Contributions to robotic control design with formal safety and stability guarantees

Philipp Schlehuber-Caissier

Sorbonne Université
Institut des Systèmes Intelligents et de Robotique

PhD Dissertation

Contributions to robotic control design with formal safety and stability guarantees

Philipp Schlehuber-Caissier

Advisors:
Vincent Padois
Nicolas Perrin

Reviewers:
Sylvain Calinon
Rafael Wisniewski

Examiners:
Nathalie Bertrand
Pascal Morin

November 12, 2018

Contents

List of Figures

List of Tables

Notation

Groups and sets

γ scalars are usually lowercase Greek letters, sometimes Latin letters

$\boldsymbol{v}$ vectors are usually bold lowercase letters, sometimes Greek letters

M matrices are usually uppercase Latin letters

$\mathbb{N}$ denotes the natural numbers

$\mathbb{Z}$ denotes the integers

$\mathbb{Q}$ denotes the rational numbers

$\mathbb{R}$ denotes the real numbers

$\mathbb{X}^{-|+}$ with $\mathbb{X} \in \{\mathbb{Z}, \mathbb{Q}, \mathbb{R}\}$ denotes the negative | positive integers, rational or real numbers

$\mathbb{X}_0^{-|+}$ with $\mathbb{X} \in \{\mathbb{Z}, \mathbb{Q}, \mathbb{R}\}$ denotes the non-positive | non-negative integers, rational or real numbers

$\mathbb{X}^n$ with $\mathbb{X} \in \{\mathbb{Z}, \mathbb{Q}, \mathbb{R}\}$ and $n \in \mathbb{N}$ denotes a column-vector of natural, rational or real numbers with n-elements

$\mathbb{X}^{n \times m}$ with $\mathbb{X} \in \{\mathbb{Z}, \mathbb{Q}, \mathbb{R}\}$ and $n, m \in \mathbb{N}$ denotes a matrix of n-rows and m-columns over the integers, rational or real numbers

$\mathbb{R}$ denotes the real numbers, scalars are usually lowercase Latin font

$\mathbb{S}^n$ denotes a quadratic symmetric matrix, so if $A \in \mathbb{S}^n$ then $A^{\mathrm{T}} = A$

$\mathbb{S}^n_{+|-}$ denotes the cone of symmetric semidefinite positive or negative matrices, so symmetric matrices with only non-negative or non-positive eigenvalues

$\mathbb{S}^n_{++|--}$ denotes the cone of symmetric definite positive or negative matrices, so symmetric matrices with only strictly positive or negative eigenvalues

$2^{\mathbb{X}}$ denotes the power set of the set $\mathbb{X}$, so the set of all possible sets that can be formed from the elements in $\mathbb{X}$. By abuse of notation we note $2^{\mathbb{R}^n}$ the (infinite) set of closed subset of $\mathbb{R}^n$.

∂S denotes the boundary of the set S

Algorithms

$\lambda(A)$ denotes the vector of eigenvalues of the square matrix A

$\lambda_{Max}(A)$ denotes the largest eigenvalue of the square matrix A

$\lambda_{Min}(A)$ denotes the smallest eigenvalue of the square matrix A

$C = \mathrm{chol}(A)$ denotes the cholesky factorization of a matrix $A \in \mathbb{S}^n_{++}$ such that $C^{\mathrm{T}}.C = A$

Chapter 1

Introduction

One of the key issues for robotic applications, or in a larger sense for all cyber-physical systems, is safety. Safety can have different implications depending on the field of work and the system concerned. However a very broad definition of safety can be given as ensuring that certain states or events never occur or that the consequences resulting from such unsafe events are bounded to an acceptable level, no matter the circumstances. For instance, in an environment shared by robots and humans, one can either demand that the robot never collides with any human or one could demand that if a collision occurs, the maximal impact force must be low in order to avoid injuries.
There are different ways to achieve safety, and one of them can be labelled as *passive* safety. It can either be directly ensured by the hardware, using for instance compliant actuators or shock absorbing coating, or by making appropriate changes to the control strategy. Adapting the hardware is not always possible or reasonable, as the solution has to be explicitly designed to prevent a specific consequence. This approach therefore provides no generic way to deal with safety concerns. For instance the collaborative robots Baxter and Sawyer have serial elastic actuators (SEA) to reduce the maximal impact force and absorb energy on (unexpected) impact. However, if the robotic arm is near a singular configuration during the impact, the SEA may not be able to reduce the impact force to an acceptable level. Passive safety can also be obtained in control laws by imposing constraints on the actuation and formally proving that, under an assumption of faithfulness for some physical model, these constraints do not allow undesirable behaviours such as excessive velocity (eventually relative to the obstacle) or force (see for instance Meguenani et al. [2016]). These approaches often lead to conservative solutions as they typically only reason about the near future. It can even occur that multiple, concurrently active safety constraints are incompatible, for instance in the case of multi-object avoidance, causing the problem to be infeasible . Such situations can occur as this approach so to speak misses informations about the context and (future) world state.

In this thesis a more active type of safety is pursued, strengthening the constraints on the control *strategy* rather than directly on the control outputs. The goal is to provide means to automatically generate control strategies that provide formal, model-based guarantees that no undesired states or events occur, considering all possible circumstances. This is a somewhat more ambitious goal than passive safety, in the sense that it does not seek to modify the inputs of the system based on additional constraints and an existing control strategy, but to *generate* a control strategy that can be safely executed without additional "run-time" constraints or hardware based safety. It is very important to note that what we aim to do is to generate controllers that can be considered safe *under the hypothesis* that the model used for the dynamics of the robot and its environment is sufficiently correct (assume-guarantee approach). With model-based approaches, making this assumption is unavoidable, but in real applications it should never be taken for granted. Making sure that the model is good enough, or refining it via extensive testing, is necessary, and only then can model-based approaches for safety such as ours be of relevance. In this thesis we use classical models of the robot dynamics, and suppose that they have been empirically verified. The approaches presented rely on formal proofs of safety based on a model, and while they give

absolute certainty for evolutions that happen according to the model, they are no more valid than the model itself, and therefore their use in practice must always be accompanied with important efforts of modelling. Even though this implies that the presented approaches do not provide the certainty of safety when used on a real robot, one of their interesting strengths is precisely that, by having a formal proof of safety, any failure brings valuable information on the limits of the model (or, as we will see in section 2.7, approximate abstractions of the model). Therefore, a simultaneous effort on empirical verifications of the model and on synthesis and tests of formally proven control strategies can be beneficial for the improvement of both the model and the controller and therefore increase the overall confidence in the considered system. Here we do not focus on designing and testing models (except for section 4.5.7), even though we present ways to abstract or approximate conveniently some of their properties. Our main focus is on the automated synthesis of control strategies that have formal guarantees of safety.

Generating such strategies is a computationally challenging task as synthesis is in general significantly more complex than verification problems when considering the same system. We are therefore exploring ways to define model-based computationally efficient abstractions for robotic systems that can be used to generate control strategies for a priori defined specifications. Such efficient constructions are possible in the case of controlled linear dynamics and to some extent for nonlinear dynamics. Moreover some of the proposed methods are also applicable in the context of general cyber-physical systems, considerably enlarging the possible field of application.

To this end, this thesis presents contributions to three different but interconnected research topics:

(1) Control law synthesis via timed automata abstraction

(2) Stabilizability of dynamical systems

(3) Learning stable vector fields from demonstration

for which the motivation and interconnections are summed up in the remainder of the introduction.

1.1 Control law synthesis via timed automata

The first approach considered in this thesis is to perform controller synthesis with the help of *formal methods*. The initial goal of formal methods is verification, which consists of automatically verifying that a given object (for instance a graph, or a finite automaton) complies with some specification expressed as a logical formula (e.g. “every state is reachable from any state”). Such methods were initially developed in the context of “verified software”, where some properties of the discrete steps of an algorithm are translated into , for instance, a finite state machine, which can then be used for verification.

More advanced formal methods include controller synthesis, a problem in which the interaction between a controller and the environment is modelled as a two-player game, one player being the controller and the other the environment, the adversary. The goal is to design algorithms that can generate controller strategies that are proven to respect a given specification no matter which action the adversary chooses (e.g. whatever happens, the controller can always force the system to eventually go back to its initial state). These tools are powerful because they can produce strategies that verify potentially complex specifications.

There is a long and rich history of scientific works concerned with the problem of using formal methods for controller synthesis, dating back as far as the early 1960’s (Church [1962]). A broad variety of techniques and more or less restrictive specification languages were henceforth developed, with a major breakthrough being the works of Rabin (Rabin [1969]) and Büchi (Büchi and Landweber [1969]) that independently proved the decidability of important synthesis problems using tree automata and infinite games respectively. Since then many works tackled the problem of synthesizing controller strategies for specifications given in different types of logics (Kress-Gazit et al. [2009]; Smith et al. [2011]) and for a multitude of different problems like scheduling or correct multi-agent behaviour. However, even though

these specification languages do allow for a rich variety of specifications and impressive results have been obtained, they lack the possibility to express *quantitative* constraints or dependencies on time. Indeed the specification languages used in the works cited above can only incorporate *qualitative* constraints on time (like linear temporal logic (LTL, see Pnueli [1977])), such as "state B is the successor of state A", but no quantitative statement like "state B is the successor of state A and has to be reached before x-time units have passed", can be made. This is partly due to the fact that such model checking tools were originally developed to analyse sequential computer programs or software and therefore the qualitative notion of time was sufficient. This changed with the occurrence and growing complexity of cyber-physical and hybrid systems. These systems have both, discrete states which can change instantaneously *and* continuous states or variables that often evolve according to dynamics described by differential equations. In this context it can be crucial to have a quantitative notion of time in order to describe the behaviour of the system in a correct or sufficiently precise manner. Therefore we pursue a different approach within this thesis relying on timed automata (TA), which provide an expressive framework to describe timed events. Also timed automata are a well established research field, providing many theoretical results and mature tools like UPPAAL (Behrmann et al. [2006]).

Timed automata (TA) have been introduced in the 90's by Rajeev Alur and David L. Dill (Alur and Dill [1994b]) and allow for the description and analysis of systems that have quantitative constraints on time. To be more specific, a timed automaton is a finite automaton equipped with a finite set of clocks that continuously evolve at a constant rate, but which can be reset during transitions. The verification of logical constraints on dynamical systems by abstracting them to timed automata (see Maler and Batt [2008]; Sloth and Wisniewski [2013]) and then checking reachability conditions on the resulting automata is a field of active research and yielded interesting approaches. These approaches are often computationally intensive as they rely on some form of discretization of the state-space which tends to scale badly with the size of the system. The usage of timed automata to represent controlled dynamical systems in order to generate control laws or strategies has not been explored as much, but nonetheless attracted a certain amount of attention (see Asarin et al. [1998]; Sloth and Wisniewski [2010b]). These approaches vary in the types of allowed specifications and the "coarseness" of abstraction, suiting the targeted application.

We developed a new representation for cyber-physical systems based on Lyapunov stability which is compatible with the theory of timed automata. In this representation the discrete states correspond to time-dependent invariant subsets in the state space of the system. Typically we use stabilisable regions around reference trajectories and derive conditions allowing to safely switch between them. The safety of these transitions can then be translated into guards on the transitions in the timed automaton, making the timed automaton a sound abstraction, as shown in Figure 1.1

Once the timed automata is constructed, a control law can be synthesized by specifying the set of undesired events or states as logical formula or auxiliary automaton and solving the reachability problem for the TA under these constraints. Therefore in this framework controller synthesis corresponds to the verification of the TA. As verification is in general less complex than synthesis we reduce the computational complexity, however at the price of possibly obtaining larger automata.

This approach has been implemented in Python and the verification software UPPAAL. Several numerical examples have been realised concerning, for instance, the control of a mass point in acceleration for non-trivial planing or the path-planning for the Dubins' car. Note that the dynamical system describing the Dubins car is a nonlinear and nonholonomic system, for which formally proven abstractions generated from Lyapunov theory are difficult to obtain. In order to treat this kind of systems, the definition of the discrete states is adapted: instead of formally proving their properties using Lyapunov theory, they are conjectured based on numerical results. Therefore the resulting control strategy is only guaranteed to be safe if the conjecture holds, but it potentially allows to synthesize control laws for systems for which proofs based on Lyapunov theory cannot be automatically generated. In this case each failure of the control strategy directly provides a trace pinpointing the violating conjecture and allowing to improve the coherence between the model and the abstraction by (repeated) refinement.

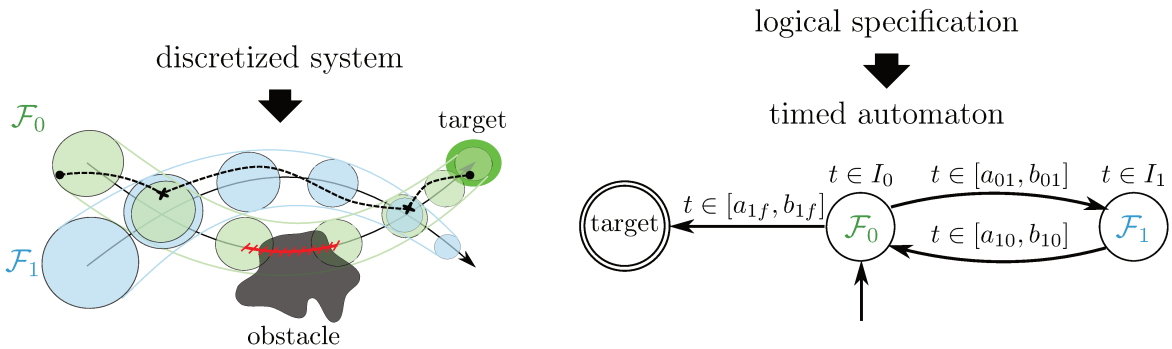


Figure 1.1 – Discretization of the dynamical system into two invariants (left) and the corresponding automaton (right) using standard syntax (presented in detail in section 2.2.1). The invariants I_1 and I_2 associated to the discrete states $\mathcal{F}_0$ and $\mathcal{F}_1$ guarantee that the system cannot collide with the obstacle. The dashed line on the left image shows a possible trajectory of the actual system. The initial and final state is indicated a the dot and the crosses correspond to a change of invariant. The times at which such a change can occur is constrained by the automaton.

1.2 Stabilizability of dynamical systems

This research topic directly emerged from needs encountered in topic 1) *Control law synthesis via timed automata*. To guarantee the specifications imposed onto the control law synthesis formulated as a reachability problem in a timed automaton, one needs correct, or more specifically sound, abstractions of the real system to construct the timed automaton. In our case the abstractions are sound if the discrete states correspond to positively invariant subsets of the dynamical system. This property is fairly easy to obtain for LTI-systems, but there exists no generic way to compute such invariants for arbitrary nonlinear systems especially when they are subjected to (input) constraints. Unfortunately, most cyber-physical systems, like robots, fall into this category. To tackle this problem we have developed a new approach to prove the convergence, or better stabilizability, of a dynamical, or more precisely polynomial, system with respect to a quadratic Lyapunov function. This approach combines ideas from optimal control and convex optimization and distinguishes itself from existing methods by the derived state space partitioning based on the optimal control input for each region as shown in Figure 1.2. We transform the question of stabilizability on a subset of the state space into an optimization problem. Due to the restriction to polynomial systems, efficient optimization methods exist to solve these problems, while granting a sufficiently high expressiveness. Specifically the differential equations describing robot movements contain many trigonometric functions, which can be approximated with a high accuracy using a truncated Taylor expansion resulting in a polynomial system further validating this choice.

As stability proofs are one of the core topics of control theory, a broad variety of approaches seeking to answer similar problems exists. Notably the use of Sum-of-Squares (SoS) methods gained a lot of attention during recent years (such as Tedrake et al. [2010b]; Majumdar et al. [2013a]), partly due to the appearance of readily available solvers for the associated convex optimization problems (Sturm [1999]; Andersen et al. [2013]). The advantage of the proposed formulation over such approaches is the separation of the proof of stabilizability from the synthesis of an actual control law. In SoS-approaches the stability of the system is proven by providing a tuple Lyapunov function, size of the region and control law and therefore the optimization problems are more complex as the introduced (polynomial) control law is part of the decision variables. In contrast the approach proposed in this thesis directly reasons on stabilizability, dissociating the question of proving stability from the synthesis of an actual control law, by partitioning the state space into input optimal regions. In these regions the control input maximizing

instantaneous convergence is the same, given a quadratic Lyapunov function. Note that, in contrast to SoS-techniques, we do not seek to modify the Lyapunov function, but only seek an as large as possible region of stabilizability for the given Lyapunov function. In order for this to work, the Lyapunov function has to be "suitable" for the considered system. This might seem like a drawback, but it will be shown that good results can be obtained relying on simple control techniques for linear systems, such as the Linear Quadratice Regulator (LQR).

The arising optimization problem corresponding to stabilizability on one of these subsets has a significantly simpler structure. However it can no longer be transposed into a standard semidefinite program, as the associated polynomial expression is nonconvex. Nonetheless it can be efficiently solved using techniques based on existing Reformulation-Linearization techniques, which solve nonconvex problems by constructing a convex (or even linear) underapproximation in a higher dimensional space. By constructing additional valid constraints the gap, that is the difference between the optimum of the original problem and the underapproximation, can be reduced to an acceptable level. In fact the gap often vanishes in practice and computation times can be reduced due to the simpler structure.

