}$$

However, to find $L^*_{x_s}$ as in (4.20), an optimization problem has to be solved, which might not be practical. For this reason we approximate $L^*_{x_s}$ with a value $\overline{L}_{x_s}$, as in Remark 4.5.1.

Remark 4.5.1 If we set $F(x) := W(x)$ then we can find, as in (3.17), via interval analysis, the parameters a_{x_s} and b_{x_s} such that

$$|W(x) - W(x_s)| \leq a_{x_s}\|x - x_s\| + b_{x_s} \tag{4.21}$$

Algorithm 4 Sampling–driven estimation of a domain of attraction $\mathbb{W}$.

Input: $extrabound$, $good$, W, $\mathcal{A}$, $\mathcal{L}$
Output: $\mathbb{W}$

1: $\partial(\mathcal{A} \cup \mathcal{L})_s \leftarrow \varnothing$,
2: **for all** $i = 1 : length(extrabound)$ **do**
3: **for all** $j = 1 : length(good)$ **do**
4: $del \leftarrow |extrabound(i).spoint - good(j).spoint|$
5: **if** $del \leq extrabound(i).tau + good(j).tau$ **then**
6: $\partial(\mathcal{A} \cup \mathcal{L})_s \leftarrow \partial(\mathcal{A} \cup \mathcal{L})_s \cup \{extrabound(i).spoint\}$
7: **for all** $x_s \in \partial(\mathcal{A} \cup \mathcal{L})_s$ **do**
8: Compute a_{x_s}, b_{x_s}, s.t. $|W(x) - W(x_s)| \leq a_{x_s}\|x - x_s\| + b_{x_s}$
9: $\bar{L}_{x_s} \leftarrow W(x_s) - a_{x_s} \max(|\delta_{x_s}|) - b_{x_s}$
10: $\overline{L} = \min_{x_s \in \partial(\mathcal{A} \cup \mathcal{L})_s} \overline{L}_{x_s}$
11: **if** $\overline{L} > 0$ **then**
12: $\mathbb{W} := \{x \in \mathbb{R}^n : W(x) \leq \overline{L}\}$

for all $x \in \mathcal{B}_{\delta_{x_s}}(x_s)$. For all $x_s^* \in \mathbb{W}_{x_s}^*$, the inequality in (4.21) becomes $W(x_s) - W(x_s^*) \leq a_{x_s} \max(|\delta_{x_s}|) + b_{x_s}$, which implies that $W(x_s^*) = L_{x_s}^* \geq W(x_s) - a_{x_s} \max(|\delta_{x_s}|) - b_{x_s}$. Denote $\overline{L}_{x_s} := W(x_s) - a_{x_s} \max(|\delta_{x_s}|) - b_{x_s}$. □

As in steps 11–12, if $\overline{L} > 0$, the set $\mathbb{W} := \{x \in \mathbb{R}^n : W(x) \leq \overline{L}\}$ is an invariant subset of the DOA of the origin for the nonlinear system ((4.1) or (4.2)). If δ_{min} and $\delta_{x_{sb}}$ are small enough, it is expected that $\overline{L}$ is an accurate approximation of L^*, as confirmed by the following illustrative examples.

Remark 4.5.2 (Numerical complexity analysis) As in Algorithm 3 in Chapter 3, the main scalability challenges in applying Algorithm 4 come from the number of selected samples. In this sense, decentralized computation is again beneficial. To assess the computational load of Algorithm 4, let us assume that the computational cost of steps 8–9 for one sampling point is $c \in \mathbb{R}_+$. The number of samples on the boundary of the set $\mathcal{A}$, which results from steps 1–6, is n_s. Also, assume that there exists a number $p \in \mathbb{Z}_+$ of parallel threads. Then, the computational complexity of the for loop at steps 7–12 in Algorithm 4 is of the order $C = c\lceil n_s/p \rceil$. If the number of threads is unlimited, i.e., $p \to \infty$, then $C = c$. Assuming that the sampling density on one dimension is constant and given by $\delta \in \mathbb{R}_+$, and the length in one dimension of an axis of the state space up to the constraint boundary is a constant l, then, the number of samples n_s is $n_s = 2n(n-1)\lceil \frac{l}{2\delta} \rceil$. □

4.6 Reflection on the sampling–driven verification framework

We have shown so far how to verify Lyapunov's inequality for both discrete–time as well as continuous–time systems, and a sampling–driven strategy to compute a DOA estimation for

constrained nonlinear systems has been developed. In this section we reflect on the methods developed so far in terms of tractability, the level of automation, advantages and drawbacks and we put them in the context of the existing literature.

4.6.1 On automating the process of stability domains computation

Coming back to the discussion on the level of automatization of Algorithm 3 from Chapter 3, when applied for verifying stability in a set $\mathcal{S}$, it is necessary to mention the following (we refer, for simplicity, only to the discrete–time systems case):

1. Any equilibrium which is non–zero can be translated to 0, which implies that also the set $\mathcal{S}$ of constraints can be translated in 0, and therefore, $0 \in \text{int}\,(\mathcal{S})$.

2. Any candidate FSLF function V can be chosen as in (4.4), with an "arbitrary" function $\eta \in \mathcal{K}_\infty$.

3. The parameter $\rho \in \mathcal{K}$ can be chosen also arbitrarily such that $\rho < id$ is satisfied. However, in general we choose, for simplicity, $\rho(y) = cy$ for all $y \in \mathbb{R}_+$, with a very large $c \in \mathbb{R}_{(0,1)}$, e.g., $c = 0.99$, to obtain a smaller step M for the FSLF.

4. Once a candidate FSLF V and a $\mathcal{K}$–function ρ are fixed, to obtain $F(x)$, which is the property function to be verified, one needs to fix M as well. At this step we can follow an iterative approach. M can start from 1, and increase up to a value $M_{max} \in \mathbb{Z}_{\geq 1}$. For each M, the property function $F_M(x) = V(G^M(x)) - \rho(V(x))$ is constructed and a set $\mathcal{A}_M$ is computed via Algorithm 3. The value of M which provides the largest set $\mathcal{A}_M$, or the largest level set $\mathbb{W}$ of the corresponding LF, can be chosen, or the bound on M from (Gielen and Lazar, 2015) can be used.

5. The set $\mathcal{N}_1(0)$ is selected such that the hole of the annulus is covered by $\mathcal{L}$. If the set $\mathcal{N}_1(0)$ is too large, then the sampling density should be reduced.

6. The computation of the constant a_{x_s}, b_{x_s} and $\varepsilon(x_s)$ can be done automatically.

Among the points enumerated above, the values of δ_{min} and M are critical for the tractability of Algorithm 3 and Algorithm 4. In what concerns δ_{min}, considering that the increase in the number of samples is exponential with the system dimension, and the fact that performing too many refinements might not increase significantly the size of the set $\mathcal{A}$, and implicitly, $\mathbb{W}$, a trade–off has to be performed between the desired accuracy and the acceptable computational load.

In what concerns M, it has been observed that a large value of M impacts strongly the tractability of Algorithm 3. This is partly because of too many iterations of the map G over intervals, but also because the interval analysis tools over–approximate the computed intervals. Thus, a small value of M_{max} is recommended. Additionally, choosing the candidate V arbitrarily is not always wise, because some functions V might require a large M. In that case it is recommended to learn a LF candidate via, e.g., the simulation based approach in (Kapinski et al., 2014). Even if the function computed in this manner is not a true LF, it might be a FSLF with a very small M. Also, an increase in the state space dimension, n, impacts rapidly the applicability of the interval analysis tool for Algorithm 3.

For Algorithm 4, the inputs $good$, W, $\mathcal{A}$ and $\mathcal{L}$ are by–products of Algorithm 3 from Chapter 3. Once a desired sampling density is given for the set $\partial\mathcal{S}$ to obtain the set $extrabound$, then Algorithm 4 is fully automatic.

The tractability of Algorithm 3 and Algorithm 4 is affected by the fact that we require computing the constants a_{x_s} and b_{x_s}, which embeds the difficulties mentioned above with interval analysis, the constants M and δ_{min} and the system dimension n. For this reason, in Chapter 5 we propose an alternative randomized approach to computing FSLFs, which avoids the requirement to compute a_{x_s} and b_{x_s} and scales better with M.

4.6.2 Comparison with existing methods

Note that hyper–rectangles and multi–resolution together with interval analysis have been used before for DOA computation, see, e.g., (Ratschan and She, 2010). Among the approach therein and the framework presented in this thesis, we distinguish the following features:

- Algorithm 3, due to its generality, is applicable to the verification of any property of the type $F(x) < (\leq)0$, with $x \in \mathcal{S}$. The approach in (Ratschan and She, 2010) is applied for DOA computation. It is recognized, however, in (Ratschan and She, 2010), that the method therein may be applicable to other properties as well, such as, e.g., computation of barrier certificates for safety verification (Prajna and Jadbabaie, 2004), or invariant generation (Sankaranarayanan et al., 2004).

- In what concerns stability analysis, it is known that most nonlinear stability analysis problems are NP–hard. In fact, even for some linear systems, stability analysis is NP–hard see, e.g., (Blondel and Tsitsiklis, 1997), the results on marginally stable linear systems. This thesis treats the NP–hard problem of finding the DOA of nonlinear systems in general. In comparison, the result in (Ratschan and She, 2010) treats only continuous–time polynomial systems with polynomial LFs. In fact, there exist GAS systems with polynomial vector field which do not admit polynomial LFs, see (Ahmadi et al., 2011), in which case, the approach in (Ratschan and She, 2010) suffers from conservatism. However, for polynomial systems which admit polynomial LFs, the method in (Ratschan and She, 2010) is able to compute a DOA without the computational load associated with a potentially large step M, as in this thesis.

- The choice of a candidate polynomial LF in (Ratschan and She, 2010) is still a user–controlled heuristic process, while in this thesis we show that, at least for exponentially stable systems, any candidate FSLF V defined as in (4.4) is a valid FSLF with a specific step M as in (Gielen and Lazar, 2015), which, in turn, generates a true LF automatically.

- The approach in (Ratschan and She, 2010) is provided for continuous–time systems only, while in this thesis we treat both continuous–time, as well as discrete–time systems.

The work in this thesis for stability domains computation is a fusion of the sampling–driven Algorithm 3 and FSLFs. With respect to the work presented in (Kapinski et al., 2014) we distinguish the following:

- The approach in (Kapinski et al., 2014) restricts the candidate LFs to the class of SOS polynomials. As mentioned above, there exist GAS systems with polynomial vector field which do not admit polynomial LFs. Thus, when a candidate LF can not be verified, there is no alternative presented therein to continue the search for a true LF. In contrast, in our method we rely on FSLFs to compute LFs, in which case the LF can incorporate nonlinearities which are inherent to the considered nonlinear system. Thus, if a candidate FSLF is not a LF, we need to increase M until the chosen function is a certified FSLF, which in turn provides a true LF as in (4.6). This process is not straightforward for continuous–time systems, though. In this case, any of the methods in Section 4.4 can be used, however, without guarantees for success, due to approximations in either time discretization of the original continuous–time system, or the fact that the solution at time d, i.e., $x(d)$, can not be analytically computed, but only numerically.
- Simulations are used in (Kapinski et al., 2014) together with LP, to generate refined candidate LFs. The algorithm therein requires also a falsifier to search for simulations which do not satisfy Lyapunov's inequality. This falsifier is a non–convex, global optimizer. In our method no optimization is required to validate the end result of Algorithm 3.
- To validate the results of the simulation–guided Lyapunov analysis techniques in (Kapinski et al., 2014), SMT solvers are used. Instead, in our algorithms we use interval analysis. Additionally, by the multi–resolution sampling technique in this thesis, we are able to find a set $\mathcal{A}$ inside the set $\mathcal{S}$, even when not all the points in the set $\mathcal{S}$ satisfy the desired inequality, as discussed in Chapter 1, Figure 1.4. This is not the case in (Kapinski et al., 2014). However, the approach in (Kapinski et al., 2014) could be modified to follow a multi–resolution approach, as in this thesis.

4.7 Examples

This section presents examples for the proposed sampling–driven stability domains computation method on several benchmark nonlinear systems from the literature. For simplicity, we used `fmincon` where local problems needed to be solved; under local Lipschitz continuity, global optimum can be attained, see (Bobiti and Lazar, 2014b). Other methods for local verification, for example, randomized methods (as we will show in the next chapter), can be used. The computations have been performed in Matlab on a Windows PC with processor Intel Core i7–3770 CPU 3.40 GHz.

4.7.1 Discrete–time 2D continuous dynamics

This example illustrates the methodology developed so far for a 2D discrete–time system. Consider the system provided in (Giesl, 2007):

$$x^+ := G(x), \tag{4.22}$$

where $x \in \mathcal{S} \subset \mathbb{R}^2$ with $\mathcal{S} := \{x \in \mathbb{R}^2 : |x_1| \leq 1, |x_2| \leq 1.3\}$, and

$$G(x) := \begin{bmatrix} \frac{1}{2}x_1 + x_1^2 - x_2^2 \\ -\frac{1}{2}x_2 + x_1^2 \end{bmatrix}.$$

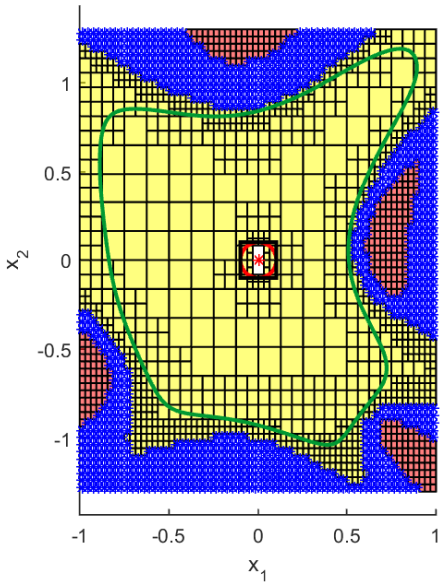


Figure 4.3: DOA for the origin of the 2D system (4.22) with hyper–rectangles.

We will try to find a LF for system (4.22) in the set $\mathcal{S}$.

Select $\delta_{min} = 0.02$ and $M_{max} = 4$. Choose a candidate function $V(x) = x^T P x$ with $P = \begin{bmatrix} 10 & 0 \\ 0 & 1 \end{bmatrix}$, which is two times differentiable, to satisfy Assumption 3.3.9. Fix $M = 4$ as a starting value for M. Fix $F(x)$ as in (4.9), with $\rho = 0.999$.

For step 1 in Algorithm 3 we choose $\mathcal{A}_s = \{0\}$. We sample the set $\mathcal{S}$ by hyper–rectangles such that initially $\mathcal{B}_\delta(0) = \mathcal{S}$, where $\delta = [1 \quad -1 \quad 1.3 \quad -1.3]^T$. We obtain $M = 4$ and the set $\mathcal{A}$ for which $F(x) \leq 0$ is verified according to Algorithm 3 is the yellow set illustrated in Figure 4.3. The white set around the origin is $\mathcal{N}_2(0)$. The red hyper–rectangles are the corresponding hyper–rectangles around sample points x_s for which it was verified via Algorithm 3 that $F(x) > 0$, as in Corollary 3.3.5 in Chapter 3.

Fix the neighborhood $\mathcal{N}_1(0) = \{x \in \mathbb{R}^2 : \|x\|_\infty \leq 0.1\}$. We verify via `fmincon`, in Matlab, that the quadratic LF V_L found for the system linearized in 0 (via `dlyap`, in Matlab) is also a LF for the nonlinear system in $\mathcal{N}_1(0)$. The maximum level set of the LF $V_L(x) = 1.3333 x^T x$, of value 0.0133, which is inside $\mathcal{N}_1(0)$, is an invariant set $\mathcal{L}$. $\mathcal{N}_1(0)$ is illustrated with black boundary, $\mathcal{L}$ with red boundary. The LF, as in (4.6), is $W(x) = \sum_{i=0}^{3} G^i(x)^T P G^i(x)$, and the set $\mathbb{W}$ illustrated with green boundary, computed as in Algorithm 4, where $\overline{L} = 9.2933$, is subset of the DOA of the origin.

With blue we have illustrated the points which satisfy neither (3.3) in Theorem 3.3.4, nor (3.9) in Corollary 3.3.5 after multi–resolution sampling with $\delta_{min} = 0.02$. Convergence to the maximum $\mathcal{A}$ can be achieved for $\delta_{min} \to 0$. Notice in Figure 4.3 also the multi–

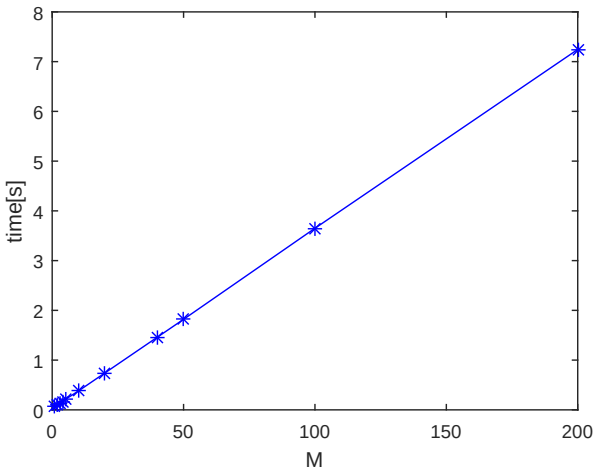


Figure 4.4: Computational time to evaluate $\bar{\gamma}(max|\delta_{x_s}|, x_s)$ at step 5 in Algorithm 3 of Chapter 3 for a given sample point $x_s \in \mathcal{S}_s$ for system (4.22).

resolution sampling of the set $\mathcal{A}$. As expected, a more fine resolution is needed towards the boundary of $\mathcal{A}$, both towards the outer and the inner boundary.

Notice that the set $\mathbb{W}$ obtained with the sampling–guided method presented in this thesis is larger than the DOA estimate obtained in (Giesl, 2007). A total of 1711 sample points have been explored in this example. If we use hyper–cubes instead of hyper–rectangles, as in (Bobiti and Lazar, 2016), we obtain a considerably larger number of sample points. With hyper–rectangles we can explore less sample points, because of the freedom of choosing any rectangle as a sampling unit, which we can maximize in such a manner that we obtain no overlay of the yellow hyper–rectangles, in the construction of $\mathcal{A}$, as opposed to the unnecessary intersections for hyper–cubes, which appear due to imperfect fitting of hyper–cubes in the constraint set $\mathcal{S}$, which is a hyper–rectangle.

In this example, as well as all the subsequent examples, a linear increase of the computational complexity of evaluating $\bar{\gamma}(max|\delta_{x_s}|, x_s)$ at step 5 in Algorithm 3 of Chapter 3 for a given sample point $x_s \in \mathcal{S}_s$ with the value of M has been observed. For instance, in this example, the computational time variation is depicted in Figure 4.4. For $M = 4$, as obtained above, the computational time of evaluating $\bar{\gamma}(max|\delta_{x_s}|, x_s)$ is on average 0.16 seconds. Then, for Algorithm 4, the computational time of steps 8 and 9 is in average 0.23 seconds.

4.7.2 Discrete–time 2D piecewise continuous dynamics

To illustrate the method proposed in this chapter for verification of a piecewise continuous nonlinear system, consider again Example 1 in Chapter 3. Let the search space be

$$\mathcal{S} := \{x \in \mathbb{R}^2 : \|x\|_\infty \leq 1.5\}.$$

Select $\delta_{min} = 0.1$ and $M_{max} = 3$. Choose a candidate function $V(x) = x^T x$ and

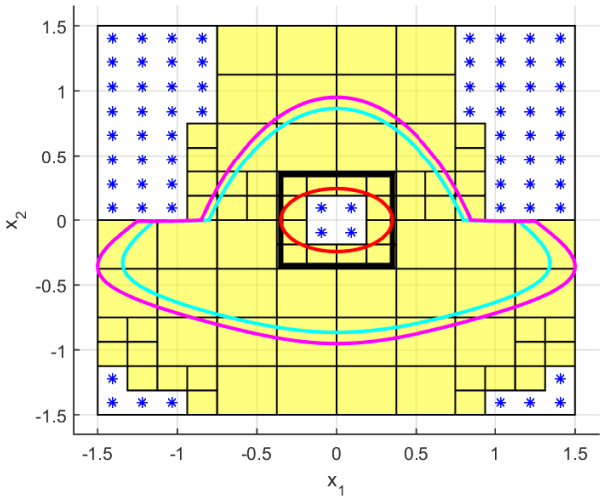


Figure 4.5: DOA of the origin for the piecewise continuous system.

construct $F(x)$ as in (4.9). By applying Algorithm 3 to Example 1, we obtain the results illustrated in Figure 4.5. $\mathcal{A}$ is the yellow set. With blue we have illustrated the points which do not satisfy (3.3). The LF, found as in (4.6), is $W(x) = \sum_{i=0}^{2} G^i(x)^T G^i(x)$. The white set around the origin is $\mathcal{N}_2(0)$.

We linearize G in 0 by linearizing both dynamics G_1 and G_2 in 0. We obtain a switched linear system, for which $V_L(x) = x^T P_L x$ with $P_L = \begin{bmatrix} 26668 & 0 \\ 0 & 55558 \end{bmatrix}$ is a common LF. We choose $\mathcal{N}_1(0) := \{x \in \mathbb{R}^2 : |x_1| \leq 0.35, |x_2| \leq 0.35\}$, and via `fmincon`, in Matlab, we verify that the quadratic LF V_L found for the system linearized in 0 is also a LF for the nonlinear system in the neighborhood $\mathcal{N}_1(0)$. The maximum level set of V_L in $\mathcal{N}_1(0)$ is $L = 3266.8$, which gives the local invariant set $\mathcal{L}$, illustrated with red.

The set $\mathbb{W}$ illustrated with green boundary is the level set of the LF W, computed according to Algorithm 4, and it is subset of the DOA of the origin, with $\overline{L} = 2.3208$. We sampled $\partial\mathcal{S}$ with a distance between samples of value 0.01. Therefore, for a point x_s on the vertical boundary of $\partial\mathcal{S}$, $\delta_{x_s} = [0 \quad 0 \quad 0.01 \quad -0.01]^T$, while for a point x_s on the horizontal boundary of $\partial\mathcal{S}$, $\delta_{x_s} = [0.01 \quad -0.01 \quad 0 \quad 0]^T$. $\overline{L}$ provides an underestimation of the true largest level set of W, of value $L^* \approx 2.805$, depicted in Figure 4.5 with magenta, which is still contained in $\mathcal{A} \cup \mathcal{L}$. The relatively low quality of the estimation of the level set, in this case, is due to the large sampling density.

In this example, in step 1 of Algorithm 3 we choose $\mathcal{A}_s = \{[0 \quad 0]^T\}$, which is on the switching boundary. The basic sampling unit is a hyper–cube. Note, however, that $\varepsilon([0 \quad 0]^T) = 0$. Moreover, by multi–resolution, the samples chosen by the algorithm are not on the switching boundary. Therefore, $\varepsilon(x_s) = 0$ for all $x_s \in \mathcal{S}_s$. In comparison to the results obtained in (Luk, 2015, Example 4), the DOA computed here is larger in set S_1, but

smaller in the set S_2.

The computational time of evaluating $\bar{\gamma}(max|\delta_{x_s}|, x_s)$ is on average 0.09 seconds. The computational time of running Algorithm 3 with the parameters given in this example is 15.72 seconds. Then, for Algorithm 4, the computational time of steps 8 and 9 is in average 0.08 seconds, with a total time for Algorithm 4 of 54 seconds.

4.7.3 Continuous–time 3D continuous dynamics

The following example illustrates the developed methodology on a 3D system, both in discrete–time, and in continuous–time. The system, see (Björnsson et al., 2015), is defined by:

$$\dot{x} := G_c(x) = \begin{bmatrix} x_1(x_1^2 + x_2^2 - 1) - x_2(x_3^2 + 1) \\ x_2(x_1^2 + x_2^2 - 1) + x_1(x_3^2 + 1) \\ 10x_3(x_3^2 - 1) \end{bmatrix}, \tag{4.23}$$

which will be discretized via the Euler method, i.e., $G(x) = x + hG_c(x)$, where $h = 0.1$ is the discretization step. Let the search space be $\mathcal{S} := \{x \in \mathbb{R}^3 : |x_1| \leq 0.9, |x_2| \leq 0.9, |x_3| \leq 0.98\}$.

Consider $\delta_{min} = 0.1$ and $M_{max} = 2$. Choose a candidate FSLF, $V(x) = x^T P x$ with $P = \begin{bmatrix} \frac{1}{0.9^2} & 0 & 0 \\ 0 & \frac{1}{0.9^2} & 0 \\ 0 & 0 & \frac{1}{0.98^2} \end{bmatrix}$. The choice is motivated by the fact that the largest possible ellipsoid that can be contained in the given box $\mathcal{S}$ is a level set of V. Take $M = 2$ and construct $F(x)$ as in (4.9).

Hyper–cubes are used for sampling, and, by applying Algorithm 3, we obtain the following: $M = 2$ and the points in the box $\mathcal{S}$ which do not satisfy (3.3) in Theorem 3.3.4 are illustrated with blue in Figure 4.6.

The computational time for step 5 of Algorithm 3 for a given $x_s \in \mathcal{S}_s$, is of around 0.26 seconds, and for steps 8 and 9 of Algorithm 4, 0.12 seconds.

Fix the neighborhood $\mathcal{N}_1(0) = \{x \in \mathbb{R}^3 : |x_1| \leq 0.6, |x_2| \leq 0.6, |x_3| \leq 0.9\}$. Again, we check via `fmincon`, in Matlab, that the quadratic LF

$$V_L(x) = x^T \begin{bmatrix} 5.5556 & 0 & 0 \\ 0 & 5.5556 & 0 \\ 0 & 0 & 1 \end{bmatrix} x$$

found for the system linearized in 0 (via `dlyap`, in Matlab) is also a LF for the nonlinear system in $\mathcal{N}_1(0)$. The maximum level set of the LF V_L, of value 0.81, which is inside $\mathcal{N}_1(0)$, is an invariant set $\mathcal{L}$, illustrated in Figure 4.6 by the red ellipsoid.

The LF found with this procedure is $W(x) = \sum_{i=0}^{1} G^i(x)^T P G^i(x)$. The set $\mathbb{W}$, computed according to Algorithm 4 and illustrated with green, is the largest estimated level set of W, with $\overline{L} = 1.8459$, which is still contained in $\mathcal{A} \cup \mathcal{L}$. $\mathbb{W}$ is subset of the DOA of the origin.

Moreover, via the approach in Section 4.4.1, we obtain that $\mathbb{W}$ is also a subset of the DOA of the original continuous–time system, i.e., $\mathbb{W}_c = \mathbb{W}$.

It is noticeable that the set $\mathbb{W}$ contains also regions of the state space which are not found in the DOA computed in (Björnsson et al., 2015) for the original continuous–time system,

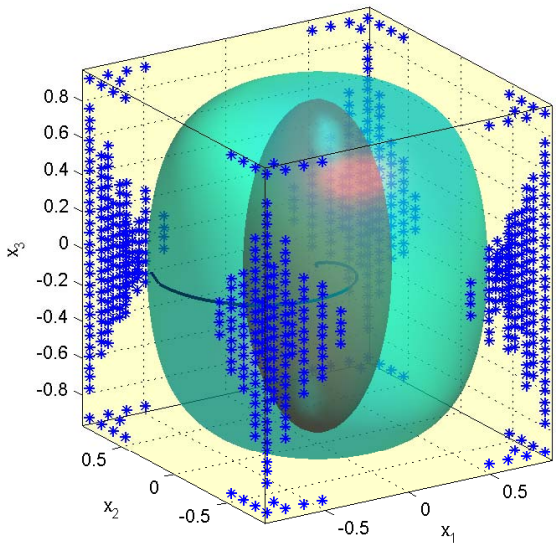


Figure 4.6: DOA for the origin of the 3D system.

see, e.g., the black trajectory illustrated in Figure 4.6, having as initial state one of the points which was not captured in (Björnsson et al., 2015), but which belongs to $\mathbb{W}$.