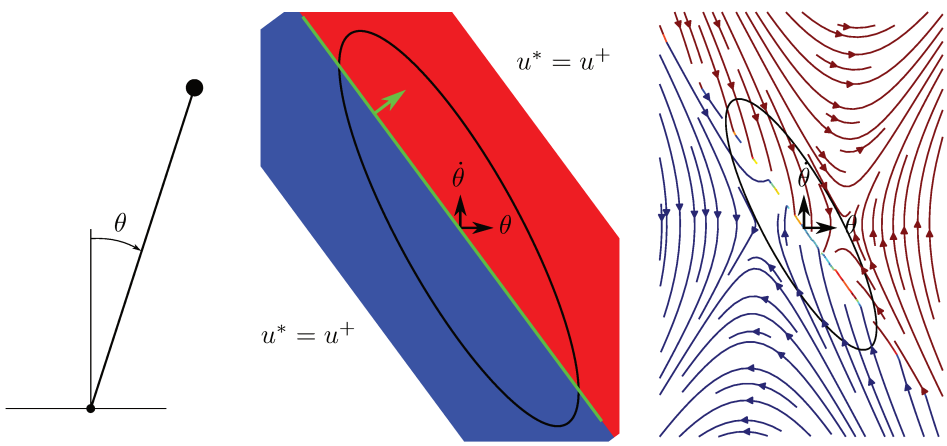


Figure 1.2 – Depiction of the proposed approach to prove stabilizability of a controlled dynamical system. The approach is based on the idea that the state space can be partitioned into subsets, within which the control input maximizing convergence is independent of the actual state considered (As long as the state belongs to the same subset). This partition is shown in the middle for the example of a torque controlled pendulum (depicted on the left) seeking to stabilize the unstable, upright position. On the right the streamlines of the system when using optimal control input (the control input is equivalent to the torque applied onto the pendulum in this example) are shown. In this case the state space is partitioned into two sets by a hyperplane (green line) passing through the origin and defined by its normal vector (green arrow), for which either the maximal control input u^+ (blue zone) or the minimal control input u^- (red zone) is optimal. The partitioning of the state space in this approach results from the system dynamics and a given Lyapunov function. A level-set of Lyapunov function considered here is shown as black ellipsoid.

This approach performs well in common test cases and often provides significantly larger regions of stabilizability (compared to the region of attraction derived with Sum-of-Squares methods).

1.3 Learning stable vector fields from demonstration

In the third research topic we investigate means to learn globally asymptotically stable nonlinear vector fields from demonstration. This is an interesting approach for multiple reasons. Firstly such vector fields provide a formal guarantee that all states are driven to a unique position: the equilibrium point of the vector field. Such vector fields have therefore a natural correspondence with reaching or grabbing motions, by interpreting the vector field as a velocity field for the robot in the task or joint space. Therefore learning nonlinear vector fields from demonstrations can be seen as a type of robot programming by demonstration and the difficulty is to guarantee stability while also faithfully recreating the demonstrations and to provide suitable generalization of the movement into neighbouring regions. Secondly, seen from a formal method point of view, such vector fields naturally translate certain linear temporal logic specifications to a corresponding velocity for a dynamical system. So given for instance the formula "eventually **position A**" and a globally asymptotically stable vector field with its equilibrium point at **position A**, then it is possible, independently of the current state of the system, to use the vector field to define the (desired) velocity of the system and it is guaranteed that **position A** will be reached and the specification is verified. Finally such learned vector fields can also be seen as generators of reference trajectories, which can then be used within the timed automata abstraction (research topic (1)) by constructing funnels around them. As it is assumed that the given demonstrations correspond to feasible trajectories for the concerned dynamical system, deducing an underlying vector field with good generalization properties allows to create reference trajectories between initial points "close" to the demonstrations ending at the equilibrium point. Due to the generalization of the movement by the vector field, these are likely to be suitable for the dynamical system. This is an interesting property as obtaining such trajectories is a generally a difficult task (for motion planning under dynamic constraints see for instance Plaku et al. [2010]). In this thesis the focus lies on the first point mentioned above, teaching a robot new movements via learning from demonstration, as it is the most direct application and also a large and very active research topic which we introduce hereafter.

The learning from demonstration (LfD) paradigm allows non-expert users to conveniently teach the robot new skills by constructing goal-driven behaviour from demonstration. Teaching a robot by demonstration is a very promising route to increase the flexibility and ease of use of robots that frequently encounter new tasks and for which no specifically trained staff is available, like service robots or manufacturing robots in small enterprises. Teaching someone/something by demonstration is very natural for humans and resolves many frequently encountered problems in motion planning like singular configurations, self collision and redundancy. The paradigm has a long and rich history in the robotics community with more and more commercial applications appearing (Baxter, Sawyer and others). The learning is normally based on well-known machine learning techniques training models like Gaussian mixture models (GMM) or Markov decision process (MDP) on the given data. An interesting way to encode motions is to represent them as (autonomous) dynamical systems, as this increases the motions inherent robustness to spatial and temporal perturbations. Despite the significant progress made in the last ten years, guaranteeing (global exponential) stability for the learned dynamical system without deteriorating the quality of reproduction remains a challenging problem.
To tackle this problem we propose a solution that takes advantage of the conservation of topological properties, such as convergence, under (smooth) diffeomorphic transformations. We can learn complex, highly nonlinear vector fields by constructing a diffeomorphic transformation that maps simple curves (mostly straight lines) which can be easily and faithfully reproduced by a "simple", globally exponentially stable vector field onto the given demonstration, see Figure 1.3. In this way, the diffeomorphic transformation increases the expressiveness of the simple dynamical system, while preserving the stability properties. In order to contruct the diffeomorphism we present a novel method to perform (approximative) diffeomorphic point matching, resulting in smooth transformations, which is advantageous when seeking to control second order dynamical systems (depicted in Figure 1.3).

Nonetheless this approach has several drawbacks, which are mainly due to the simplicity of the source curves used to construct the diffeomorphism and the corresponding simplicity of the control space

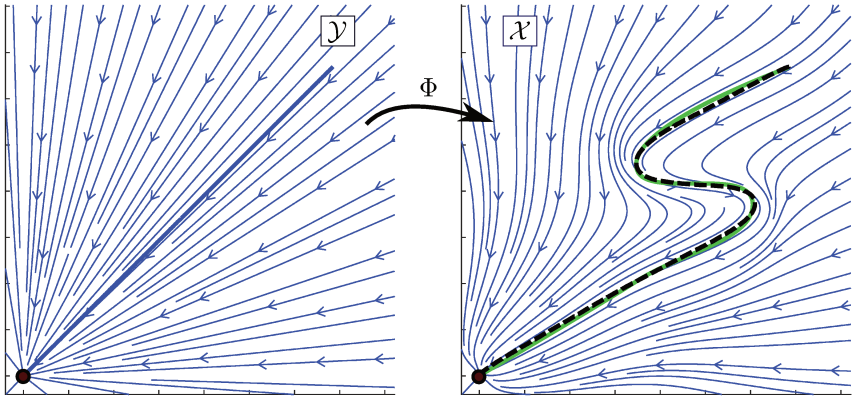


Figure 1.3 – This approach takes advantage of two spaces, the "naturally existing" demonstration space $\mathcal{X}$ (right image), the "artificially introduced" control space $\mathcal{Y}$ (left image), and the diffeomorphism Φ defined between them. We construct a *simple*, globally exponentially stable vector field in the control space reproducing the source curves (left image, blue line). Our method then constructs a diffeomorphic transformation Φ such that the image of this trajectory under the transformation (right image, dashed black line) matches the demonstration (right image, green line). The vector field in the demonstration space is the result of the control-space vector field and the transformation Φ. As one can see the resulting vector field in the demonstrations space represents well the demonstration while being stable.

dynamics, see Figure 1.3. It cannot take advantage of multiple demonstrations to better generalise the movement to unseen regions in the state space and often causes the high gradients in the diffeomorphism. To tackle these disadvantages, we developed several major modifications to the proposed approach based on the main idea of coupling machine learning techniques and diffeomorphic transformations, as shown Figure 1.4. This allows for learning more complex, but still globally asymptotically stable, control-space dynamics (described in detail in section 4.5.5) based on the demonstrations. Instead of constructing the diffeomorphism between a "simple" curve and the demonstration, in this approach we seek to directly construct the diffeomorphism between forward trajectories of the (learned) control space dynamics and the given demonstrations.

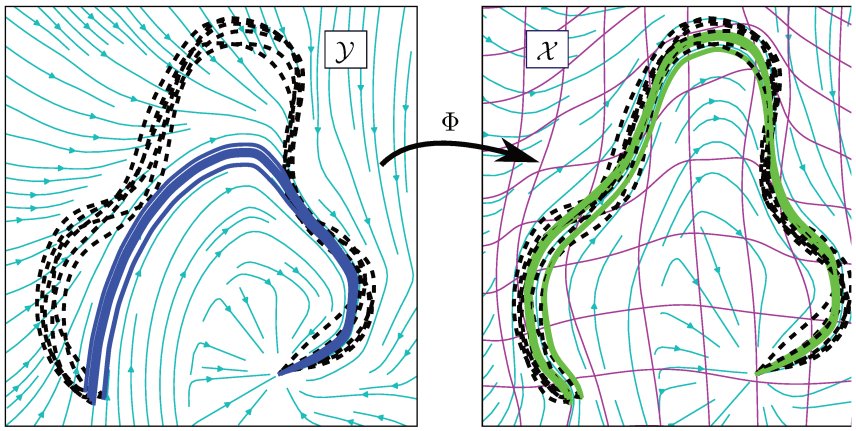


Figure 1.4 – The left image depicts the control space: We allow for more complex control-space dynamics learned from the given demonstrations while guaranteeing global asymptotic stability. The diffeomorphism Φ is then constructed between the forward trajectories in the control space (blue lines, left image) and the given demonstrations (dashed black lines, both images). The images of the control-space trajectories (green lines, right image) and the transposed control-space dynamics, called demonstration-space dynamics (right image, streamlines) faithfully reproduce the demonstrations and moreover generalise them to some neighbourhood. As the distance between the source (control-space trajectories) and the target curves (demonstrations) is reduced compared to the first approach, the diffeomorphism matching them is less complex and with smaller gradients. This is showcased by the image of a regular grid under the diffeomorphism shown in the right image (magenta lines).

Chapter 2

Timed-automata Abstraction of Controlled Systems

The development of formal methods for control design is an important challenge with potential applications in a wide range of safety-critical cyber-physical systems. Focusing on switched dynamical systems, we propose a new abstraction, based on time-varying regions of invariance (the *control funnels*), that models behaviours of systems as timed automata. The main advantage of this method is that it allows automated verification of formal specifications and reactive controller synthesis without discretizing the evolution of the state of the system. Efficient constructions are possible in the case of controlled linear and to some extent for nonlinear dynamics. We demonstrate the potential of our approach with two examples.

2.1 Introduction

Verification and synthesis are notoriously difficult for hybrid dynamical systems, i.e. systems that allow abrupt changes in continuous dynamics. For instance, reachability is already undecidable for 2-dimensional piecewise-affine maps (Koiran et al. [1994]), or for 3-dimensional dynamical systems with piecewise-constant derivatives (Asarin et al. [1995]). For systems governed by state-dependent differential equations, there also exist many negative results. The case of general nonlinear differential equations is only treatable using (conservative) approximations as in Asarin et al. [2003] or Althoff et al. [2008]. Even for the case of linear differential equations, exact reachability for the system $\dot{\boldsymbol{x}} = A.\boldsymbol{x}$ is decidable only if A has a certain eigenstructure[1], affecting its applicability for real-world systems (see Lafferriere et al. [1998]).

To enlarge the class of systems considered, switched (nonlinear) systems of the form $\dot{\boldsymbol{x}} = f_i(\boldsymbol{x})$ with $\boldsymbol{x} \in \mathbb{R}^d$ denoting a point in the state space, $\boldsymbol{x} \in \mathbb{R}^d$ being the velocity of the system and $f_i \in \{f_0, f_1, \cdots, f_N\}$ denoting the current (nonlinear) dynamics, belonging to the finite set $\{f_0, f_1, \cdots, f_N\}$. To enable automated logical reasoning on switched dynamical systems, most methods tend to entirely discretize the dynamics, for example by approximating the behaviour of the system with a finite-state machine. Alternatively, early work pointed out links between hybrid and timed systems (Maler et al.

[1] it has to be either nilpotent, have only real or only imaginary eigenvalues

[1992]), and several methods have been designed to create formal abstractions of dynamical systems that do not rely on a discretization of time, such as Frazzoli et al. [2005].

The contribution proposed in this chapter is a novel timed-automata abstraction of switched dynamical systems based on a particular kind of time-varying regions of invariance: control funnels. Recent results have shown that these invariants are very useful for robust motion planning and control (see for instance Tedrake et al. [2010a]; Majumdar and Tedrake [2013]; Majumdar et al. [2013a]), and that funnels or similar concepts related to the notion of Lyapunov stability or contraction control analysis can be used for formal verification of hybrid systems (Julius and Pappas [2009]; Duggirala et al. [2013]), and for reactive controller synthesis as in DeCastro and Kress-Gazit [2014].

The chapter is organized as follows: after a brief introduction to timed automata and an overview of related approaches in section 2.2, section 2.3 describes how control funnels, in particular for trajectory tracking controllers, can be used to create timed transition systems that abstract the behaviour of a given switched dynamical system, and as a result can potentially allow the use of verification tools to solve Reach-Avoid problems for this kind of systems. In section 2.4, we show how these timed transition systems can be encoded as timed automata. In section 2.5, we consider the case of linear dynamics and introduce the notion of fixed size LQR funnel, a convenient form of funnel used within the examples and present efficient algorithms to construct the funnel system and the corresponding timed automaton. In section 2.6 we present a variety of problem cases for which we can successfully perform controller synthesis reformulated as a reachability problem in a timed automaton deduced from a LQR-funnel system. These cases comprise motion planning for the Dubins' car, a nonholonomic system, and a Pick-And-Place scenario under logical constraints. In all cases the model checker Uppaal (Behrmann et al. [2006]) or its extension to timed games Uppaal-Tiga (Behrmann et al. [2007]) are used to solve the resulting TA. Finally section 2.8 concludes the chapter and presents some avenues for future work.

The contributions presented can be summed up as

Contributions in this chapter

- Presentation of a novel abstraction method for switched dynamical systems
- Reduction of such abstractions to timed automata
- Proof-of-concept by synthesizing control laws and strategies for different systems and problems

The material presented in this chapter is (partially) published in Bouyer et al. [2015, 2017].

2.2 Related Work and Background

2.2.1 Timed Automata

Timed automata were introduced in the early 90's as an extension to finite automata in Alur and Dill [1994b] and have gained great popularity within the formal verification community. But let us first introduce finite automata and build on that.

A finite automaton is described by a tuple $\mathcal{A} = (L, L_0, L_F, \Sigma, E)$ where L is a finite set of locations or states, $L_0 \in L$ is the initial state, $L_F \subset L$ is a set of final states, meaning that a run terminates when attaining any of these states, Σ is called the alphabet of the automaton and contains all labels associated to the edges and finally the set of edges $E \subset L \times \Sigma \times L$ of the form (ℓ, δ, ℓ'). Two locations $\ell, \ell' \in L$ are said to be connected if there exists an edge (ℓ, δ, ℓ'), also denoted $\ell \xrightarrow{\delta} \ell'$, in E for some letter δ of the alphabet Σ. We also say that there exists a transition between ℓ and ℓ' in the automaton. Here we are only interested in *deterministic finite* automata. That is the set L contains only a finite number of states and it is deterministic in the sense that all outgoing edges from a state have different labels. A

configuration of such an automaton is entirely described by the current location and a run corresponds to the list of labels associated to the edges or transitions that are consecutively taken. As this is a list of letters from the alphabet Σ, it is called a word over the alphabet Σ and the word is accepted, or in the language of the automaton, if it leads from the initial to a final state, as shown in Figure 2.1.

The reachability problem on such an automaton deals precisely with the question of finding such accepted words or runs. Such automata enjoy many decidable properties and reachability is one of them. There moreover exist many extensions to finite automata and some of them would be interesting in the context of this work, such as the extension to infinite words and Büchi acceptance conditions (Büchi and Landweber [1969]) as they could be used to describe repeated tasks in a robotic scenario, but for the moment we concentrate on the simplest form of reachability.

Such automata can describe the change of the state given an action to perform (among a finite number of possible actions represented by labels) and can therefore encode a *qualitative* notion of time, such as "first open the door then go through it then close the door". But such automata fail to express a *quantitave* notion of time, which is necessary to express conditions like "go through the door before 5 time units have passed". Such questions are however crucial as they naturally arise in many realistic scenario for robotic or other cyber-physical systems. For instance, consider the growing sector of embedded electronics and controlled systems. In such systems there exists a continuous evolution of the controlled system *and* discrete switches due to a change of the control strategy or a sudden change in the environment.

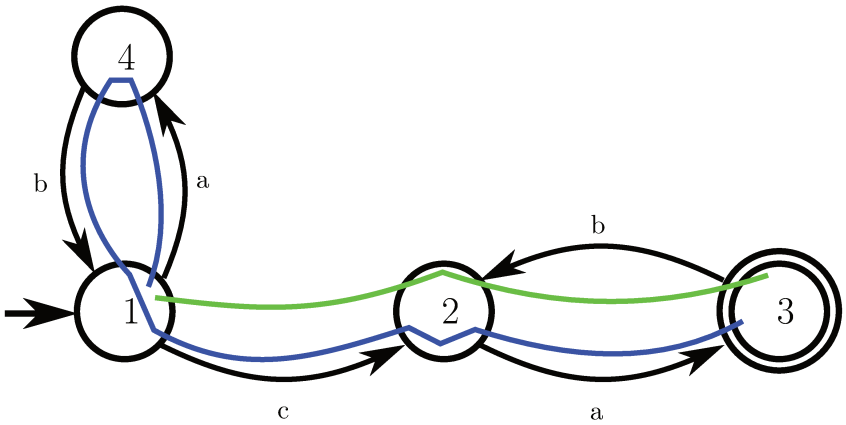


Figure 2.1 – A simple example of a deterministic finite automaton. The set of locations is given as $L = \{1,2,3,4\}$, the initial location is $L_0 = 1$ (indicated by the arrow without origin), the set of final locations is $L_F = \{3\}$. The alphabet is $\Sigma = \{a,b,c\}$. Two examples runs are showcased, given by the words (c,a) (green run) and (a,b,c,a) (blue run).

In order to be able to represent such cases, timed automata (TA) have been introduced in the early 90's. They correspond to a timed extension of finite automata and are defined by a tuple $\mathcal{A} = (L, L_0, L_F, C, \Sigma, E, \mathsf{Inv})$ where L, L_0, L_F and Σ are defined as before. C denotes a set of real-valued variables called *clocks*, $E \subset L \times \mathcal{C}(C) \times R(C) \times \Sigma \times L$ is the set of (timed) labelled edges defined by tuples of the form $(\ell, g_{\ell,\ell'}, \mathsf{res}_{\ell,\ell'}, \delta, \ell')$, extending the edges of the finite automaton by the clock guards $(g_{\ell,\ell'})$ and resets $(\mathsf{res}_{\ell,\ell'})$ detailed hereafter.

The (finite) set of clocks will allow for a quantitative notion of time. A *clock valuation*, equivalent to determining the current value the clocks, over the set C is a mapping $v\colon C \to \mathbb{R}_0^+$. We write $\mathbb{R}^C$ for the set of clock valuations over C. If $\Delta \in \mathbb{R}^+$, we write $v + \Delta$ for the clock valuation defined by $(v + \Delta)(c) = v(c) + \Delta$ for every $c \in C$, this corresponds to letting a certain amount of time pass. A *clock constraint* over C is a boolean combination of expressions of the form $c \sim \alpha$ where $\alpha \in \mathbb{Q}$, and $\sim \in \{\leq, <, =, >, \geq\}$. Note that it is only allowed to compare the clocks to constants and not to another clock, like $(c, c', \alpha) \in C \times C \times \mathbb{Q}$: $c \sim c' + \alpha$. This is called a diagonal guard and causes the

verification of the automaton to be more involved (see Bouyer et al. [2005]), leading to higher complexity of the verification process. As we will see, our approach does not depend on this kind of guards and we therefore only consider so-called diagonal-free TAs. We denote by $\mathcal{C}(C)$ the set of clock constraints over C. We write $v \models g$ if v satisfies g. That is if the current values of all clocks satisfy all atomic constraints, or better propositions, in g. A reset of the clocks is an element res of $(\mathbb{Q} \cup \{\perp\})^C$ (which we may write $R(C)$), and if v is a valuation, its image by res, denoted $\mathsf{res}(v)$, is the valuation mapping c to $v(c)$ whenever $\mathsf{res}(c) = \perp$ (the clock is unchanged), and to some predefined constant $\mathsf{res}(c) \in \mathbb{Q}$ otherwise (deterministic reset). In the framework of timed automata such resets can only take place during a transition.

Finally Inv is called the invariant labelling function, which assigns to each location a clock constraint that has to be verified while staying in this location. Note that different possibilities to define the tuple representing the timed automaton exist, including or excluding the explicit labelling of the transitions and the presence of invariants. These are however only differences in notation and do not restrict or extent the expressiveness of the automaton. Here the most explicit version is chosen, as in Sloth and Wisniewski [2010a].

In contrast to finite automata the configuration in a TA is no longer entirely determined by the current location ℓ, but by the pair specifying the current state and clock valuation, so by the pair $(\ell, v) \in L \times \mathbb{R}^C$. Or informally by the state and the "current time". The invariant labelling function $\mathsf{Inv}\colon L \to \mathcal{C}(C)$ assigns an invariant to each location, that is a set of clock constraints that has to be satisfied when in this state. Therefore the configuration pair introduced above always has to be such that $v \models \mathsf{Inv}(\ell)$. Finally the finite set of edges E is a subset off all possible transitions in the TA, defined by the tuples $(L \times \mathcal{C}(C) \times R(C) \times \Sigma \times L)$. That is, a tuple describing the initial location, the clock constraints (defining "when" the transition can be taken), the clock reset (defining the clock valuation after the transition) and the letter or symbol associated to this edge as well as the target location.

The configuration given configuration (ℓ, v) of the timed automaton $\mathcal{A}$ evolves according to the following two rules:

- *time-elapsing transition:* $(\ell, v) \to (\ell, v + \Delta)$ whenever $v + \delta \models \mathsf{Inv}(\ell)$ for every $0 \leq \delta \leq \Delta$;
- *switching transition:* $(\ell, v) \to (\ell', v')$ whenever there exists $(\ell, g, \mathsf{res}, \ell') \in E$ such that $v \models g \wedge \mathsf{Inv}(\ell)$, $v' = \mathsf{res}(v)$, and $v' \models \mathsf{Inv}(\ell')$.

A run in such an automaton is, much similar to the one in the case of finite automata, a sequence of consecutive transitions, or more specifically, a sequence of time-elapsing and switching transitions. This sequence, due to the labelling and the timing requirements is called a timed word, so a list of actions to take (element of Σ) and times to let pass ($\Delta \in \mathbb{R}^+$), as shown in Figure 2.2. Equally, a run can also be identified as a sequence of configurations, that is the run r is defined by $((\ell_0, v_0), (\ell_1, v_1), \ldots, (\ell_N, v_N))$ and for each run the associated timed word can be found and vice-versa.

Similar to the case of the finite automaton, the reachability problem in a TA asks to find an accepted timed-word for the given timed automaton with all clocks initialised to zero. This property is shown to be decidable and PSPACE-complete (Relying on the region constructiong, see Alur and Dill [1994b]), making timed automata an interesting framework for our use-case.

2.2.2 Related Work

In the literature a number of different approaches exist for the verification of (switched) dynamical systems and some also seek to perform formally verified controller synthesis, and reflect the growing interest in such approaches as well as the broad variety of different problems falling into this category.

Firstly the existing approaches for verification can roughly be divided into two groups, direct and indirect methods. Direct methods are able to reason directly about nonlinear differential equations by computing (over-)approximations of the reachable set, given the set of possible initial conditions. The main computational complexity of such approaches lies in the computation of these sets and several

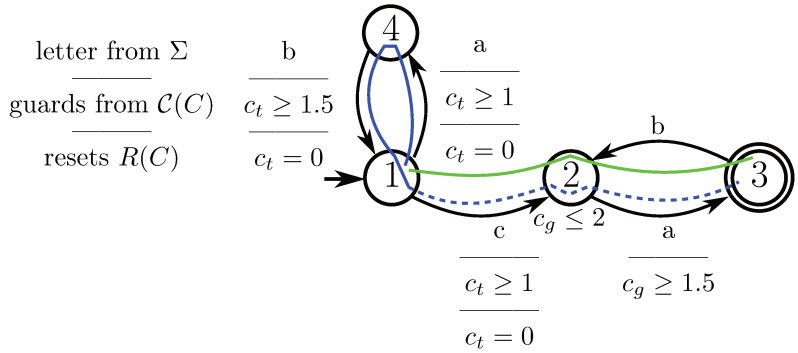


Figure 2.2 – A simple example for a timed automaton. The set of final states, the set of states, the initial state and the alphabet are the same as for the finite automaton. There are two clocks for this automaton, so $C = \{c_t, c_g\}$, initialized to zero when entering location 1. Location 2 is the only location with a non-trivial invariant. For the first run of the finite automaton (green), there exist (infinitely many) corresponding timed words, for instance ((1.1,c),(.6,a)). This means that we first let 1.1 time units pass while being in location 1 before taking the transition labelled with c to location 2. Note that before the transition we have $v(c_t) = 1.1$ and $v(c_g) = 1.1$ and after the transition (so after the reset) we have $v'(c_t) = 0$. and $v'(c_g) = 1.1$. Therefore $v(C)$ satisfies the transition guard and $v'(C)$ satisfies the invariant of location 2. Then we let another 0.6 time units pass by before taking the next transition labelled with a in order to attain a final state. The second example run in the finite automaton (blue) has no corresponding timed word. After taking the first two transitions labelled a and b to and from location 4, we inevitably end up with $c_g \geq 2.5$. As the c_g clock is not reset during the transition from 1 to 2, this would violate the invariant of location 2 and therefore the transition cannot be taken.

different methods for doing so have been proposed, see for instance Asarin et al. [2003] for nonlinear systems, Althoff et al. [2008] for linear hybrid systems or Althoff et al. [2010] for nonlinear hybrid systems.

The second, the indirect approach, is concerned with building a "correct" abstraction of the system, which then allows to formally reason about its properties. Typically finite state machines (De Jong et al. [2001]), timed automata (Maler and Batt [2008]) or hybrid automata (Henzinger et al. [1998]) are used as abstraction. For such abstractions the notions of soundness and completeness are important with respect to the true system, which will be detailed afterwards. The method proposed here falls into this category.

In general, a trade-off has to be made between three properties: the expressiveness of the specification to be verified, the difficulty or generality of the considered dynamical system together with the fineness of the representation and the overall computational complexity. In the following several works setting different priorities and the connections to the approach presented in this chapter are given. Finally approaches tackling the synthesis problem for switched dynamical systems are presented.

Before introducing the related works, a short note on soundness and completeness in the context of dynamical system abstraction is given, as they somewhat differ from the standard definitions in computer science. Throughout this chapter we say that an abstraction is sound if the non-reachability of a state in the abstraction implies that there exists no trajectory of the dynamical system reaching this state. Or, to put it differently, if there exists a trajectory for the dynamical system reaching a state, then the corresponding abstract state must also be reachable in the automaton. This is closely related to safety, as it guarantees that if an unsafe state is not reachable in the abstraction, the dynamical system will not reach it either. Note that this does not imply the inverse implication, so soundness does not guarantee that if a state is reachable in the abstraction, then there exists a trajectory for the dynamical system reaching this state. This property in combination with soundness is sometimes called completeness (used in this sense in Sloth and Wisniewski [2013]), but has different meanings in other works. The notion of soundness can, depending on the context, also be refined by taking into consideration the time or duration of the trajectory, which will be detailed when necessary.

To study the reachability property of a continuous dynamical system, an abstraction to a timed automaton is presented in Maler and Batt [2008]. Here the state space is decomposed into a regular grid and each hypercube of this grid corresponds to a discrete state in the automaton. For each dimension of the considered system there is an associated clock in the automaton and the guards and invariants correspond to the maximal and minimal dwell time of the system in the current hypercube. As this work is principally concerned with reachability objectives, the set of reachable states has to be overapproximated by the abstraction to be sound. Therefore the minimal dwell time in a hypercube ensured by the guards is the side-length of the hypercube divided by the maximal velocity in the dimension corresponding to the clock attained anywhere within the hypercube. Similarly, the maximal dwell time is insured by the invariants and is set to the side-length of the hypercube divided by the minimal velocity (in the dimension corresponding to the clock attained anywhere within the hypercube). Transitions only exist between hypercubes that share a facet, and each time a transition is taken, the corresponding clock is set to zero. By adding these time-restrictions, the over-approximation of the timed-automata abstraction is obviously tighter than an abstraction without it, like finite state-machines (De Jong et al. [2001]), however due to the use of the minimal and maximal occurring velocity within each hypercube, the "quality" of the timing constraints is directly linked to the size of hypercubes (excepts for systems with constant derivatives) and the local Lipschitz constant of the system. This causes this approach to be computationally expensive especially for highly "dynamic" systems. The number of clocks is equal to the dimension of the system, the number of hypercube grows exponentially with the dimension and finally small hypercubes are required to obtain expressive timing constraints, making the approach intractable for medium-sized problems.

A different line of work that has a similar flavour is proposed in Sloth and Wisniewski [2010a, 2013]. Here the state space is also entirely decomposed into cells (bounded, simply connected subsets of the state space) which serve then as discrete states in the timed automaton, however they are constructed in a more sophisticated way resulting in tighter bounds. In Sloth and Wisniewski [2010a] so called "slice-families", a collection of sublevel-sets of a Lyapunov like function, are generated such that each sublevel-set is an invariant for the considered dynamical system. Taking one slice of each slice family and forming the intersection, one gets, in general, several disjoint subsets. Defining a cell as one of the subsets, which is then simply connected, one obtains a partition of the state-space defined by these cells. By associating a clock to each slice-family generating the partitioning, guards and invariants for each state can be deduced. Due to the construction of the slices using invariant sets, better bounds (upper and lower bounds on the dwell-time in a slice) can be derived. Even though these bounds are tighter than the ones obtained in Maler and Batt [2008], they remain conservative. In Sloth and Wisniewski [2013], the authors address this problem by building on this approach while restricting the dynamics to polynomial systems. By restricting the so-called partitioning functions to be polynomial as well, the demand that all level-sets of different partitioning functions intersect each other transversally becomes computationally tractable. Transversally in this context means that the combined tangent space of the intersecting level-sets spans the entire state space and this is a necessary condition to prevent degenerated cells. This approach is able to generate sound, as the works cited above, but in certain cases even complete timed automata abstractions of dynamical systems. This is worth mentioning as completeness is significantly harder to obtain than soundness. Soundness in this context means that if there is a trajectory for the dynamical system from some state $\boldsymbol{x}_0$ in some cell c_0 to another state $\boldsymbol{x}_1$ in some other cell c_1 in time T, then there exists a run in the automaton of total time T from the state corresponding to c_0 to the state corresponding to c_1. This is also the case for the approach presented in Maler and Batt [2008]. But the abstraction is, in certain cases, also complete in the sense that if there exists a run in the automaton of duration T from a state corresponding to cell c_0 to a state corresponding to c_1, then there exists also a trajectory of duration T for the dynamical system starting at some point in c_0 and ending somewhere in c_1, which is not the case for Maler and Batt [2008]. If an abstraction is not complete, there are more transitions in the abstraction then actually realisable by the dynamical system. This is also known as *false transitivity*. To guarantee completeness one has to prove that the bounds on the minimal and maximal dwell-time are tight, that is they coincide and therefore all states have to traverse a slice in the same time, independently

of their position. This is not achievable in all cases, depending on the given system, and if it is infeasible to derive such partitioning functions the presented approach seeks to obtain the tightest possible bounds.

Such approaches are interesting as they allow to formally reason about the dynamics of a system and allow to verify timed specifications on the system, but do not address controlled or switched dynamical systems. Such synthesis problems ask to generate a control law or switching sequence which formally verifies a given specification for the model of the dynamical system under all possible realisations of a given environment, equally given as a specification. Such problems are addressed with a variety of different approaches and techniques. Most of these approaches are concerned with generating control laws for linear temporal logic or fragments of it and therefore only provide a qualitative notion of time. As these are nonetheless interesting approaches and the approach proposed in this chapter draws some inspiration from them, they are detailed in the following.

In Kloetzer and Belta [2008] an approach to derive a continuous control law for controlled linear systems under polyhedral input constraints ($\dot{\boldsymbol{x}} = A.\boldsymbol{x} + B.\boldsymbol{u} + b$, $C.\boldsymbol{u} \leq \boldsymbol{g}$) satisfying a LTL specification is presented. The approach is constructive in the sense that it will terminate if such a control law exists. It relies on first partitioning the state space into polyhedrons with respect to the propositions in the given formula, then these polyhedrons are subdivided with respect to the dynamics of the system, such that there exists an admissible control input that drives all states in the polyhedron through a specific facet. This way the polyhedrons form an abstraction for the dynamical system and a corresponding automaton respecting the possible transitions between the polyhedrons and the specification given can be constructed. This way, a low-level control law is generated such that the resulting dynamics verify the specification. In this work too, only qualitative timing constraints can be accounted for.

The approach proposed in Kress-Gazit et al. [2007, 2009] and the method presented in this chapter have in common that the abstraction rather relies on guarantees provided by some low-level control law instead of seeking to directly synthesize the low-level control. The problem addressed in these works corresponds to motion planning under LTL constraints for service robots. In a first step the workspace is decomposed into polyhedrons and by relying on the control law described in Conner et al. [2003], it can be assured that all states within a polyhedron can be driven through a prespecified facet into an adjacent polyhedron.[2] Then again the polyhedrons in the workspace correspond to the discrete states of an automaton and a transition between them exists if and only if they are adjacent and there exists a control law that drives all states in the outgoing polyhedron through the facet shared with the ingoing polyhedron in finite time. By restricting the specification to a fragment of LTL, known as General Reactivity (1), a reactive control law can be synthesized in reasonable time for fairly complicated task and environment specifications (see Piterman et al. [2006]). The drawback of this approach is the simplicity of the admissible dynamics (it is restricted to so-called kinematic models $\boldsymbol{x} = \boldsymbol{u}$ in the proposed version).

In Sloth and Wisniewski [2010b] the idea of using a state space partitioning induced by the intersection of partitioning functions is extended to controlled systems. To achieve this, a set of inputs or control laws is defined under the constraint that the dynamics conditioned by each of these control laws is transversal to any sublevel-set of any partitioning function, a non-trivial constraint. Once this achieved, the path-planning problem for dynamical systems can be reformulated as a game on this timed game automaton and additional specifications in terms of Timed Computation Tree Logic can be defined. This approach allows for quantitative timing constraints, however, these can not be very precise without the goal becoming unreachable due to the conservative bounds on the transition times between the cells. Another drawback of the presented approach is that it is not fully implementable within existing verification tools due to its specific updates.

Closer to the approach proposed in this thesis are the ones in Tedrake et al. [2010b]; Le Ny and Pappas [2012]; Majumdar and Tedrake [2017]. They all have in common that they use some sort of time-dependent positive invariant set (funnel) to abstract the possible system trajectories. In Tedrake et al. [2010b] pure motion planning without other timing constraints is considered. In order to be able to guarantee that the all states can be driven to a target zone, first reference trajectories are generated

[2]The control law used in Kress-Gazit et al. [2007, 2009] shows many similarities to the one presented in Rimon and Koditschek [1991].

ending in the target zone, then "large" funnels are constructed around these trajectories, such that their final zone is a subset of the target zone. In order to cover the entire state space, new reference trajectories are generated leading to the initial zone of existing funnels and new funnels are created around them such that their final zone is in turn a subset of the initial zone of the existing funnels (connectivity condition) in an iterative process. As all funnels either end in a successor funnel or the target zone, and due to the positive invariance of the funnels, all states can be driven to the target zone. This gives the set of so created funnels the structure of a directed graph with the funnels being the vertices and the existence of an edge between two vertices v and v' depends on whether or not the final zone of the funnel associated to v is englobed by the initial zone of the funnel associated to v'. In Le Ny and Pappas [2012], similar ideas are used, but also take into account partially observability of the system and (bounded) disturbances and modelling errors. Instead of generating reference trajectories, here a finite set of predefined controllers (supposed to come with guarantees on convergence) is assumed and instead of generating a library covering the entire state space, the sequential composition of the funnels is integrated into a rapidly-exploring random tree (RRT, LaValle [1998]) like approach. Here the addressed problem is steering a given initial set into a final region.