4.7.4 Powertrain Control System

This example illustrates the potential of the methodology developed in this chapter for computing the DOA of the origin for a system inspired by a real–life application. Consider a 3D simplified version of a Powertrain Control system, inspired by Example 5 of (Kapinski et al., 2014):

$$\dot{x} := G_c(x) = \begin{bmatrix} c_1\left(2u_1\sqrt{\frac{p}{c_{11}} - \left(\frac{p}{c_{11}}\right)^2}\right) - \\ \qquad -c_1(c_3 + c_4c_2p + c_5c_2p^2 + c_6c_2^2p) \\ 4\left(\frac{1}{c_{13}(1+i+c_{14}(r-c_{16}))} - r\right) \\ c_{15}(r - c_{16}) \end{bmatrix}, \tag{4.24}$$

where $x = [p\ r\ i]^T$ is the state vector. Here p is the pressure manifold, r is the air–to–fuel ratio and i is a PI controller, designed to maintain the air–to–fuel ratio within 10% of the optimal value. The corresponding parameters are: $c_1 = 0.41328$, $c_2 = 200$, $c_3 = -0.366$, $c_4 = 0.08979$, $c_5 = -0.0337$, $c_6 = 0.0001$, $u_1 = 16$, $c_{11} = 1$, $c_{13} = 0.9$, $c_{14} = 0.4$, $c_{15} = 0.4$, $c_{16} = 1$. The aim is to compute a set $\mathbb{W}$ where the control system maintains the performance specification of keeping the air–to–fuel ratio within 10% of the optimal value.

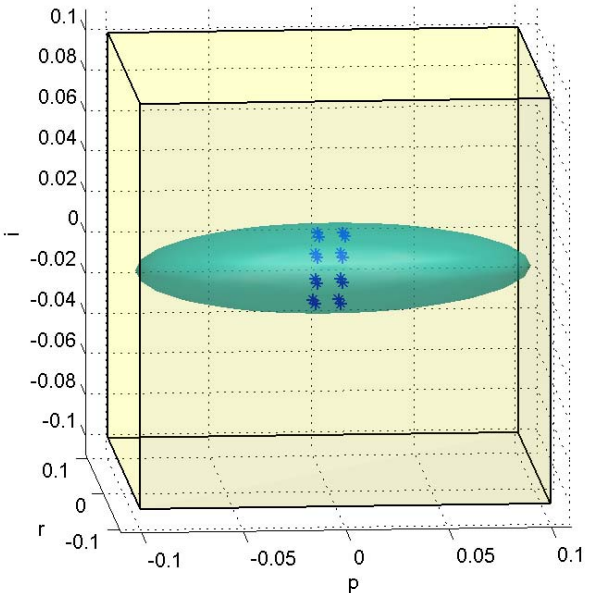


Figure 4.7: DOA for the origin of the Powertrain Control System.

The continuous–time system is again discretized via the Euler method, i.e., $G(x) = x + hG_c(x)$, where $h = 0.01$ is the discretization step. The equilibrium $x_0 = [0.7975\ 1\ 0.1111]^T$ is translated in 0, and we study the $\mathcal{KL}$–stability of 0, and its corresponding DOA.

Let the search space be $\mathcal{S} := \{x \in \mathbb{R}^3 : \|x\|_\infty \leq 0.1\}$. Choose $\delta_{min} = 0.01$, $M_{max} = 3$ and a candidate FSLF given by $V(x) = x^T P x$ with $P = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 4 & 2 \\ 0 & 2 & 14 \end{bmatrix}$, and $M = 3$.

Then, by applying Algorithm 3 with $F(x)$ computed as in (4.9), we obtain the following: $M = 3$, $\mathcal{A}$ is the yellow set illustrated in Figure 4.7, excluding the hyper–rectangles $\mathcal{B}_{\delta_{x_s}}(x_s)$, where x_s are the blue points in the box $\mathcal{S}$ which do not satisfy (3.3) in Theorem 3.3.4. `fmincon` fails to provide a certification for a local LF V_L. Thus we have no invariant set $\mathcal{L}$. Therefore, Theorem 4.3.1 is not used here.

However, we certify $V(G^3(x)) - \rho(V(x)) < 0$ with $\rho = 0.999id$ by optimization in each of the sets $\mathcal{B}_{\delta_{x_s}}(x_s)$ via `fmincon`. Because set $\mathcal{S}$ is the union of all the sets $\mathcal{B}_{\delta_{x_s}}(x_s)$ certified via `fmincon` and the set $\mathcal{A}$ certified via Theorem 3.3.4, then

$$V(G^3(x)) - \rho(V(x)) < 0$$

holds for all $x \in \mathcal{S}$.

The LF we find is $W(x) = \sum_{i=0}^{2} G^i(x)^T P G^i(x)$. The set $\mathbb{W}$ computed via Algorithm 4 is plotted in green color. Here we obtain $\overline{L} = 0.0209$, computed with $\max(|\delta_{x_s}|) = 0.02$

for the points on the boundary of $\mathcal{S}$. $\mathbb{W}$ is subset of the DOA of the origin for the discretized system.

We obtain a computational time for evaluating $\bar{\gamma}(max|\delta_{x_s}|, x_s)$ at step 5 in Algorithm 3 for a given sample point $x_s \in \mathcal{S}_s$, of around 0.29 seconds, and for steps 8 and 9 of Algorithm 4, an average of 0.33 seconds.

Moreover, we obtain that $\mathbb{W}$ is also subset of the DOA of the original continuous–time system, because $\mathcal{A}_c = \mathcal{A}$, which means also that the new level set of W is the same as computed previously for the discretized system, i.e., $\overline{L} = 0.0209$. This fact guarantees that, for any initial condition starting in $\mathbb{W}$, $\|r(t) - x_0(2)\|_\infty \leq 0.1$, for all $t \in \mathbb{R}_+$, which means that, indeed, for any initial condition starting in $\mathbb{W}$, air–to–fuel ratio is maintained within 10% of the optimal value.

4.7.5 Continuous–time systems with polynomial solution approximation

Consider the continuous–time system provided in (Giesl and Hafstein, 2015):

$$\dot{x}(t) := G(x(t)) = \begin{bmatrix} x_2(t) \\ -x_1(t) + \frac{1}{3}x_1(t)^3 - x_2(t) \end{bmatrix}. \tag{4.25}$$

We will try to find a LF and a corresponding DOA of the origin for (4.25), via FTLFs and polynomial approximation of the dynamics, as in Section 4.4.2. Let the search space be

$$\mathcal{S} := \{x \in \mathbb{R}^2 : \|x\|_\infty \leq 1.5\}.$$

We construct an analytical polynomial $f(t, x, x_s)$ of order 2, which is identical for every $x_s \in \mathcal{S}_s$ by selecting a pool of 100 samples uniformly distributed in the set $\mathcal{S}$. Simulations are performed in a time interval $[0, 3]$ starting from each of the 100 samples. The polynomial $f(t, x, x_s)$ is computed via the function `polyfitn` in Matlab. The estimated parameters are saved in the following matrices:

$$C = \begin{pmatrix} 0.0073 & -0.0104 \\ -0.4264 & 0.0139 \\ -0.0153 & -0.4134 \\ -0.0219 & 0.0313 \\ -0.0055 & -0.0021 \\ -0.0072 & -0.0029 \\ 1.0657 & -0.3834 \\ -0.0031 & -0.0013 \\ 0.3913 & 0.6770 \\ 0.0108 & -0.0151 \end{pmatrix}, T_1 = \begin{pmatrix} 2 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 2 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix},$$

where $T = [\ T_1 \quad T_1\ ]$. Select a candidate FTLF $V(x) = x^T x$ and find $d = 3$. A LF $W(x)$ is therefore constructed as in (4.19), with parameters indicated in the matrices C and T.

Let us fix a neighborhood $\mathcal{N}_1(0) = \{x \in \mathbb{R}^2 : \|x\| \leq 0.3\}$. It can be checked via `fmincon`, in Matlab, that the quadratic LF V_L found for the system linearized in 0 (via `dlyap`, in Matlab) is also a LF for the nonlinear system in the neighborhood $\mathcal{N}_1(0)$. The maximum level set of the LF V_L, which is inside $\mathcal{N}_1(0)$, is an invariant set $\mathcal{L}$. By applying Algorithm 3 with $F(x)$ defined as in (4.15), for the LF candidate computed as in (4.19),

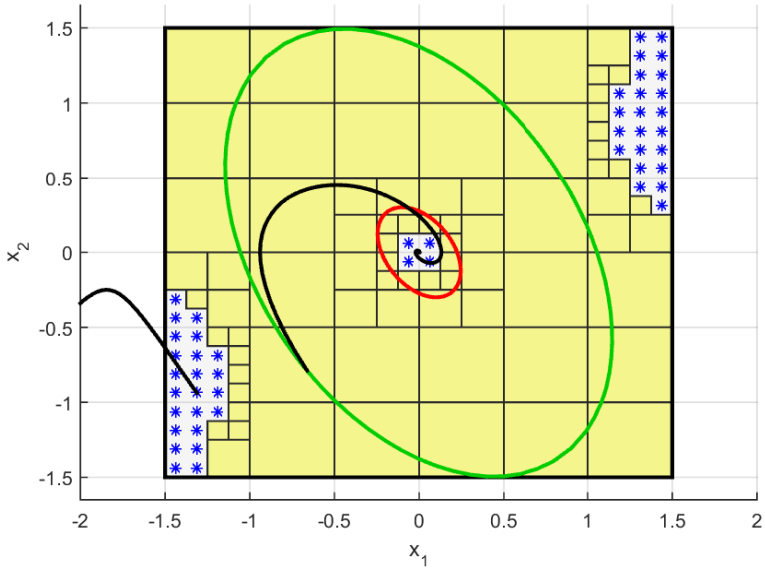


Figure 4.8: DOA for the origin of system (2.22).

we obtain the following: $\mathcal{A}$ is the yellow set illustrated in Figure 4.8, $\mathcal{L}$ is illustrated with red boundary. With blue we have illustrated the points which do not satisfy (3.3) in Theorem 3.3.4 after multi–resolution sampling from $\delta_{max} = 0.5$ to $\delta_{min} = 0.05$.

The LF found with this procedure is as in (4.19) and the set $\mathbb{W}$ illustrated with green boundary, i.e., the largest level set of W (of value 1.5) which is still contained in $\mathcal{A} \cup \mathcal{L}$, is a subset of the DOA of the origin, according to Proposition 4.4.1. Notice in Figure 4.8 the multi–resolution sampling of the set $\mathcal{A}$. The boundary of $\mathcal{A}$ requires a finer resolution. Convergence to the maximum set $\mathcal{A}$ can be achieved if $\delta_{min} \to 0$. Figure 4.8 shows also a trajectory starting from the boundary of $\mathbb{W}$ converging to 0, while the trajectory starting in a blue point exits and never returns to $\mathcal{S}$.

4.8 Conclusions

The sampling–driven strategy developed in Algorithm 3 of Chapter 3 can be adopted for constructing DOAs for nonlinear systems. This is particularly beneficial when difficulties with optimization–based stability verification occur. In this chapter we have illustrated this adaptation for both discrete– and continuous–time systems. Particularly, we have introduced a result which offers a solution to the problem that the LF is zero at the origin. This fact does not allow verification of the inequality $F(x) < (\leq) 0$ at the origin via the inequality $F(x_s) < (\leq) - \overline{\gamma}(\max|\delta_{x_s}|, x_s)$, which needs to be verified for each sample point x_s in Algorithm 3 of Chapter 3.

For discrete–time systems we use a converse result for constructing a LF from a FSLF. For continuous–time systems we have proposed two different alternatives. A first possibil-

ity is to compute a LF for the discretized system and then verify its validity for the original continuous–time system via the sampling–driven Algorithm 3. Alternatively, we can construct approximations of a FTLF via polynomial approximations of the continuous–time dynamics. Then again, from the converse theorem, this FTLF can be exploited to compute a LF, which is verified via Algorithm 3.

A deterministic verification for a LF provides the advantage of a rigorous certificate. However, this comes at a price of conservatism, through the bounds a_{x_s} and b_{x_s}. Besides conservatism, these bounds also reduce significantly the scalability of this deterministic certificate. In the following chapter, a non–deterministic alternative is provided.

Chapter 5

On randomized stability analysis for large scale nonlinear systems

This chapter presents probabilistic strategies to compute DOAs, as an alternative method to the deterministic methods discussed so far based on optimization, in Chapter 2, and sampling–driven, in Chapter 4, to open up the applicability of sampling–driven DOA computation for large–scale nonlinear systems. Based on randomized methods, we verify Lyapunov's inequality at a number of sampling points randomly generated and provide a certificate in terms of reliability, depending on the ratio between the number of points where the property was satisfied and the total number of points tested. However, when the reliability is not maximum, this certificate does not have a practical value and interpretation. For this reason, by exploiting level sets of candidate LFs, we show how to actually construct a set $\mathbb{W}$ where Lyapunov's inequality is satisfied, and which is a subset of the DOA with a given probability. Then, a probabilistic method is provided to iteratively compute a candidate LF based on FSLFs. The methodology is applicable to systems of increasingly large dimensions, as it will be illustrated through examples.

5.1 Introduction

In the theory of systems and control, probabilistic approaches in the form of randomized methods have been used primarily in analyzing robustness, where the uncertainty is a non–deterministic variable, or on systems with uncertain parameters. A vast literature on the matter illustrates the versatility of randomized methods for such topics, see, e.g., the wide exposition in (Tempo et al., 2012) and the survey in (Calafiore et al., 2007) and the references therein. Due to the inherent uncertainty in such systems, randomized methods are accepted.

However, it is less common to treat an originally deterministic problem via a probabilistic approach. When evaluating the question "Is resorting to fate wise?" in (Campi, 2010), and the related discussion therein with respect to possible problems which justify the use of probabilistic methods, we observe the following. It is understood that randomized methods are suitable to increase solvability when any deterministic algorithm fails. Failure or computational intractability of deterministic methods which provide an absolute certificate for stability (a yes or a no) of nonlinear large scale systems is expected, as shown in the

previous chapters. In that case, the requirement for an absolute certificate of stability can be deliberately relaxed to a probabilistic certificate, for the sake of obtaining at least a partial certificate. However, in real–life systems, absolute certificates can rarely be obtained and for this reason, in general, bounds are set on the probability of failure, or on a minimum satisfactory level.

For the problem of stability domains computation for constrained nonlinear systems, a deterministic verification provides the advantage of a rigorous certificate. For systems of small dimensions with safety priorities, the deterministic approach is a suitable choice. However, this comes at a price of conservatism, through the bounds a_{x_s} and b_{x_s} exploited in inequality (3.3) in Chapter 3. Besides conservatism, these bounds also reduce the scalability of the deterministic certificate in Chapter 3. Scalability is affected also by the refinement of hyper–rectangles. Alternatively, in (Najafi et al., 2016) it was shown that sampling can be used to obtain fast computation of DOA. In (Najafi et al., 2016), even though empirically it was concluded that sampling has potential for scalable and high performance DOA computation, no certificate was provided, even in a probabilistic sense.

In this chapter, we turn to the results obtained originally for probabilistic robustness analysis reported in (Tempo et al., 2012), to adapt them for verification of properties of the type $F(x) \leq 0$ for all x in a set $\mathcal{S}$ by verifying $F(x) \leq 0$ only for N samples in a discrete subset $\mathcal{S}_s$ of $\mathcal{S}$, and particularly for DOA computation with probabilistic certificates. Explicit bounds on the sample size N can be computed as in (Tempo et al., 1997) and they depend solely on pre–specified accuracy and confidence requirements.

The contribution of this chapter is in exploiting the existing results in randomized methods to computing DOAs for nonlinear systems. The methods developed in this chapter are constructive and they exploit FSLFs to compute a candidate LF. Level sets are used to iteratively prune out samples which would eventually end up in reducing the reliability of the resulting DOA. An iterative algorithm selects a FSLF which could generate the largest DOA in a probabilistic sense.

The resulting mechanism generates promising results for constrained nonlinear systems of large dimensions, as examples illustrate. The pitfalls and points of precaution in these methods are also exposed through examples.

5.2 Preliminaries on randomized certification of property functions

To perform certification of $F(x) \leq 0$ for $F(x) : \mathbb{R}^n \to \mathbb{R}$ all $x \in \mathcal{S}$, in a probabilistic manner, we follow the problem setting in (Tempo et al., 2012). Let us recall from classic probability theory that a collection of observations $\{x^{(1)}, \ldots, x^{(N)}\}$ with $N \in \mathbb{Z}_+$ of a random vector x is independent and identically distributed (i.i.d.) if each sample is drawn from the same probability distribution as the others, and all are mutually independent, i.e., the probability of observing two samples $x^{(1)}$ and $x^{(2)}$ is the probability of observing $x^{(1)}$ multiplied by the probability to observe $x^{(2)}$. In this chapter we use a uniform distribution for the selection of samples. Other distributions may be used, such as the beta distribution, as in, e.g., (Calafiore, 2010). However, the approach therein is only applicable for convex functions $F(x)$, e.g., when estimating the level set of a LF which is convex, thus, not a LF constructed via FSLFs.

Define the probability of performance, or reliability, as:

$$R = Pr\left\{F(x) \leq 0\right\}. \tag{5.1}$$

Note that R is a deterministic value and if $R = 1$, then $F(x) \leq 0$ holds for all $x \in \mathcal{S}$. As explained in (Tempo et al., 2012, page 93), the exact evaluation of this integral requires computation of the solution of multiple integrals, which is difficult to achieve. Therefore, R is typically estimated by the empirical reliability $\hat{R}_N$, as follows.

Problem 5.2.1 Assume R is the unknown reliability. Fix a desired accuracy $\varepsilon \in \mathbb{R}_{(0,1)}$ and a confidence $\delta \in \mathbb{R}_{(0,1)}$. Compute an estimate of R, i.e., $\hat{R}_N$, such that

$$Pr\left\{|R - \hat{R}_N| \leq \varepsilon\right\} \geq 1 - \delta.$$

Problem 5.2.1 is solved via a Monte Carlo experiment, with an experiment size N of at least the size of the additive Chernoff bound (Chernoff, 1952):

$$N \geq N_{ch} := \left\lceil \frac{\log \frac{2}{\delta}}{2\varepsilon^2} \right\rceil. \tag{5.2}$$

The Monte Carlo experiment, see (Tempo et al., 2012, Chapter 7.1) for details, comprises of the following steps:

1. Draw $N \in \mathbb{Z}_+$ i.i.d. samples of the random vector x with uniform distribution within the set $\mathcal{S}$: $x^{(1)}, \ldots, x^{(N)}$. Evaluate $F(x^{(1)}), \ldots, F(x^{(N)})$.

2. Construct the empirical reliability

$$\hat{R}_N = \frac{1}{N} \sum_{i=1}^{N} I(F(x^{(i)})),$$

where $I : \mathbb{R} \to \{0, 1\}$ denotes the indicator function:

$$I(F(x^{(i)})) = \begin{cases} 1 & \text{if } F(x^{(i)}) < 0 \\ 0 & \text{otherwise,} \end{cases}$$

for all $i \in \mathbb{Z}_{[1,N]}$. Notice that $\hat{R}_N = \frac{N_{good}}{N}$, where N_{good} is the number of samples from the N samples drawn in 1., which did satisfy $F(x^{(i)}) \leq 0$, and $\hat{R}_N$ is a random variable.

Remark 5.2.2 If the set $\mathcal{S}$ on which we draw samples is a hyper–rectangle describing the constraints on the system states, we can draw N i.i.d. samples of x in the set $\mathcal{S}$, via the `rand` function in Matlab, as follows: $x_i^{(j)} = a_i + (b_i - a_i) * rand$, where a_i and b_i are upper and lower bounds for the interval on which x_i may take values on the i–th axis, intervals which can be retrieved from the set $\mathcal{S}$, with $i \in \mathbb{Z}_{[1,n]}$ and $j \in \mathbb{Z}_{[1,N]}$. This method is a valid strategy to obtain uniformly distributed random samples. However, if the considered application requires high accuracy, we refer the reader to (Tempo et al., 2012, Chapter 14) for alternative Random Number Generators. □

This randomized sampling–driven verification method does not suffer from the "curse of dimensionality" as the deterministic counterpart, which means that the method is more scalable with the system dimension, see the examples in Section 5.6. Nevertheless, for high accuracy one might need to draw a very large number of samples, as shown in the next example.

Example 2 Consider again the continuous–time system example in Chapter 4.7.5. Suppose we want to provide a probabilistic certificate for the satisfaction of $\dot{W}(x) < 0$ in $\mathcal{S} \setminus \{0\}$, with the function W computed in Chapter 4.7.5, by performing a Monte Carlo experiment as mentioned previously. Let us first compute the size N of the Monte Carlo experiment such that $|R - \hat{R}_N| \leq \varepsilon$ holds with probability at least $1 - \delta$, where the accuracy is fixed to $\varepsilon = 0.5\%$ and the confidence is $\delta = 1.5\%$, which gives a probability $1 - \delta = 98.5\%$. According to the additive Chernoff bound, $N_{ch} = 42499$. After performing the Monte Carlo experiment with $N = 42499$, we obtain $N_{good} = N = 42432$, which gives $\hat{R}_N = 0.9984$. Thus, by an experiment with $N = 42499$ samples uniformly distributed in $\mathcal{S} \setminus \{0\}$ we have proven with an accuracy of 0.5% that 99.84% of all the points in the set $\mathcal{S} \setminus \{0\}$ satisfy the Lyapunov inequality $\dot{W}(x) < 0$ with a probability of 98.5%. Notice that, if we repeat the Monte Carlo experiment, this time on the set $\mathcal{N}_1(0)$ (the neighborhood of the origin as defined in Chapter 4.7.5), then we obtain $\hat{R}_N = 1$, with the corresponding accuracy and probability. □

For verifying stability with a given LF W, if we want to find a subset $\mathbb{W}$ of the set $\mathcal{S}$ where the inequality $W(G(x)) - W(x) \leq 0$ holds ($\hat{R}_N = 1$), and with significantly lower sample size, then we might need to resort to other methods, with lower bounds on the number of samples, as shown in the next subsection. Moreover, to design constructive methods of obtaining DOA estimations $\mathbb{W} \subseteq \mathcal{S}$, the inherent properties of a stability verification problem have to be exploited.

In the following section we will show that the level set of the candidate LF can be used to shape the set $\mathbb{W}$. The foundation for this result lays in the results in (Tempo et al., 1997) and (Calafiore et al., 2007, Algorithm 2) on randomized worst–case performance evaluation.

As defined in (Calafiore et al., 2007), the problem of worst–case performance evaluation is the following. Let $p^*, \delta \in \mathbb{R}_{(0,1)}$ be assigned probability levels. The randomized algorithm should estimate a performance level γ_N such that

$$Pr\{Pr\{F(x) \leq \gamma_N\} \geq p^*\} \geq 1 - \delta, \tag{5.3}$$

for all $x \in \mathcal{S}$. This means that $F(x) \leq \gamma_N$ with a probability at least p^*, and this event has a probability of occurence of at least $1 - \delta$. The value of γ_N can be estimated by maximizing $F(x^{(i)})$ over all the samples $x^{(i)}$ with $i \in \mathbb{Z}_{[1,N]}$. The corresponding lower bound on N is now

$$N \geq N_{wc} := \left\lceil \frac{ln\frac{1}{\delta}}{ln\frac{1}{p^*}} \right\rceil, \tag{5.4}$$

which gives considerably less sample complexity than the Chernoff bound. For instance, if we write $p^* = 1 - \varepsilon$, with $\varepsilon = \delta = 0.005$ we obtain a bound of 1058 with (5.4) and 119830

with (5.2). Also, with $\varepsilon = \delta = 0.001$, the sample bound is 6905 with (5.4) and 3800500 with (5.2).

In the next section we will show how this setting, formulated in general for any function $F(x)$, which is not particularly related to a dynamical system, can be instrumentally used in a framework for DOA estimation with probabilistic guarantees.

5.3 Constructive randomized algorithm for verifying Lyapunov's inequality and finding DOA with a low number of samples

Consider again, as in the previous chapter, the autonomous nonlinear system in discrete–time

$$x_{k+1} = G(x_k), \quad k \in \mathbb{Z}_+, \tag{5.5}$$

and in continuous–time

$$\dot{x}(t) = G_c(x(t)), \quad t \in \mathbb{R}_+, \tag{5.6}$$

where $x_k \in \mathcal{S}$ (resp. $x(t) \in \mathcal{S}$) is the state, $\mathcal{S}$ is a proper set denoting a subset of the set of constraints and $G : \mathbb{R}^n \to \mathbb{R}^n$, $G_c : \mathbb{R}^n \to \mathbb{R}^n$ are nonlinear functions.

Assume a LF candidate, $W : \mathbb{R}^n \to \mathbb{R}_+$, is given. For the discrete–time system (5.5) we aim to verify

$$F(x) = W(G(x)) - \rho(W(x)) \leq 0, \quad \forall x \in \mathcal{S}, \tag{5.7}$$

where $\rho < id$ is a $\mathcal{K}$ function, while for the continuous–time system (5.6) we verify

$$F(x) = \dot{W}(x) < 0, \quad \forall x \in \mathcal{S} \setminus \{0\}. \tag{5.8}$$

For simplicity of exposition, we develop the methods in this chapter for discrete–time systems. For continuous–time systems we recommend a similar approach. In this section, we aim at solving the problem of randomized stability analysis and estimation of a DOA, i.e.:

Problem 5.3.1 Fix a desired accuracy $p^* \in \mathbb{R}_{(0,1)}$ and a confidence $\delta \in \mathbb{R}_{(0,1)}$. Construct a subset $\mathbb{W}$ of $\mathcal{S}$, such that for $x \in \mathbb{W}$:

$$Pr\{Pr\{F(x) \leq 0\} \geq p^*\} \geq 1 - \delta, \tag{5.9}$$

where $F(x)$ is given by (5.7). □

The aim of Problem 5.3.1 is to obtain the set $\mathbb{W}$ as a candidate invariant set and subset of the DOA of the origin. Notice that Problem 5.3.1 is similar to Problem 3.3.3 in Chapter 3, for the case when $F(x)$ is (5.7). However, in Problem 5.3.1, we aim at constructing a set $\mathbb{W}$ which is a subset of the DOA with given probability levels, while Problem 3.3.3 requires an absolute guarantee that $\mathbb{W}$ is a subset of the DOA.

We develop a solution to Problem 5.3.1 via the iterative steps in Algorithm 5. In there, a high accuracy $p^* \in \mathbb{R}_{(0,1)}$ and a small confidence constant $\delta \in \mathbb{R}_{(0,1)}$, are given. The

sample size N is computed according to the bound in (5.4), to ensure probabilistic guarantees. Other inputs to Algorithm 5 are the candidate LF W, the system dynamics, i.e., G, or G_c, for discrete–time or continuous–time systems respectively. The set $\mathcal{S}$ is given, for simplicity, as a polytope:

$$\mathcal{S} := \{x \in \mathbb{R}^n : Hx \leq K\},$$

with $H \in \mathbb{R}^{m \times n}$ and $K \in \mathbb{R}^m$, where $m \in \mathbb{Z}_{\geq n+1}$. Also, μ and τ are defined as very small constants, of which μ is used to select points on the boundary of the set $\mathcal{S}$, and τ is used to reduce slightly the current level set of the LF, i.e., c. This is necessary because

Algorithm 5 Randomized estimation of the set $\mathbb{W}$ for Problem 5.3.1.