In the here presented approach we also rely on funnels, however their reduction to timed-automata does not result in a loss of exact, or quantitative, timing properties. Therefore a richer variety of tasks can be addressed in theory, such as "pick up object A in position $\boldsymbol{x}$ at the exact time-point T", however at the cost of a possibly higher computational complexity.

A different approach to formally verify hybrid systems is based on differential dynamical logic (d$\mathcal{L}$), an extension to first-order logic. This approach is successfully used in the verification tools KeYmaera (Platzer and Quesel [2008]; Fulton et al. [2015]) and SpaceEx (Frehse et al. [2011]). Using such frameworks to directly synthesize control strategies for complex systems and task is (currently) beyond their scope. The abstraction proposed within this chapter could also be translated into d$\mathcal{L}$ formulas but this is less natural. It might be interesting to consider connections between d$\mathcal{L}$ formulas and our approach in the future, but we focus on timed automata in this chapter.

In this chapter the main idea behind the proposed approach to synthesize control strategies verifying logical constraints is presented before showcasing its application.

2.3 Graphs of Control Funnels

2.3.1 Control Funnels

Consider a controlled dynamical system governed by the general nonlinear differential equation:

$$\dot{\boldsymbol{x}} = \tilde{f}(\boldsymbol{x}) + g(\boldsymbol{x}, u(\boldsymbol{x}, t)) = f(\boldsymbol{x}, t, u(\boldsymbol{x}, t)) = f_u(\boldsymbol{x}, t), \tag{2.1}$$

where $\boldsymbol{x} \in \mathbb{R}^d$ is the state of the system, $t \in \mathbb{R}_0^+$ is a real (clock) value corresponding to an internal controller time, $\tilde{f}\colon \mathbb{R}^d \to \mathbb{R}^d$ is the system dynamic, $g\colon \mathbb{R}^d \times \mathbb{R}^k \to \mathbb{R}^d$ is the input dynamic and $u(\boldsymbol{x}, t)\colon \mathbb{R}^d \times \mathbb{R}_0^+ \to \mathbb{R}^k$ is the control law. To shorten notations, this can be equally written as $f_u(\boldsymbol{x}, t)$, denoting the dynamics of the system conditioned by the control law u. To ensure uniqueness of the solution for a given initial condition, we suppose that f_u is continuously differentiable from $\mathbb{R}^d \times \mathbb{R}_0^+$ to $\mathbb{R}^d$, meaning that $\tilde{f}$ and g are continuously differentiable from $\mathbb{R}^d$ to $\mathbb{R}^d$ and from $\mathbb{R}^d \times \mathbb{R}^k$ to $\mathbb{R}^d$ and $u(\boldsymbol{x}, t)$ is continuously differentiable form $\mathbb{R}^d \times \mathbb{R}_0^+$ to $\mathbb{R}^k$.

In this work, we mostly consider state spaces that describe the position and velocity of systems controlled in acceleration, such as articulated robots or autonomous vehicles. The continuity of trajectories in the state space ensures that the position is always a continuously differentiable function of time. Moreover, without loss of generality, we restrict ourselves to nonnegative time values. It is worth noting that, since t is an internal controller time, it can have a discontinuous evolution with discrete resets to any value in $\mathbb{R}_0^+$. However, except for these resets, the controller time is assumed to continuously increase at the same rate as physical time.

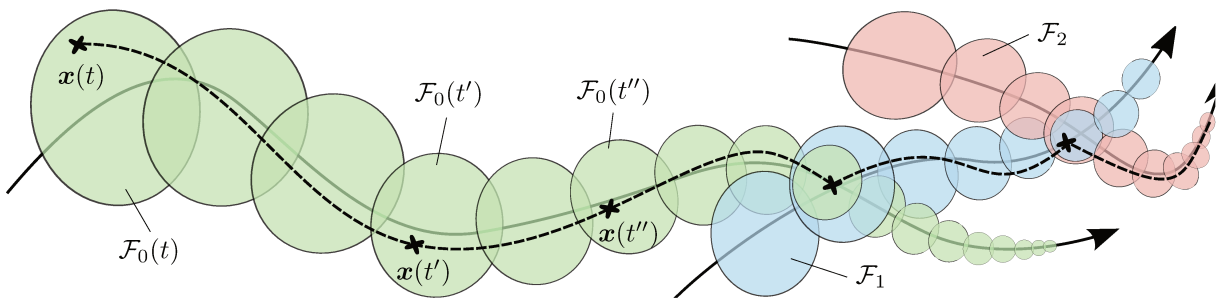


Figure 2.3 – An example of a control funnel for a controller tracking a reference trajectory. Here three different funnels $\mathcal{F}_0$ (green), $\mathcal{F}_1$ (blue) and $\mathcal{F}_2$ (red) are defined around their respective reference trajectory (black lines). The dashed line is a trajectory of the controlled system in the state space. The current state and the current region is shown for three particular time instances t, t' and t'' by the cross on the trajectory and the coloured ellipsoid while tracking the reference trajectory associated with $\mathcal{F}_0$. Therefore no discrete change in the dynamics occurs between the time instances, and $\mathcal{F}_0$ is positively invariant for the system. On the right side, switching transitions between control funnels, inducing discrete changes in the dynamics, are depicted.

A *control funnel* for the above dynamical system with a fixed control law u is a function $\mathcal{F}\colon I \to 2^{\mathbb{R}^d}$ such that $I \subseteq \mathbb{R}_0^+$ and for any solution $\boldsymbol{x}(t)$ of (2.1) with no discret resets of the control time t, the following property holds:

$$\forall t_1 \in I.\ \forall t_2 \in I.\ (t_2 > t_1 \text{ and } \boldsymbol{x}(t_1) \in \mathcal{F}(t_1)) \Rightarrow \boldsymbol{x}(t_2) \in \mathcal{F}(t_2). \tag{2.2}$$

Equation (2.2) defines a property known as positive invariance, and the funnel $\mathcal{F}$ corresponds to a time-varying region of invariance. That is, once the system has entered the funnel, it cannot leave the funnel by simply letting time pass by.

Example 1

A typical example of a control funnel based on a trajectory tracking controller is shown in Fig. 2.3. Such pairs of tracking controller plus reference trajectory lend themselves well for the construction of control funnels asymptotically converging towards a reference trajectory in the state space.

Example 2

For a concrete example, consider the simple system whose trajectories are of the form $\exp(-t) \cdot \boldsymbol{x}$, or equally governed by $\dot{\boldsymbol{x}} = -\boldsymbol{x}$, describing a uniform exponential convergence. Then any set $W \subseteq \mathbb{R}^d$ defines a control funnel $\mathcal{F}_W\colon t \mapsto \{\exp(-t) \cdot \boldsymbol{w} \mid \boldsymbol{w} \in W\}$ as seen in Figure 2.4.

The notion of funnel was popularized by Mason [1985], and it usually specifically refers to operations that eliminate uncertainty (as is the case in the example of Figure 2.3) by collapsing a larger set of initial conditions into a smaller set of final conditions (see for instance Tedrake et al. [2010a]). In our case, the control funnel may or may not reduce uncertainty, and it is important to note that the set $\mathcal{F}(t)$ does not have to decrease in size, over time.

This more general concept is closer to the definition of *viability tubes* (Aubin [1988]) or the so-called *flowpipes* used within the verification tool Flow* (see Chen et al. [2013] and Chen [2015]), but we nevertheless use the term control funnel as some reduction of uncertainty is often essential to the

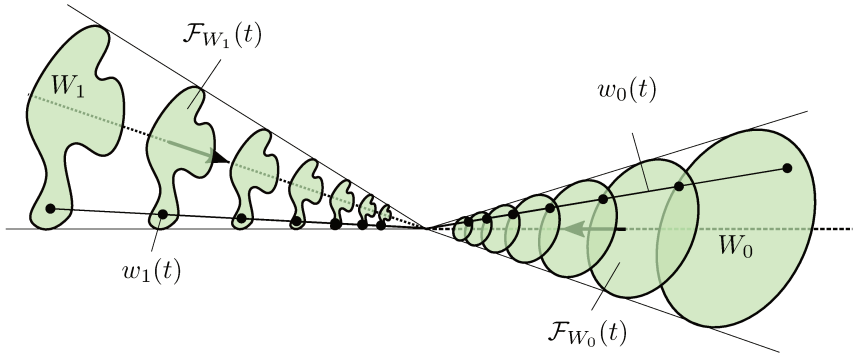


Figure 2.4 – Depiction of control funnels for uniform exponential convergence for two different initial sets W_0 and W_1 as well as an example trajectory of one point of each set as function of time $w_0(t)$ and $w_1(t)$.

usefulness of our abstractions. We address the computation of control funnels in section 2.5 for linear systems and in Chapter 3 for polynomial systems, and leave them as relatively abstract objects for now. For the moment it is sufficient to keep in mind the property of positive invariance.

2.3.2 Formalizing the Reach-Avoid Problem for Controlled Systems

To be able to generate motions based on funnels, we need to establish the relations between them. So suppose that we have a finite set U of control laws $u_0(\boldsymbol{x}, t)$, $u_1(\boldsymbol{x}, t)$, ..., $u_n(\boldsymbol{x}, t)$ that respectively set the dynamics of a given system to $\dot{\boldsymbol{x}} = f_{u_0}(\boldsymbol{x}, t)$, $\dot{\boldsymbol{x}} = f_{u_1}(\boldsymbol{x}, t)$, ..., $\dot{\boldsymbol{x}} = f_{u_n}(\boldsymbol{x}, t)$.

We say that the system can switch to the control law $u_i(\boldsymbol{x}, t)$ at some state $\boldsymbol{x}'$ whenever there is $t_0 \in \mathbb{R}_0^+$ and a solution $\boldsymbol{x}(t)$ of $\dot{\boldsymbol{x}} = f_{u_i}(\boldsymbol{x}, t)$ with initial condition $\boldsymbol{x}' = \boldsymbol{x}(t_0)$. Typically, if $u_i(\boldsymbol{x}, t)$ corresponds to a trajectory tracking controller, t_0 identifies the point of the trajectory where the tracking is triggered.

Informally, the *Reach-Avoid* problem asks, given a finite set of control laws as above, an initial point $\boldsymbol{x}_0$, a target zone $T_f \subseteq \mathbb{R}^d$, and an obstacle $\Omega \subseteq \mathbb{R}^d$, whether there exists a sequence of control law switches that generates a trajectory reaching from $\boldsymbol{x}_0$ to T_f while avoiding the obstacle Ω. Several variants of this problem can be considered, that vary on the objective (for instance some tasks can be expressed as ω-regular, so periodically reoccurring objectives, see for instance Rozenberg and Salomaa [2012]) which could also be solved using the proposed approach, however we focus here on a pure reachability with avoidance objective. As shown later on, many applications of practical importance can be defined as such reachability problems.

More formally, the Reach-Avoid problem asks for a finite sequence of time values $t_0^1 < t_1^1$, $t_0^2 < t_1^2$, ..., $t_0^P < t_1^P$, a finite sequence of control laws indices j_1, ..., j_P, and a finite sequence $\boldsymbol{x}_1$, ..., $\boldsymbol{x}_P$ of points in $\mathbb{R}^d$, such that:

(a) the goal is reached, $\boldsymbol{x}_P \in T_f$.

(b) for every $0 \leq p \leq P$, the trajectory portion $\boldsymbol{x}^p(t)$ is the unique solution to $\dot{\boldsymbol{x}} = f_{u_{j_p}}(\boldsymbol{x}, t)$ with initial condition $\boldsymbol{x}^p(t_0^p) = \boldsymbol{x}_{p-1}$ and $\boldsymbol{x}^p(t_1^p) = \boldsymbol{x}_p$.

(c) for every $0 \leq p \leq P$, for every $t_0^p \leq t \leq t_1^p$, the trajectory does not intersect with the obstacle $\boldsymbol{x}^p(t) \notin \Omega$.

Intuitively, this means that we can switch conveniently between all the control laws, causing discrete changes in the system dynamic, and ensure the global (reachability with avoidance) objective. The continuous trajectory generated by the solution above is the concatenation of the trajectory portions

$\{x^p(t) \mid t_0^p \leq t \leq t_1^p\}$ for $1 \leq p \leq P$. Therefore the resulting trajectory is almost everywhere differentiable, except at the time points where switching occurs.

Example 3

Consider the case of the two-dimensional controlled kinematic system $\dot{\boldsymbol{x}} = u_i$ with the finite set of control laws $u_0 = \begin{bmatrix}1 & 0\end{bmatrix}^{\mathrm{T}}$ ("go right"), $u_1 = \begin{bmatrix}-1 & 0\end{bmatrix}^{\mathrm{T}}$ ("go left"), $u_2 = \begin{bmatrix}0 & 1\end{bmatrix}^{\mathrm{T}}$ ("go up") and $u_3 = \begin{bmatrix}0 & -1\end{bmatrix}^{\mathrm{T}}$ ("go down"). The problem is further defined by its initial point $\boldsymbol{x}_0 = \begin{bmatrix}0 & 1\end{bmatrix}^{\mathrm{T}}$, target zone $T_f = \left\{\begin{bmatrix}0 & -1\end{bmatrix}^{\mathrm{T}}\right\}$ and obstacle $\Omega = \{\boldsymbol{x} \mid \|\boldsymbol{x}\|_2 \leq 0.5\}$. Then one can see that the time values $((0,1),(0,2),(0,1))$ together with the control law indices $(0,3,1)$ and the corresponding sequence of points $\left(\begin{bmatrix}1\\1\end{bmatrix}, \begin{bmatrix}1\\-1\end{bmatrix}, \begin{bmatrix}0\\-1\end{bmatrix}\right)$ is a solution for the Reach-Avoid problem. The trajectory of the system results from applying control law u_1 for 1 time unit (internal controller clock going from 0 to 1), resulting in the trajectory portion $\boldsymbol{x}^0(t) = \boldsymbol{x}_0 + \begin{bmatrix}1 & 0\end{bmatrix}^{\mathrm{T}} t$ (with $0 \leq t \leq 1$). Then a discrete change in the dynamics occurs when switching from control law u_1 to u_3. At the same time as the switching occurs, the internal controller clock is reset to zero. The control law u_3 is then applied for 2 time units resulting in the trajectory portion $p^1(t) = \begin{bmatrix}1 & 0\end{bmatrix}^{\mathrm{T}} + \begin{bmatrix}0 & -1\end{bmatrix}^{\mathrm{T}} t$ (with $0 \leq t \leq 2$). Finally, at $t = 2$, a second discrete change of the dynamics occurs, when switching to control law u_1, which is then applied for 1 time unit. This results in the third trajectory portion $p^2(t) = \begin{bmatrix}1 & -1\end{bmatrix}^{\mathrm{T}} + \begin{bmatrix}-1 & 0\end{bmatrix}^{\mathrm{T}} t$ (with $0 \leq t \leq 1$) and the system safely attains T_f, as $p^2(1) \in T_f$.
Note that there are often infinitely many solutions to a specific problem and in order to rank them some sort of optimality is often demanded. The most common condition is to demand the fastest solution, so the solution necessitating the least amount of time. As we will see later on this problem is decidable for timed automata.

2.3.3 Reach-Avoid Objectives on Graphs of Control Funnels

We now explain how the Reach-Avoid problem can be abstracted using timed transition systems based on control funnels, which can then be translated to a timed automaton.

For each control law $u_i(\boldsymbol{x}, t)$, we assume that we have a finite set of control funnels $\mathcal{F}_i^0, \mathcal{F}_i^1, \ldots, \mathcal{F}_i^{m_i-1}$, respectively defined over $I_i^0 \subseteq \mathbb{R}_0^+$, $I_i^1 \subseteq \mathbb{R}_0^+$, $\ldots$, $I_i^{m_i-1} \subseteq \mathbb{R}_0^+$. We assume that for every $0 \leq i \leq n-1$, for every $0 \leq j \leq m_i - 1$, for every $t \in I_i^j$, it holds $\mathcal{F}_i^j(t) \cap \Omega = \emptyset$, which means that trajectories contained in these funnels always avoid the obstacle Ω.

Consider a control law switch at position $\boldsymbol{x}'$ to law $u_i(\boldsymbol{x}, t)$ with clock value t_0. If there exists a control funnel $\mathcal{F}_i^j$ such that $t_0 \in I_i^j$, and $\boldsymbol{x}' \in \mathcal{F}_i^j(t_0)$, then we know that the state of the system will remain inside $\mathcal{F}_i^j(t)$ for any $t > t_0$ in I_i^j (as long as the control law $u_i(\boldsymbol{x}, t)$ is used). To always keep the system inside one of the control funnels, we can impose sufficient conditions on the switches. For instance, if the state is inside $\mathcal{F}_i^j(t_0)$, and if for some future clock value t_1, there exists a control funnel $\mathcal{F}_k^l$ and $t_2 \in I_k^l$ such that $\mathcal{F}_i^j(t_1) \subseteq \mathcal{F}_k^l(t_2)$, then when the clock value is t_1 we can safely switch to the control law $u_k(\boldsymbol{x}, t)$ while setting the clock to t_2. Indeed, we know that the state of the system at the switch instant will be inside $\mathcal{F}_k^l(t_2)$, and therefore it will remain inside $\mathcal{F}_k^l(t)$ after the switch. Such transitions from a funnel to another are illustrated on the right side of Figure 2.3, or in greater detail in Figure 2.5. It is worth noting that similar transitions could be achieved with, instead of control funnels, controller specifications as introduced in Le Ny and Pappas [2012], which are more conservative but directly account for disturbances in the actuation and sensor noise.

For some control funnels $\mathcal{F}_i^j$ and $\mathcal{F}_i^k$ associated to the same control law, it is the case (see section 2.5) that when funnel $\mathcal{F}_i^j$ is entered at time t, then at any time $t' \geq t + h_i^{j \to k}$ (where $h_i^{j \to k}$ is a constant), the state of the system is inside $\mathcal{F}_i^k(t')$ (and obviously also inside $\mathcal{F}_i^j(t')$ if $t' \in I_i^j$ as the control law

does not change). In that case, we say that the funnel $\mathcal{F}_i^k$ $h_i^{j\to k}$-absorbs the funnel $\mathcal{F}_i^j$. As we will see, we will typically construct funnels in practice such that $\mathcal{F}_i^{k=j+1}$ is *smaller* than $\mathcal{F}_i^j$. As they are constructed around the same reference trajectory they also share the same invariant except in the case that at least one of them intersects with the obstacle. In that case the invariants differ. Therefore $\forall t \in I_i^j$, $\mathcal{F}_i^{j+1}(t) \subset \mathcal{F}_i^j(t)$ and $h_i^{j\to j+1}$ is strictly positive. Note that $\mathcal{F}_i^{j+1}(t) \subset \mathcal{F}_i^j(t)$ also implies that $h_i^{j+1\to j}$ can be chosen equal to zero, meaning that one can always switch from the smaller to the larger funnel.

These rules for navigating between control funnels give to the set of control funnels the structure of an infinite graph, or, more precisely, of a timed transition system with real-valued clocks. One of the clocks of this timed transition system is c_t, the *controller clock*. We add two other clocks: a global clock c_g, and a local clock c_h.

The *funnel timed transition system* $\mathcal{T}_{U,F}$ associated with the family of laws $U = (u_i(\boldsymbol{x}, t))_{0\le i\le n-1}$ and the family of funnels $F = ((\mathcal{F}_i^j, I_i^j))_{0\le i\le n-1, 0\le j\le m_i-1}$ is defined as follows. The configurations are pairs $(\mathcal{F}_i^j, v)$ where v assigns a non-negative real value to each of the clocks c_t, c_g and c_h, with $v(c_t) \in I_i^j$, and its transition set contains three types of elements as depicted in Figure 2.5:

- the *time-elapsing transitions*: $(\mathcal{F}_i^j, v) \to (\mathcal{F}_i^j, v + \Delta)$ whenever $[v(c_t), v(c_t) + \Delta] \subseteq I_i^j$ (where $v + \Delta$ denotes the valuation that maps each clock c to $v(c) + \Delta$);
- the *switching transitions*: $(\mathcal{F}_i^j, v) \to (\mathcal{F}_k^l, v')$ whenever $v'(c_g) = v(c_g)$, $v'(c_h) = 0$, $v(c_t) \in I_i^j$, $v'(c_t) \in I_k^l$, and $\mathcal{F}_i^j(v(c_t)) \subseteq \mathcal{F}_k^l(v'(c_t))$;
- the *absorbing transitions:* $(\mathcal{F}_i^j, v) \to (\mathcal{F}_i^k, v')$ whenever $\mathcal{F}_i^k$ $h_i^{j\to k}$-absorbs $\mathcal{F}_i^j$, $v(c_t) \in I_i^j$, $v(c_t) \in I_i^k$, $v(c_h) \ge h_i^{j\to k}$, $v'(c_h) = 0$, $v'(c_g) = v(c_g)$ and $v'(c_t) = v(c_t)$.

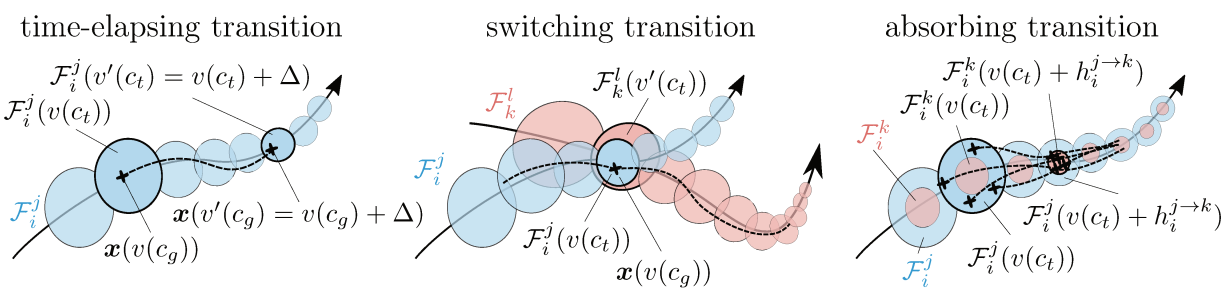


Figure 2.5 – Depiction of the three different types of transitions within a funnel timed transition system.

A run in this timed transition system is a finite sequence of configurations $\big((\mathcal{F}_{i_0}^{j_0}, v_0), (\mathcal{F}_{i_1}^{j_1}, v_1), \ldots, (\mathcal{F}_{i_P}^{j_P}, v_P)\big)$ such that $v_0(c_h) = v_0(c_g) = 0$, $v_0(c_t) \in I_{i_0}^{j_0}$, and all the transitions $(\mathcal{F}_{i_p}^{j_p}, v_p) \to (\mathcal{F}_{i_{p+1}}^{j_{p+1}}, v_{p+1})$ for $0 \le p < P$ are valid transitions that belong to $\mathcal{T}_{U,F}$.

Such a run is of total duration $v_P(c_g)$, and it corresponds to a schedule of control law switches that results from the following rules: initially, the control law is set to $u_{i_0}(\boldsymbol{x}, t)$, and the controller clock c_t is set to $v_0(c_t)$. For every time-elapsing transition $(\mathcal{F}_i^j, v) \to (\mathcal{F}_i^j, v + \Delta)$, the same control law $u_i(\boldsymbol{x}, t)$ is kept for a duration of Δ time units, and for every switching transition $(\mathcal{F}_i^j, v) \to (\mathcal{F}_k^l, v')$, the control law is switched from $u_i(\boldsymbol{x}, t)$ to $u_k(\boldsymbol{x}, t)$, with an initialization of the controller clock to $v'(c_t)$. Absorbing transitions are discarded, as they just correspond to an instantaneous change of funnels for the same control law.

Let us denote this sequence of switches by r. Then, it is fundamental to notice that for every $\boldsymbol{x} \in \mathcal{F}_{i_0}^{j_0}(v_0(c_t))$, if we follow the schedule of control law switches just described, then the system remains inside control funnels and reaches at the end of the run a unique point of $\mathbb{R}^d$, that we denote $r(\boldsymbol{x})$. The

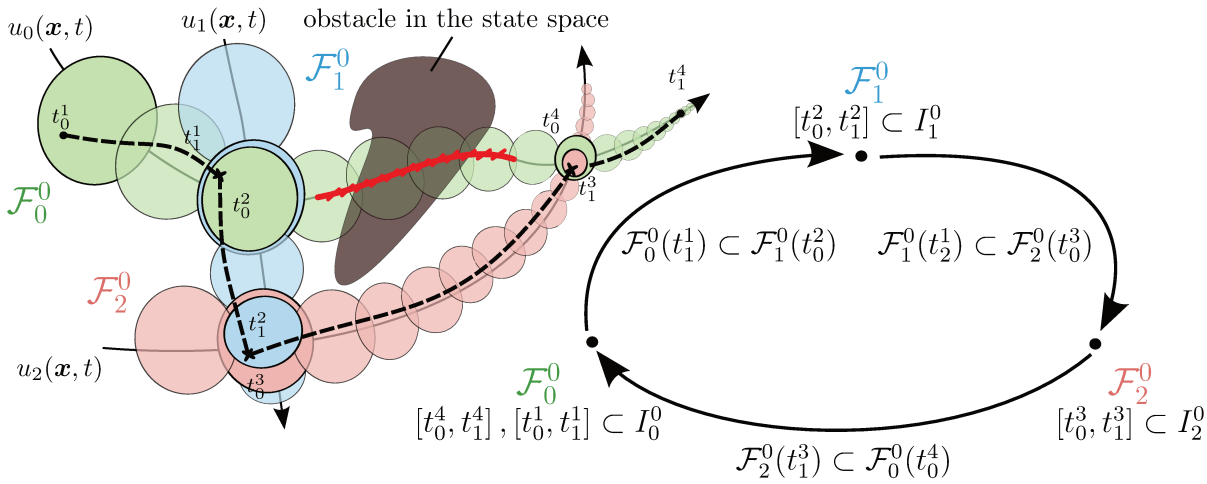


Figure 2.6 – On the left: graphical depiction of a funnel system and an obstacle. On the right its abstraction as funnel timed transition system. A run of this system comprising three control funnels: $r = \big((\mathcal{F}_0^0, v_0^1), (\mathcal{F}_0^0, v_1^1), (\mathcal{F}_1^0, v_0^2), (\mathcal{F}_1^0, v_1^2), (\mathcal{F}_2^0, v_0^3), (\mathcal{F}_2^0, v_1^3), (\mathcal{F}_0^0, v_0^4), (\mathcal{F}_0^0, v_1^4)\big)$, with: $\forall 1 \leq i \leq 4$, $v_0^i(c_t) = t_0^i$, $v_1^i(c_t) = t_1^i$, $v_0^i(c_h) = 0$, $v_1^i(c_h) = t_1^i - t_0^i$, $v_1^i(c_g) = v_0^i(c_g) + v_1^i(c_h)$, and $v_0^1(c_g) = 0$, and $\forall 2 \leq i \leq 4$, $v_0^i(c_g) = v_1^{i-1}(c_g)$.

trajectory going from $\boldsymbol{x}$ to $r(\boldsymbol{x})$ is also uniquely defined, continuous and almost everywhere continuously differentiable.

The funnel timed transition system $\mathcal{T}_{U,F}$ satisfies the following property:

Theorem 2.1. *Let* $r = \big((\mathcal{F}_{i_0}^{j_0}, v_0), (\mathcal{F}_{i_1}^{j_1}, v_1), \ldots, (\mathcal{F}_{i_P}^{j_P}, v_P)\big)$ *be a run in* $\mathcal{T}_{U,F}$. *If* $\boldsymbol{x} \in \mathcal{F}_{i_0}^{j_0}(v_0(c_t))$, *then* $r(\boldsymbol{x}) \in \mathcal{F}_{i_P}^{j_P}(v_P(c_t))$.

Proof. This directly follows from the guaranteed positive invariance of the funnel for the considered system dynamics conditioned by the control law associated to the funnel. The positive invariance guarantees that any state of the system inside a funnel remains inside of it. Additionally all transitions in the run r have to be valid transitions of $\mathcal{T}_{U,F}$ guaranteeing a correct succession of funnels. □

In some sense, the funnel timed transition system $\mathcal{T}_{U,F}$ is a correct, or sound, abstraction of trajectories that can be generated by the set of control laws: if $\boldsymbol{x}_0 \in \mathcal{F}_{i_0}^{j_0}(v_0(c_t))$ and $\mathcal{F}_{i_P}^{j_P}(v_P(c_t)) \subseteq T_f$, then such a run witnesses a solution to the Reach-Avoid problem. However, this abstraction is obviously not complete.

Example 4

The example in Figure 2.6 shows a run with three control laws $u_0(\boldsymbol{x},t)$, $u_1(\boldsymbol{x},t)$ and $u_2(\boldsymbol{x},t)$, three control funnels $\mathcal{F}_0^0$, $\mathcal{F}_1^0$ and $\mathcal{F}_2^0$, and an obstacle in the state space. The domains of definition of the control funnels I_0^0, I_1^0 and I_2^0 are such that for all $\alpha \in \{0,1,2\}$ and all $t \in I_\alpha^0$, $\mathcal{F}_\alpha^0(t)$ does not intersect the obstacles.
With the previous property, any run in the corresponding funnel timed transition system leads to a trajectory that avoids the obstacles. The example of Figure 2.6, where reaching $\mathcal{F}_1^1(t_1^4)$ from $\mathcal{F}_1^1(t_0^1)$ requires a series of switches between the different control funnels, shows the potential interest of automated verification in timed transition systems, as it can result in the generation of obstacle-free dynamic trajectories via appropriate control law switches.

2.4 Reduction to Timed Automata

Timed automata (Alur and Dill [1994a]), as introduced in section 2.2, provide an expressive formalism for modelling and reasoning about real-time systems, and enjoy decidable reachability properties; much efforts have been invested over the last 20 years for the development of efficient algorithms and tools for their automatic verification such as Kronos (Bozga et al. [1998]) and Uppaal (Behrmann et al. [2006]). Due to its more recent implementation and sophisticated GUI, Uppaal is the tool of choice in this chapter. Its efficient data-structures and implementation allow for the analysis of large scale automata in reasonable time and allow to obtain the later presented results.

We define a slightly modified variant of timed automata with rational constants, general boolean combinations of clock constraints and extended clock resets; those timed automata are as expressive as standard timed automata (see Bouyer et al. [2004]), but they will be useful for expressing funnel timed transition systems. These extensions are also directly supported by the tool Uppaal, with the exception of rational constants. In Uppaal all constants have to be integers, which can however be easily achieved with a preprocessing step to almost arbitrary precision.

The *timed automaton* is defined by the tuple $\mathcal{A} = (L, L_0, L_F, C, E, \Sigma, \mathsf{Inv})$ where L is a finite set of locations, $L_0 \subseteq L$ is a set of initial locations, $L_F \subseteq L$ is a set of final locations, C is a finite set of clocks, Σ is the alphabet and $E \subseteq L \times \mathcal{C}(C) \times R(C) \times \Sigma \times L$ is a finite set of edges, and $\mathsf{Inv}\colon L \to \mathcal{C}(C)$ is an invariant labelling function.

A configuration of $\mathcal{A}$ is a pair $(\ell, v) \in L \times \mathbb{R}^C$ such that $v \models \mathsf{Inv}(\ell)$, and the timed transition system generated by $\mathcal{A}$ is given by the following two rules:

- *time-elapsing transition:* $(\ell, v) \to (\ell, v + \Delta)$ whenever $v + \delta \models \mathsf{Inv}(\ell)$ for every $0 \leq \delta \leq \Delta$;
- *switching or absorbing transition:* $(\ell, v) \xrightarrow{\alpha} (\ell', v')$ whenever there exists $(\ell, g, \mathsf{res}, \alpha, \ell') \in E$ such that $v \models g \wedge \mathsf{Inv}(\ell)$, $v' = \mathsf{res}(v)$, and $v' \models \mathsf{Inv}(\ell')$.

Or informally

- *time-elapsing transition:* the configuration can stay in the current state and let time pass by as long as the invariant associated to this state is not violated;
- *switching or absorbing transition:* the configuration can change instantaneously from (ℓ, v) to (ℓ', v') if there exist an edge between these states and the corresponding (transition) guard as well as the invariant of the new state after the transition is satisfied.

A run in $\mathcal{A}$ is a sequence of consecutive transitions. The most fundamental result about timed automata is the following:

Theorem 2.2 (Alur and Dill [1994a]). *Reachability in timed automata is PSPACE-complete.*

By comparing the structure of the timed automaton and the funnel timed transition system, the similarities are numerous. The states of the timed automaton correspond to the different funnels, the edges correspond to the transitions and the constraints defining the validity of a transition can be translated into guards. The domain over which the funnel is defined can be interpreted as the associated invariant.

We consider again the family of control laws $U = (u_i(\boldsymbol{x}, t))_{0 \leq i \leq n-1}$, and the family of funnels $F = ((\mathcal{F}_i^j, I_i^j))_{0 \leq i \leq n-1, 0 \leq j \leq m_i - 1}$, as in the previous section. For every pair $0 \leq i, k \leq n-1$, and every $0 \leq j \leq m_i - 1$ and $0 \leq l \leq m_k - 1$, we select finitely many tuples $(\mathsf{switch}, [\alpha, \beta], (i, j), \gamma, (k, l))$ with $\alpha, \beta, \gamma \in \mathbb{Q}$ such that

(i) $[\alpha, \beta] \subseteq I_i^j$ (α and β satisfies the invariant of the current state)

(ii) $\gamma \in I_k^l$ (γ satisfies the invariant of the next state)

(iii) for every $t \in [\alpha, \beta]$, $\mathcal{F}_i^j(t) \subseteq \mathcal{F}_k^l(\gamma)$ (for every time instant between α and β the current funnel has to be englobed by the next funnel at time point γ)

This allows us to under-approximate the possible switches between funnels, as there can often be infinitely many such switches. An empirically justified way to select the switches is detailed in section 2.5. For every $0 \leq i \leq n-1$, for every pair $0 \leq j, k \leq m_i - 1$ we select at most one tuple $(\mathsf{absorb}, \nu, (i, j, k))$ such that $\nu \in \mathbb{Q}$ and $\mathcal{F}_i^k(t)$ ν-absorbs $\mathcal{F}_i^j(t)$. This allows us to under-approximate the possible absorbing transitions. For every $0 \leq i \leq n-1$ and every $0 \leq j \leq m_i - 1$, we fix a finite set of tuples $(\mathsf{initial}, \alpha, (i, j))$ such that $\alpha \in \mathbb{Q} \in I_i^j$ and $\boldsymbol{x}_0 \in \mathcal{F}_i^j(\alpha)$. This allows us to under-approximate the possible initialization to a control funnel containing the initial point $\boldsymbol{x}_0$. For every $0 \leq i \leq n-1$ and $0 \leq j \leq m_i - 1$, we fix finitely many tuples $(\mathsf{invariant}, S_{i,j}, (i, j))$, where $S_{i,j} \subseteq I_i^j$ is a finite set of closed intervals with rational bounds. This allows us to under-approximate the definition set of the funnels. Finally, for every $0 \leq i \leq n-1$ and $0 \leq j \leq m_i - 1$, we fix finitely many tuples $(\mathsf{target}, [\alpha, \beta], (i, j))$, where $\alpha, \beta \in \mathbb{Q}$ and $[\alpha, \beta] \subseteq I_i^j \cap \{t \mid \mathcal{F}_i^j(t) \subseteq T_f\}$. This allows us to under-approximate the target zone. We denote by K the set of all tuples we just defined.