Input: $p^* \in \mathbb{R}_{(0,1)}$, $\delta \in \mathbb{R}_{(0,1)}$, W, G (or G_c), $\mathcal{S} := \{x \in \mathbb{R}^n : Hx \leq K\}$, $\mu, \tau \in \mathbb{R}_+$.
Output: $\mathbb{W}$

1: $\mathcal{S}_0 \leftarrow \mathcal{S}$, $iter \leftarrow 1$, $verif \leftarrow 0$, $N \leftarrow N_{wc} \leftarrow \frac{ln\frac{1}{\delta}}{ln\frac{1}{p^*}}$
2: **while** $verif == 0$ **do**
3: $k \leftarrow 0$, $U \leftarrow \emptyset$, $J \leftarrow \emptyset$
4: Select N uniformly distributed samples in $\mathcal{S}_0$: $\mathcal{S}_{s0} := \{x^{(1)}, \ldots, x^{(N)}\} \subset \mathcal{S}_0$
5: **for all** $i = 1 : N$ **do**
6: **if** $F(x) > 0$ **then**
7: $k \leftarrow k + 1$
8: $U \leftarrow U \cup \{x^{(i)}\}$
9: **else**
10: $J \leftarrow J \cup \{x^{(i)}\}$
11: **if** $iter == 1$ **then**
12: $P \leftarrow \{x \in J : \min(|Hx - K|) \leq \mu\}$
13: $iter \leftarrow 0$
14: **else**
15: $P \leftarrow \emptyset$
16: **if** $k == 0$ **then**
17: $verif \leftarrow 1$
18: **if** $verif == 0$ **then**
19: $c \leftarrow \min_{x \in U \cup P} W(x)$
20: $c \leftarrow c - \tau$
21: $\mathcal{S}_0 := \{x \in \mathcal{S} : W(x) \leq c\}$
22: $\mathbb{W} \leftarrow \mathcal{S}_0$

the level set computed according to step 19, is possibly evaluated in a point $x \in U$, i.e., a point which does not satisfy $F(x) \leq 0$. After the selection of N samples within the set $\mathcal{S}_0$, we store these samples $x^{(i)}$ for all $i \in \mathbb{Z}_{[1,N]}$ in two sets, namely U, if $F(x^{(i)}) > 0$ and J if $F(x^{(i)}) \leq 0$, see steps 4–10. Additionally, at the first iteration, we collect in a set P samples from J which lay on the boundary of the set $\mathcal{S}$, according to steps 11–15 in

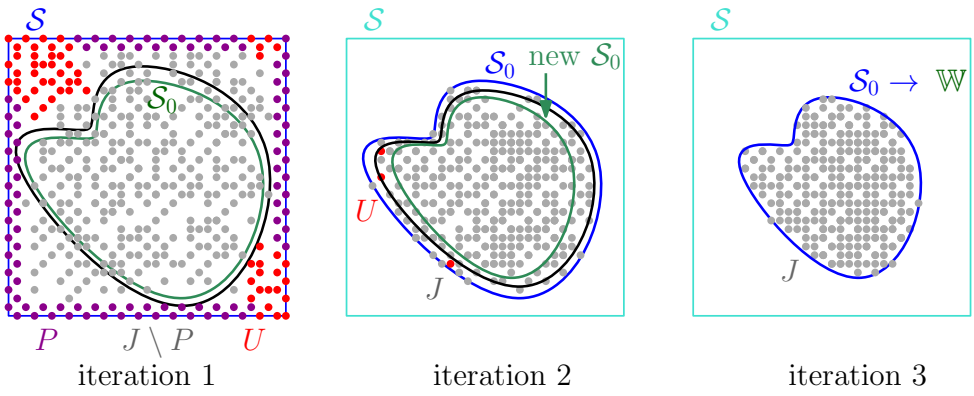


Figure 5.1: Illustration of the construction of the sets U, J, P, $\mathcal{S}_0$, $\mathbb{W}$ for an example where $\mathbb{W}$ is computed in 3 iterations.

Algorithm 5. Here we use the constant μ to define a vicinity around $\partial\mathcal{S}$ from where we select samples near the boundary of $\mathcal{S}$. Other methods for selecting points x_s on, or near the boundary of $\mathcal{S}$ may be used as well to obtain a set P at step 12. At step 19 we compute the smallest level set which passes through the points in $U \cup P$. The necessity for this step stems from the same principles as exposed earlier, in Chapter 4.5. A candidate DOA, i.e., the set $\mathcal{S}_0$, is computed at each iteration as in step 21. At the last iteration, when all the N sample points in the set $\mathcal{S}_{s0}$ belong to the set J, then, $\mathcal{S}_0$ is the candidate set $\mathbb{W}$ which solves Problem 5.3.1.

In Figure 5.1 we illustrate the computation of the sets U, J, P, $\mathcal{S}_0$, $\mathbb{W}$ for an example where $\mathbb{W}$ is computed in three iterations. Notice, in one iteration, the similarity of the construction of the candidate DOA $\mathcal{S}_0$ with the construction of the DOA with the deterministic approach in Chapter 4, Figure 4.2. Then, a new iteration starts, with other N samples in the new set $\mathcal{S}_0$. If all these samples satisfy the property $F(x) \leq 0$, then, the currently available set $\mathcal{S}_0$ provides a solution to Problem 5.3.1 and the iterations stop. Otherwise the process is repeated by decreasing the level set of the candidate LF W until we find a set $\mathcal{S}_0$ for which all the N samples selected in $\mathcal{S}_{s0}$ satisfy Lyapunov's inequality.

Remark 5.3.2 Given the fact that Problem 5.3.1 is the probabilistic equivalent of Problem 3.3.3 in Chapter 3, there exist similarities also between Algorithm 5 in this chapter and Algorithm 3 in Chapter 3. More exactly, Algorithm 5 is the probabilistic counterpart of Algorithm 3, together with Algorithm 4 in Chapter 4. Similarly to Algorithm 3, in Algorithm 5, for each sample in $\mathcal{S}_{s0}$, the value of F is evaluated in the current sampling point and the available samples are stored in two sets, i.e., one set of samples which do satisfy Lyapunov's inequality, and another set, with points which do not satisfy Lyapunov's inequality. Thus, steps 5–10 of Algorithm 5 are similar to steps 6–17 in Algorithm 3. In Algorithm 3, however, the refinement is performed with respect to the current hyper–rectangle, while in Algorithm 5, the refinement is performed with respect to the level sets of the candidate LF.

Thus, steps 11–21 of Algorithm 5 resemble Algorithm 4, where the purpose of the level set minimization over the set $\partial(\mathcal{A} \cup \mathcal{L})$ at step 10 is to compute the true DOA estimation, i.e., $\mathbb{W}$. In Algorithm 5, a similar minimization is performed to compute a new candidate set $\mathcal{S}_0$, which has to be tested again for satisfaction of Lyapunov's inequality, until (5.9) holds. This was not necessary in Algorithm 4, because therein, the satisfaction of Lyapunov's inequality in the set $\mathbb{W}$ is certain, due to the deterministic certificate obtained in Algorithm 3. □

Remark 5.3.3 Recall, from Example 2 in Section 5.2, that verifying a property $F(x) \leq 0$ with a high confidence and a good accuracy of the reliability in (5.1), as in Problem 5.2.1, requires a very large number of samples given by the Chernoff bound in (5.2). In Algorithm 5 we avoid the costly computation of a reliability by iteratively refining a set $\mathcal{S}_0$ until it becomes highly probable that $F(x) \leq 0$ for all $x \in \mathcal{S}_0$. This problem can thus be fit in the context of a worst–case performance evaluation problem, as in (5.3), rather than the problem of verifying a property $F(x) \leq 0$ with high reliability. Indeed, when all the N samples in the set $\mathcal{S}_0$ give an $F(x) \leq 0$, then we have obtained a bound γ_N on $F(x)$ which satisfies $\gamma_N \leq 0$, and therefore (5.3) holds, which in turn implies that (5.9) holds for $\mathbb{W} = \mathcal{S}_0$. □

As described above, the same procedure can be used for certifying LFs for discrete–time systems as well as continuous time systems. Algorithm 5 can be applied for a given LF W. Now we need to answer the question of how to find a LF. A possible answer is provided via FSLFs, in the next section.

5.4 Finding a Lyapunov function candidate

If a LF W is not known, and we want to also find the LF using a randomized approach, then we can follow the steps we develop in Algorithm 6. The approach in Algorithm 6 relies on finding an M–step LF V and computing a LF W using (4.6), i.e.:

$$W(x) := \sum_{i=0}^{M-1} V(G^i(x)).$$

An arbitrary candidate FSLF V is fixed as in (4.4), i.e., $V(x) := \eta(\|x\|)$ with $\eta \in \mathcal{K}_\infty$. The FSLF V, together with a number of samples N (as large as possible), the discrete–time dynamics G, a polytopic set of constraints $\mathcal{S}$, a maximum desirable step M_{max} and a set $\mathcal{B}_\omega(0)$ are given as inputs to Algorithm 6.

The set $\mathcal{B}_\omega(0)$ with a $\omega \in \mathbb{R}_+$ is a minimum set which we want to cover with a potential LF, i.e., a minimum admissible region of convergence. This set is required to force the candidate LF to be a LF over at least a small set which contains the origin in its interior. The outputs of Algorithm 6 are the step M for the FSLF and the candidate LF W.

Algorithm 6 selects a set $\mathcal{S}_s$ of N samples in $\mathcal{S}$. Then, the algorithm iterates over all the values of M from 1 until M_{max} and, for each value of M, a reliability $R(M)$ is computed at step 13 as the proportion from all the points N of sample points $x^{(i)}$ in $\mathcal{S}_s$ which do satisfy the inequality $F(x^{(i)}) = V(G^M(x^{(i)})) - V(x^{(i)}) \leq 0$, which were counted at step 12.

From all values of M from 1 to M_{max}, select the candidate with the largest reliability, as in step 15. Note that, for the values of M for which there exist sample points $x^{(i)} \in$

$\mathcal{S}_s \cap \mathcal{B}_\omega(0)$ for which the inequality $F(x) \leq 0$ does not hold, the reliability $R(M)$ will be 0, and thus, they will not be selected as suitable steps M for computing the LF candidate W, see steps 8–10 in Algorithm 6. The LF candidate W, computed in step 16, can be used for

Algorithm 6 Randomized computation of a LF candidate.

Input: $N, V, G, \mathcal{S} := \{x \in \mathbb{R}^n : Hx \leq K\}, \mathcal{B}_\omega(0), M_{max}$.
Output: M, W

1: $M \leftarrow 1$
2: Select N uniformly distributed samples in $\mathcal{S}$: $\mathcal{S}_s := \{x^{(1)}, \ldots, x^{(N)}\} \subset \mathcal{S}$
3: **while** $M \leq M_{max}$ **do**
4: $\quad k \leftarrow 0$
5: $\quad$ **for all** $i = 1 : N$ **do**
6: $\quad\quad F(x^{(i)}) \leftarrow V(G^M(x^{(i)})) - V(x^{(i)})$
7: $\quad\quad$ **if** $F(x^{(i)}) > 0$ **then**
8: $\quad\quad\quad$ **if** $x^{(i)} \in \mathcal{B}_\omega(0)$ **then**
9: $\quad\quad\quad\quad k \leftarrow 0$
10: $\quad\quad\quad\quad$ Break the current **for** loop
11: $\quad\quad$ **else**
12: $\quad\quad\quad k \leftarrow k + 1$
13: $\quad R(M) \leftarrow \frac{k}{N}$
14: $\quad M \leftarrow M + 1$
15: $M \leftarrow arg\max_{\tilde{M} \in \mathbb{Z}_{[1, M_{max}]}} \{R(\tilde{M})\}$
16: $W(x) := \sum_{i=0}^{M-1} V(G^i(x))$

the verification program in Algorithm 5. However, from Algorithm 6 we obtain a candidate LF, which can be falsified by Algorithm 5, or which may provide a very small DOA. For this reason, in the next section we propose a merging of Algorithm 5 and Algorithm 6 to obtain from the start a LF and a DOA with probabilistic guarantees.

5.5 Randomized DOA estimation

The algorithms in Subsection 5.3 and Subsection 5.4 can be combined to simultaneously construct LF and maximize DOAs for nonlinear systems on a compact (polytopic) set $\mathcal{S}$, as formalized in the following problem:

Problem 5.5.1 Fix a desired accuracy $p^* \in \mathbb{R}_{(0,1)}$ and a confidence $\delta \in \mathbb{R}_{(0,1)}$. Construct a candidate LF W and a candidate subset $\mathbb{W} \subseteq \mathcal{S}$ of the DOA of the origin for system (5.5), such that for $x \in \mathbb{W}$:

$$Pr\{Pr\{x \in DOA(0) \cap \mathbb{W}, F(x) \leq 0, G(x) \in \mathbb{W}\} \geq p^*\} \geq 1 - \delta, \tag{5.10}$$

where $F(x)$ is defined as in (5.7). In (5.10) we express the fact that, with the given probability bounds, the function W is a LF and $\mathbb{W}$ is an invariant subset of the DOA of the origin

Algorithm 7 Randomized computation of a LF.

Input: $p^* \in \mathbb{R}_{(0,1)}$, $\delta \in \mathbb{R}_{(0,1)}$, V, G, $\mathcal{S} := \{x \in \mathbb{R}^n : Hx \leq K\}$, $\mathcal{B}_\omega(0)$, M_{max}, $\mu, \tau \in \mathbb{R}_+$, f.

Output: $M, W, \mathbb{W}$

$N \leftarrow N_{wc} \leftarrow \frac{ln\frac{1}{\delta}}{ln\frac{1}{p^*}}$, $L \leftarrow f\mathbf{1}_{M_{max}}$, $R \leftarrow \mathbf{0}_{M_{max}}$, $verif \leftarrow 0$, $M \leftarrow 1$
Select N uniformly distributed samples in $\mathcal{S}$: $\mathcal{S}_s := \{x^{(1)}, \ldots, x^{(N)}\} \subset \mathcal{S}$
while $M \leq M_{max}$ **do**
    $k \leftarrow 0$, $q \leftarrow 0$, $U \leftarrow \emptyset$, $J \leftarrow \emptyset$, $W_M(x) := \sum_{j=1}^{M-1} V(G^j(x))$
    for all $i = 1 : N$ **do**
        $F(x^{(i)}) \leftarrow W_M(G(x^{(i)})) - W_M(x^{(i)})$
        if $F(x^{(i)}) > 0$ **then**
            if $x^{(i)} \in \mathcal{B}_\omega(0)$ **then**
                $k \leftarrow 0$
                Break the current for loop
            $L(M) \leftarrow \min\{L(M), W_M(x^{(i)})\}$
        else
            $k \leftarrow k + 1$, $J \leftarrow J \cup \{x^{(i)}\}$
            if $\min(|Hx^{(i)} - K|) \leq \mu$ **then**
                $L(M) \leftarrow \min\{L(M), W_M(x^{(i)})\}$
    $L(M) \leftarrow L(M) - \tau$
    for all $x^{(j)} \in J, j = 1 : k$ **do**
        if $W_M(x^{(j)}) \leq L(M)$ **then**
            $q \leftarrow q + 1$
    $R(M) \leftarrow \frac{q}{N}$, $M \leftarrow M + 1$
$M \leftarrow arg\max_{\tilde{M} \in \mathbb{Z}_{[1, M_{max}]}} \{R(\tilde{M})\}$,
$W(x) := \sum_{i=0}^{M-1} V(G^i(x))$, $\mathcal{S}_0 := \{x \in \mathcal{S} : W(x) \leq L(M)\}$
while $verif == 0$ **do**
    $k \leftarrow 0$, $U \leftarrow \emptyset$, $J \leftarrow \emptyset$
    Select N uniformly distributed samples in $\mathcal{S}_0$: $\mathcal{S}_{s0} := \{x^{(1)}, \ldots, x^{(N)}\} \subset \mathcal{S}_0$
    for all $i = 1 : N$ **do**
        if $F(x^i) > 0$ **then**
            $k \leftarrow k + 1$, $U \leftarrow U \cup \{x^{(i)}\}$
        else
            $J \leftarrow J \cup \{x^{(i)}\}$
    if $k == 0$ **then**
        $verif \leftarrow 1$

33: **if** $verif == 0$ **then**
34: $c \leftarrow \min_{x \in U \cup P} W(x)$
35: $c \leftarrow c - \tau$
36: $\mathcal{S}_0 := \{x \in \mathcal{S} : W(x) \leq c\}$
37: $\mathbb{W} \leftarrow \mathcal{S}_0$

for system (5.5). □

Problem 5.5.1 can be solved via Algorithm 7. In steps 1–22, by iterating M from 1 to M_{max}, an M is selected, which provides the highest reliability for the candidate FSLF V. In this algorithm, the reliability $R(M)$ is computed in a manner that the resulting DOA estimation, $\mathcal{S}_0$, would provide the largest possible set. For this reason, $R(M) = \frac{q}{N}$, where q is computed as the total number of samples in $\mathcal{S}_s$ which satisfy $F(x) \leq 0$ and belong to the largest level set of the current LF W_M. The value of M with the largest q provides thus the largest DOA candidate with the available samples. At step 22, a LF candidate exists, with a corresponding set $\mathcal{S}_0$. From step 23 onward, we try to certify that $\mathcal{S}_0$ is the candidate set $\mathbb{W}$, or to refine $\mathcal{S}_0$ until $\mathbb{W}$ is found, which solves Problem 5.5.1. These steps are similar to Algorithm 5. However, this time, the starting search set $\mathcal{S}$ for Algorithm 5 is in fact $\mathcal{S}_0$, which is already inside $\mathcal{S}$, and therefore there is no need to sample the boundary of the set $\mathcal{S}$ anymore.

The main elements which influence the computational complexity for the algorithms proposed in this chapter are:

- The number of samples, N; this is given as in (6.30), and it depends on the parameters p^* and δ, chosen by the user;
- The complexity of selecting a sample point in the current set;
- The complexity of computing $F(x)$.

The other operations are simple computations, such as comparisons, which are fast compared with the operations mentioned above. Computing the minimum from a finite number of values, as in step 19 of Algorithm 5, step 15 in Algorithm 6, or step 34 in Algorithm 7, is not performed for every sample. In the following two remarks we address the last two items mentioned above.

Remark 5.5.2 In the algorithms proposed in this chapter, we assume that one can sample uniformly in a set $\mathcal{S}$, but also in $\mathcal{S}_0$. In the examples we propose in Section 5.6 the set of constraints $\mathcal{S}$ is considered to be a hyper–rectangle, because it is common to have constraints on states to be represented via intervals. Sampling uniformly on hyper–rectangles is simple, as illustrated in Section 5.2. If the set $\mathcal{S}$ is a convex polytope, as required by the algorithms presented in this chapter, the problem of selecting uniform samples in the convex polytope can be approached, e.g., via the method presented in (Rubin, 1984). In this thesis, any set which is not a hyper–rectangle is sampled via rejection methods (Tempo et al., 2012,

pag. 207). In these methods, a set $\mathbb{X}$ is upper–bounded by another set $\mathbb{X}_d$ on which uniform sampling can be performed. However, as mentioned therein, the rejection rate of this method is given by the ratio of volumes: $\frac{Vol(\mathbb{X}_d)}{Vol(\mathbb{X})}$. While any bounded set can be upper–bounded by a hyper–rectangle, the resulting rejection rate might be very large. In this case, we can talk about a "curse of dimensionality" of the rejection methods, as illustrated, e.g., in (Tempo et al., 2012, Table 16.2, pag. 233). Instead of upper–bounding the set with a hyper–rectangle, other, more tight polytopes can be used as in, e.g., (Sijs, 2012, Figure 3.6, pag. 63). However, in this case, a trade–off is necessary between the size of $\mathbb{X}_d$ and the computational complexity of obtaining $\mathbb{X}_d$. Additionally, in (Tempo et al., 2012), alternative efficient methods are presented to sample in sets generated by p–norms. This might be useful when we verify, e.g., quadratic LFs. However, in general, for LF generated via FSLFs, which inherently integrate the system dynamics G, the resulting level set is not given by a norm. An alternative method for random uniform sampling of bounded sets is provided in (Smith, 1984). □

Remark 5.5.3 The computational complexity of computing $F(x)$ is expected to grow linearly with the state space dimension, because the evaluation is performed independently for each state x_j, with $j \in \mathbb{Z}_{1,n}$. This will be illustrated via the example in Section 5.6.2. □

5.6 Examples

This section presents the potential of the algorithms developed in this chapter for systems of increasing dimension. The computations have been performed in Matlab, on a Windows PC with processor Intel Core i7–3770 CPU 3.40 GHz.

5.6.1 2D system

By the help of a 2D system, we illustrate visually the consequences of the algorithms developed in this chapter. Consider again the example in Section 4.7.1.

To Illustrate Algorithm 5, consider the set $\mathcal{S}$ given in Section 4.7.1. The LF candidate is the function W computed therein. To solve Problem 5.3.1, fix a desired accuracy $p^* = 0.999$ and a confidence $\delta = 0.001$. Consider also $\mu = 0.01$ and $\tau = 0.0001$.

Then, we construct a subset $\mathbb{W}$ of $\mathcal{S}$, such that for $x \in \mathbb{W}$, (5.9) holds, with $F(x)$ given as in (5.7). The construction of the set $\mathbb{W}$ is performed via Algorithm 5 with the above parameters and we obtain the results illustrated in Figure 5.2. The sample size with the desired accuracy and confidence is $N = 6904$, and the level computed at step 20 is $c = 11.7212$, which is considerably larger than the one obtained in Chapter 4, i.e., $\overline{L} = 9.2933$. The set $\mathbb{W}$ with $c = 11.7212$ does not exceed the boundary of the set $\mathcal{S}$. The evaluation time for the function $F(x)$ is around $1.5[ms]$.

Algorithm 7, with the same $p^*, \delta, \mu, \tau, \mathcal{S}$ as above, with V as in Section 4.7.1 and with $f = 10^6$ (a very large number), $M_{max} = 10$, provides the following results: $M = 4$, $c = 11.5264$ (close to the previous value), the same W as above and $\mathbb{W}$ as in step 36 of Algorithm 7. The computational times for evaluating $F(x)$ for each value of M, in order, from 1, to M_{max}, are given in Figure 5.3. Notice a reasonable linear increase of the computational time with the value of M. The probabilistic certificate holds provided that

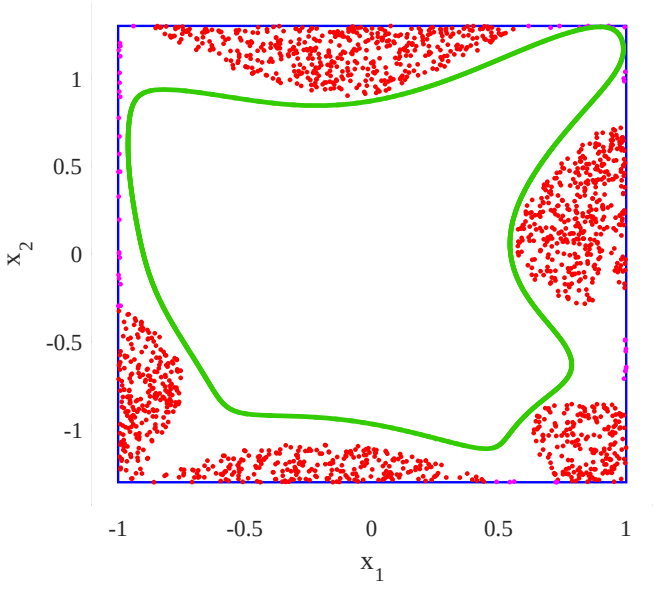


Figure 5.2: DOA for the origin of the 2D system with Algorithm 5: $\mathcal{S}$ (blue boundary), $\mathbb{W}$ (green boundary), and the points $x_s \in P \cup U$ (red and magenta).

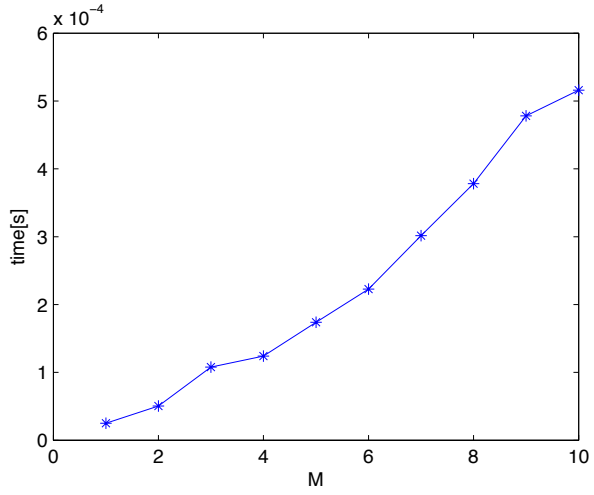


Figure 5.3: Computational time to evaluate $F(x)$ as a function of the value of the step M.

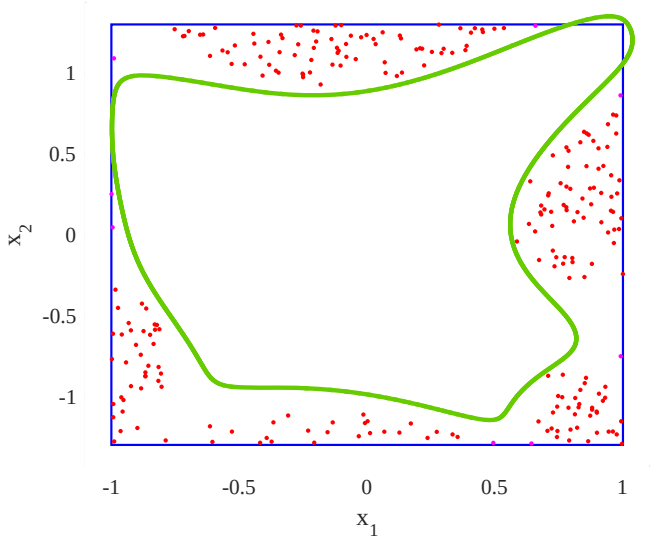


Figure 5.4: A case in which the level set exceeds $\partial\mathcal{S}$: $\mathcal{S}$ (blue boundary), $\mathbb{W}$ (green boundary), and the points $x_s \in P \cup U$ (red and magenta).

the current set can be uniformly sampled. It is important to make the following observation. Notice in Figure 5.4, when the number N of samples is small and there aren't enough samples to represent the boundary, the boundary might be exceeded by the level set which we compute. Also, by the rejection method, we still select the random samples from the whole set $\mathcal{S}$ and then consider only the samples which are inside the current level set, but we ignore samples outside the set $\mathcal{S}$ which are inside the current level set. Therefore we do not know that we have exceeded the set $\mathcal{S}$. In Figure 5.2, with a larger number of samples, we notice that the level set does not exceed the boundary of $\mathcal{S}$.

A possible solution to this problem is to select samples x exactly on the boundary of $\mathcal{S}$. Consider that $\partial\mathcal{S}$ is an $(n-1)$–dimensional set. These samples focus more on the boundary region, and increase the chance of avoiding level sets which exceed the boundary of $\mathcal{S}$.

5.6.2 Cascade cart—spring–damper systems

To illustrate the scalability with the state space dimension of the methods developed in this chapter we consider a system composed of q carts connected as cart–spring–damper systems. This model is relevant, for example, for car platooning, or robots with flexible joints.