We can now define a timed automaton that conservatively computes the runs generated by the funnel timed transition system. It is defined by $\mathcal{A}_{U,F,K} = (L, L_0, L_F, C, E, \Sigma, \mathsf{Inv})$ with:

- $L = \{\mathcal{F}_i^j \mid 0 \leq i \leq n-1,\ 0 \leq j \leq m_i - 1\} \cup \{\mathsf{init}, \mathsf{stop}\}$; $L_0 = \{\mathsf{init}\}$; $L_F = \{\mathsf{stop}\}$;
- $C = \{c_t, c_g, c_h\}$;
- E is composed of the following edges:
  (a) for every $(\mathsf{initial}, \alpha, (i, j)) \in K$, we have an edge $(\mathsf{init}, \mathtt{true}, \mathsf{res}, \mathcal{F}_i^j)$ in E, with $\mathsf{res}(c_t) = \alpha$ and $\mathsf{res}(c_g) = \mathsf{res}(c_h) = 0$;
  (b) for every $(\mathsf{switch}, [\alpha, \beta], (i, j), \gamma, (k, l)) \in K$, we have an edge $(\mathcal{F}_i^j, \alpha \leq c_t \leq \beta, \mathsf{res}, \mathcal{F}_k^l)$ with $\mathsf{res}(c_t) = \gamma$, $\mathsf{res}(c_h) = 0$ and $\mathsf{res}(c_g) = \perp$;
  (c) for every $(\mathsf{target}, [\alpha, \beta], (i, j)) \in K$, we have an edge $(\mathcal{F}_i^j, \alpha \leq c_t \leq \beta, \mathsf{res}, \mathsf{stop})$ in E, with $\mathsf{res}(c_t) = \mathsf{res}(c_g) = \mathsf{res}(c_h) = \perp$;
  (d) for every $(\mathsf{absorb}, \nu, (i, j, k)) \in K$, we have an edge $(\mathcal{F}_i^j, c_h \geq \nu, \mathsf{res}, \mathcal{F}_i^k)$ with $\mathsf{res}(c_h) = 0$ and $\mathsf{res}(c_t) = \mathsf{res}(c_g) = \perp$;
- for every $(\mathsf{invariant}, S_{i,j}, (i, j)) \in K$, we let $\mathsf{Inv}(\mathcal{F}_i^j) \triangleq \bigvee_{[\alpha,\beta] \in S_{i,j}} (\alpha \leq c_t \leq \beta)$.

Note that the invariant associated to a state is the intersection between the domain of the reference trajectory and the time intervals for which the funnel constructed around the reference trajectory does not intersect with the obstacle.

With the so defined automaton we easily get the following result:

Theorem 2.3. *Let* $(\mathsf{init}, v_0) \to (\ell_1, v_1) \to \cdots \to (\ell_P, v_P) \to (\mathsf{stop}, v_P)$ *be a run in* $\mathcal{A}_{U,F,K}$ *such that* v_0 *assigns* 0 *to every clock. Then* $r = ((\ell_1, v_1),\ \ldots,\ (\ell_P, v_P))$ *is a run of the funnel timed transition system* $\mathcal{T}_{U,F}$ *that brings* $\boldsymbol{x}_0$ *to* $r(\boldsymbol{x}_0) \in T_f$ *while avoiding the obstacle* Ω.

This shows that the reachability of stop in $\mathcal{A}_{U,F,K}$ implies that there exists an appropriate schedule of control law switches that safely brings the system to the target zone. Of course, the method is not complete, not all schedules can be obtained using the timed automaton $\mathcal{A}_{U,F,K}$. But if $\mathcal{A}_{U,F,K}$ is precise enough, it will be possible to use automatic verification techniques for dynamic trajectory generation.

Before continuing with the effective construction of funnels for linear systems, we would like to give some additional remarks.

Remark 2.1. The absorbing transitions in (d) are defined as $(\mathcal{F}_i^j, c_h \geq \nu, \mathsf{res}, \mathcal{F}_i^k)$. By adding this transition only if $\nu > 0$, that is if $\mathcal{F}_i^k$ is *smaller* than $\mathcal{F}_i^j$ and by ensuring that all transitions that exist for $\mathcal{F}_i^k$ also exist for $\mathcal{F}_i^j$ if $j > k$, we can change the transition to $(\mathcal{F}_i^j, c_h = \nu, \mathsf{res}, \mathcal{F}_i^k)$. This simplifies the automaton without loosing expressiveness. In Uppaal we can even enforce this transition by making it urgent (see Barbuti and Tesei [2004]), further reducing the verification complexity without reducing the expressiveness.

Remark 2.2. We could be more precise in the modelling as a timed automaton, if we could use non-deterministic clock resets (see Bouyer et al. [2004]); but we should then be careful with decidability issues. Additionally, non-deterministic resets are not implemented in UPPAAL, which is why we have chosen timed automata with deterministic resets only.

Remark 2.3. The invariants are constructed such that they represent a finite set of closed intervals with rational bounds on the controller clock c_t. This can be loosened into boolean combinations of expressions of the form $c_x \sim \alpha$, with c_x being either the controller or world clock, α is again a constant $\mathbb{Q}$ and $\sim \in \{\leq, <, =, >, \geq\}$. This could be useful to simulate a changing environment, but was not necessary for the examples presented.

Remark 2.4. As we show with some examples in section 2.6, our timed-automata abstraction can be used for other types of objectives than just reachability with avoidance. In particular, the approach can be extended to *timed games* (Asarin et al. [1998]), where special uncontrollable transitions model uncertainty in the environment. In that case, the aim is not to synthesize one single run in the system, but rather a *strategy* that dictates how the system should be controlled, depending on how the environment evolves. It is worth knowing that winning strategies can be computed in exponential time in timed games, and that the tool UPPAAL-TIGA (Behrmann et al. [2007]) computes such winning strategies. In section 2.6.1, we give an example of application where timed games and UPPAAL-TIGA are used.

2.5 LQR FUNNELS

2.5.1 LYAPUNOV STABILITY AND CONSTRUCTION

In this section we consider the particular case of linear time-invariant stabilizable systems whose dynamics are described by the following equation:

$$\dot{\boldsymbol{x}} = A.\boldsymbol{x} + B.\boldsymbol{u}, \tag{2.3}$$

where $A \in \mathbb{R}^{d\times d}$ and $B \in \mathbb{R}^{d\times k}$ are two constant matrices, and $\boldsymbol{u} \in \mathbb{R}^k$ is the control input. We also consider reference trajectories that can be realized with controlled systems described by (2.3), i.e. trajectories $\boldsymbol{x}_{\text{ref}}(t)$ defined for $t \in [0, T]$ for which there exists $\boldsymbol{u}_{\text{ref}}(\boldsymbol{x}, t)$ such that $\dot{\boldsymbol{x}}_{\text{ref}}(t) = A.\boldsymbol{x}_{\text{ref}}(t) + B.\boldsymbol{u}_{\text{ref}}(t)$ for all $t \in [0, T]$. We can combine this equation with (2.3) and get $\dot{\boldsymbol{x}}(t) - \dot{\boldsymbol{x}}_{\text{ref}}(t) = A.(\boldsymbol{x}(t) - \boldsymbol{x}_{\text{ref}}(t)) + B.(\boldsymbol{u}(t) - \boldsymbol{u}_{\text{ref}}(t))$, which rewrites

$$\dot{\boldsymbol{x}}_\Delta(t) = A.\boldsymbol{x}_\Delta(t) + B.\boldsymbol{u}_\Delta(t). \tag{2.4}$$

To track $\dot{\boldsymbol{x}}_{\text{ref}}$, we compute $\boldsymbol{u}_\Delta$ using the linear quadratic regulator in its infinite horizon version (LQR,see Sontag [1998]), i.e. a minimization of the quadratic cost: $J = \int_0^\infty \left(\boldsymbol{x}_\Delta{}^{\mathrm{T}} Q \boldsymbol{x}_\Delta + \boldsymbol{u}_\Delta{}^{\mathrm{T}} R \boldsymbol{u}_\Delta\right) dt$, where Q and R are respectively positive-semidefinite and positive-definite matrices, see section 3.2 for a more detailed introduction. The solution is a time-independent feedback control matrix K defined as $K = R^{-1}.B^{\mathrm{T}}.P$ with P being the unique positive-definite matrix solution to the continuous time algebraic Riccati equation: $PA + A^{\mathrm{T}}P - PBR^{-1}B^{\mathrm{T}}P + Q = 0$. By applying the control law $\boldsymbol{u}_\Delta = -K\boldsymbol{x}_\Delta$, the cost J is minimized and moreover the quadratic function $V(\boldsymbol{x}_\Delta(t)) = \boldsymbol{x}_\Delta(t)^{\mathrm{T}}.P.\boldsymbol{x}_\Delta(t)$ is a Lyapunov function for the closed loop system with some minimal exponential convergence rate, so some $\gamma > 0$ for which $\dot{V}(\boldsymbol{x}_\Delta(t)) \leq -\gamma V(\boldsymbol{x}_\Delta(t))$ or equally $V(\boldsymbol{x}_\Delta(t+\delta t)) \leq V(\boldsymbol{x}_\Delta(t)).\exp(-\gamma\delta t)$ holds. This means that the error term $\boldsymbol{x}_\Delta$ tends to $\mathbf{0}$ exponentially fast and therefore all trajectories converge to the reference trajectory $\boldsymbol{x}_{\text{ref}}(t)$.

This convergence property can be used to define control funnels as follows. For $\alpha > 0$, we define:

$$\mathcal{F}_\alpha(t) = \{\boldsymbol{x}_{\text{ref}}(t) + \boldsymbol{x}_\Delta(t) \mid V(\boldsymbol{x}_\Delta(t)) \leq \alpha\} \tag{2.5}$$

where $\mathcal{F}$ is a control funnel defined over $t \in [0, T]$ (the duration of the reference trajectory): if $\boldsymbol{x}_\Delta(t) = \boldsymbol{x}(t) - \boldsymbol{x}_{\text{ref}}(t)$ is a solution of equation (2.4) such that $\boldsymbol{x}(t_1) \in \mathcal{F}_\alpha(t_1)$, then for any $t_2 > t_1$, since $V(\boldsymbol{x}_\Delta)$ only decreases, $V(\boldsymbol{x}_\Delta(t_2)) \leq V(\boldsymbol{x}_\Delta(t_1)) \leq \alpha$, and thus $\boldsymbol{x}(t_2) = \boldsymbol{x}_{\text{ref}}(t_2) + \boldsymbol{x}_\Delta(t_2) \in \mathcal{F}_\alpha(t_2)$.

The funnel $\mathcal{F}_\alpha(t)$ is therefore defined by a fixed d-dimensional ellipsoid, which is a sublevel-set for the associated Lyapunov function, translated along the reference trajectory as it is at each time point centred at $\boldsymbol{x}_{\text{ref}}(t)$. Due to the guaranteed minimal convergence rate γ, for any solution $\boldsymbol{x}_\Delta(t)$ of (2.4) we have:

$$\forall t \in [0, T]\,, \forall \delta t \in]0, T-t]\,,\; V(\boldsymbol{x}_\Delta(t+\delta t)) \leq \exp(-\gamma \delta t) V(\boldsymbol{x}_\Delta(t)). \tag{2.6}$$

This proves that if the system is inside the control funnel $\mathcal{F}_\alpha(t)$ at a given instant, then after letting time elapse for a duration of δt, the system will be inside the control funnel $\mathcal{F}_{\alpha \exp(-\gamma\delta t)}(t)$. This means that the funnel $\mathcal{F}_{\alpha'}(t)$ absorbs the funnel $\mathcal{F}_\alpha(t)$ and the corresponding constant is derived in section 2.5.2. Thanks to this property, for a given LQR controller and a reference trajectory $\boldsymbol{x}_{\text{ref}}(t)$, we can define a finite set of fixed-size control funnels $\mathcal{F}_{\alpha_0}(t), \mathcal{F}_{\alpha_1}(t), \ldots, \mathcal{F}_{\alpha_q}(t)$, with $\alpha_0 > \alpha_1 > \cdots > \alpha_q > 0$, and absorbing transitions between them in the corresponding timed automaton. For such families of funnels we will adopt the notation that $\mathcal{F}_i^j$ denotes the funnel constructed around the ith reference trajectory as the sublevel-set $\left\{\boldsymbol{x}_{\text{ref}}^i(t) + \boldsymbol{x}_\Delta(t) \mid \boldsymbol{x}_\Delta(t)^{\mathrm{T}}.P.\boldsymbol{x}_\Delta(t) \leq \alpha_j\right\}$.

Here we have deliberately chosen to consider only disturbance free and fully observable linear systems. However, the decisive properties of positive and invariance and minimal convergence rate can also be computed for systems with (bounded) disturbance and partial observability of the form

$$\dot{\boldsymbol{x}} = A.\boldsymbol{x} + B.\boldsymbol{u} + D.\boldsymbol{\omega} \tag{2.7a}$$

$$\boldsymbol{y} = H.\boldsymbol{x} \tag{2.7b}$$

where D determines the channels of the vector of (bounded) disturbances and $\boldsymbol{y}$ is the output, given as a linear transformation of the system state by the H. This necessitates more involved stability proofs and induces increased conservativeness in order to deal with the disturbances, but does not fundamentally change the approach of how the funnels are constructed.

In the remainder of this chapter, we will mainly use this kind of fixed size control funnels, which we call "LQR funnels". They are convenient because the larger ones can be used to "catch" other control funnels, and the smaller ones can easily be caught by other control funnels or used when a "precise" location of the system (in the state space) has to be defined. Figure 2.7 depicts a typical sequence, where first a large control funnel $\mathcal{F}_1^0$ (in green) catches the system which previously evolved in $\mathcal{F}_0^0$ via a switching transition. The bounds α and β on $\mathcal{F}_0^0$ are shown as the ellipsoids with dashed line. Then after some time longer than $h_0^{0\to 1}$, an absorbing transition can be triggered and the system switches from $\mathcal{F}_1^0$ to $\mathcal{F}_1^1$. Finally, a new switching transition brings the system to a larger control funnel $\mathcal{F}_2^0$ (in blue) defined around another reference trajectory.

Testing for inclusion between fixed-size ellipsoids can be done very efficiently (by introducing some conservativeness, see section 2.4), and therefore LQR funnels allow for efficient algorithms for the computation of the tuples needed for the timed-automaton reduction $((\mathsf{switch}, [\alpha, \beta], (i,j), \gamma, (k,l)), (\mathsf{invariant}, S_{i,j}, (i,j)), \ldots,$ also see section 2.4).

It should be noted that the concepts of fixed size control funnels and absorbing transitions, introduced here for linear systems, are also suitable for nonlinear and in particular polynomial systems. Lyapunov functions in general, and quadratic ones in particular, can be computed via convex optimization for polynomial systems, for example with Sum-of-Squares techniques (Majumdar et al. [2013a]), contraction theory (Lohmiller and Slotine [1998]) or convexification and state-space seperation (chapter 3). In all these approaches, the minimial exponential convergence necessary for the absorbing transitions can be imposed within the framework of each of the approaches cited above.

2.5.2 Computing the Tuples

In order to reduce the funnel timed transition system we need to compute constants within the tuples defining the absorbing and switching transitions.

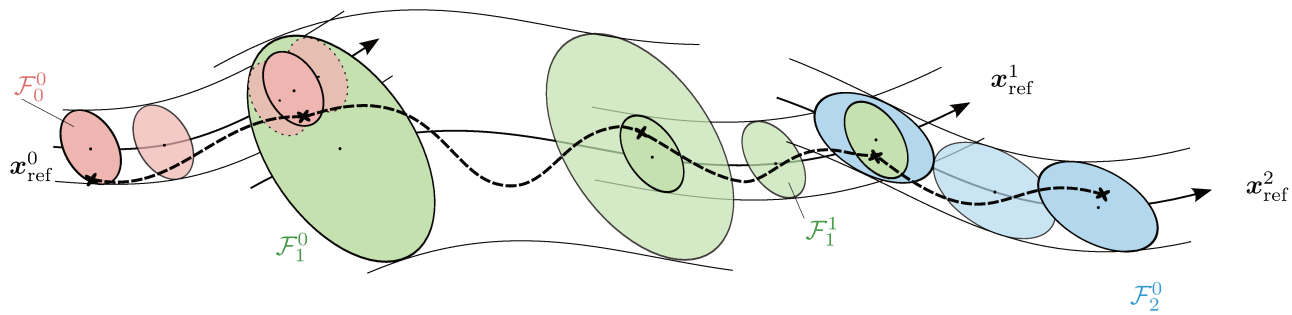


Figure 2.7 – An absorbing transition (in green) between two switching transitions.