The model of a cart with mass M_i, with $i \in \mathbb{Z}_{[1,q]}$, which is moving on a plane is inspired by the model in (Raimondo et al., 2009). The first cart is attached to a wall via a spring with elastic constant k_1 varying with the first state $k_1 = k_0 e^{-x_1}$, where x_1 stands for the displacement of the first cart from the equilibrium position. Thus, the system incorporates exponential nonlinearities. Similarly, x_{2j-1} represents the displacement of the $j-th$ cart

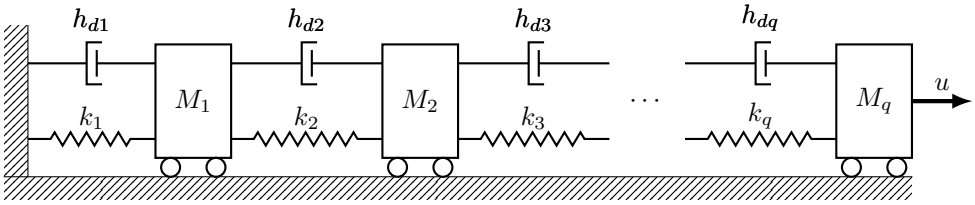


Figure 5.5: Cart–spring damper connected systems.

from its equilibrium position. A damper acts as a resistor in the system, with damping h_{d1}. The other $q-1$ carts are connected to each other, via the same springs and dampers, and the last one is pulled by a force $u(t)$, as illustrated in Figure 5.5. All the carts have identical parameters, i.e., $M_i = M$, $k_i = k$, $h_{di} = h_d$ for all $i \in \mathbb{Z}_{[1,q]}$. The continuous–time nonlinear model of the q cart–spring–damper connected systems is the following:

$$\dot{x}(t) = f(x(t)) + Fu(t), \tag{5.11}$$

where

$$f(x(t)) = \begin{bmatrix} x_2(t) \\ -\frac{\rho_0}{M} e^{-x_1(t)} x_1(t) - \frac{h_d}{M} x_2(k) + \frac{\rho_0}{M} e^{-x_3(t)} x_3(t) + \frac{h_d}{M} x_4(k) \\ \cdots \\ x_{2i}(t) \\ -\frac{\rho_0}{M} e^{-x_{2i-1}(t)} x_{2i-1}(t) - \frac{h_d}{M} x_{2i}(k) + \frac{\rho_0}{M} e^{-x_{2i-1}(t)} x_{2i-1}(t) + \frac{h_d}{M} x_{2i}(k) \\ \cdots \\ x_{2q}(t) \\ -\frac{\rho_0}{M} e^{-x_{2q-1}(t)} x_{2q-1}(t) - \frac{h_d}{M} x_{2q}(k) \end{bmatrix}, \tag{5.12}$$

and

$$F = \begin{bmatrix} 0 & \dots & 0 & \frac{1}{M} \end{bmatrix}^T \in \mathbb{R}^{2q}, \tag{5.13}$$

where x_{2i} is the velocity of the i–th cart and u is an external force which acts as an input to the last cart system. The parameter values are $\rho_0 = 0.33$, $M = 1$, $h_d = 1.1$. The system, discretized with the Euler method is

$$x(k+1) = G(x(k)) = f_1(x(k)) + F_2 u(k), \tag{5.14}$$

where $f_1(x(k)) = x(k) + T_s f(x(k))$ and $F_2 = T_s F$ and $T_s = 0.4s$.

In this example we linearize the system (5.14) in 0 and we compute a linear quadratic regulator (LQR) for the linear system to obtain a controller $u(k) = -Kx(k)$ and a quadratic LF $V(x) = x^T P x$. Knowing that, if the origin is stable, then the LF computed for the linearized system is a LF for the original nonlinear system in a neighborhood of the origin

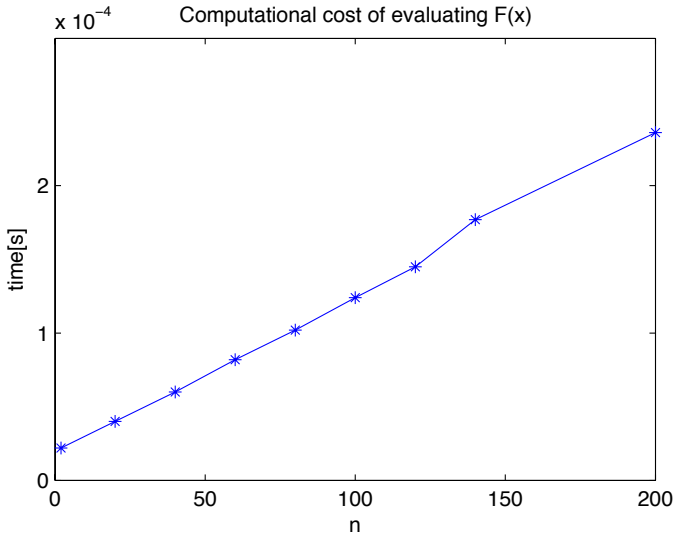


Figure 5.6: Linear increase in the computational time to evaluate $F(x)$ with the number of states n.

(Khalil, 2002), we will try to compute a subset of the DOA of the origin for the nonlinear system via the approach in Algorithm 5, in a constraint set defined by

$$\mathcal{S} := \{x \in \mathbb{R}^{2q} : \|x\|_\infty \leq 1\}.$$

Let us first analyze the average computational time involved in computing $F(x) = V(G(x)) - \rho V(x)$, with V computed above and $\rho = 0.9999$ as a function of the number n of states. Notice that the total number of states is $n = 2q$. The computational time increase with the system dimension is illustrated in Figure 5.6, and it is linear. This plot suggests that, in what concerns the computation of $F(x)$, the methods developed in this chapter are scalable with the system dimension. However, the complexity of sampling uniformly contributes also to the dimensionality problem.

To compute a DOA estimate for a nonlinear system of relatively large dimension, we fix $q = 10$, which gives a state space dimension $n = 20$, and apply Algorithm 5 with $p^* = 0.999$, $\delta = 0.001$, $\mu = 0.01$, $\tau = 0.0001$ and the LF candidate $W = V$. We obtain $L_1 = 40.8705$ after only two iterations of the `while` loop. This shows a relatively fast convergence of the algorithms developed in this chapter in terms of obtaining a DOA candidate even for systems of large dimension ($n > 10$).

5.7 Reflection on randomized methods for DOA estimation

To address the problem of non–scalability with the system dimension in the deterministic method presented in Chapter 4 for DOA computation, in this chapter we have proposed methods for computing candidate LFs and DOAs for nonlinear systems in a constraint set $\mathcal{S}$, with probabilistic guarantees. The developed methods scale linearly with the step M

	Randomized verification: Chapter 5	Deterministic verification: Chapter 4
Scalability with n in computing $F(x)$	linear	exponential
Scalability with N in computing $F(x)$	From experiments: linear	From experiments: linear
Complexity of computing samples	Sampling in a hyper–rectangle $\mathcal{S}$: simple uniform sampling, as in Remark 5.2.2. Sampling in a level set $\mathcal{S}_0$ of a (possibly non–convex) LF W: "curse of dimensionality".	Sampling only in a hyper–rectangle $\mathcal{S}$: simple. Refinement of the sampling: exponential with n_r.
Validity of the stability certificate	probabilistic	deterministic
Other observations	It is necessary that the set $\mathcal{S}_0$ is uniformly sampled, and it does not exceed the boundary of the set $\mathcal{S}$, see Example 5.6.1.	

Table 5.1: Randomized versus deterministic methods for DOA estimation: a complexity assessment.

and the state space dimension n. However, the algorithms developed in this chapter suffer from the "curse of dimensionality" due to the rejection methods employed for sampling in the set $\mathbb{W}$. Also, the validity of the certificate of stability is probabilistic, with accuracy and confidence dictated by the user's requirements. A summary of the findings of this chapter, in comparison to the methods developed in Chapter 4 is found in Table 5.1.

Based on this summary of the results obtained in this chapter we can answer the question Q_4 posed in Chapter 1 by the fact that the scalability issues reported for the deterministic method developed in Chapter 4 can be overcome by the randomized methods presented in Chapter 5 by reducing the exponential increase of the computational complexity of evaluating $F(x)$ for a given point x with a linear increase with respect to n. From this perspective, we can answer positively to the question Q_5 posed in Chapter 1 as well, when a probabilistic certificate for the DOA suffices. Precaution is necessary though when employing rejection methods for sampling in level sets of LFs.

Chapter 6

Sampling–driven nonlinear model predictive control

This chapter develops a new sampling–driven nonlinear model predictive control (SDNMPC) algorithm, with a bound on complexity which is quadratic in the prediction horizon N and linear in the number of samples. The idea of the proposed algorithm is to use the sequence of predicted inputs from the previous time step as a warm start, and to iteratively update this sequence by changing its elements one by one, either backward or forward along the prediction horizon. The backward strategy resembles the dynamic programming (DP) principle, while the forward implementation resembles the rollout algorithms for nonlinear model predictive control (NMPC). Both versions allow for parallelization up to a certain level and yield a suboptimal NMPC algorithm with guaranteed recursive feasibility, stability and improved cost function at every iteration, which is suitable for real–time implementation. The complexity of the algorithm per each time step in the prediction horizon depends only on the horizon, the number of samples and parallel threads, and it is independent of the measured system state. Conditions for convergence of the SDNMPC are discussed in this chapter, as well as recommendations for obtaining a smooth input sequence. Comparisons with the *fmincon* nonlinear optimization solver on benchmark examples indicate that as the simulation time progresses, the proposed algorithm converges rapidly to the "optimal" solution, even when using a small number of samples.

6.1 Introduction

NMPC is the most straightforward control strategy which can provide optimal online control for constrained nonlinear systems. However, classically it requires solving online a nonlinear optimization problem. An alternative strategy in NMPC is to draw samples from either the state or input space, to design computationally feasible NMPC methods, see for example, (Piovesan and Tanner, 2009), which proposes a randomized approach to sampling the space of predicted input sequences. Alternatively, in (Chakrabarty et al., 2016), a method is proposed for explicit NMPC (ENMPC) based on sampling of the state space for continuously differentiable nonlinear systems. The method therein solves optimization problems offline to find optimal control sequences, which are used to construct the ENMPC strategy. While there are still concerns in ENMPC related to robustness, feasibility of the offline op-

timization and finding the neighbors in the sampled grid for the off–grid states, ENMPC, when successful, reduces significantly the computational load of MPC at the expense of an acceptable cost degradation.

Input and state space sampling–methods for solving NMPC via approximate DP (ADP) have also been proposed, see (Bertsekas, 2005a), though they inherit the dimensionality issues of DP (Lee and Lee, 2004). Another relevant sampling–based strategy, the so–called sampling based MPC (SBMPC) (Dunlap et al., 2010), is applicable to nonlinear systems in general, though, its performance is dependent on a user–specified heuristic. A more ample discussion on sampling–based ADP and SBMPC, in the light of the method proposed in this chapter, is reported in Section 6.2.2.

A common problem of sampling–guided methods for NMPC is the sampling strategy. For example, with each input sample, a tree is expanded. After the tree is built, the path of least cost in the tree is selected from the initial state to the desired state. If the sampling is performed over the input space, and each sample is connected to all the samples in the input space for the next time step in the control horizon, then the tree growth is exponential with the horizon. Alternatively, as in randomized MPC (Piovesan and Tanner, 2009), sampling randomly in the input space, of dimension m, augmented to the horizon of dimension N requires a large number of samples, in an mN dimensional space, to achieve a significant accuracy.

In this chapter we adopt a suboptimal formulation of NMPC, as originally proposed in (Scokaert et al., 1999), where it was shown that feasibility of a solution implies stability under suitable conditions. This, together with the fact that suboptimal NMPC has the same inherent robustness properties as optimal NMPC, see (Pannocchia et al., 2011) and (Lazar and Heemels, 2009), suggest that suboptimal NMPC is a viable approach when a sampling–guided MPC strategy is undertaken for the control of nonlinear systems. Furthermore, we aim at a sampling method which provides a suboptimal solution that yields good control performance, has a reasonable computational complexity increase with the prediction horizon and allows for parallel implementation up to some level.

In this chapter, the main idea for achieving this goal is to use the shifted sequence of predicted inputs from the previous time step as a warm start, and to iteratively update this sequence by changing its elements one by one in a backward manner, i.e., starting from the last predicted input and ending with the first predicted input, or the other way around for a forward strategy. The backward method resembles the DP principle and the forward approach is similar to the rollout algorithms, see (Bertsekas, 2005b). Both the backward and forward strategies improve a heuristic base policy for optimal control, as in rollout algorithms. Additionally, as opposed to the ADP, for instance, in this chapter we sample only the input space, which is typically represented by a proper set $\mathbb{U} \subset \mathbb{R}^m$. Sampling allows for parallelization of the calculations performed for updating each of the elements of the predicted sequence of inputs and it enables limiting the computational time according to the requirements of the considered application. An upper–bound on the complexity of the overall algorithm is quadratic with the prediction horizon N and linear with the number of samples in $\mathbb{U}$. This enables the usage of long prediction horizons or real–time implementation on inexpensive computing devices such as ASIC and FPGA. The suitability for real–time implementation is also enhanced by the fact that the algorithm can be stopped at any iteration performed within a sampling period, while the complexity of the calcu-

lations per iteration depends only on the horizon N, the number of samples and parallel threads, and it does not depend on the measured state of the system. Moreover, the updated predicted sequence of inputs obtained at any iteration will guarantee recursive feasibility, stability and an improved cost function under the same conditions as in suboptimal NMPC (Scokaert et al., 1999).

This chapter analyzes also the convergence of the input sequence computed via this algorithm to the optimal input sequence, with a detailed comparison to the DP solution and the rollout algorithms. Additionally, because of the sampled nature of the input sequence that we compute, we provide also an alternative for input smoothing.

6.2 Suboptimal MPC problem formulation

6.2.1 Problem formulation

Let us consider the discrete–time system described by

$$x_{k+1} = f(x_k, u_k), \tag{6.1}$$

where $x_k \in \mathbb{R}^n$ is the state and $u_k \in \mathbb{R}^m$ is the control vector at discrete–time $k \in \mathbb{Z}_+$. We assume that the map $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ satisfies $f(0, 0) = 0$, which is, the origin is an equilibrium point for system (6.1).

The goal of MPC is to regulate the state to the origin while satisfying control and state constraints, i.e., $u_k \in \mathbb{U} \subset \mathbb{R}^m$ and $x_k \in \mathbb{X} \subset \mathbb{R}^n$ for all $k \in \mathbb{Z}_+$, where $\mathbb{U}$ and $\mathbb{X}$ are proper sets. MPC relies on a receding–horizon control law in order to determine, for each k, a finite–sequence of control inputs

$$U(k) = \{u_{k|k}, u_{k+1|k}, \ldots, u_{k+N-1|k}\},$$

where N is the control and prediction horizon, which are considered equal in this chapter, for simplicity of exposition. If the initial state is $x_{k|k} = x_k$ and the control sequence is $U(k)$, the solution of system (6.1) in closed–loop with $U(k)$ at time $k + i$ is denoted by $\phi(x_{k|k}, U(k), i)$. The current control action u_k, is selected as the first control action in $U(k)$, i.e., $u_k = u_{k|k}$.

To achieve this, at each discrete–time k, optimal MPC computes the global minimizer of a cost function of the type

$$J(x_{k|k}, U(k)) = V_f(x_{k+N|k}) + \sum_{i=0}^{N-1} L(x_{k+i|k}, u_{k+i|k}), \tag{6.2}$$

where $J : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}_+$ is the total cost function, $V_f : \mathbb{R}^n \to \mathbb{R}_+$ is a terminal cost and $L : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}_+$ is a stage cost. The minimization is performed with respect to $U(k)$ and it is subject to

$$x_{k+j|k} = f(x_{k+j-1|k}, u_{k+j-1|k}), \quad \forall j \in \mathbb{Z}_{[1,N]}, \tag{6.3}$$

and to the state and input constraints:

$$x_{k+i|k} \in \mathbb{X}, u_{k+i|k} \in \mathbb{U}, \quad \forall i \in \mathbb{Z}_{[0,N-1]}, \tag{6.4}$$

$$x_{k+N|k} \in \mathbb{X}_T, \tag{6.5}$$

where $\mathbb{X}_T \subseteq \mathbb{X}$ is a proper set which represents a terminal constraint. Moreover, define by $\mathcal{U}(x_{k|k})$ the set of control sequences $U(k)$ which, applied on $x_{k|k}$, satisfy (6.3), (6.4) and (6.5).

When the dynamics f is a nonlinear, possibly non–convex function, the optimization of the cost (6.2) cannot be guaranteed to converge to a global optimum, in general. Suboptimal MPC is a viable alternative, see, e.g., (Scokaert et al., 1999), to deal with this inherent shortcoming of nonlinear global optimization.

Suboptimal MPC relies on an initial feasible solution, a warm start sequence $U_{warm}(k) \in \mathcal{U}(x_{k|k})$ at each step k, which is improved iteratively. The suboptimal MPC problem considered in this chapter is formulated as follows:

Problem 6.2.1 For each $k \in \mathbb{Z}_{\geq 1}$, given a sequence $U_{warm}(k)$ which is different than the globally optimal input sequence, find a sequence $U(k) \in \mathcal{U}(x_{k|k})$ such that

$$J(x_{k|k}, U_{warm}(k)) > J(x_{k|k}, U(k)), \tag{6.6}$$

and the constraints (6.3), (6.4) and (6.5) are satisfied. □

Remark 6.2.2 Consider a locally stabilizing control law $k_f : \mathbb{X}_T \to \mathbb{U}$. Assume that $\mathbb{X}_T$ is a sublevel set of V_f and the following properties hold:

- $V_f(f(x, k_f(x))) + L(x, k_f(x)) \leq V_f(x)$ for all $x \in \mathbb{X}_T$;
- there exist $\alpha_1, \alpha_2 \in \mathcal{K}_\infty$ such that $\alpha_1(\|x\|) \leq V_f(x) \leq \alpha_2(\|x\|)$ for all $x \in \mathbb{X}_T$;
- there exist $\alpha_3 \in \mathcal{K}_\infty$ such that $L(x, u) \geq \alpha_3(\|x\|)$ for all $(x, u) \in \mathbb{X} \times \mathbb{U}$.

Then, the MPC closed loop system is stable. The first property above listed implies that $\mathbb{X}_T$ is positively invariant for the closed–loop system $x_{k+1} = f(x_k, k_f(x_k))$. The second and third properties can be satisfied if, for example V_f and L are positive definite quadratic functions. □

These properties imply that the cost function $J(\cdot, \cdot)$ is a LF, see (Mayne and Rawlings, 2009, Lemma 2.14). As such, if $\mathbb{F}$ is the set of states in $\mathbb{X}$ for which there exists a control sequence $U(k)$ which satisfies the constraints (6.3), (6.4) and (6.5), then the solution to Problem 6.2.1 provides an asymptotically stabilizing controller with a region of attraction $\mathbb{F}$.

In (Mayne and Rawlings, 2009), an algorithm is proposed for suboptimal MPC with stability guarantees. Given the current state $x_{k|k}$ and the previous control sequence $U(k-1)$ as an input, the steps of the algorithm therein can be summarized as follows:

- If $x_{k|k} \notin \mathbb{X}_T$, use the warm start sequence:

$$\begin{aligned} U_{warm}(k) = \{&u_{k|k-1}, u_{k+1|k-1}, \dots, \\ &u_{k+N-2|k-1}, k_f(x_{k+N-1|k-1})\}. \end{aligned} \tag{6.7}$$

Solve iteratively Problem 6.2.1 via optimization to improve $U_{warm}(k)$ with a $U(k) \in \mathcal{U}(x_{k|k})$. Apply control $u_k = u_{k|k}$.

- If $x_{k|k} \in \mathbb{X}_T$, set $u_{k|k} = k_f(x_{k|k})$, or, similarly to the previous case, use the warm start and solve an optimization algorithm iteratively to find an improved control sequence $U(k) \in \mathcal{U}(x_{k|k})$.

Optimization solvers, in both optimal and suboptimal MPC, present difficulties in terms of parallelization and a priori known execution time independently of the current state x_k. To circumvent these drawbacks and enable a computationally efficient and parallelizable NMPC algorithm, we develop a sampling–driven approach to solving Problem 6.2.1.

6.2.2 Existing NMPC approaches based on sampling

This section provides a brief in depth review of two main existing approaches for NMPC based on sampling, namely ADP and SBMPC, which were also mentioned in Section 6.2.

An approximate version of DP, as a tool for solving optimization problems, is proposed in (Bertsekas, 2005a, Section 6.6.1). DP has been successfully applied for determining explicit solutions for linear MPC controllers, see, e.g., (Muñoz de la Peña et al., 2004). In NMPC, the state and input states are typically discretized to apply DP algorithms. The main idea is to discretize the state space with a finite grid and to express each state outside of the grid as an interpolation of nearby grid elements. The same interpolation law is applied to compute the cost of the current nongrid state as a function of the costs of the nearby grid states. As such, by discretizing both the state space for each time in the control horizon and the input space, a transition diagram is obtained which approximates the dynamics of the system in the continuous space. On this discrete transition system, DP is applied to determine the path with the smallest cost, which, for a given initial state $x_{k|k}$, provides the control sequence $U(k)$.

Solving MPC with DP via discretization suffers from the “curse of dimensionality", due to sampling of both state and input spaces and the requirement for constructing the complete transition diagram, by evaluating the subsequent state and cost for each sampled state and all the samples in the input space.

An alternative to ADP, namely SBMPC, has been developed within the area of robotics, where typically optimization problems arising in control are non–convex, due to either kinematic constraints or constraints posed by obstacle avoidance. Sampling–based motion planning such as rapidly–exploring random trees (RRTs) (LaValle, 1998) or randomized A^* algorithms (Likhachev and Stentz, 2008), have been extensively used to construct trees which connect an initial state to a final state based on sampling states in the search space and searching for feasible inputs to connect these states. To approach issues related to the, possibly unfeasible, search for an input after sampling only in the state space, SBMPC, proposed in (Dunlap et al., 2010), samples the input space at each sampling period and creates trees that contain feasible state trajectories. The optimal path to a goal in the state–space is then searched for within the tree using goal–directed search algorithms, such as LPA^*. Such algorithms rely on computing a heuristic measure of the distance from the current sample to the goal. Selecting the heuristic is, however, not an obvious task for general nonlinear systems.

Therefore, a desirable feature of a sampling–based suboptimal NMPC algorithm is a non–exponential growth in the tree generated through sampling of the state or input space at each step in the horizon. Furthermore, it is also desirable to reduce the dependency of the algorithm on the non–obvious selection of a heuristic, which significantly impacts the performance of the sampling–based strategy. To circumvent these issues, an alternative suboptimal strategy for sampling the input space is developed in the next section, based on sequentially updating a warm start feasible sequence of predicted inputs.

6.3 Prototype algorithm

We develop a sampling–based solution to Problem 6.2.1. By the mechanism involved in the iterative improvement of the initial feasible control sequence $U_{warm}(k)$, this solution has a low increase of the computational complexity with the control horizon.

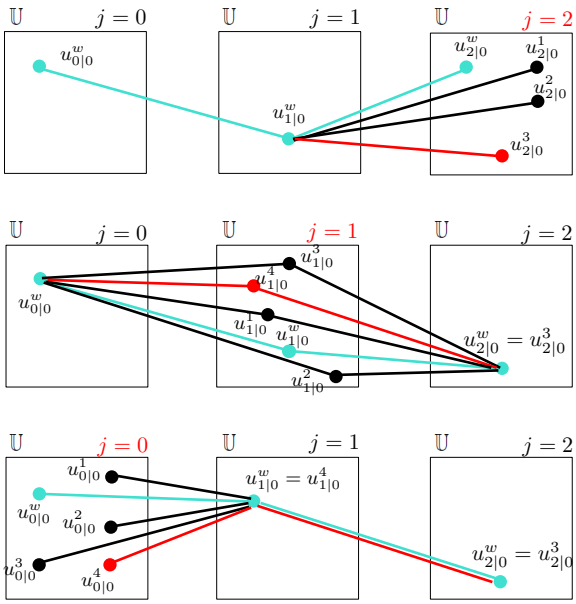


Figure 6.1: An example of the sampling principle employed by Algorithm 8, for SDNMPC, for the case when $N = 3$. At each step, $U_{warm}(k)$ (blue input sequence), is updated with the new best input from among the selected samples (red sample).

The principle behind the proposed sampling–driven approach is illustrated in Figure 6.1 and formalized in Algorithm 8. In Figure 6.1, the iterative improvement of an initial cost provided by an initial feasible control sequence $U_{warm}(k)$ is illustrated for the case when $N = 3$. The algorithm keeps always $U_{warm}(k)$ (blue) as a reference sequence, and it covers the horizon in a backward fashion, in N iterations. Starting with $j = N-1$, at each iteration step, n_j samples $\{u^q_{k+j|k}\}_{q\in\mathbb{Z}_{[1,n_j]}}$ are drawn from the input constraint set $\mathbb{U}$ (black and red points). For each sample, the reference sequence $U_{warm}(k)$ is modified in the j^{th} location, and a new sequence, $U(k)^{jq}$ is obtained. Suppose the following properties hold:

- $U(k)^{jq}$ is a feasible sequence, i.e., the constraints (6.3), (6.4) and (6.5) hold;
- the new cost, $J_{new} = J(x_{k|k}, U(k)^{jq})$ decreases with respect to $J(x_{k|k}, U_{warm}(k))$.

Algorithm 8 Backward sampling–driven suboptimal NMPC algorithm (SDNMPC).

Input: N, $\{n_j\}_{j\in\mathbb{Z}_{[0,N-1]}}$, $x_{k|k}$, $\mathbb{X}$, $\mathbb{U}$, $\mathbb{X}_T$, $J(\cdot,\cdot)$
$U_{warm}(k) = \{u^w_{k|k}, \ldots, u^w_{k+N-1|k}\}$
Output: $U(k)$, u_k

1: $J_{sub} \leftarrow J(x_{k|k}, U_{warm}(k))$;
2: **for all** $j = N-1 : -1 : 0$ **do**
3: Select n_j samples $u^q_{j+k|k} \in \mathbb{U}$, $q \in \mathbb{Z}_{[1,\ldots,n_j]}$;
4: **for all** $q = 1 : 1 : n_j$ **do**
5: **if** $j \geq 1$ **then**
6: $U(k)^{jq} = \{u^w_{k|k}, \ldots, u^w_{k+j-1|k}, u^q_{k+j|k}, u^w_{k+j+1|k}, \ldots, u^w_{k+N-1|k}\}$;
7: **else**
8: $U(k)^{jq} = \{u^q_{k|k}, u^w_{k+1|k}, \ldots, u^w_{k+N-1|k}\}$;
9: **if** $\phi(x_{k|k}, U(k)^{jq}, i) \in \mathbb{X}, \forall i \in \mathbb{Z}_{[j+1,N-1]}$ and $\phi(x_{k|k}, U(k)^{jq}, N) \in \mathbb{X}_T$ **then**
10: $J_{new} \leftarrow J(x_{k|k}, U(k)^{jq})$;
11: **if** $J_{new} < J_{sub}$ **then**
12: $J_{sub} \leftarrow J_{new}$;
13: $U_{warm}(k) \leftarrow U(k)^{jq}$;
14: $U(k) = U_{warm}(k)$, $u_k = u^w_{k|k}$;

Then $U_{warm}(k)$ is replaced by $U(k)^{jq}$ and the algorithm continues backwards (in Figure 6.1, we switch to the text horizontal line) with respect to the prediction time j, in a similar manner. With this approach, at any point in time, if the maximally allowed computational time is reached, a feasible, improved control sequence exists and it can be utilized as a suboptimal MPC solution.

The forward approach differs in the fact that the covering of the horizon starts from $j = 0$ and ends with $j = N-1$. More specifically, in step 2 of Algorithm 8 we have used a backward navigation of the control horizon ($j = N-1 : -1 : 0$). However, covering the horizon in a forward manner, i.e., $j = 0 : 1 : N-1$, as in the rollout algorithms, can as well be a solution, with similar characteristics. An example is provided in Section 6.7.1.

At each step j, the states $\phi(x_{k|k}, U_{warm}(k), i)$ for all $i \in \mathbb{Z}_{[1,j]}$ already satisfy the state and input constraints, by the feasibility of $U_{warm}(k)$. This holds not only for the original $U_{warm}(k)$, but for any subsequent improvement of $U_{warm}(k)$. As such, also the stage costs up to the j^{th} state can be recovered from previous computations. This suggests intuitively that the proposed cost improvement method could deliver good performance, which is supported by results obtained in Section 6.7 on non–trivial case studies.

When $k = 0$, we can select an initial sequence $U_{warm}(0)$ as the solution of the optimal MPC problem. In this case, we can proceed with $k = 1$. Alternatively, we can select a feasible sequence $U_{warm}(0)$, an "oracle", by randomly selecting sequences of inputs in $\mathbb{U}$ until a feasible $U_{warm}(0)$ is found. In this case, if it is feasible to afford the computational time, we can proceed with Algorithm 8 in an attempt of obtaining an improved sequence. For $k \in \mathbb{Z}_{\geq 1}$, to choose the input $U_{warm}(k)$ for Algorithm 8, one can use the receding horizon principle of MPC. As such, the input sequence

$$U_{warm}(k) = \{u_{k|k-1}, \ldots, u_{k+N-2|k-1}, u\}$$

is a warm start at time k. If $\phi(x_{k-1|k-1}, U(k-1), N) \in \mathbb{X}_T$ and $\mathbb{X}_T$ is positively invariant for the system $x_{k+1} = f(x_k, k_f(x_k))$, then one can select $u = k_f(x_{k+N-1|k-1})$. In this case, $U_{warm}(k)$ is a feasible solution, and therefore a candidate warm start for every $k \in \mathbb{Z}_{\geq 1}$. If there exists no terminal set $\mathbb{X}_T$ and no $k_f(\cdot)$, then one can select $u \in \mathbb{U}$ such that $U_{warm}(k)$ remains feasible, i.e., $\phi(x_{k-1|k-1}, U_{warm}(k), N) \in \mathbb{X}$. In these circumstances, however, stability of the closed loop is not guaranteed. Such an example is illustrated in Section 6.7.3.

Remark 6.3.1 Common sampling schemes are employed for sampling of the input space $\mathbb{U}$ at each iteration, among which we consider deterministic uniform sampling, which places each sample at equal distance from each other, to cover uniformly the space $\mathbb{U}$. Alternatively, "true" random samples can be selected, which are simpler to draw in higher dimensional spaces. Quasi–random low–discrepancy sequences, see (Chakrabarty et al., 2016), may be used as well, considering the fact that they appear to be random for multiple purposes, such as Monte Carlo simulations. Such sampling methods, e.g., Sobol or Halton sequences, have been shown, see, e.g., (de Dios Ortúzar and Willumsen, 1994), to better cover the space than random sequences. □

Remark 6.3.2 Assume that k_f is a locally stabilizing control law on $\mathbb{X}_T$, a sublevel set of V_f, which is positively invariant for the system $x_{k+1} = f(x_k, k_f(x_k))$ and V_f and L are, e.g., positive definite quadratic functions. Considering that the sequence $U(k)$ provided by Algorithm 8 is a suboptimal solution solving Problem 6.2.1, then, by Remark 6.2.2, Problem 6.2.1 is recursively feasible and it ensures stability of system (6.1) in closed loop with $u_k = u^w_{k|k}$. □

Remark 6.3.3 The working mechanism of Algorithm 8 resembles the principle of the rollout algorithm described in the survey paper (Bertsekas, 2005b). Therein, a rollout algorithm improves iteratively a base policy (here, the feasible control sequence $U_{warm}(k)$) to provide a suboptimal solution to an optimal control problem via ADP and suboptimal control. The working principle of the rollout algorithm with SDNMPC, i.e., choosing at time k the iterated solution of the previous time instance, $k-1$, as a warm start, and a sampling strategy, is not provided therein. A discussion on SDNMPC in comparison with DP and rollout is available in Appendix A.2. □

Remark 6.3.4 Note that here we assume that the initial state is known exactly. However, in the case of modeling error or exogenous noise this is probably not achieved. In (Lazar and Heemels, 2009) it is proven that if recursive feasibility is ensured, then inherent input–to–state stability holds for suboptimal NMPC. To ensure recursive feasibility when the exact state is now known, new methods, outside of the scope of this thesis, have to be developed, possibly by recomputing a terminal set and tightening constraints. □

6.4 Complexity analysis

In order to analyze the complexity of Algorithm 8, the following assumptions are undertaken, for a given state x, input u and input sequences U, U_1, U_2:

1) The cost of evaluating $f(x,u)$ and performing a feasibility test $f(x,u) \in \mathbb{X}$ is c_1;

2) The cost of evaluating one stage cost $L(x,u)$ for a given state $x \in \mathbb{X}$ and input $u \in \mathbb{U}$ is c_2;

3) The cost of comparing $J(x,U_1) < J(x,U_2)$ and changing J_{sub} and $U_{warm}(k)$ if necessary, i.e., steps 11–13 in Algorithm 8, is negligible.

4) The current cost J_{sub} is instantaneously available for comparison with each of the n_j samples according to step 11 in Algorithm 8 and each of the new sequences $U(k)^{jq}$ may modify J_{sub} if the new cost J_{new} is smaller than J_{sub}.

The complexity of Algorithm 8 for a given $x_{k|k}$ is the following:

$$C = (c_1 + c_2)\left(\sum_{j=0}^{N-1}(N-j)n_j\right). \tag{6.8}$$

If we assume $n_j \leq \overline{n}$ for all $j \in \mathbb{Z}_{[0,N-1]}$, then the complexity (6.8) can be upper bounded by

$$C = \overline{n}(c_1 + c_2)\frac{N(N+1)}{2}. \tag{6.9}$$

A possible reduction of the bound (6.9) might be attained, considering the fact that all the states subsequent to a non–feasible state are no longer evaluated and checked for feasibility. This means that, in step 9 of Algorithm 8, if $\phi(x_{k|k}, U(k)^{jq}, i) \notin \mathbb{X}$ for a specific $i \in \mathbb{Z}_{[j+1,N-1]}$, then $\phi(x_{k|k}, U(k)^{jq}, r)$ for all $r \in \mathbb{Z}_{[i+1,N]}$, are no longer evaluated, in which case steps 2) and 3) above are skipped all together.

Consider now that many threads are available, from multiple processors. Notice also that at each time in the horizon, all n_j computations can be performed separately. In these conditions, assuming we have $\overline{n}$ threads available, then the complexity of Algorithm 8 is upper bounded by

$$C = (c_1 + c_2)\frac{N(N+1)}{2}. \tag{6.10}$$

The complexity bound given in (6.10), though quadratic in the prediction horizon, yields a reasonable complexity, considering that, in NMPC, a horizon $N = 10$ is considered a reasonably large horizon.

In general, if we have $p \in \mathbb{Z}_{[2,\overline{n})}$ processors, then the complexity of Algorithm 8 is

$$C = \lceil \overline{n}/p \rceil (c_1 + c_2) \frac{N(N+1)}{2}, \tag{6.11}$$

where the term $\lceil \overline{n}/p \rceil$ appears due to the fact that a thread can not engage in computations related to a subsequent time horizon until all the threads have finalized the computations related to the current time horizon j.

6.5 Convergence analysis

To evaluate the performance of the developed method, it is of interest to analyse the extent to which the resulting sequence $U(k)$ obtained via Algorithm 8 converges to the optimal sequence $U^*(k)$. For this reason, in this section, the evolution with the time k of the distance between the cost functions $J(x_{k|k}, U(k))$ and $J(x_{k|k}, U^*(k))$ will be analyzed.

6.5.1 Sufficient conditions for convergence

The following assumption is necessary for the convergence analysis in this subsection.

Assumption 6.5.1 A global optimal input sequence $U^*(k)$ exists at each step $k \in \mathbb{Z}_+$.

The following theorem defines conditions under which the sequence $U(k)$ converges to the optimal sequence $U^*(k)$, and the rate of convergence.

Theorem 6.5.2 *Suppose Assumption 6.5.1 holds. For each time step k, if there exists a scalar $\sigma_k \in \mathbb{R}_{[0,1)}$ such that:*

$$\begin{aligned} &J(x_{k+1|k+1}, U^w(k+1)) - J(x_{k+1|k+1}, U^*(k+1)) \leq \\ &\sigma_k (J(x_{k|k}, U^w(k)) - J(x_{k|k}, U^*(k))), \end{aligned} \tag{6.12}$$

then the cost $J(x_{k|k}, U^w(k))$ converges monotonically to $J(x_{k|k}, U^*(k))$ with a convergence rate σ_k and $\lim_{k\to\infty}(J(x_{k|k}, U^w(k)) - J(x_{k|k}, U^*(k)) = 0$. □

Proof: Consider the function $V : \mathbb{X} \to \mathbb{R}_+$ defined as $V(x_{k|k}) := J(x_{k|k}, U^w(k)) - J(x_{k|k}, U^*(k))$. Notice that $V(x_{k|k}) \geq 0$ for all $k \in \mathbb{Z}_+$, because the cost $J(x_{k|k}, U^*(k))$ corresponds to the optimal input sequence $U^*(k)$, while $U^w(k)$ is suboptimal. (6.12) can then be rewritten as:

$$V(x_{k+1|k+1}) \leq \sigma_k V(x_{k|k}).$$

Thus, it follows that

$$\begin{aligned} 0 \leq V(x_{k+1|k+1}) &\leq \sigma_k V(x_{k|k}) \\ &\leq \sigma_k \sigma_{k-1} V(x_{k-1|k-1}) \\ &\leq \dots \\ &\leq \prod_{i=0}^{k} \sigma_i V(x_{0|0}). \end{aligned} \tag{6.13}$$

Since $\sigma_k \in \mathbb{R}_{[0,1)}$, if there exists $\bar{\sigma} < 1$ such that $\sigma_k \leq \bar{\sigma}$ for all $k \in \mathbb{Z}_+$, then

$$\lim_{k\to\infty} \prod_{i=0}^{k} \sigma_k \leq \lim_{k\to\infty} \bar{\sigma}^k = 0. \tag{6.14}$$

Therefore, from (6.14) and (6.13) it follows that

$$\lim_{k\to\infty} V(x_{k|k}) = 0, \tag{6.15}$$

and consequently,

$$\lim_{k\to\infty} (J(x_{k|k}, U^w(k)) - J(x_{k|k}, U^*(k)) = 0. \tag{6.16}$$

■

Notice that V acts like a LF. Alternative, relaxed conditions for convergence can be imposed, such as a p–step decrease, with $p \in \mathbb{Z}_{\geq 1}$, of $V(x_{k|k})$ of the type

$$V(x_{k+p|k+p}) \leq \sigma_k V(x_{k|k}),$$

with $\sigma_k \in \mathbb{R}_{[0,1)}$.

In Algorithm 8 there are three sources for the improvement of the cost $J(x_{k|k}, U(k))$ during the same time instant k, but also as the iteration k progresses. The decrease sources, and therefore, the construction of σ_k, are detailed in the following lemma.

Lemma 6.5.3 *For each $k \in \mathbb{R}_+$ there exist $\alpha_k \in \mathbb{R}_{[0,1]}$, $\beta_k \in \mathbb{R}$, $\rho_{jk} \in \mathbb{R}$ for all $j \in \mathbb{Z}_{[1,N]}$ such that* (6.12) *holds with*

$$\sigma_k = \alpha_k \beta_k \left(\prod_{j=1}^{N} \rho_{jk} \right) \in \mathbb{R}_{\geq 0} \in \mathbb{R}_+.$$

Proof: The three sources for the improvement of the cost $J(x_{k|k}, U(k))$ in Algorithm 8 during the same time instant k, but also as the iteration k progresses are the following:

1. At the same time instance k, due to the fact that we use samples, we improve the input sequence with every step, from $j+1$ to j. Then, for each of the elements j from N up to 1, i.e., step 2 in Algorithm 8, a real value ρ_{jk} exists such that:

$$\begin{aligned} J(x_{k|k}, U^j(k)) - J(x_{k|k}, U^*(k)) \leq \\ \rho_{jk}(J(x_{k|k}, U^{j+1}(k)) - J(x_{k|k}, U^*(k))), \end{aligned} \tag{6.17}$$

   where $U^{N+1}(k) = U^w(k)$ and $U(k) = U^1(k)$. Notice that $\rho_{jk} \in \mathbb{R}_{[0,1]}$, because $J(x_{k|k}, U^j(k)) - J(x_{k|k}, U^*(k)) \geq 0$, $J(x_{k|k}, U^{j+1}(k)) - J(x_{k|k}, U^*(k)) \geq 0$ and $J(x_{k|k}, U^j(k)) \leq J(x_{k|k}, U^{j+1}(k))$. If no sample improves the previously feasible sequence, i.e., $U^{j+1}(k)$, then $\rho_{jk} = 1$, otherwise $\rho_{jk} \in \mathbb{R}_{[0,1)}$.

2. As the time k progresses, if the MPC feedback closed–loop system is stable, due to the fact that J is a LF, there exists a decrease in J with the increase of k. This decrease occurs with respect to the input sequences $U(k)$ and $U^w(k+1)$ resulting from Algorithm 8, thus, there exists a real value β_k such that:

$$J(x_{k+1|k+1},U^w(k+1)) - J(x_{k|k},U^*(k)) \leq \\ \beta_k(J(x_{k|k},U(k)) - J(x_{k|k},U^*(k))). \tag{6.18}$$

The true optimal sequence generates a stable MPC feedback closed–loop system as well. Therefore, the decrease of J occurs with respect to the optimal input sequence as well, i.e., from $U^*(k)$ to $U^*(k+1)$, and therefore, there exists a real value α_k such that:

$$J(x_{k+1|k+1},U^w(k+1)) - J(x_{k+1|k+1},U^*(k+1)) \leq \\ \alpha_k(J(x_{k+1|k+1},U^w(k+1)) - J(x_{k|k},U^*(k))). \tag{6.19}$$

By embedding (6.18) and (6.17) in (6.19) we obtain

$$\begin{aligned} &J(x_{k+1|k+1},U^w(k+1)) - J(x_{k+1|k+1},U^*(k+1)) \leq \\ &\leq \alpha_k(J(x_{k+1|k+1},U^w(k+1)) - J(x_{k|k},U^*(k))) \\ &\leq \alpha_k\beta_k(J(x_{k|k},U(k)) - J(x_{k|k},U^*(k))) \\ &\leq \alpha_k\beta_k\rho_{1k}(J(x_{k|k},U^2(k)) - J(x_{k|k},U^*(k))) \\ &\leq \dots \\ &\leq \alpha_k\beta_k\left(\prod_{j=1}^{N}\rho_{jk}\right)(J(x_{k|k},U^w(k)) - J(x_{k|k},U^*(k))). \end{aligned} \tag{6.20}$$

In (6.20), notice that

$$J(x_{k+1|k+1},U^w(k+1)) - J(x_{k+1|k+1},U^*(k+1)) \geq 0$$

and

$$J(x_{k|k},U^w(k)) - J(x_{k|k},U^*(k)) \geq 0,$$

and therefore

$$\sigma_k = \alpha_k\beta_k\left(\prod_{j=1}^{N}\rho_{jk}\right) \in \mathbb{R}_{\geq 0}.$$

■

Remark 6.5.4 As to what concerns the equations (6.18) and (6.19), notice that the term $J(x_{k+1|k+1},U^w(k+1)) - J(x_{k|k},U^*(k))$ might be negative, in which case $\alpha_k \leq 0$, because $J(x_{k+1|k+1},U^w(k+1)) - J(x_{k+1|k+1},U^*(k+1)) \geq 0$. In that case the value of β_k has to be negative, which is possible, if we analyse (6.18). If the term $J(x_{k+1|k+1},U^w(k+1)) -$

$J(x_{k|k}, U^*(k))$ is positive, then, by the stability analysis formulation from Section 6.2.1, where it was established that the cost J is a LF, we have the necessary condition that $J(x_{k+1|k+1}, U^w(k+1)) < J(x_{k|k}, U(k))$, and therefore $\beta_k \in \mathbb{R}_{[0,1)}$. Also, by the same argument, the inequality $J(x_{k+1|k+1}, U^*(k+1)) < J(x_{k|k}, U^*(k))$ holds. Then, in (6.19), $\alpha_k \in \mathbb{R}_{>1}$. If no samples are selected, which means that only the warm start sequence is used, then $\sigma_k = \alpha_k \beta_k$. □

From Theorem 6.5.2 it follows that, if $\sigma_k \in \mathbb{R}_{[0,1)}$, then the convergence of Algorithm 8 is guaranteed. A question we may ask is, how many input samples n_j shall we select to ensure that ρ_{jk} in Lemma 6.5.3 achieves a value that enables σ_k to remain in an interval $[0, 1)$? For answering this question, we formulate the following assumptions:

Assumption 6.5.5 For each j from N up to 1, there exists a sequence $U_j(k)$ such that (6.17) holds with $\rho_{jk} \leq \rho_k$, where $\rho_k \in \mathbb{R}_{[0,1)}$. □

Assumption 6.5.6 The map f in (6.1) is continuous with respect to u_k in the set $\mathbb{U}$. □

In what follows we consider separately the two cases when the sampling is uniform deterministic and randomized sampling.

6.5.2 Convergence conditions under uniform deterministic sampling

In this subsection we consider that the sampling of the set $\mathbb{U}$ is uniform.

Definition 6.5.7 The sampling density η of a set $\mathbb{U}$ is the distance between sample points on any of the axes of the set $\mathbb{U}$. □

More specifically, denote by $\Upsilon \in \mathbb{R}^{m\times 2}$ the matrix containing on each line $i \in \mathbb{Z}_{[1,m]}$ the bounds on the i–th element of the input vector u_k, i.e., $\Upsilon(i,1) \leq u_k(i) \leq \Upsilon(i,2)$. By exploiting the sampling density η and the intervals $[\Upsilon(i,1),\ \Upsilon(i,2)]$, we can obtain the number of samples drawn on each axis, a_i as follows:

$$[\Upsilon(i,2) - \Upsilon(i,1)] = (a_i + 1)\eta,$$

which in turn allows to compute the total number of sample points n_j which we draw in the set $\mathbb{U}$ via the equation

$$n_j = \prod_{i=1}^{m} a_i = \prod_{i=1}^{m} \left(\frac{\Upsilon(i,2) - \Upsilon(i,1)}{\eta} - 1 \right). \tag{6.21}$$

As mentioned previously, the values of α_k and β_k do not depend on the number of samples n_j we select in the set $\mathbb{U}$. As such, in the process of computing the number of samples, we consider α_k and β_k to be known.

To be able to decide on the sampling density at the current step, j, see step 2 in Algorithm 8, we need to connect the cost function J for a specific sample with the sampling density. This is necessary in order to ensure the contraction of the cost function J of the current input sequence to the cost of the optimal input sequence.

Lemma 6.5.8 *Suppose Assumption 6.5.5 and Assumption 6.5.6 hold. Define by $U^{j*}(k)$ the argument of the minimum cost at the step j, i.e., if*

$$U^{j+1}(k) = \{u^w_{k|k}, \dots, u^w_{k+j|k}, \dots, u^w_{k+N-1|k}\},$$

then $U^{j*}(k) = \{u^w_{k|k}, \dots, u^*_{k+j|k}, \dots, u^w_{k+N-1|k}\}$, where

$$u^*_{k+j|k} = arg \min_{u(k+j|k)\in\mathbb{U}} J(x_{k|k}, U(k))$$
$$\text{s.t. } U(k) = \{u^w_{k|k}, \dots, u_{k+j|k}, \dots, u^w_{k+N-1|k}\}.$$

If the inequality

$$\frac{J(x_{k|k}, U^{j*}(k)) - J(x_{k|k}, U^*(k))}{J(x_{k|k}, U^{j+1}(k)) - J(x_{k|k}, U^*(k))} < \left(\frac{1}{\alpha_k \beta_k}\right)^{\frac{1}{N}} \tag{6.22}$$

holds, then there exists a finite number of samples n_j, such that (6.12) *holds with $\sigma_k \in \mathbb{R}_{[0,1)}$.* □

Proof: Starting from Assumption 6.5.6, we can construct a bound on the difference between the cost of the current sampled sequence and the cost of the optimal sequence. The bound is a function of the distance from the sampled sequence to the optimal sequence. It is known (Lazar et al., 2013b, Corollary III.10) that a map f which is continuous on a compact set $\mathbb{U}$ is also $\mathcal{K}$–continuous on the set $\mathbb{U}$. Then, by the construction of the cost function, from Assumption 6.5.6, for each $x_{k|k}$, denote by d_k the $\mathcal{K}$ function which ensures

$$\|J(x_{k|k}, U(k)) - J(x_{k|k}, V(k))\| \leq d_k(\|U(k) - V(k)\|),$$

for each $U(k), V(k) \in \mathbb{U}^N$. Then, it follows directly that

$$J(x_{k|k}, U^j(k)) - J(x_{k|k}, U^{j*}(k)) \leq d_k(\|U^j(k) - U^{j*}(k)\|). \tag{6.23}$$

From (6.17) we obtain ρ_k as follows:

$$\frac{J(x_{k|k}, U^j(k)) - J(x_{k|k}, U^*(k))}{J(x_{k|k}, U^{j+1}(k)) - J(x_{k|k}, U^*(k))} \leq \rho_k. \tag{6.24}$$

To achieve contractiveness of the cost function J, it is sufficient to require $\alpha_k \beta_k \rho_k^N < 1$, which implies that

$$\rho_k < \left(\frac{1}{\alpha_k \beta_k}\right)^{\frac{1}{N}}. \tag{6.25}$$

Finally, via (6.24) and (6.25), the inequality to satisfy is

$$\frac{J(x_{k|k}, U^j(k)) - J(x_{k|k}, U^*(k))}{J(x_{k|k}, U^{j+1}(k)) - J(x_{k|k}, U^*(k))} < \left(\frac{1}{\alpha_k \beta_k}\right)^{\frac{1}{N}}. \tag{6.26}$$

In (6.23), if we have selected $U^j(k)$ as the argument for the best cost function achievable with the sample points, then, by the continuity of J, $\|U^j(k) - U^{j*}(k)\| \leq \eta$. Therefore, by (6.23), it follows that if

$$\frac{J(x_{k|k}, U^{j*}(k)) + d_k(\eta) - J(x_{k|k}, U^*(k))}{J(x_{k|k}, U^{j+1}(k)) - J(x_{k|k}, U^*(k))} < \left(\frac{1}{\alpha_k \beta_k}\right)^{\frac{1}{N}}, \tag{6.27}$$

then also (6.26) holds. Now, since the number of samples n_j is a function of η, if we find a feasible η, then n_j can be computed via (6.21). We can find the largest value of η by using a bisection strategy in the interval

$$\left(0, \min_{i=1:m} \frac{\Upsilon(i,2) - \Upsilon(i,1)}{2}\right].$$

A value of η can be found in this manner only under the condition that

$$\frac{J(x_{k|k}, U^{j*}(k)) - J(x_{k|k}, U^*(k))}{J(x_{k|k}, U^{j+1}(k)) - J(x_{k|k}, U^*(k))} < \left(\frac{1}{\alpha_k \beta_k}\right)^{\frac{1}{N}} \tag{6.28}$$

holds. ■

6.5.3 Convergence conditions under random sampling

The method for bounding the number of samples n_j as proposed in the previous subsection is valuable for deterministic, uniform sampling. However, it is more practical to sample randomly, especially in higher dimensional spaces. For this reason, in this section, a bound on the number of samples is proposed for the case of random sampling. First, inspired by (6.26), consider the following notation:

$$v_{jk} := J(x_{k|k}, U^*(k)) + \left(\frac{1}{\alpha_k \beta_k}\right)^{\frac{1}{N}} (J(x_{k|k}, U^{j+1}(k)) - J(x_{k|k}, U^*(k))).$$

Then, the problem to solve is defined as follows:

Problem 6.5.9 Find (assuming that it exists) a sequence $U^j(k)$ such that

$$J(x_{k|k}, U^j(k)) \leq v_{jk}. \tag{6.29}$$

Solution 1: A possible solution to Problem 6.5.9 is to perform a randomized verification of the inequality $J(x_{k|k}, U^j(k)) - v_{jk} \leq 0$ over the set $\mathbb{U}$, by computing a probability of performance, or reliability, as:

$$R = Pr\left\{F(x) \leq 0\right\},$$

where $F(u^q_{j+k|k}) = J(x_{k|k}, U^j(k)) - v_{jk}$.

Note that if $R = 1$, then $F(x) \leq 0$ holds for all $x \in \mathbb{U}$. We estimate R by the empirical reliability $\hat{R}_{n_j}$, as in Chapter 5.2.1.

However, this solution might be over–pessimistic as to what concerns the number of samples, because it requires an estimation of the reliability R on the whole set $\mathbb{U}$, while in fact when a non–zero value of the estimation $\hat{R}_{n_j} = \frac{1}{n_j}$ is achieved, then, no further sampling of the set $\mathbb{U}$ is necessary, because the solution to Problem 6.5.9 was obtained. ■

Solution 2: A solution to Problem 6.5.9 can be achieved also by following, as in Chapter 5.2.1, a randomized worst–case performance evaluation, as follows.

Let $p^*, \delta \in \mathbb{R}_{(0,1)}$ be assigned probability levels. The randomized algorithm should estimate a performance level γ such that

$$Pr\{Pr\{F(x) \leq \gamma\} \geq p^*\} \geq 1 - \delta,$$

for all $x \in \mathbb{U}$, where, in this case $F(u^q_{j+k|k}) = v_{jk} - J(x_{k|k}, U^j(k))$. The corresponding lower bound on n_j is now

$$\frac{ln\frac{1}{\delta}}{ln\frac{1}{p^*}}, \tag{6.30}$$

which gives considerably less samples than the Chernoff bound. It is expected that, with a sufficiently low δ and large p^*, the bound γ is close enough to the true maximum bound on $F(x)$ for $x \in \mathbb{U}$, and therefore we expect $\gamma \geq 0$, which is sufficient to guarantee that the $U^j(k)$ required by Problem 6.5.9 has been achieved. If $\gamma \geq 0$ has not yet been satisfied, then new samples have to be drawn, which in turn improves the probability levels. ■

6.6 Input smoothing

Due to the sampling procedure having a discontinuous behaviour, the inputs might vary more than it is safe for the actuators. This problem could be alleviated by either smoothing the inputs via interpolation of the obtained previous sequence with the initial sequence, or by penalizing $\Delta u_k = u_k - u_{k-1}$ via constraints or via the cost function, such that the input variability is limited to acceptable bounds.

In this section we propose input smoothing by penalizing Δu_k via an inequality

$$|\Delta u_k| \leq \alpha, \tag{6.31}$$

where $\alpha \in \mathbb{R}^m_{>0}$. Then, we can add (6.31) as an additional constraint to the set of constraints (6.3) and (6.4). The value of α can be set constant, or it can varied to an extent that the problem remains feasible. Note that, if the problem is not feasible due to the additional constraint (6.31) for any sample, then the iterated warm sequence still remains a feasible input sequence.

Notice that the constraint (6.31) affects step 3 of Algorithm 8 in the following way. We can distinguish two cases:

- If $k \in \mathbb{Z}_{\geq 1}$ and $j = 0$:

$$\mathbb{U}_0 = \left\{u_{k|k} : |u_{k|k} - u_{k-1}| \leq \alpha, \alpha \in \mathbb{R}^m_{>0}\right\}.$$

- Else, for each $k \in \mathbb{Z}_{\geq 0}$, if $j \in \mathbb{Z}_{[1,N-1]}$:

$$\mathbb{U}_j = \left\{u_{k+j|k} : |u_{k+j|k} - u_{k+j-1|k}| \leq \alpha, \alpha \in \mathbb{R}^m_{>0}\right\}.$$

For each case, in step 3 of Algorithm 8 we will sample in $\mathbb{U} \cap \mathbb{U}_j$ for all $j \in \mathbb{Z}_{[1,N-1]}$ instead of sampling in the complete constraint set $\mathbb{U}$. An illustrative example will be provided in Section 6.7.2.

6.7 Illustrative examples

The following examples illustrate the algorithm SDNMPC developed in this chapter and all the related concepts: backward and forward implementation, input smoothing, complexity and convergence analysis, SDNMPC with and without stability guarantees.

These examples have a state space dimension of maximum three. For a more complex illustration, inspired by the real time model of an interventional X–ray machine, a robotic arm with six states, complex constraints and obstacles and strict constraints on the computational time, we suggest the interested reader to consult the Master of Science (MSc.) thesis (Groen, 2017). This reference provides also an indepth analysis on computational aspects and a comparison with the IPOPT solver (Wächter and Biegler, 2006), instead of `fmincon`.

6.7.1 Cart–spring system

SDNMPC is applied first to a system incorporating an exponential nonlinearity, i.e., the model of a cart with mass M, which is moving on a plane, see (Raimondo et al., 2009). This cart is attached to a wall via a spring with elastic constant k varying with the first state $k = k_0 e^{-x_1}$, where x_1 stands for the displacement of the carriage from the equilibrium position. A damper acts as a resistor in the system, with damping h_d. The discretized nonlinear model of the cart and spring system is the following:

$$x(k+1) = \begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = f_1(x(k)) + F_2 u(k), \tag{6.32}$$

where

$$\begin{aligned} f_1(x(k)) &= \begin{bmatrix} x_1(k) + T_s x_2(k) \\ x_2(k) - T_s \frac{\rho_0}{M} e^{-x_1} x_1(k) - T_s \frac{h_d}{M} x_2(k) \end{bmatrix}, \\ F_2 &= \begin{bmatrix} 0 & \frac{T_s}{M} \end{bmatrix}^T, \end{aligned} \tag{6.33}$$

where x_2 is the velocity of the cart and u is an external force which acts as an input to the system. The parameter values are $T_s = 0.4s$, $\rho_0 = 0.33$, $M = 1$, $h_d = 1.1$.