Computing the time-constant $h_i^{j\to k}$ for the absorbing transition is straightforward using the guaranteed minimal convergence γ. As all states in $\mathcal{F}_\alpha(t) = \left\{\boldsymbol{x} \mid (\boldsymbol{x}-\boldsymbol{x}_{\text{ref}}(t))^{\text{T}}.P.(\boldsymbol{x}-\boldsymbol{x}_{\text{ref}}(t)) \leq \alpha\right\}$ are guaranteed to be in $\left\{\boldsymbol{x} \mid (\boldsymbol{x}-\boldsymbol{x}_{\text{ref}}(t+\delta t))^{\text{T}}.P.(\boldsymbol{x}-\boldsymbol{x}_{\text{ref}}(t+\delta t)) \leq \alpha\exp(-\gamma\delta t\right\}$ after δt time units, we can immediately see that the funnel defined as $\mathcal{F}_{\alpha'}(t) = \left\{\boldsymbol{x} \mid (\boldsymbol{x}-\boldsymbol{x}_{\text{ref}}(t))^{\text{T}}.P.(\boldsymbol{x}-\boldsymbol{x}_{\text{ref}}(t)) \leq \alpha'\right\}$ with $0 < \alpha' < \alpha$, $\left[\frac{1}{\gamma}\log(\frac{\alpha}{\alpha'})\right]$-absorbs the control funnel $\mathcal{F}_\alpha(t)$. So we have $h^{\alpha\to\alpha'} = \left[\frac{1}{\gamma}\log(\frac{\alpha}{\alpha'})\right]$ if the funnels of different size share the same reference trajectory and Lyapunov function to generate the sublevel-sets. In the case that $\alpha' \geq \alpha$ the constant $h^{\alpha\to\alpha'}$ would be nonpositive. Allowing and adding such transitions unnecessarily complicates the automaton without gaining expressiveness with respect to the reach-avoid problem, as it is always "better" to be in a smaller funnel (If a larger funnel avoids an obstacle/reaches an target, so does the smaller funnel) and are therefore not taken into account. The more interesting problem is if the two funnels defined on the same reference trajectory are generated by different Lyapunov functions, that is $\mathcal{F}_\alpha(t) = \left\{\boldsymbol{x} \mid (\boldsymbol{x}-\boldsymbol{x}_{\text{ref}}(t))^{\text{T}}.P.(\boldsymbol{x}-\boldsymbol{x}_{\text{ref}}(t)) \leq \alpha\right\}$ and $\mathcal{F}_{\alpha'}(t) = \left\{\boldsymbol{x} \mid (\boldsymbol{x}-\boldsymbol{x}_{\text{ref}}(t))^{\text{T}}.P'.(\boldsymbol{x}-\boldsymbol{x}_{\text{ref}}(t)) \leq \alpha'\right\}$ with $P, P' \in \mathbb{S}_{++}$ but there exists no $a \in \mathbb{R}^+$ such that $P = aP'$ (P and P' are truly different and not just scaled versions of each other). In this case the above definition for $h^{\alpha\to\alpha'}$ is not suitable and one needs to solve the following problem:

$$h^{\alpha\to\alpha'} = \max_{\boldsymbol{x}(t)\in\mathcal{F}_\alpha(t)} \min_{\delta t} \boldsymbol{x}(t) \in \mathcal{F}_\alpha(t) \implies \boldsymbol{x}(t+\delta t) \in \mathcal{F}_{\alpha'}(t+\delta t) \tag{2.8}$$

with $\boldsymbol{x}(t)$ being the solution to the system dynamic conditioned by the current control law. This problem is as such not directly tractable and we therefore use the following conservative approximation. Consider the funnel $\mathcal{F}_\beta(t) = \left\{\boldsymbol{x} \mid (\boldsymbol{x}-\boldsymbol{x}_{\text{ref}}(t))^{\text{T}}.P.(\boldsymbol{x}-\boldsymbol{x}_{\text{ref}}(t)) \leq \beta\right\}$, where β is chosen as the maximal value for which $\mathcal{F}_\beta(t) \subset \mathcal{F}_{\alpha'}(t)$ (largest inscribed ellipsoid problem), then we can define $h^{\alpha\to\alpha'}$ as $\left[\frac{1}{\gamma}\log(\frac{\alpha}{\beta})\right]$ and obtain a correct (but conservative) abstraction of the behaviour as shown in Figure 2.8.

Next the tuples for the switching transitions have to be computed, or more specifically, we seek an efficient way to compute the constants α and β for a given γ in $(\mathsf{switch}, [\alpha,\beta], (i,j), \gamma, (k,l))$. As the reference trajectory has bounded velocity and acceleration, it is sufficient to test finitely many points to conclude the inclusion of one funnel in another within a (continuous) interval. Therefore computing α and β comes down to (efficiently) test the inclusion of ellipsoids. The inclusion can be tested relying on convex optimization, or more precisely semidefinite programming (modified Löwner-John ellipsoid, Boyd and Vandenberghe [2004]), which is however prohibitively expensive as we seek to test many points. We therefore use, again, a conservative approximation. Consider the funnels $\mathcal{F}_i^j$ and $\mathcal{F}_k^l$ defined on the reference trajectories $\boldsymbol{x}_{\text{ref}}^i$ and $\boldsymbol{x}_{\text{ref}}^k$ using sublevel-sets of quadratic Lyapunov functions defined by P and

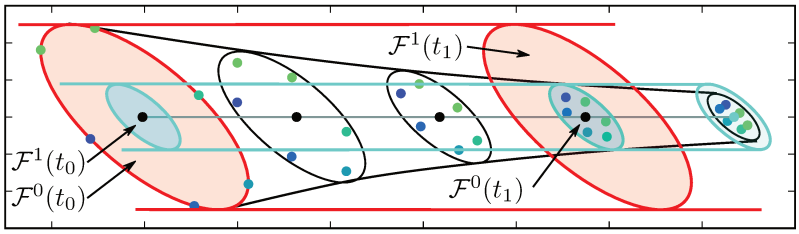


Figure 2.8 – In this images the convergence of a funnel and the resulting absorption transitions for fixed size funnels is shown. At t_0 the initial points (coloured dots) are distributed on the boundary of the larger fixed size funnel ($\mathcal{F}^0$, red). In black the conservative convergence of the funnel using the guaranted minimal convergence rate γ is shown. We see that the trajectories starting at the boundary are in the interior of the converging funnel for $t > t_0$. At t_1 the size of the converging funnel is equal to the smaller funnel ($\mathcal{F}^1$, cyan), and therefore $h^{0\to 1} = t_1 - t_0$.

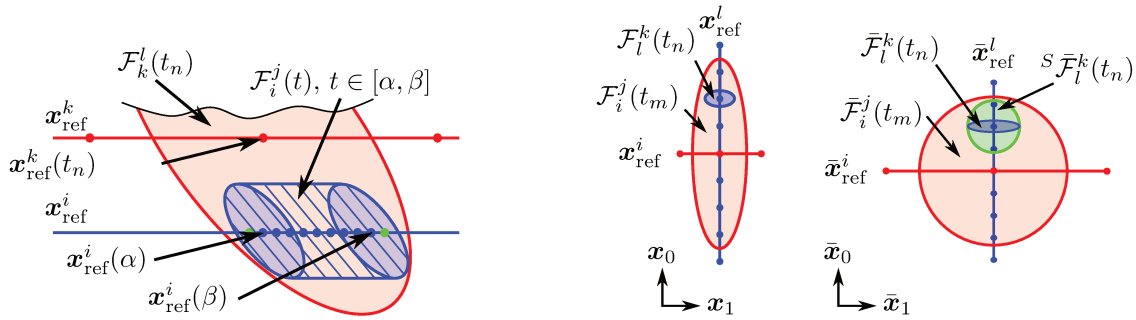


Figure 2.9 – On the left: checking finitely many time points to obtain α and β for a given γ. The inclusion of each checked time point (blue dots) implies the inclusion $\forall t \in [\alpha, \beta]$. On the right: Overapproximation of the funnel in the case of non-identical funnel shapes. The shown case induces "much" conservativeness as the eigenvectors of the largest eigenvalues of P and P' are orthogonal.

P' and the positive scalars ρ and ρ'. By using the transformation

$$\bar{\boldsymbol{x}} = \text{chol}(P').\boldsymbol{x} \tag{2.9}$$

we obtain transformed coordinates in which the funnel $\mathcal{F}_k^l$ is a hypersphere translated along the (transformed) reference trajectory $\bar{\boldsymbol{x}}_{\text{ref}}^k$. By over-approximating the (transformed) funnel $\bar{\mathcal{F}}_i^j$ by the smallest circumscribed hypersphere, denoted ${}^S\bar{\mathcal{F}}_i^j$ of radius $\bar{r}$ the inclusion test of $\mathcal{F}_i^j(t)$ in $\mathcal{F}_k^l(\gamma)$ comes down to checking

$$\sqrt{\alpha'} \geq \bar{r} + \left\| \bar{\boldsymbol{x}}_{\text{ref}}^k(\gamma) - \bar{\boldsymbol{x}}_{\text{ref}}^i(t) \right\|_2 \tag{2.10}$$

as depicted in Figure 2.9. The conservativeness introduced by this approach depends on the "distance" between P and P' and vanishes if they are simply scaled versions of one another. In general the matrices obtained for LTI-systems using LQR derived controllers are fairly similar further motivating this approach.

Before continuing with the first examples, a special type of switching transition is presented, as it is useful in some cases where fine timing constraints have to be met: switching from a smaller funnel to a larger funnel defined on the same reference trajectory while modifying the controller clock c_t. Such transitions $(\mathsf{switch}, [\alpha, \beta], (i, j), \gamma, (i, k))$ are always possible, so for all γ in I_i there exist an α and β such

that $\alpha \leq \gamma \leq \beta$ with all three satisfying I if $\mathcal{F}_i^j(t) \subset \mathcal{F}_i^k(t)$ for all $t \in I$. Such transitions allow to "accelerate" or "decelerate" on the reference trajectory.

2.6 Examples of Application

In this section we will show how to apply the proposed reduction of a funnel system constructed with fixed size LQR funnels for fairly different problem settings. Before showing how to relax certain constraints in the next section.

2.6.1 Synchronization of Sine Waves

In this example, there is a unique reference trajectory: $x_{\text{ref}}(t) = \sin(\frac{2\pi}{\tau}t)$, for $t \in [0, \tau]$ and $\tau \in \mathbb{Q}$. The controlled system corresponds to a linear second order system controlled in acceleration, so

$$\dot{\boldsymbol{x}} = \begin{pmatrix} \dot{x} \\ \ddot{x} \end{pmatrix} = A.\boldsymbol{x} + B.\boldsymbol{u} = A.\boldsymbol{x} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u. \tag{2.11}$$

By computing a unique LQR controller (choosing the matrices Q and R), and introducing the reference trajectory the system becomes

$$\ddot{x} = \ddot{x}_{\text{ref}} + A[1,:].(\boldsymbol{x} - \boldsymbol{x}_{\text{ref}}) - B[1,:].K.(\boldsymbol{x} - \boldsymbol{x}_{\text{ref}}) \tag{2.12}$$

where $X[k,:]$ for some matrix $X \in \mathbb{R}^{m \times n}$ denotes the extraction of the kth row from X, interpreted as a $1 \times n$ matrix.

We define two fixed size LQR funnels $\mathcal{F}^0$ (the large one) and $\mathcal{F}^1$ (the small one) defined over $[0, \tau]$ such that $\mathcal{F}^1$ γ-absorbs $\mathcal{F}^0$ for some $\gamma \in \mathbb{Q}^+$. The size of $\mathcal{F}^0$ is computed such that an upper bound on the acceleration is always ensured, as long as the state of the system remains inside the control funnel. More importantly, limiting the size of $\mathcal{F}^0$ also allows to limit the maximal control input u, so $u_m \leq u \leq u_M$ (see section 3.9.3) holds for all states inside the funnel, which is a very important property for real applications.

The set $\mathcal{F}^0(\tau/2)$ englobes the smaller control funnel $\mathcal{F}^1(t)$ for a range of time values $[\alpha, \beta]$ for some $\alpha < \frac{\tau}{2} \in \mathbb{Q}$ and $\beta > \frac{\tau}{2} \in \mathbb{Q}$. This allows switching transitions from $\mathcal{F}^1$ to $\mathcal{F}^0$ with abrupt modifications of the controller clock c_t. Together with the absorbing transition and "cyclic transitions" that come from the equalities $\mathcal{F}^0(0) = \mathcal{F}^0(\tau)$ and $\mathcal{F}^1(0) = \mathcal{F}^1(\tau)$, the timed automaton shown on the left Figure 2.10 corresponds to the reduction of the timed transition system.

The goal is to synchronize the controlled signal to a fixed signal $\sin(\frac{2\pi}{\tau}t + \varphi_0)$. The phase φ_0 is initially unknown, which we model using an adversary: we use a new clock c'_t, and an opponent transition as in the timed automaton on the right of Figure 2.10.

With these two timed automata, we can use the tool Uppaal-Tiga (see Behrmann et al. [2007]) to synthesize a controller that reacts to the choice of the adversary, and performs adequate switching transitions until $c_t = c'_t$. It is even possible to generate a strategy that guarantees that the synchronization can always be performed in a bounded amount of time.

We show in Fig. 2.11 a trajectory generated by the synthesized reactive controller. In this example, the phase chosen by the adversary is such that it is best to "accelerate" the controlled signal. Therefore, the controller uses twice the switching transition from $\mathcal{F}^1$ to $\mathcal{F}^0$ with a reset of the controller clock from α to $\tau/2$ (1 and 2 in Fig. 2.11). Between these switching transitions, an absorbing transition is taken to go back to the control funnel $\mathcal{F}^1$ (A in Figure 2.11) before taking the "cyclic transition". After the first two switching transitions, the remaining gap $\epsilon = c'_t - c_t$ is smaller than $\frac{\tau}{2} - \alpha$, and therefore the controller waits a bit longer (until $\frac{\tau}{2} - \epsilon$) to perform the switching transition that exactly synchronizes the two signals (3 in Figure 2.11).

This example shows that our abstraction can be used for reactive controller synthesis via timed games. The main advantage of our approach over methods based on full discretization is that, since a "continuous"

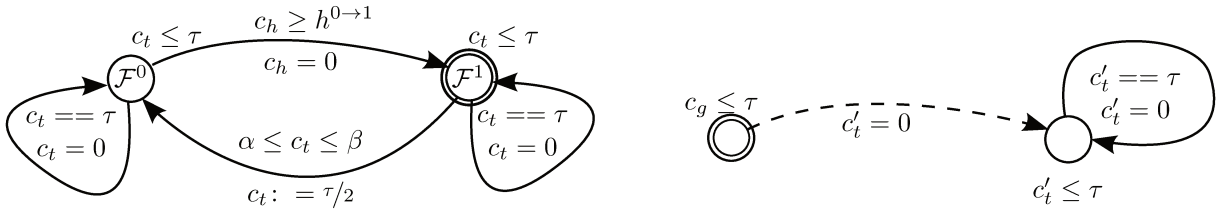


Figure 2.10 – On the left: the timed automaton for the controlled signal (the system). On the right: the timed automaton used to model the target signal with an initially unknown phase φ_0. The opponent transition (dashed) is the one used to set φ_0.

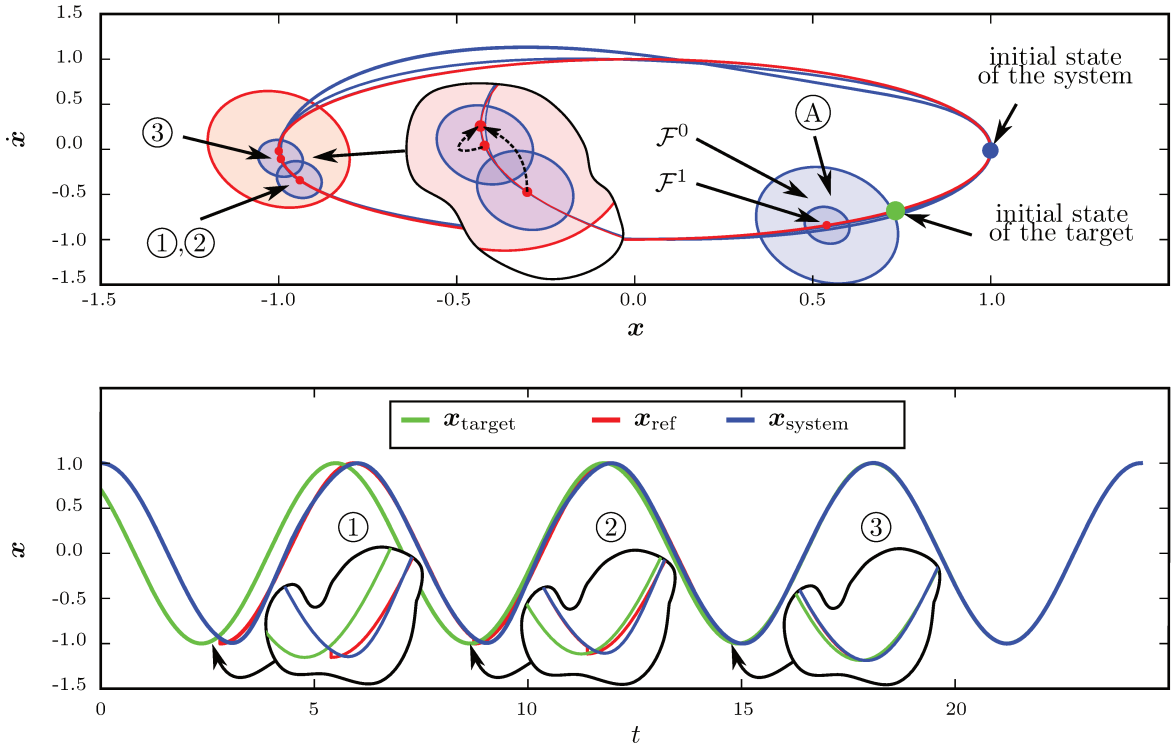


Figure 2.11 – The reactive controller performs three switching transitions to exactly adjust its phase to that of the target signal.

notion of time is kept in our abstraction, the reactive strategy is theoretically able to *exactly* synchronize the controlled signal to any rational value of φ_0 without augmenting the necessary size of the automaton as necessary by approaches using a full discretization.

2.6.2 A 1D Pick-and-Place Problem

In this second example, we show that timed-automata abstractions based on control funnels can be used to perform non-trivial planning under logical constraints. We propose a one-dimensional pick-and-place scenario. The set-up consists of a linear system controlled in acceleration moving along a straight line.

On this line, four positions are defined as lanes (see Figure 2.12). On three of these lanes (1, 2 and 3), packages arrive that have to be caught at the right time by the system and later delivered to lane 0. The system has limited acceleration, control input and velocity, and can carry at most two packages at a time. While the boundedness of the velocity, acceleration and control input to certain predefined values can be directly ensured during the construction of the funnel system, the logical constraint that a maximum of two items can be carried at a time corresponds to a "run-time" constraint and cannot be ensured via the funnel construction. Therefore an auxiliary automaton encoding this logical constraint as well as the task of picking up the objects itself is created. By solving the product automaton of the one encoding the task plus constraints and the one representing the funnel timed transition system, one obtains a feasible strategy for the given task and considered dynamical system, as depicted in Figure 2.12.