The MPC controller has to steer the cart to the origin from a non–zero initial state, while satisfying the input and state constraints, which are

$$|u| \leq 4.5, |x_1| \leq 2.65, \tag{6.34}$$

and reducing the cost (6.2), where the stage cost and terminal cost are quadratic functions, i.e., $L(x,u) = x^T Q x + u^T R u$ and $V_f(x) = x^T P x$. Choose the following parameters for the MPC problem:

$$M = 4, P = \begin{bmatrix} 7.0814 & 3.3708 \\ 3.3708 & 4.2998 \end{bmatrix}, Q = diag(1,1), R = 1.$$

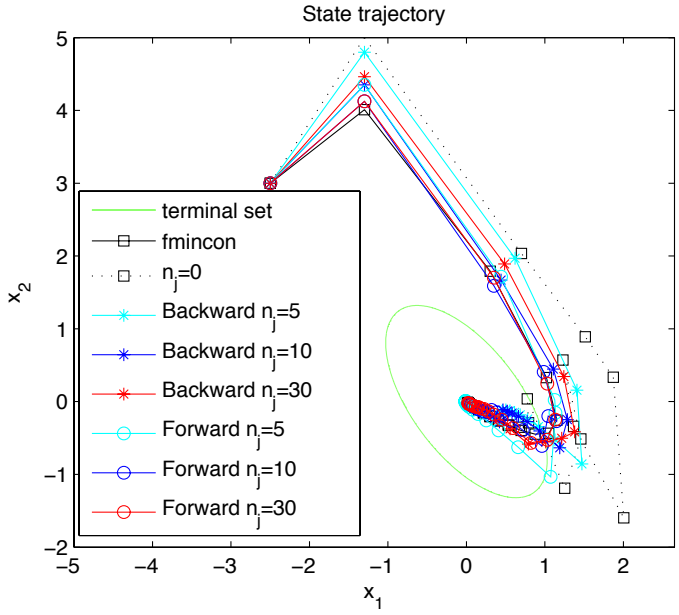


Figure 6.2: State trajectories x_k of the system (6.32), with input sequences computed according to both backward and forward SDNMPC, with different number of sample points n_j.

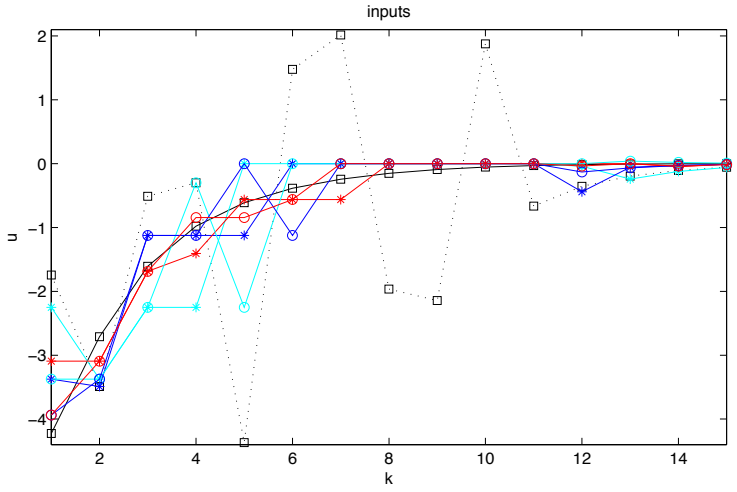


Figure 6.3: Inputs $u_{k|k}$ applied to system (6.32), with input sequences computed with backward and forward SDNMPC, with different number of sample points n_j.

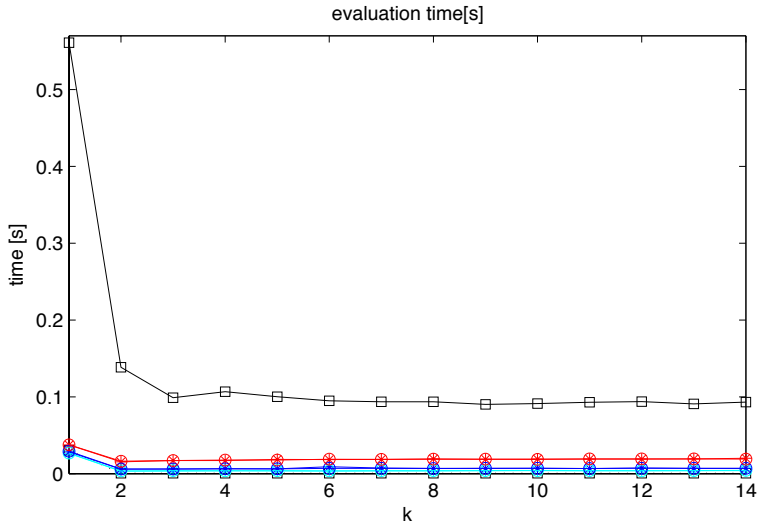


Figure 6.4: The computational time of applying Algorithm 8 to system (6.32) for different number of sample points n_j, at each time step k.

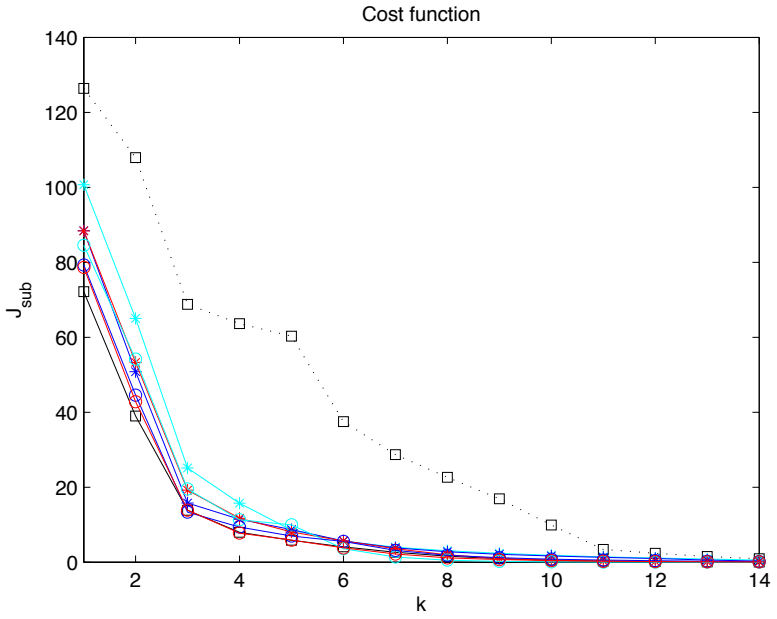


Figure 6.5: The cost J_{sub} of the suboptimal NMPC solution computed according to Algorithm 8 for system (6.32), with different number of sample points n_j.

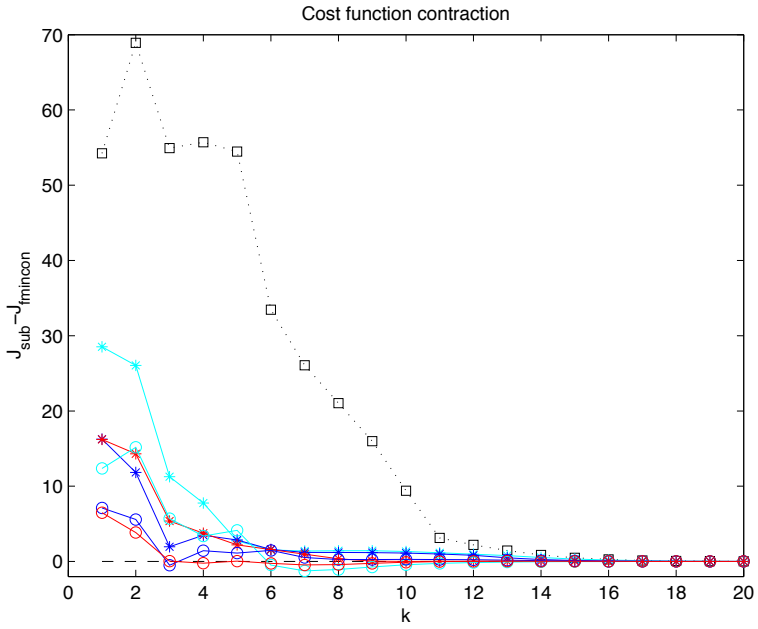


Figure 6.6: The value of $J(x_{k|k}, U(k)) - J(x_{k|k}, U_{fmincon}(k))$ for system (6.32), to illustrate the convergence rate of Algorithm 8 with different number of sample points n_j.

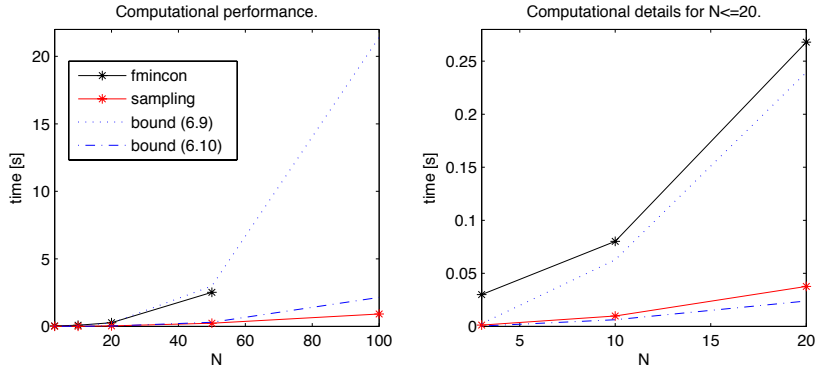


Figure 6.7: The computational complexity of Algorithm 8 as a function of the control horizon N versus the bounds computed in Section 6.4, for system (6.32).

In (Raimondo et al., 2009) it is shown that the control law

$$u = k_f(x) = -\begin{bmatrix} 0.8783 & 1.1204 \end{bmatrix} f_1(x),$$

is locally stabilizing in the set

$$\mathbb{X}_T = \{x | V_f(x) \leq 4.7\},$$

which is a terminal set where the conditions for stability of MPC mentioned in Remark 6.2.2 are satisfied.

Algorithm 8 is applied for the MPC control of system (6.32) in a both forward as well and backward implementation. We compare the results of this method with the results provided by `fmincon` in Matlab, even though, as it will be seen later, the optimization tool does not always provide the optimum, due to local minima. The scalability of the algorithm is tested by varying both the number of samples $\overline{n}$ and the control horizon N. The tests are performed on a feasible initial state $x_{0|0} = [-2.5, 3]$. The choice of the initial condition does not influence greatly the results, which are similar for other feasible initial states.

For the first experiment, fix $N = 10$. An initial $U_{warm}(0)$ is provided by a random "oracle", with values

$$\begin{aligned} U_{warm}(0) = &[-1.7441, -3.4905, -0.5104, -0.2991, -4.3680, \\ &1.4765, 2.0166, -1.9653, -2.1436, 1.8762] \end{aligned} \tag{6.35}$$

Consider $n_j = \overline{n}$, for all $j \in \mathbb{Z}_{[0,N-1]}$, taking various values in the set $\{0, 5, 10, 30\}$. For $\overline{n} = 0$, $U_{warm}(0)$ is propagated though iterations without any intervention or change from the sampling mechanism. This serves as a reference, to notice the improvements brought in by Algorithm 8. The results are illustrated in Figure 6.2–6.6, where the legend from Figure 6.2 holds until Figure 6.6. Iterations are considered from $k = 1$ until $k = 20$. Though the different sampling options proposed in Remark 6.3.1 provide in this example similar outcomes, Halton sample points have been used here for a comparison between the backward and forward versions of the SDNMPC algorithm. This choice is motivated by the practical feature that, adding extra points only when they are required does not have impact on the coverage of the set $\mathbb{U}$.

In Figure 6.2 and Figure 6.3 the state trajectory and the inputs $u(k)$ applied to the system are illustrated. The constraint specifications (6.34) are satisfied for all cases, at all times. Notice the immediate smoothing of the trajectories and input sequence even for a small $\overline{n}$, and even though in this example no input smoothing was used, as in Section 6.6. In Figure 6.3 a convergence of the inputs to the optimal input is observed for both the backward and the forward implementation of Algorithm 8. In Figure 6.4, the computational time, without parallelization, is illustrated. At $k = 1$, the computational cost of finding an "oracle" is included. Notice that a backward or forward implementation has the same complexity for the same number of samples $\overline{n}$. Figure 6.5 illustrates the values of J_{sub} for each iteration. Notice, overall, that even for a small number of samples, the performance of the closed loop system is significantly improved and the computational time is promising, even for a non–parallel implementation. Also, as the iteration k advances, the initially modest performance increases significantly, converging to the optimal sequence as discussed in

Section 6.5.2. Interestingly, even for $\overline{n} = 5$, at iteration $k = 7$, the cost $J_{sub}(7)$ is smaller than the cost computed via `fmincon`, which, due to local minima, provided a feasible but not optimal solution. Figure 6.6 illustrates $J(x_{k|k}, U(k)) - J(x_{k|k}, U_{fmincon}(k))$, to gain insight in the contraction of $J(x_{k|k}, U(k)) - J(x_{k|k}, U^*(k))$ with increasing time k. Most of the input sequences, computed with any number of samples, converge generally to the optimal sequence.

In this experiment, due to the starting $U_{warm}(0)$, the cost J_{sub} for the forward implementation decreases faster than for the backward implementation. However, for other $U_{warm}(0)$, the cost for the backward implementation has a faster convergence. For this reason, there exists no clear distinction in terms of convergence for the two implementations.

For the second experiment we aim to test the computational complexity of Algorithm 8 in terms of the control horizon. We use a backward implementation without parallelization. Fix $\overline{n} = 10$ and N takes values in the set $\{3, 10, 20, 50, 100\}$. The results are illustrated in Figure 6.7. With red, the computational complexity for Algorithm 8 is depicted, and it is always smaller than the bound (6.9). It is expected that on dedicated devices, the complexity of Algorithm 8 is even smaller, due to the processors not running in parallel threads related to other system applications. With parallelization, further reduction in complexity is expected, as described in Section 6.4 and illustrated in Figure 6.7. for large horizons in fact, the computational complexity achieved with sampling is already smaller than the bound for parallel computation (6.10). Notice that, for all the horizons, the complexity of Algorithm 8 is smaller than the complexity of `fmincon`, even without parallelization. For $N = 100$, `fmincon` provides solutions which are not feasible, while the proposed Algorithm 8 still terminates in less than 0.05 seconds.

6.7.2 Buck–Boost power converter

The bilinear model of a Buck–Boost power converter is considered, as in (Spinu et al., 2011):

$$x(k+1) = Ax(k) + Bu(k) + \begin{bmatrix} x(k)^T C_1 \\ x(k)^T C_2 \end{bmatrix} u(k), \tag{6.36}$$

where $x := \begin{bmatrix} v_C & i_L \end{bmatrix}^T \in \mathbb{X} \subset \mathbb{R}^2$ is the state vector consisting of the voltage across the output capacitor and the current through the filter inductor. The input $u := \begin{bmatrix} d_1 & d_2 \end{bmatrix}^T \in \mathbb{U} \subset \mathbb{R}^2$ stands for the duty–cycle ratio of the control signal applied to the switching node. The parameters are

$$A = \left(I_2 + T_s \begin{bmatrix} -\frac{1}{R_H C} & 0 \\ 0 & -\frac{R_L}{L} \end{bmatrix} \right), B = \begin{bmatrix} 0 & 0 \\ \frac{v_s}{L} & 0 \end{bmatrix} T_s,$$

$$C_1 = \begin{bmatrix} 0 & 0 \\ 0 & \frac{1}{C} \end{bmatrix} T_s, C_2 = \begin{bmatrix} 0 & -\frac{1}{L} \\ 0 & 0 \end{bmatrix} T_s,$$

with the values $R_L = 0.2\Omega$, $C = 22\mu F$, $L = 220\mu H$, $T_s = 10\mu s$.

The aim of the control loop is to stabilize the system to the equilibrium point $x_e := \begin{bmatrix} 20 & 0.5 \end{bmatrix}^T$, $u_e := \begin{bmatrix} 0.81 & 0.4 \end{bmatrix}^T$, under the constraints $i_L \in \mathbb{R}_{[0,3]}$, $v_C \in \mathbb{R}_{[-0.1,22.5]}$,

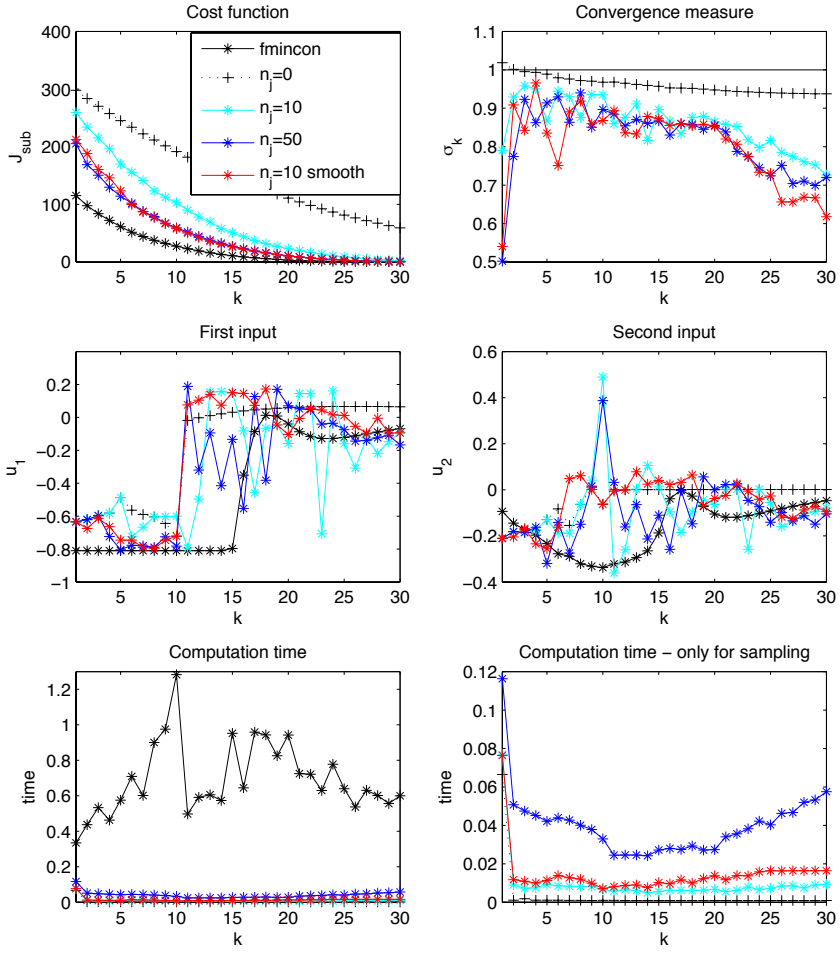


Figure 6.8: An illustration of the results of applying Algorithm 8 for the power converter, with varying number of sample points n_j, at each time step k; from left to right and up–down: the cost function J_{sub}, the measure of convergence σ_k from (6.12), the inputs u_1 and u_2, the computational time of SDNMPC in comparison with the solution provided by fmincon, and a close–up illustration of the computational time only on the SDNMPC solution.

$u \in \mathbb{R}^2_{[0,1]}$. The terminal controller

$$u = u_e + K(x - x_e), K = \begin{bmatrix} -0.0014 & -0.3246 \\ 0.0001 & -0.0055 \end{bmatrix},$$

stabilizing the system in the terminal set $\mathbb{X}_T$ given in (Spinu et al., 2011, set $\mathbb{P}$ in Section IV.C), and the quadratic cost with the matrices

$$Q = diag(1, 2), R = diag(1, 1), P = \begin{bmatrix} 46.6617 & 42.8039 \\ 42.8039 & 69.4392 \end{bmatrix},$$

satisfies all the conditions for stability formulated in Remark 6.2.2.

Similarly to the previous example, we apply Algorithm 8 for the MPC control of system (6.36), with only the backward implementation. Fix the initial state $x_{0|0} = \begin{bmatrix} 1 & 2 \end{bmatrix}^T + x_e$, which is outside of the terminal set $\mathbb{X}_T$, and $N = 10$. $U_{warm}(0)$ is given by an oracle and we use random sampling of $\mathbb{U}$. One of the experiments uses the same number of samples as a previous experiment, i.e., $n_j = 10$, but the inputs are smoothened as in Section 6.6, with $\alpha = [0.1\ 0.1]^T$.

See in Figure 6.8 the effect of varying $\overline{n}$ on the cost function J_{sub} and the convergence measure σ_k, the inputs and the computation time, per iteration. Notice that, for all the considered cases, the convergence is maintained at all times. Increasing the number of samples generally increases the convergence of the current cost function to the optimal cost function. However, this is not always the case, especially in randomized sampling. Smoothing the inputs, in this case, had a positive influence on the convergence rate as well, which became at times better than the convergence rate for an experiment with five times more samples. This is caused by the fact that smoothing had an effect of reducing the strong variations of the inputs caused by the sampling mechanism. Thus, when we have already reached a region in the set $\mathbb{U}$ which is close to the optimum, choosing the new sample input in a neighborhood of the previous input increases the chance that the new input is close to the optimal input as well.

The last two plots in Figure 6.8 show the effect of varying $\overline{n}$ on the time necessary, per iteration, to compute the corresponding control law. Notice the effect of the unknown termination time on the evaluation time for the optimization performed through `fmincon` and the relatively equal computational time of Algorithm 8 over the iterations k. Also, the computational time in case of the sampling–based strategy is significantly smaller compared to `fmincon`. It is expected that, through implementation on dedicated multi–thread systems, the computational time for the control law decreases to the extent of fitting the tight sampling period of the power converter.

6.7.3 Wheeled mobile robot

The last example illustrates the methodology developed in this chapter for an obstacle avoidance task by a nonholonomic system with trigonometric nonlinearities, due to kinematics, i.e., a model of the wheeled mobile robot (WMR), as described in (Kuhne et al., 2005):

$$x(k+1) = \begin{bmatrix} x_1(k) + u_1(k)\cos x_3(k)T_s \\ x_2(k) + u_1(k)\sin x_3(k)T_s \\ x_3(k) + u_2(k)T_s \end{bmatrix}. \tag{6.37}$$

In (6.37), the state $x \in \mathbb{R}^3$ describes the position and the orientation of the robot with respect to a global inertial frame $\{O, X, Y\}$, and the input $u \in \mathbb{R}^2$ gives the linear and angular velocity, respectively. The parameter $T_s = 0.1s$ is the discretization period of system (6.37).

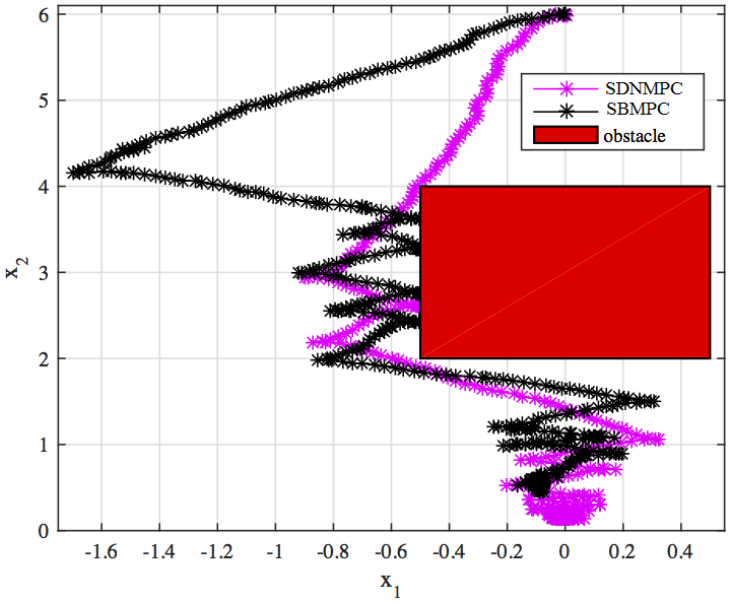


Figure 6.9: Trajectories of WMR.

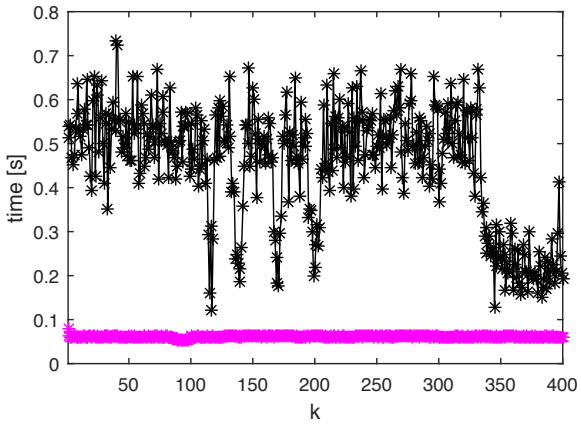


Figure 6.10: Time performance of WMR.

The MPC strategy aims at driving the WMR from an initial state $x_{0|0} = \begin{bmatrix} 0 & 6 & 0 \end{bmatrix}^T$

to the origin of the inertial frame, i.e., $x_g = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T$, while satisfying the input constraints $u_1 \in \mathbb{R}_{[-0.47,0.47]}$, $u_2 \in \mathbb{R}_{[-3.77,3.77]}$. A quadratic cost function of the form

$$J(x_{k|k}, U(k)) = x_{k+N|k}^T P x_{k+N|k} + \sum_{j=1}^{N-1} x_{k+j|k}^T Q(j) x_{k+j|k} + \sum_{j=0}^{N-1} u_{k+j|k}^T R u_{k+j|k}$$

is considered, with the parameters: $Q(j) = 2^{j-1}Q$, $P = 50Q(N)$, $Q = diag(1, 1, 0.5)$, $R = diag(0.1, 0.1)$, $N = 5$.

In this example we illustrate Algorithm 8 with the above parameters. Consider $U_{warm}(0)$ generated by an "oracle", and sampling of $\mathbb{U}$ based on random sequences. The result for $\overline{n} = 30$ is illustrated in Figure 6.9. `fmincon` could not be applied due to feasibility issues related to non–convexity caused by the presence of the obstacle. Notice the avoidance of the obstacle of the WMR under the control law generated by Algorithm 8. For comparison we illustrate also the result using SBMPC (Dunlap et al., 2010), with the same horizon N and $\overline{n} = 30$. Notice, in the state trajectory plot in Figure 6.9 the fact that, after 400 iterations, the WMR did not reach x_g yet, and the input values are not yet 0, which means that the sampling mechanism did not consider that the WMR is close to the goal. In this case, it is recommended to sample more densely around 0 in the set $\mathbb{U}$, rather than uniformly covering $\mathbb{U}$ with samples.

Notice in Figure 6.10 that, due to the building of a tree and the search in a tree for the best path, SBMPC is more computationally demanding. Even though in this example a more conservative case was implemented, with computing the complete cost $J_{sub}(k)$ for each time in the horizon, the complexity of Algorithm 8 with the given $\overline{n}$ fits the sampling period $T_s = 0.1s$ of the WMR, which makes the method applicable for real–time control of this system.

6.8 Conclusion

In this chapter, an algorithm, SDNMPC, based on sampling of the input space at each time in the horizon has been proposed, which iteratively improves in terms of cost an initially feasible control sequence. This suboptimal NMPC strategy provides a promising computational complexity even for large control horizons, with good perspectives for parallel implementation. This chapter has introduced also conditions for convergence of the solution of SDNMPC to the optimal solution of the NMPC problem, as well as a proposal for smoothing of the sampled inputs, to avoid over–use of the actuators. The algorithm provides a recursively feasible solution and the sampling of the input space can be stopped at any moment, as required in real–time by the sampling frequency of the system for which we design the control. The low computational complexity, finite–termination time, the convergence properties, the opportunities for achieving a smooth input and the real–time stability guarantees of SDNMPC make it a suitable solution for real–time NMPC. Future work aims at investigating the scalability of SDNMPC in terms of system dimension and the effect of parallelization on computational time for real–life applications.

Chapter 7

Conclusions and recommendations

7.1 Overview of the results

In this thesis we have considered the following two problems: (i) stability analysis (ii) control synthesis via MPC, with formal guarantees of real–time feasibility and stability, for nonlinear systems. Both problems require solving non–convex optimization problems. To avoid this, a sampling approach has been undertaken. For problem (i), we have developed methods for sampling–driven stability verification that verify the satisfaction of Lyapunov's inequality on a finite number of samples selected from the constraint set, and then extend the verification to an infinite, but bounded set inside the set of constraints. Then, based on the samples previously generated, a subset of the DOA is computed. Methods have been developed for the verification of both discrete–time as well as continuous–time systems. Deterministic methods have been developed for stability verification of nonlinear systems of relatively small state space dimension, and probabilistic sampling–based stability verification was proposed for nonlinear systems of larger dimension. For problem (ii), we have developed methods for sampling–driven NMPC with a bound on complexity quadratic in the prediction horizon and linear in the number of samples. Conditions for stability and convergence to the optimum have been established as well. The SDNMPC approach developed in this thesis guarantees recursive feasibility and it is suitable for real–time implementation of MPC for nonlinear systems. More precisely, the contributions for each problem can be summarized as follows.

7.1.1 Sampling–driven analysis

The first question that has been investigated in this thesis was question Q_1 posed in Chapter 1. Answering this question involves formulating the stability analysis problem via optimization problems and evaluating the limitations in solving analysis problems via optimization for constrained nonlinear systems. More specifically, for stability domains computation, LFs have to be computed. However, before even formulating a typical optimization–based stability analysis, we need to show how to constructively compute a LF for nonlinear systems, which is in itself an NP–hard problem, for general nonlinear systems. For this purpose, Chapter 2 develops a systematic approach for computing a LF on a compact set for nonlinear discrete–time systems. The developed solution relies on nonlinear optimization

and on a converse theorem, which is applied to obtain an explicit LF from a finite–step LF. It is shown in Chapter 2 that this approach can be applied to any Lipschitz continuous nonlinear dynamics with convex constraints. If the system dynamics is not Lipschitz continuous or the constraints are not convex, then the optimization problems may still be used, at the expense of a loss in formal guarantees. Additionally, there are limitations on the level of scalability of the corresponding optimization problems with the state space dimension. This chapter creates though the foundation for the following three chapters of the thesis, where a similar approach is undertaken to computing a subset of the DOA of the origin, due to the advantage that we can construct a LF from a "freely chosen" FSLF.

To avoid the limitations related to non–Lipschitz dynamics, non–convex optimization problems, feasibility and scalability, solving the corresponding optimization problem is replaced in the following chapters by a constructive, sampling–driven approach. As such, in Chapter 3, we have formulated a sampling–driven algorithm for verification of generic properties of the type $F(x) \leq (<)0$ on compact sets $\mathcal{S}$. This property can represent properties of interest for analyzing safety of constrained nonlinear, possibly discontinuous systems, which opens up the application of sampling–driven verification to hybrid systems. The proposed approach distributes the verification of the property on a finite sampling of a bounded set of states of interest. Then, it extends the validity of the property to an infinite, bounded set of states by automatically exploiting local continuity properties via interval analysis. Efficient state–space exploration is achieved using multi–resolution sampling and hyper–rectangles as basic sampling blocks. The operations that need to be performed for each sampling point in the state–space can be carried out in parallel, which improves scalability. The procedure returns a subset $\mathcal{A}$ of points in $\mathcal{S}$ which satisfy the given property.

The sampling–driven strategy developed in Chapter 3 is adopted in Chapter 4 for constructing DOAs for nonlinear systems, in both discrete–time, as well as continuous–time settings. A result has been introduced which offers a solution to the problem that the LF is zero at the origin, which does not allow verification of the inequality $F(x) < (\leq)0$ at the origin via a conservative inequality as proposed in Chapter 3. For discrete–time systems, the same converse result as in Chapter 2 is used to construct a LF from a FSLF. For continuous–time systems two different alternatives have been explored. The first proposal is to compute a LF for the discretized system and then verify its validity for the original continuous–time system via the sampling–driven Algorithm 3 in Chapter 3. This approach relies on the same arguments discussed in Chapter 1 about the legitimacy of abstractions. The alternative proposal is to construct approximations of a FTLF via polynomial approximations of the continuous–time dynamics. Then again, from the converse theorem, this FTLF can be exploited to compute a LF, which is verified as in Chapter 3. To this point, we use only a deterministic approach for computing a DOA estimate, because it provides the advantage of a rigorous certificate. However, the bounds a_{x_s} and b_{x_s} introduce conservatism, and they also reduce the scalability of this deterministic certificate with both the state space dimension, n, and with the step of the FSLF, i.e., M. The sampling strategy based on hyper–rectangle refinement has also an exponential growth with the number of axes n_r on which the refinement is performed (which is smaller or equal than n).

The problem of non–scalability with the system dimension and the value of the step M in the deterministic method presented in Chapter 4 for DOA computation is addressed in Chapter 5 by a non–deterministic alternative. Thus, we propose methods for computing

candidate LFs and DOAs for nonlinear systems in a constraint set $\mathcal{S}$, with probabilistic guarantees. These methods scale linearly with the step M and the state space dimension n. However, due to the rejection methods employed for sampling, the algorithms developed in this chapter may still suffer from the "curse of dimensionality". The number of samples which are selected to draw the conclusions regarding the stability certificate depends only on the desired accuracy and confidence, which are both specified by the user. The iterative strategy employed in these algorithms allows for constructively finding the set $\mathcal{S}_0$ which is an estimate of a DOA of the origin, with a small bound on the number of samples. This bound can be computed as in randomized methods for worst–case performance evaluation, rather than the Chernoff bound, which is used to compute the maximum number of samples for performance verification. Resorting to a probabilistic certificate, as in this chapter, is wise to the extent to which a relaxation in the certainty of the certificate is allowed in the system of interest.

7.1.2 Sampling–driven nonlinear model predictive control

In Chapter 5 we evaluate the extent to which we can attain guarantees of stability and real–time feasibility for NMPC when using a sampling–driven approach. It is observed that a sampling–driven NMPC can be posed in a suboptimal NMPC strategy, which we integrate via an algorithm, SDNMPC. This algorithm is based on sampling of the input space at each time in the horizon, to iteratively improve in terms of cost a warm start, i.e., an initially feasible control sequence. This suboptimal NMPC strategy provides resursive feasibility and complexity which is quadratic with the horizon and linear with the number of samples. This low complexity makes the approach computationally suitable for systems of large dimensions and with large control horizons. Also, conditions for convergence of the SDNMPC output to the optimal solution are discussed in this chapter. Many times it is necessary to have a smooth input sequence, which does not allow for excessively high variations in the input, to avoid damaging the actuators. To achieve this, we recommend bounding both along the time k, but also with the control horizon, the increase in the input amplitude. The guaranteed recursive feasibility, stability and improved cost function at every iteration, make SDNMPC suitable for real–time implementation.

7.2 Recommendations and future work

This thesis has explored the different aspects involved in solving problems (i)–(ii), which generally involve nonlinear, even non–convex optimization problems. These problems are mostly NP–hard. We have shown that sampling–driven methods can be effective in avoiding solving nonlinear optimization problems. The complex optimization problem has been distributed to verification on samples, which does not involve solving optimization problems, but introduces particular challenges. Thus, new questions have been revealed, as well as specific precautions and recommendations.

7.2.1 Recommendations and extensions

We start with recommendations for stability domains computation, with the deterministic approach, as follows.

- Notice that the sampling–driven methods formulated in Chapter 4 can compute automatically a stability domain in a set $\mathcal{S}$ where there might exist regions where Lya-

punov's inequality is not satisfied. This is achieved by the multi–resolution iterative sampling strategy. In contrast, for the optimization–based results in Chapter 2, the refinement of the candidate finite–step invariant set $\mathbb{X}$ in Algorithm 2 is not obvious. The same holds for the choice of another candidate FSLF V in Algorithm 1. As such, future work deals with the problems of automatically choosing $\mathbb{X}$ and maximizing $\mathbb{W}$. For the example in Chapter 2.4.2, the refinement of $\mathbb{X}$ is performed by selecting the new set $\mathbb{X}$ as $\frac{1}{2}\mathbb{X}$. Of course, a bisection type of approach can be followed for finding a finite–step invariant set $\mathbb{X}$, which exists, according to Remark 2.3.6 in Chapter 2. However, one should consider that existence of a set is guaranteed theoretically, but it may be practically very difficult to construct, in the case when the resulting step M is very large. This brings us to the next point of discussion.

- Particularly because of the use of FSLFs and finite–step invariant sets, the iterated maps, for instance, $V \circ G^M$, pose difficulties when M is large. In what concerns the deterministic sampling–driven result of Chapter 4, the difficulty is computational, because with each extra function in the map composition, the interval analysis problem becomes more computationally demanding. Secondly, because of iterating these maps over intervals, and because a function of an interval provides another interval, which is over–aproximating, when we have a map composition, then the over–approximation aggravates. In this case, the conservatism of the inequality (3.3) increases. The size of M impacts also the computability of the optimization programs in Chapter 2 for the verification of finite–step Lyapunov's inequality and finite–step invariance. A solution is to use from the start a FSLF V which gives a small M. This can be achieved by using simulation traces of the dynamics starting from a set of initial conditions and solving an LP problem to fit the parameters of a, e.g., polynomial candidate LF, as in (Kapinski et al., 2014), or by statistical learning strategies, see, e.g., the support vector machine approach in (Prokhorov and Feldkamp, 1999). Both these methods fit data from simulations to obtain polynomially parameterized LFs, which can be close to a true LF. This gives a high chance that the computed candidate function is a FSLF with a small step M, which can further be used within the verification methods proposed in this thesis. Also, notice that, from all the methods in this thesis, the probabilistic approach in Chapter 5 is the least affected by the size of M.

As to what concerns the DOA estimation via the non–deterministic approach in Chapter 4, notice the following:

- When using rejection methods, the rate of rejection has to be evaluated. If the rate of rejection is large, then the approach will suffer from dimensionality issues. In such a case we recommend upper–bounding the set in which we have to sample uniformly with a more tight polytope, as in (Sijs, 2012), or a set defined via suitable p–norms, for which there exist uniform sampling strategies, as explained in (Tempo et al., 2012). Generating methods for upper–bounding a set given by the level set of a non–convex function constitutes a research topic in itself. Alternative methods, which are not based on rejection, as in (Smith, 1984), should also be explored.

- If the rejection rate is small, then the computability of new samples will not suffer from dimensionality. However, along the iterations of Algorithm 7, the level set c

decreases. It is recommended, at each iteration step, to apply the rejection method not from the large set $\mathcal{S}$, but from a set which more tightly includes the new level set of the LF.

- Probably the most serious concern in the probabilistic approach is the issue revealed by the example in Chapter 5.6.1. The probabilistic certificate holds provided that the current set can be uniformly sampled. If the boundary of the set $\mathcal{S}$ is not sufficiently dense sampled, then the level set of the LF, which is a new set $\mathcal{S}_0$ to be sampled, might exceed the boundary of the set $\mathcal{S}$, especially in the cases when the set $\mathcal{S}_0$ is highly non–convex and non–uniform. Future work has to develop methods that avoid the situation when the boundary of the set $\mathcal{S}$ is exceeded by the new level set $\mathcal{S}_0$.

For SDNMPC, future work aims at investigating the scalability of SDNMPC in terms of system dimension and the effect of parallelization on computational time for real–life applications. Additional research is necessary to investigate:

- alternative solutions to guarantee recursive feasibility when a terminal set can not be found; for example, see the approach in (Ding et al., 2014), which imposes a constraint of decreasing energy of the control system at each time step, instead of terminal constraints. More specifically, for an energy function V, the constraint $V(x_{k+N|k}) \leq V(x_{k+N-1|k-1})$ is imposed. This approach is helpful also when the terminal set is small, or when we can not afford a large horizon. That is the case, in general, for path planning.

- adaptive smoothening; the additional constraints introduced for smoothening have to be relaxed when feasibility is threatened. The constraints can be reinforced when sampling on the complete input range is not necessary, or when it affects the performance of the SDNMPC algorithms, as in the last steps of the example in Section 6.7.3. In that case, for instance, when the system is already close to the goal, sampling of the complete input range introduces excessive wobbling around the goal, while the goal is found already.

- concrete strategies for obtaining a warm start, such as, e.g., the heuristic in (Dunlap et al., 2010), which should be automatically generated.

7.2.2 Future outlook

The methods developed in this thesis address systems without disturbances or model uncertainties. However, the problem addressed by this thesis in Chapter 1 is: *Can formal guarantees be attained for (complex) nonlinear systems in terms of stability and DOA estimation, and real–time feasibility and stability of NMPC, using a sampling–driven approach?* To fully address this question, it is necessary to consider nonlinear systems in real–life, in cases when the model of the system is not fully known, or when disturbances act on the system.

To analyze nonlinear systems with disturbances, we have to extend the stability verification to input–to–state stability (ISS). In this case, the disturbance space can also be sampled, as we do with the state space when analyzing stability. The same holds for systems with parametric uncertainty, when the interval on which the parameters vary is known. Then,

this interval may be sampled as well. However, additional research is necessary to evaluate the corresponding implications.

The formal guarantees in Chapters 2–4 require knowledge of an explicit model of the nonlinear system, through the map G. Many real–life nonlinear systems have at most a complex description in Matlab/Simulink, instead of a model defined via differential/difference equations, see, e.g., the Toyota engine model in (Watanabe and Ohata, 2014). Other systems, like in power networks, lack information about the dynamic subsystems in the model, due to either confidentiality, or complexity and heterogeneity in the system. In that case it is necessary to use simulations of the system trajectories directly to learn information about the system. In (Wang and Liu, 2013) for instance, for a certain class of nonlinear systems with only one equilibrium point, at the origin, the authors recommend using directly measured data from the system, relying on the high precision of the modern sensors or state estimation algorithms. Other methods at the core of data–based analysis are: computational intelligence, data–mining approaches and machine learning methods. Also, an extension of the methods developed in this thesis to stochastic systems is of interest.

In SDNMPC we have assumed also a fixed system model, G. If the model G is known and it is deterministic, then the methods developed in Chapter 6 apply with recursive feasibility guarantees. Else, if the model G is not explicitly known, assume that it can be used as a black box to obtain a predictive model. Then, if a terminal set, terminal cost and a terminal controller are known and the system is deterministic, then recursive feasibility can still be guaranteed. However, for systems with uncertainty or disturbances, methods for stochastic and respectively robust SDNMPC need to be further developed.

Appendix A

Detailed elaborations

A.1 Mean Value Theorem to obtain the Lagrange remainder

In what follows we will need to use the Mean Value Theorem to process the Taylor series expansion in (3.11), see (Berz and Hoffstätter, 1998, Section 2). However, the Mean Value Theorem can not be directly applied to multivariate functions, and therefore a one–dimensional function $f_R : [0,1] \to \mathbb{R}$ is introduced by the formula $f_R(s) = F_i(x_s + s(x - x_s))$, where x and x_s are given. Note that

$$f_R^{(v)}(s) = ([(x - x_s)\nabla]^v F_i)(x_s + s(x - x_s)),$$

and

$$f_R^{(v)}(0) = ([(x - x_s)\nabla]^v F_i)(x_s).$$

Denote

$$T_R(s, s_0, m) := \sum_{v=0}^{m} \frac{f_R^{(v)}(s_0)}{v!}(s - s_0)^v.$$

Notice that the formulae of T and T_R are similar, but T is multivariate, while T_R is univariate. Furthermore

$$\begin{aligned} T_R(1, 0, m) &= \sum_{v=0}^{m} \frac{f_R^{(v)}(0)}{v!} = \sum_{v=0}^{m} \frac{([(x - x_s)\nabla]^v F_i)(x_s)}{v!} \\ &= T(x, x_s, m). \end{aligned}$$

Apply Taylor expansion formula to f_R around 0 and evaluate in $s = 1$:

$$\begin{aligned} f_R(1) &= T_R(1, 1, m) = T_R(1, 0, \infty) \\ &= T_R(1, 0, m) + remainder. \end{aligned}$$

Let us apply the Mean Value Theorem to T_R and an arbitrary function $g : \mathbb{R} \to \mathbb{R}$ with $g'(x) \neq 0$ on $(0, 1)$. There exists $\xi \in (0, 1)$ such that:

$$\frac{T_R(1, 1, m) - T_R(1, 0, m)}{g(1) - g(0)} = \frac{T_R'(1, \xi, m)}{g'(\xi)},$$

and therefore

$$T_R(1,1,m) = T_R(1,0,m) + \frac{g(1)-g(0)}{g'(\xi)} T'_R(1,\xi,m). \tag{A.1}$$

To analyse the expression of $T'_R(1,\xi,m)$ when we derivate T_R with respect to the second argument we see that

$$\begin{aligned} T'_R(1,\xi,m) &= \sum_{v=0}^{m} \left(\frac{f_R^{(v+1)}(\xi)}{v!}(1-\xi)^v - \frac{f_R^{(v)}(\xi)}{v!}(1-\xi)^{v-1} \right) \\ &= \frac{f_R^{(m+1)}(\xi)}{m!}(1-\xi)^m. \end{aligned} \tag{A.2}$$

If $g(\xi) = (1-\xi)^{m+1}$, then $g'(\xi) = -(m+1)(1-\xi)^m$. Notice that $g'(\xi) \neq 0$ for $\xi \in (0,1)$. Then, from (A.1) and (A.2) it follows that

$$T_R(1,1,m) = T_R(1,0,m) + L_m(x, x_s, \xi),$$

where

$$\begin{aligned} L_m(x,x_s,\xi) =& \frac{-1}{-(m+1)(1-\xi)^m}(1-\xi)^m \frac{f_R^{(m+1)}(\xi)}{m!} \\ =& \frac{f_R^{(m+1)}(\xi)}{(m+1)!} \\ =& \frac{([(x-x_s)\nabla]^{m+1} F_i)(x_s + \xi(x-x_s))}{(m+1)!} \end{aligned} \tag{A.3}$$

is the Lagrange remainder and $\xi \in (0,1)$.

A.2 Comparison between backward DP, rollout and SDNMPC

A.2.1 Backward DP

Considering the cost function (6.2) and the state prediction (6.3), in backward DP, the optimization problem which provides the optimal input sequence $U^*(k)$ for a given initial state $x_{k|k}$ is computed via backward iterations, as follows (Bertsekas, 2005a):

$$\begin{aligned} \min_{U(k)\in\mathbb{U}^N} J(x_{k|k}, U(k)) = & \min_{u_{k|k}, x_{k+1|k}, \ldots, u_{k+N-2|k}, x_{k+N-1|k}} \left[\sum_{i=0}^{N-2} L(x_{k+i|k}, u_{k+i|k}) \right] + \\ & + \min_{u_{k+N-1|k}, x_{k+N|k}} \left[L(x_{k+N-1|k}, u_{k+N-1|k}) + V_f(x_{k+N|k}) \right] \\ & \text{s.t. } x_{k+j|k} = f(x_{k+j-1|k}, u_{k+j-1|k}), \quad \forall j \in \mathbb{Z}_{[1,N]}, \\ & \quad x_{k+i|k} \in \mathbb{X}, u_{k+i|k} \in \mathbb{U}, \quad \forall i \in \mathbb{Z}_{[0,N-1]}, x_{k+N|k} \in \mathbb{X}_T. \end{aligned}$$

At the first step of a DP iteration, the problem to be solved is

$$\begin{aligned} J^{\circ}_{N-1}(x_{k+N-1|k}) = \min_{u_{k+N-1|k}, x_{k+N|k}} & \left[L(x_{k+N-1|k}, u_{k+N-1|k}) + V_f(x_{k+N|k})\right] \\ \text{s.t. } & x_{k+N|k} = f(x_{k+N-1|k}, u_{k+N-1|k}), \\ & u_{k+N-1|k} \in \mathbb{U}, x_{k+N|k} \in \mathbb{X}_T, \end{aligned}$$

where $x_{k+N-1|k}$ is a parameter which is not known at this step. Thus, in backward DP, the optimal cost and decisions at the last stage are parameterized by the state at the previous stage. Then, $J^{\circ}_{N-1}(x_{k+N-1|k})$ is called the optimal cost to go from state $x_{k+N-1|k}$ to the last state, $x_{k+N|k}$, under optimal control law $u^{\circ}_{N-1}(x_{k+N-1|k})$.

Similarly, at the next stage of the DP recursion, the problem to be solved is

$$\begin{aligned} J^{\circ}_{N-2}(x_{k+N-2|k}) = \min_{u_{k+N-2|k}, x_{k+N-1|k}} & \left[L(x_{k+N-2|k}, u_{k+N-2|k}) + J^{\circ}_{N-1}(x_{k+N-1|k})\right] \\ \text{s.t. } & x_{k+N-1|k} = f(x_{k+N-2|k}, u_{k+N-2|k}), \\ & x_{k+N-1|k} \in \mathbb{X}, u_{k+N-2|k} \in \mathbb{U}, \end{aligned}$$

where again $x_{k+N-2|k}$ is a parameter. The procedure repeats until $J^{\circ}_{N-1}(x_{k|k})$, where the first state $x_{k|k}$ is known and the input sequence $U^*(k)$ can be computed. The DP solution to MPC yields the implicit MPC control law $u^{\circ}_i(\cdot)$ for all $i \in \mathbb{Z}_{[0,N-1]}$, thus providing the optimal policy $U^*(\cdot) = \left\{u^{\circ}_0(\cdot), u^{\circ}_1(\cdot), \ldots, u^{\circ}_{N-1}(\cdot)\right\}$. DP is a general solution, however, it is more practical for linear systems. When the system is nonlinear, the implicit MPC control law is very difficult to obtain. For that reason, in NMPC with DP, ADP is used, with the difficulties mentioned in Section 6.2.2.

We specify, however, that the backward version of SDNMPC resembles DP because of selecting, at each stage j, the best cost (from within the n_j available samples), only from the part of the total cost $J(x_{k|k,U(k)})$ from j to $N-1$.

A.2.2 Rollout

In rollout algorithms, see, e.g., (Bertsekas, 2005b), a trajectory is defined as a sequence of inputs and states:

$$\left(x_{k|k}, u_{k|k}, x_{k+1|k}, u_{k+1|k}, \ldots, u_{k+N-1|k}, x_{k+N|k}\right),$$

where $x_{k+j|k} = f(x_{k+j-1|k}, u_{k+j-1|k})$, for all $j \in \mathbb{Z}_{[1,N]}$. A partial trajectory, generated by a base policy $U_{warm}(k)$, is defined as

$$H(x_{k+j|k}) := \left(x_{k+j|k}, u_{k+j|k}, x_{k+1|k}, u_{k+1|k}, \ldots, u_{k+N-1|k}, x_{k+N|k}\right).$$

The cost corresponding to the partial trajectory $H(x_{k+j|k})$ is:

$$\tilde{J}(x_{k+j|k}) = \sum_{i=j}^{N-1} L(x_{k+i|k}, u_{k+i|k}) + V_f(x_{k+N|k}).$$

The rollout algorithm starts at stage 0 and it sequentially proceeds to the last stage, $N-1$. At stage j, it maintains a partial trajectory:

$$T_{k+j} := \left(x_{k|k}, \overline{u}_{k|k}, \overline{x}_{k+1|k}, \overline{u}_{k+1|k}, \ldots, \overline{u}_{k+j-1|k}, \bar{x}_{k+j|k}\right).$$

The algorithm starts with a partial trajectory $T_k = \left(x_{k|k}\right)$, at $j = 0$. For $j = 0 : N-1$, given the current T_{k+j}, we form, for each $u_{k+j|k} \in \overline{U}_{k+j}(\overline{x}_{k+j|k})$, a complete trajectory $T^c_{k+j}(u_{k+j|k})$ as follows:

$$T^c_{k+j}(u_{k+j|k}) = T_{k+j} \cup \left(u_{k+j|k}\right) \cup H(f(\overline{x}_{k+j|k}, u_{k+j|k})),$$

where $\overline{U}_{k+j}(\overline{x}_{k+j|k})$ is a chosen subset of $\mathbb{U}$, for which $T^c_{k+j}(u_{k+j|k})$ is a feasible trajectory. Then, we select from $\overline{U}_{k+j}(\overline{x}_{k+j|k})$ a control $\overline{u}_{k+j|k}$ that minimizes over $u_{k+j|k} \in \overline{U}_{k+j}(\overline{x}_{k+j|k})$ the cost from j to $N-1$, i.e.:

$$L(\overline{x}_{k+j|k}, u_{k+j|k}) + \tilde{J}(f(\overline{x}_{k+j|k}, u_{k+j|k})).$$

For this step, a DP type of recursion is used in rollout algorithms. To summarize, the rollout algorithm selects at each stage j the input

$$\overline{u}_{k+j|k} \in arg \min_{u_{k+j|k} \in \overline{U}_{k+j}(\overline{x}_{k+j|k})} L(\overline{x}_{k+j|k}, u_{k+j|k}) + \tilde{J}(f(\overline{x}_{k+j|k}, u_{k+j|k})).$$

Then, the partial trajectory T_{k+j+1} is created by adding the pair $\left(\overline{u}_{k+j|k}, \overline{x}_{k+j+1|k}\right)$ to T_{k+j} as follows:

$$T_{k+j+1} = T_{k+j} \cup \left(\overline{u}_{k+j|k}, \overline{x}_{k+j+1|k}\right),$$

where $\overline{x}_{k+j+1|k} = f(\overline{x}_{k+j|k}, \overline{u}_{k+j|k})$. In SDNMPC, according to Algorithm 8, the set $\overline{U}_{k+j}(\overline{x}_{k+j|k})$ is selected by sampling, and we do not rely on optimization to obtain the solution $u_{k_j|k}$. We only select the best candidate from $\overline{U}_{k+j}(\overline{x}_{k+j|k})$.

Additionally, in SDNMPC, conditions for stability are established as in Remark 6.2.2.

Bibliography

A. A. Ahmadi, M. Krstic, and P. A. Parrilo. A globally asymptotically stable polynomial vector field with no polynomial Lyapunov function. In *CDC-ECE*, pages 7579–7580, 2011.

B. Akbarpour and L. C. Paulson. Metitarski: An automatic theorem prover for real–valued special functions. *Journal of Automated Reasoning*, 44(3):175–205, 2010.

M. Alamir. *Stabilization of nonlinear systems using receding–horizon control schemes: a parametrized approach for fast systems*, volume 339. Springer London, 2006.

M. Althoff, O. Stursberg, and M. Buss. Safety assessment of autonomous cars using verification techniques. In *American Control Conference, 2007. ACC'07*, pages 4154–4159. IEEE, 2007.

M. Althoff, O. Stursberg, and M. Buss. Reachability analysis of nonlinear systems with uncertain parameters using conservative linearization. In *Proceedings of the 47th IEEE Conference on Decision and Control*, pages 4042–4048. IEEE, 2008a.

M. Althoff, O. Stursberg, and M. Buss. Verification of uncertain embedded systems by computing reachable sets based on zonotopes. In *Proceedings of the 17th IFAC World Congress*, pages 5125–5130, 2008b.

A. Astolfi. Optimization – An introduction. University Lecture. `http://www3.imperial.ac.uk/pls/portallive/docs/1/7288263.PDF`, 2006. Accessed: 2014-08-13.

E. M. Aylward, P. A. Parrilo, and J. J. E. Slotine. Stability and robustness analysis of nonlinear systems via contraction metrics and sos programming. *Automatica*, 44(8):2163–2170, 2008.

C. Baier, J. P. Katoen, and K. G. Larsen. *Principles of model checking*. MIT press, 2008.

F. Berkenkamp, R. Moriconi, A. P. Schoellig, and A. Krause. Safe learning of regions of attraction for uncertain, nonlinear systems with gaussian processes. In *55th IEEE Conference on Decision and Control*, Las Vegas, USA, 2016.

D. P. Bertsekas. *Dynamic programming and optimal control*, volume 1. Athena Scientific Belmont, MA, 2005a.

D. P. Bertsekas. Dynamic programming and suboptimal control: A survey from ADP to MPC. *European Journal of Control*, 11(4):310–334, 2005b.

M. Berz and G. Hoffstätter. Computation and application of Taylor polynomials with interval remainder bounds. *Reliable Computing*, 4(1):83–97, 1998.

S. P. Bhat and D. S. Bernstein. Finite–time stability of continuous autonomous systems. *SIAM Journal on Control and Optimization*, 38(3):751–766, 2000.

J. Bjornsson, P. Giesl, S. Hafstein, C. M. Kellett, and H. Li. Computation of continuous and piecewise affine Lyapunov functions by numerical approximations of the Massera construction. In *53rd IEEE Annual Conference on Decision and Control (CDC)*, pages 5506–5511, Dec 2014.

J. Björnsson, S. Gudmundsson, and S. Hafstein. Class library in C++ to compute Lyapunov functions for nonlinear systems. *IFAC-PapersOnLine*, 48(11):778–783, 2015.

F. Blanchini. Set invariance in control. *Automatica*, 35(11):1747–1767, 1999.

F. Blanchini and S. Miani. *Set–theoretic methods in control*. Springer Science & Business Media, 2007.

V. Blondel and J. N. Tsitsiklis. NP–hardness of some linear control design problems. *SIAM Journal on Control and Optimization*, 35(6):2118–2127, 1997.

R. V. Bobiti and M. Lazar. On input–to–state stability analysis of discrete–time systems via finite–time Lyapunov functions. In *19th IFAC World Congress*, Cape Town, South Africa, 2014a.

R. V. Bobiti and M. Lazar. On the computation of Lyapunov functions for discrete–time nonlinear systems. In *18th International Conference on System Theory, Control and Computing*, Sinaia, Romania, 2014b.

R. V. Bobiti and M. Lazar. A delta–sampling verification theorem for discrete–time, possibly discontinuous systems. In *18th International Conference on Hybrid Systems: Computation and Control*, Seattle, Washington, USA, 2015.

R. V. Bobiti and M. Lazar. A sampling approach to finding Lyapunov functions for nonlinear discrete–time systems. In *European Control Conference*, Aalborg, Denmark, 2016.

R. V. Bobiti, R. H. Gielen, and M. Lazar. Non–conservative and tractable stability tests for general linear interconnected systems with an application to power systems. In *4th IFAC Workshop on Distributed Estimation and Control in Networked Systems*, pages 152–159, Koblenz, Germany, 2013.

T. S. Buchanan, D. G. Lloyd, K. Manal, and T. F. Besier. Neuromusculoskeletal modeling: estimation of muscle forces and joint moments and movements from measurements of neural command. *Journal of applied biomechanics*, 20(4):367–395, 2004.

G. C. Calafiore. Random convex programs. *SIAM Journal on Optimization*, 20(6):3427–3464, 2010.

G. C. Calafiore, F. Dabbene, and R. Tempo. A survey of randomized algorithms for control synthesis and performance verification. *Journal of Complexity*, 23(3):301–316, 2007.

E. F. Camacho, T. Samad, M. Garcia-Sanz, and I. Hiskens. Control for renewable energy and smart grids. *The Impact of Control Technology, Control Systems Society*, pages 69–88, 2011.

M. C. Campi. Why is resorting to fate wise? a critical look at randomized algorithms in systems and control. *European Journal of Control*, 16(5):419–430, 2010.

M. Canale, L. Fagiano, and M. Milanese. Fast nonlinear model predictive control via set membership approximation: an overview. In *Nonlinear Model Predictive Control*, pages 461–470. Springer, 2009.

Á. Castillo and P. J. Zufiria. Cell mapping techniques for tuning dynamical systems. In *Global Analysis of Nonlinear Dynamics*, pages 31–50. Springer, 2012.

A. Chakrabarty, V. Dinh, M. Corless, A. E. Rundell, S. H. Zak, and G. T. Buzzard. Support vector machine informed explicit nonlinear model predictive control using low–discrepancy sequences. *IEEE Transactions on Automatic Control, preprint published on–line, DOI: 10.1109/TAC.2016.2539222*, PP(99), 2016.

H. Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *The Annals of Mathematical Statistics*, pages 493–507, 1952.

G. Chesi. *Domain of attraction: analysis and control via SOS programming*. Springer Science & Business Media, 2011.

T. X. T. Dang. *Verification and synthesis of hybrid systems*. PhD thesis, Institut National Polytechnique de Grenoble-INPG, 2000.

J. de Dios Ortúzar and L. G. Willumsen. *Modelling transport*. Wiley New Jersey, 1994.

L. De Moura and N. Bjørner. Z3: An efficient SMT solver. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340. Springer, 2008.

C. E. de Souza, A. Trofino, and J. De Oliveira. Robust H_∞ control of uncertain linear systems via parameter–dependent Lyapunov functions. In *Proceedings of the 39th IEEE Conference on Decision and Control*, volume 4, pages 3194–3199, 2000.

T. Deyle. `http://www.hizook.com/blog/2009/10/18/meka-robotics-humanoid-torso-and-anthropomorphic-hands`, 2009. Accessed: 2017-04-29.

X. Ding, M. Lazar, and C. Belta. LTL receding horizon control for finite deterministic systems. *Automatica*, 50(2):399–408, 2014.

A. I. Doban. *Stability domains computation and stabilization of nonlinear systems: implications for biological systems*. PhD thesis, Eindhoven University of Technology, 2016.

A. I. Doban and M. Lazar. Domain of attraction computation for tumor dynamics. In *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*, pages 6987–6992. IEEE, 2014.

A. I. Doban and M. Lazar. Computation of Lyapunov functions for nonlinear differential equations via a Massera–type construction. `arXiv:1603.03287[math.DS]`, 2016a. Accessed: 2016-03-11.

A. I. Doban and M. Lazar. Computation of lyapunov functions for nonlinear differential equations via a yoshizawa–type construction. *IFAC–PapersOnLine*, 49(18):29–34, 2016b.

dReal. Atrial Fibrillation Model. `http://dreal.github.io/benchmarks/atrial_fibrillation/`, 2017. Accessed: 2017-04-29.

P. S. Duggirala, S. Mitra, and M. Viswanathan. Verification of annotated models from executions. In *Proceedings of the Eleventh ACM International Conference on Embedded Software*, page 26. IEEE Press, 2013.

D. D. Dunlap, C. V. Caldwell, and E. G. Collins. Nonlinear model predictive control using sampling and goal–directed optimization. In *2010 IEEE International Conference on Control Applications*, pages 1349–1356. IEEE, 2010.

C. Fan and S. Mitra. Bounded verification with on–the–fly discrepancy computation. In *International Symposium on Automated Technology for Verification and Analysis*, pages 446–463. Springer, 2015.

V. Gan, G. A. Dumont, and I. Mitchell. Benchmark problem: A PK/PD model and safety constraints for anesthesia delivery. In *Proceedings of the 17th international conference on Hybrid systems: computation and control*, pages 1–8. ACM, 2014.

S. Gao, J. Avigad, and E. M. Clarke. δ–complete decision procedures for satisfiability over the reals. In *Automated Reasoning*, pages 286–300. Springer, 2012.

R. Geiselhart, R. H. Gielen, M. Lazar, and F. R. Wirth. An alternative converse Lyapunov theorem for discrete–time systems. *Systems and Control Letters*, 70(0):49 – 59, 2014. ISSN 0167-6911. doi: http://dx.doi.org/10.1016/j.sysconle.2014.05.007.

R. H. Gielen and M. Lazar. Non–conservative dissipativity and small–gain conditions for stability analysis of interconnected systems. In *51st IEEE Conference on Decision and Control*, pages 4187–4192, Maui, Hawaii, USA, 2012.

R. H. Gielen and M. Lazar. On stability analysis methods for large–scale discrete–time systems. *Automatica*, 55:66–72, 2015.

P. A. Giesl. On the determination of the basin of attraction of discrete dynamical systems. *Journal of Difference Equations and Applications*, 13(6):523–546, 2007.

P. A. Giesl and S. F. Hafstein. Revised CPA method to compute Lyapunov functions for nonlinear systems. *Journal of Mathematical Analysis and Applications*, 410(1):292–306, 2014.

P. A. Giesl and S. F. Hafstein. Computation and verification of Lyapunov functions. *SIAM Journal on Applied Dynamical Systems*, 14(4):1663–1698, 2015.

A. Girard. Reachability of uncertain linear systems using zonotopes. In *Hybrid Systems: Computation and Control*, pages 291–305. Springer, 2005.

A. Girard and G. J. Pappas. Verification using simulation. In *Hybrid Systems: Computation and Control*, pages 272–286. Springer, 2006.

M. P. C. in 't Groen. Constraint control of an interventional X–ray machine using sampling–based nonlinear model predictive control. MSc. thesis, 2017.

R. Grosu, G. Batt, F. H. Fenton, J. Glimm, C. Le Guernic, S. A. Smolka, and E. Bartocci. From cardiac cells to genetic regulatory networks. In *International Conference on Computer Aided Verification*, pages 396–411. Springer, 2011.

L. Grüne and J. Pannek. Nonlinear model predictive control. Theory and algorithms. London: Springer–Verlag, 2011.

S. Hafstein, C. Kellett, and H. Li. Computation of Lyapunov functions for discrete–time systems using the Yoshizawa construction. In *53rd IEEE Conference on Decision and Control*, Los Angeles, CA, USA, 2014a.

S. Hafstein, C. M. Kellett, and Huijuan Li. Continuous and piecewise affine Lyapunov functions using the Yoshizawa construction. In *American Control Conference (ACC), 2014*, pages 548–553, June 2014b.

L. Jaulin and E. Walter. Set inversion via interval analysis for nonlinear bounded-error estimation. *Automatica*, 29(4):1053–1064, 1993.

L. Jaulin, M. Kieffer, O. Didrit, and E. Walter. *Applied Interval Analysis*. Springer, 2001.

X. Jin, J. V. Deshmukh, J. Kapinski, K. Ueda, and K. Butts. Powertrain control verification benchmark. In *Proceedings of the 17th international conference on Hybrid systems: computation and control*, pages 253–262. ACM, 2014.

T. A. Johansen. Computation of Lyapunov functions for smooth nonlinear systems using convex optimization. *Automatica*, 36(11):1617–1626, 2000.

T. A. Johansen. Approximate explicit receding horizon control of constrained nonlinear systems. *Automatica*, 40(2):293–300, 2004.

P. Julian, J. Guivant, and A. Desages. A parametrization of piecewise linear Lyapunov functions via linear programming. *International Journal of Control*, 72(7-8):702–715, 1999.

J. Kapinski and J. V. Deshmukh. Discovering forward invariant sets for nonlinear dynamical systems. In *Proceedings of the International Conference on Applied Mathematics, Modeling and Computational Science*, 2013.

J. Kapinski, J. V. Deshmukh, S. Sankaranarayanan, and N. Arechiga. Simulation–guided Lyapunov analysis for hybrid dynamical systems. In *Proceedings of the 17th international conference on Hybrid systems: computation and control*, pages 133–142. ACM, 2014.

J.M. Keckler Medical Co. `http://hybridoperatingroom.com/`, 2017. Accessed: 2017-04-29.

C. M. Kellett. A compendium of comparison function results. *Mathematics of Control, Signals, and Systems*, pages 1–36, 2014.

H. K. Khalil. *Nonlinear systems*, volume 3. Prentice hall Upper Saddle River, 2002.

J. C. Knight. Safety critical systems: challenges and directions. In *Proceedings of the 24rd International Conference on Software Engineering, 2002. ICSE 2002.*, pages 547–550. IEEE, 2002.

I. V. Kolmanovsky and J. Sun. Parameter governors for discrete–time nonlinear systems with pointwise–in–time state and control constraints. *Automatica*, 42(5):841–848, 2006.

F. Kuhne, W. F. Lages, and J. M. G. Da Silva. Point stabilization of mobile robots with nonlinear model predictive control. In *IEEE International Conference Mechatronics and Automation, 2005*, volume 3, pages 1163–1168. IEEE, 2005.

P. Kundur. *Power system stability and control*. McGraw–Hill, 1994.

S. M. LaValle. Rapidly–exploring random trees: A new tool for path planning. *TR 98-11, Computer Science Dept., Iowa State University*, 1998.

M. Ławryńczuk. Computationally efficient nonlinear predictive control based on state–space neural models. In *International Conference on Parallel Processing and Applied Mathematics*, pages 350–359. Springer, 2009.

M. Lazar. *Model Predictive Control of Hybrid Systems: Stability and Robustness*. Ph.D. thesis. Eindhoven University of Technology, 2006.

M. Lazar and W. P. M. H. Heemels. Predictive control of hybrid systems: Input–to–state stability results for sub–optimal solutions. *Automatica*, 45(1):180–185, 2009.

M. Lazar, A. I. Doban, and N. Athanasopoulos. On stability analysis of discrete–time homogeneous dynamics. In *Proceedings of 17th International Conference on Systems Theory, Control and Computing*, pages 297–305, Sinaia, Romania, 2013a.

M. Lazar, W. P. M. H. Heemels, and A. R. Teel. Further input–to–state stability subtleties for discrete–time systems. *IEEE Transactions on Automatic Control*, 58(6):1609–1613, 2013b.

J. M. Lee and J. H. Lee. Approximate dynamic programming strategies and their applicability for process control: A review and future directions. *International Journal of Control Automation and Systems*, 2:263–278, 2004.

A. K. Lenstra, H. W. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, 1982.

M. Likhachev and A. Stentz. R* search. *Lab Papers (GRASP)*, page 23, 2008.

C. Luk. Domain of attraction in hybrid systems. *HKU Theses Online (HKUTO), The University of Hong Kong (Pokfulam, Hong Kong)*, 2015.

A. M. Lyapunov. The general problem of the stability of motion. *International Journal of Control*, 55(3):531–534, 1992.

L. Magni, D. M. Raimondo, and F. Allgöwer. *Nonlinear model predictive control: Towards new challenging applications*, volume 384. Springer, Lecture notes in Control and Information Sciences, 2009.

A. Majumdar, A. A. Ahmadi, and R. Tedrake. Control and verification of high–dimensional systems with DSOS and SDSOS programming. In *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*, pages 394–401. IEEE, 2014.

I. B. Makhlouf and S. Kowalewski. Networked cooperative platoon of vehicles for testing methods and verification tools. In *Proceedings of the 17th international conference on Hybrid systems: computation and control*, pages 37–42. ACM, 2014.

L. G. Matallana, A. M. Blanco, and J. A. Bandoni. Estimation of domains of attraction: A global optimization approach. *Mathematical and Computer Modelling*, 52(3):574–585, 2010.

D. Q. Mayne and J. B. Rawlings. Model predictive control: theory and design. *Madison, WI: Nob Hill Publishing, LCC*, 2009.

A. N. Michel, N. R. Sarabudla, and R. K. Miller. Stability analysis of complex dynamical systems. *Circuits, Systems and Signal Processing*, 1(2):171–202, 1982.

I. M. Mitchell, A. M. Bayen, and C. J. Tomlin. A time–dependent Hamilton–Jacobi formulation of reachable sets for continuous dynamic games. *Automatic Control, IEEE Transactions on*, 50(7):947–957, 2005.

Mitsubishi. `http://mitsubishi-motors.com.my/attrage#features`, 2017. Accessed: 2017-04-29.

N. Mohan. *Power Electronics: A First Course.* John Wiley & Sons, Inc, 2012.

R. E. Moore, R. B. Kearfott, and M. J. Cloud. *Introduction to interval analysis*. SIAM, 2009.

D. Muñoz de la Peña, T. Alamo, A. Bemporad, and E. F. Camacho. A dynamic programming approach for determining the explicit solution of linear MPC controllers. In *IEEE Conference on Decision and Control*, 2004.

R. M. Murray, Z. Li, and S. S. Sastry. *A mathematical introduction to robotic manipulation*. CRC press, 1994.

E. Najafi, R. Babuška, and G. A. D. Lopes. A fast sampling method for estimating the domain of attraction. *Nonlinear Dynamics*, pages 1–12, 2016.

L. V. Nguyen and T. T. Johnson. Benchmark: DC–to–DC switched–mode power converters (Buck converters, Boost converters, and Buck–Boost converters). In *Proceedings of the 17th international conference on Hybrid systems: computation and control*, pages 19–24. ACM, 2014.

S. Panikhom and S. Sujitjorn. Numerical approach to construction of Lyapunov function for nonlinear stability analysis. *Research Journal Applied Sciences, Engineering and Technology*, 4(17):2915–2919, 2012.

G. Pannocchia, J. B. Rawlings, and S. J. Wright. Conditions under which suboptimal nonlinear MPC is inherently robust. *Systems & Control Letters*, 60(9):747–755, 2011.

A. Papachristodoulou, J. Anderson, G. Valmorbida, S. Prajna, P. Seiler, and P. A. Parrilo. *SOSTOOLS: Sum of squares optimization toolbox for MATLAB*, 2013. Available: `http://www.eng.ox.ac.uk/control/sostools`.

J. L. Piovesan and H. G. Tanner. Randomized model predictive control for robot navigation. In *IEEE International Conference on Robotics and Automation, 2009*, pages 94–99, Kobe, Japan, 2009.

S. Prajna and A. Jadbabaie. Safety verification of hybrid systems using barrier certificates. In *International Workshop on Hybrid Systems: Computation and Control*, pages 477–492. Springer, 2004.

D. V. Prokhorov and L. A. Feldkamp. Application of SVM to Lyapunov function approximation. In *International Joint Conference on Neural Networks, 1999.*, volume 1, pages 383–387. IEEE, 1999.

R. Quirynen, S. Gros, and M. Diehl. Inexact Newton based lifted implicit integrators for fast nonlinear MPC. In *Proceedings of the IFAC Conference on Nonlinear Model Predictive Control (NMPC)*, pages 32–38, 2015.

D. M. Raimondo, D. Limon, M. Lazar, L. Magni, and E. F. Camacho. Min–max model predictive control of nonlinear systems: A unifying overview on stability. *European Journal of Control*, 15(1):5–21, 2009.

S. Ratschan and Z. She. Providing a basin of attraction to a target region of polynomial systems by computation of Lyapunov–like functions. *SIAM Journal on Control and Optimization*, 48(7):4377–4394, 2010.

P. A. Rubin. Generating random points in a polytope. *Communications in Statistics–Simulation and Computation*, 13(3):375–396, 1984.

S. M. Rump. INTLAB - INTerval LABoratory. In Tibor Csendes, editor, *Developments in Reliable Computing*, pages 77–104. Kluwer Academic Publishers, Dordrecht, 1999. URL `http://www.ti3.tuhh.de/rump/`.

S. Sankaranarayanan, H. B. Sipma, and Z. Manna. Constructing invariants for hybrid systems. In *International Workshop on Hybrid Systems: Computation and Control*, pages 539–554. Springer, 2004.

P. O. M. Scokaert, D. Q. Mayne, and J. B. Rawlings. Suboptimal model predictive control (feasibility implies stability). *IEEE Transactions on Automatic Control*, 44(3):648–654, 1999.

J. Sijs. *State estimation in networked systems*. PhD thesis, 2012.

R. L. Smith. Efficient Monte Carlo procedures for generating points uniformly distributed over bounded regions. *Operations Research*, 32(6):1296–1308, 1984.

S. E. Z. Soudjani and A. Abate. Adaptive gridding for abstraction and verification of stochastic hybrid systems. In *Quantitative Evaluation of Systems (QEST), 2011 Eighth International Conference on*, pages 59–68. IEEE, 2011.

V. Spinu, M. Lazar, and P. P. J. van den Bosch. An explicit state–feedback solution to constrained stabilization of DC–DC power converters. In *2011 IEEE International Conference on Control Applications (CCA)*, pages 1112–1118. IEEE, 2011.

P. Tabuada. *Verification and control of hybrid systems: a symbolic approach*. Springer Science & Business Media, 2009.

W. Tan and A. Packard. Stability region analysis using polynomial and composite polynomial Lyapunov functions and sum–of–squares programming. *IEEE Transactions on Automatic Control*, 53(2):565–570, 2008.

R. Tempo, E. Bai, and F. Dabbene. Probabilistic robustness analysis: Explicit bounds for the minimum number of samples. *Systems & Control Letters*, 30(5):237–242, 1997.

R. Tempo, G. Calafiore, and F. Dabbene. *Randomized algorithms for analysis and control of uncertain systems: with applications*. Springer Science & Business Media, 2012.

U. Topcu, A. Packard, and P. Seiler. Local stability analysis using simulations and sum–of–squares programming. *Automatica*, 44(10):2669–2675, 2008.

K. Tremblay. `https://clipartpig.com/download/yAR7TbJ`, 2017. Accessed: 2017-04-29.

R. van der Maas. *Advanced geometric calibration and control for medical X–ray systems*. PhD thesis, Eindhoven University of Technology, 2016.

J. A. W. van der Spek. *Cell mapping methods: modifications and extensions*. PhD thesis, 1994.

A. Vannelli and M. Vidyasagar. Maximal Lyapunov functions and domains of attraction for autonomous nonlinear systems. *Automatica*, 21(1):69–80, 1985.

M. Vidyasagar. *Nonlinear systems analysis*, volume 42. SIAM, 2002.

A. Wächter and L. T. Biegler. On the implementation of an interior–point filter line–search algorithm for large–scale nonlinear programming. *Mathematical programming*, 106(1): 25–57, 2006.

Z. Wang and D. Liu. Data–based stability analysis of a class of nonlinear discrete–time systems. *Information Sciences*, 235:36–44, 2013.

S. Watanabe and A. Ohata. Benchmark problem for near boundary operation control for automotive engine. In *Proceeding of 6th Conference on Simulation and Testing for Automotive Electronics*, pages 101–110, 2014.

V. M. Zavala and L. T. Biegler. Nonlinear programming strategies for state estimation and model predictive control. In *Nonlinear model predictive control*, pages 419–432. Springer, 2009.

Acknowledgements

Several efforts lay behind the research summarized in this thesis. The research could not have been fullfilled if it weren't for the continuous support of the people who have been there for me.

First and the foremost, I would like to express my gratitude to Mircea, for his continuous guidance during all the years. From the beginning of my Master studies, starting with my internship in Philips, during my Master project, and until now, I learned a lot from him. Many research discussions that we had, have been very challenging, but also very encouraging. Thank you for believing that I can make a nice contribution even in odd times. At times we did perceive things differently, but I think this has been a source of improvement for both of us. I want to thank you also for the advice during times of personal dilemmas and for many trips during conferences, of which I particularly remember Seattle, Cape Town and Sinaia.

Next, I would like to thank my promotor, Paul Van den Hof for his guidance and advice. Thank you for the support in funding my PhD research. I have particularly appreciated the discussions with you, which have always revealed missing pieces in the story of my PhD research. You have taught me how important it is to go beyond the comfort zone, question your assumptions and your line of thinking.

Also, I am very grateful to the members of my committee. Your comments have greatly improved my thesis. Particularly, I would like to thank Dr. Paul Goulart for the careful reading of my thesis and the very sharp suggestions from matters of relevant literature, to writing style. At the same time, I am grateful to Dr. Antoine Girard for pointing me to the literature on set inversion via interval analysis, as well as noticing possible extensions of the results in this thesis both in terms of alternative formulations of finite–step Lyapunov functions, and the main result of Chapter 3. Next, I would like to thank Prof. dr. Claudio De Persis for many insightful remarks. Finally, I would like to thank Dr. ir. Tamás Keviczky and Prof. dr. ir. Nathan van de Wouw for accepting to be part of the committee and for the positive evaluation of my contributions.

It serves to look back and reflect on the positive influence people had on my education. For this reason, I would like to thank my masters internship supervisor in Philips, Dr. Cristian Presura, for his great support during my internship. Cristi, your great enthusiasm for science has always inspired me, as well as many interesting discussions from outside the professional spectrum. Rob, thank you for co–supervising me during my masters and for the support after your left the university. Then, I would like to thank Dr. Ion Necoara, my supervisor in my Bachelor thesis. You have inspired me to look into model predictive con-

trol and to continue my studies in Control. Looking back on my early school, I would like to express my appreciation to Prof. Irina Zaman, for her guidance as a teacher and later as a friend. I hold very dear the memory of your visit to Eindhoven. A special thanks goes to Prof. Daniel Stretcu. You have been a great support for me to aim for performance in Math and beyond. Your dedication humbles me. Thanks to Prof. Nicolina Cârstina for supporting me in the competitions in Physics.

My four years as a PhD student have been really nice due to my colleagues. I have always appreciated our diversity and the joy we took from it! I will thank especially Alina for the nice time we had together, from the gym till coffee breaks, enjoying talking Romanian and sharing so many thoughts! Then, I want to thank Pepijn for so many discussions we had behind the wall, both technical and jokes. Thank you Tuan for believing I will be a safe, though tired, driver through Death Valley. Bahadir, I will miss the philosophical discussions with you. I would like to also thank Veaceslav, Mohsin, Dhruv, Ruben, Ryan, Edwin, Ana, David, Henrik, Sofie (our adventures in Aalborg), Can, Yanin, Marcella, Giuseppe (mamma mia, Venetia in Las Vegas), Daming (you should open a restaurant), Constantijn (you too), Esmail, Mohamed, Koen. Also special thanks go to Barbara, Diana, Lucia, Will and Udo, for making my stay in the group really smooth. Mark, thank you for being a self–driven student, in our collaboration for your masters project for control of the iXR machine. Rishi, it was very nice that we became friends so fast. Lieneke, thank you for the lunches and the concerts.

A PhD project runs at the same time with the ups and downs of our daily lives. The support of my friends throughout the years has been priceless. Alexandra, thank you for the continuous presence and friendship. George, Emilia, Andrei, knowing you since my master has been great fun! Iulia, Sonia, Madalina, Ana–Maria, Evert, Raja, Asma, Komal, Tania, Silvia, the Romanian community in Eindhoven, thank you greatly! Mehak and Shakila, shukriya! Evert, thanks for showing me how nice the dutch language can be and for the discussions on: Wie zijn wij?

This thesis would not have been possible without the support of my family. My mother, Luciana, and father, Angelu, had so much faith in me that, ever since I was a child, they trusted my actions and supported my education to, and beyond, the best of their possibilities. Thus, I want to thank my parents for giving me independence, of which I feel so much responsible. Also, I want to thank my brother Augustin and his young family for being always by my side.

Această teză nu ar fi fost posibilă fără sprijinul familiei mele. Mama mea, Luciana, şi tatăl, Angelu, au avut atât de multă credinţă în mine încât, încă de când eram copil, au avut încredere în întreprinderile mele şi mi–au susţinut educaţia dincolo de toate posibilităţile. Le mulţumesc părinţilor mei pentru dragostea şi independenţa pe care mi le–au oferit, de care mă simt profund responsabilă. În acelaşi timp îi mulţumesc fratelui meu, Augustin şi Denisei, pentru că îmi sunt mereu alături.

Finally I would like to thank to Umar for his care, for being permanently concerned with helping me achieve the best of myself. Your presence is a gift.

Ruxandra Bobiti,
Eindhoven, September 1, 2017.

Curriculum Vitae

Ruxandra Valentina Bobiti was born in Drobeta–Turnu Severin, Romania, on July 10, 1988. She has received her B.Sc. degree in System Engineering in 2011 from the University Politehnica of Bucharest, Romania. In 2011, she started her M.Sc. studies with a scholarship from the Eindhoven University of Technology, The Netherlands as part of the Talent Scholarship Program and received her (M.Sc.) degree in Systems and Control Engineering (cum laude) in 2013. Her M.Sc. thesis was focused on robustness and stability analysis of discrete–time systems via finite–step Lyapunov functions, with an application to power systems. Thereafter, she started working towards her Ph.D. degree in the Control Systems group at the same institution, under the supervision of Prof.dr.ir. P.M.J. Van den Hof and dr. M. Lazar.

Her main research interests include sampling–based methods for performance verification (particularly stability domains computation) and control of nonlinear systems. She has studied the application of these techniques on systems from domains such as biomedical, automotive, mechanical systems and power electronics.