The LQR funnels in this example are constructed based on 12 reference trajectories. The first four have different constant positive velocities ($\boldsymbol{x}^i_{\text{ref}}$ with $i \in \{1, \ldots, 4\}$, the fastest one being $\boldsymbol{x}^4_{\text{ref}}$, and the slowest one $\boldsymbol{x}^1_{\text{ref}}$). The next four are the same trajectories but with negative velocities. On each of these reference trajectories, five different control funnels of constant size are defined ($\mathcal{F}^j_i$ for $j \in \{0, \ldots, 4\}$, the largest one being $\mathcal{F}^0_i$). The control funnels with negative constant velocity are the mirror image of those with positive velocity. Additionally, four stationary trajectories $\boldsymbol{x}^{\text{L}k}_{ref}$ (with $k \in \{1, \ldots, 4\}$) at the positions of the lanes are defined. The controllers associated to these trajectories simply stabilize the system at lane positions. For each of these trajectories a small ($j = 1$) and a large ($j = 0$) control funnel are constructed. They are denoted by $\mathcal{F}^j_{\text{L}k}$. By construction, neighbouring trajectories (e.g. $\boldsymbol{x}^3_{\text{ref}}$ and $\boldsymbol{x}^2_{\text{ref}}$ or $\boldsymbol{x}^1_{\text{ref}}$ and $\boldsymbol{x}^{-1}_{\text{ref}}$) are connected, meaning that for two neighbouring trajectories $\boldsymbol{x}^i_{\text{ref}}$ and $\boldsymbol{x}^k_{\text{ref}}$, $\forall t \in I_i,\ \exists t' \in I_k$ such that $\mathcal{F}^4_i(t) \subset \mathcal{F}^0_k(t')$ (see Figure 2.12). This allows the system to reach a higher or lower velocity without the need of an explicitly defined acceleration trajectory. While the abstraction based on these control funnels does not represent all the possible behaviours of the system (it is not complete), switching between different velocity references allows the system to perform a great variety of trajectories with continuous and bounded velocity and bounded accelerations and control inputs.

To fully specify the timed-automata abstraction, the tuples defining the transition guards must be computed (see Section 2.4). For each trajectory we empirically choose a number of equidistantly distributed time points γ within the bounds of the duration of the reference trajectory. Then the approach presented in section 2.5.2 is used to compute the switching transition $((\mathsf{switch}, [\alpha, \beta], (i, j), \gamma, (k, l)))$ for each such γ.

We consider an example where three packages respectively arrive on lanes 3, 2 and 1 at times $t^1_{\text{arrive}} = 40$, $t^2_{\text{arrive}} = 111$ and $t^3_{\text{arrive}} = 122$, corresponding to the marked transitions in Figure 2.12. The goal is to find a trajectory that catches all the packages and delivers them to lane 0. At the moment of the catch ($c_g = t^p_{\text{arrive}}$), the reference $\boldsymbol{x}^i_{\text{ref}}$ tracked by the system must be exactly at the correct position (i.e. on the lane of the arriving package). Depending on the reference trajectory, this corresponds to a particular value of c_t. We add the following constraints on the catches: an upper bound on velocity such that the system cannot be tracking $\boldsymbol{x}^4_{\text{ref}}$, $\boldsymbol{x}^3_{\text{ref}}$, $\boldsymbol{x}^{-3}_{\text{ref}}$ or $\boldsymbol{x}^{-4}_{\text{ref}}$ when it catches a package, and a bound on uncertainty such that the system must be in a small control funnel to catch a package. Checking this reachability objective, Uppaal outputs a timed word that corresponds to the schedule of control-law switches and the trajectory shown on Figure 2.13, which successfully catches the packages and delivers them to lane 0.

The two upper graphs of Figure 2.13 show the evolution of the system in its state space and some of the regions of invariance when taking a *switching transition* (coloured ellipsoids). The green dots mark positions at which *absorbing transitions* take place ($\mathcal{F}^j_i \to \mathcal{F}^{j+1}_i$). Purple crosses represent a package. The lower graph compares the evolution of the position of the real system with the reference. One can see that even though the reference velocity can only take seven different values, a relatively smooth trajectory is realized.

Before catching the first package, the system switches from $\mathcal{F}^4_1$ to $\mathcal{F}^0_{\text{L}3}$ ①. It then converges to $\mathcal{F}^1_{\text{L}3}$ ② just before the catch. The difference between the real system position and the reference is very small at that point in time, as implied by the current funnel. The system then switches to $\mathcal{F}^0_{-1}$ ③ in order to return to lane 0. It is interesting to notice that the system chooses to return to lane 0 after having picked only one package, therefore adopting a non-greedy strategy. This is because it wouldn't have time

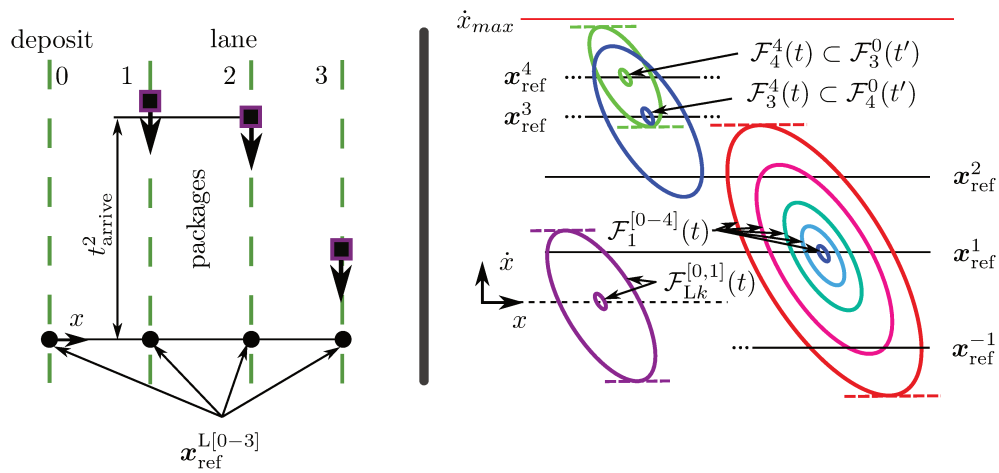


Figure 2.12 – Graphical depiction of the problem setting and funnel system used. The highest/lowest velocity can be attained in $\mathcal{F}^0_4/\mathcal{F}^0_{-4}$, which is by construction admissible.

to perform a delivery to lane 0 between the arrival of the second and third package.

When the second package arrives on lane 2, the system catches it while being in $\mathcal{F}^4_{-1}$ ④ . This is again a non-trivial behavior: in order to get both the second and the third package, the system has to first go a little bit further than lane 2 so as to be able to catch the two packages without violating the limit on acceleration. A slight adjustment of the reference position ⑤ has to be done to catch the third package exactly on time ⑥. After that, the system performs a local acceleration ⑦ to reach lane 0 as soon as possible, and delivers the two packages.

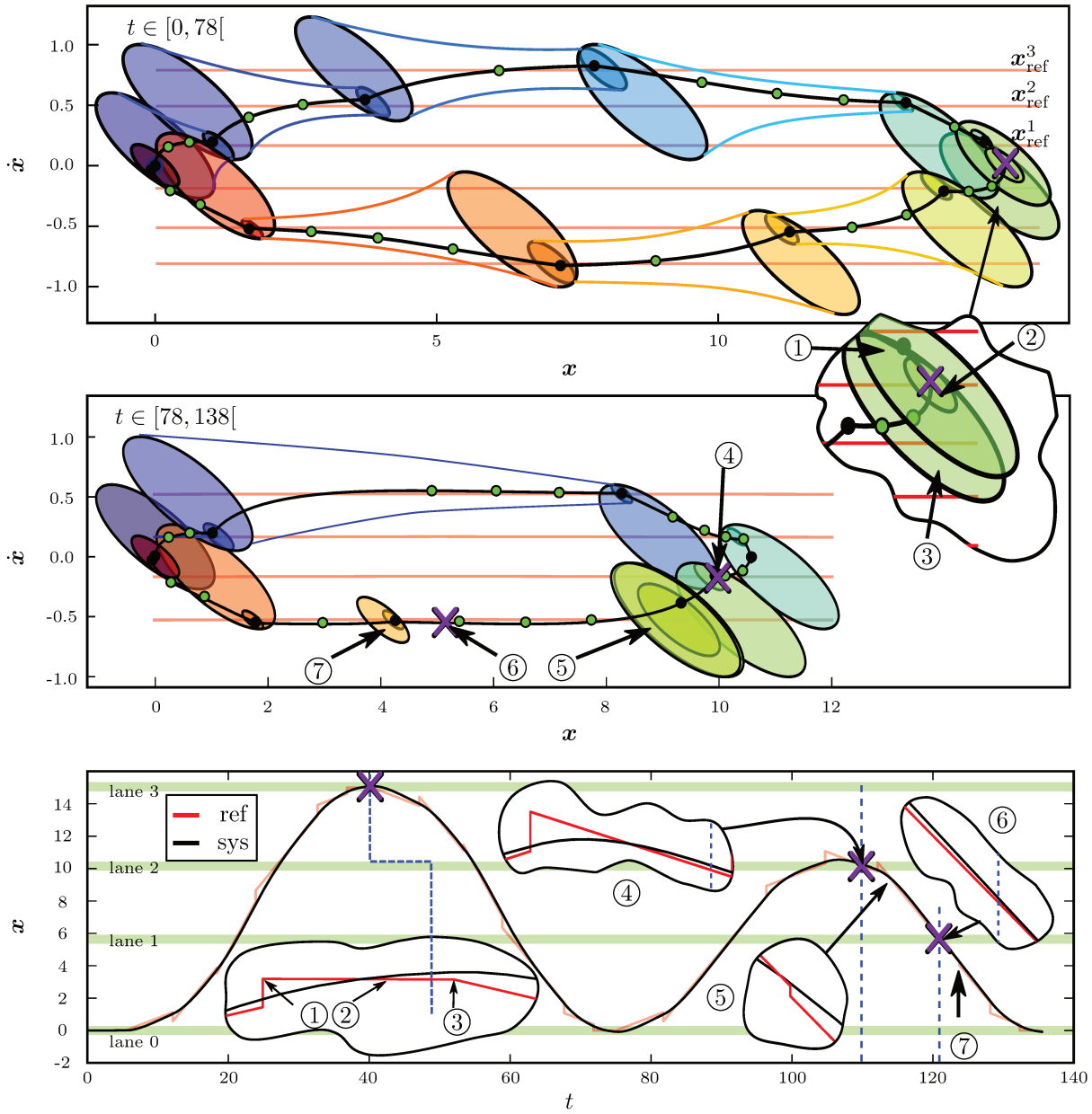


Figure 2.13 – Execution of a succeeding control strategy given as a timed word.

2.7 Bounding Funnels with Conjectured Properties

The proposed approach allows to reason about the dynamics of LTI systems by abstracting their possible behaviour into a finite set of funnels. While this approach offers the advantage of keeping a continuous notion of time, the abstraction is only correct if these funnels correspond to time-dependent zones of positive invariance. This property is comparably easy to obtain for LTI systems, but this drastically changes when considering dynamical systems described by nonlinear differential equations.

In this section, we propose a method to treat this class of systems, introducing the concept of *bounding funnels*, and using conjectured properties that are empirically verified. This approach is then used to solve a Reach-Avoid problem for a modified version of the Dubins' car, a nonlinear and nonholonomic system.

2.7.1 Introducing Bounding Funnels with Conjectured Properties

The main problem encountered when trying to construct control funnels for nonlinear systems, is the difficulty to find a suitable pair of monotonic Lyapunov function and control law. The associated optimization problem is in general nonconvex and no generic approach exists to find such pairs, even without addressing the problem of finding large regions of invariance in the case of constrained inputs or maximizing the (guaranteed) convergence rate. There exist approaches for certain subclasses of nonlinear dynamics, like semidefinite programming for polynomial Lyapunov functions and systems with polynomial dynamics as done in Prajna et al. [2004]. In Majumdar et al. [2013a], it is shown how to use sum-of-squares optimization to handle nonlinear systems by using time-dependent polynomial approximations. It is an interesting approach, but its high computational complexity and the introduced conservativeness restrain its usability, even though the computational burden is later on decreased in Ahmadi and Majumdar [2014] by using more conservative constraints. Moreover the demand of monotonic convergence somewhat contradicts the use of quadratic Lyapunov functions. In general, the (controlled) system might only converge with respect to more complex functions or demand complex control laws in order to converge with respect to a simple Lyapunov function. As the proposed approach relies on cheap inclusion testing, which is not possible for sublevel-sets for more complicated Lyapunov functions, this is not admissible.

We propose therefore a different approach: bounding funnels with conjectured properties. Bounding funnels enlarge the concept of regular funnels by weakening some of the required assumptions. The properties of these funnels are as hard to guarantee as the properties of regular funnels, but due to the weakened assumptions they are more likely to be true. We propose to conjecture these properties based on numerical simulations. With these bounding funnels, the control sequence obtained is guaranteed to satisfy a given specification provided that the conjectures hold for the nonlinear dynamics under all circumstances that can occur. However as usually only a finite number runs is actually performed on the real system, it can never be fully assured that the conjectures truly hold under all circumstances.

Bounding Funnels

The concept of bounding funnels relies on a modified concept of positive invariance, which, together with the conservative approximation of convergence time, makes funnels suitable for timed automata reduction. The property of positive invariance described by equation (2.2) is closely linked to the concept of monotonic Lyapunov functions. For general nonlinear systems this property is very difficult to obtain. There exists no generic way to generate them, so the Lyapunov functions and control laws have to be found on a case to case basis. An expressive sub-class of nonlinear dynamics allowing for a generic way to compute the control law and the Lyapunov function is the class of polynomial systems. However, constructing funnels based on proofs on the truncated Taylor expansion (or any other polynomial approximation) of the real nonlinear dynamics is in some sense also conjecturing the properties, except if one uses additional constraints as proposed in Chesi [2009] overapproximating the error between the models and therefore increasing conservativeness.

On the other hand, if a system converges asymptotically to the origin, it also eventually stays inside any neighbourhood of the origin. Or, to put it differently, if $V^*(\boldsymbol{x}, t)$ is a Lyapunov function for the dynamical system $\dot{\boldsymbol{x}} = f(\boldsymbol{x}, t)$, then the system will also converge, possibly non-monotonically, with respect to every other Lyapunov function candidate $V'(\boldsymbol{x}, t)$, see Figure 2.14. This property is very useful since it allows us to use functions with simple level sets, like ellipsoids, to construct our funnels, "independently" of dynamical system treated. In certain cases the non-monotonic convergence can even be proven (see Ahmadi [2008], Butz [1969]) and a Lyapunov function with monotonic convergence can be constructed based on intermediate results of the proof, but this is not the goal here as we seek to keep the simple shape of the sublevel-sets. For a bounding funnel $\mathcal{F}_i^j \colon I_i^j \subseteq \mathbb{R}^+ \to 2^{\mathbb{R}^d}$, the property of positive invariance is weakened in the sense that to each inner funnel $\mathcal{F}_i^j$ we associate an outer funnel $\mathcal{F}_i^{\mathrm{O},j} \colon I_i^j \to 2^{\mathbb{R}^d}$ such that the following property holds:

$$\forall t_1 \in I_i^j,\ \forall t_2 \in I_i^j,\ t_2 > t_1,\ \boldsymbol{x}(t_1) \in \mathcal{F}_i^j(t_1) \Rightarrow \boldsymbol{x}(t_2) \in \mathcal{F}_i^{\mathrm{O},j}(t_2). \tag{2.13}$$

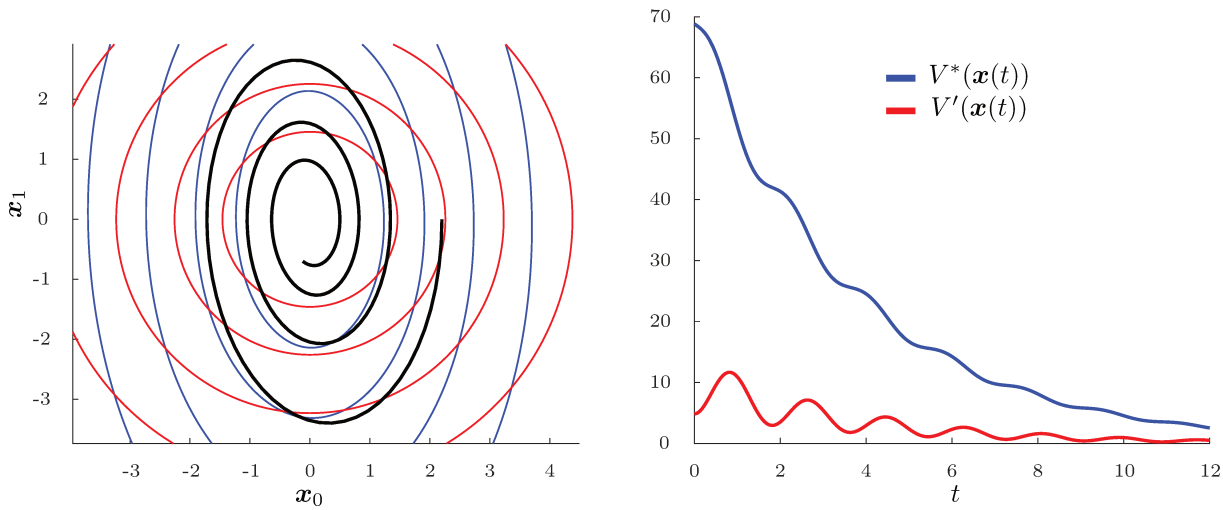


Figure 2.14 – Example for the non-monotonic convergence of a globally asymptotically stable system. On the left a trajectory of the system (black line) and the level-sets of two different (here both quadratic) Lyapunov functions are shown (red and blue lines). On the right the evolution of the respective value of the two Lyapunov function along the trajectory is shown. Both converge to zero, however the one shown in red does so in non-monotonic fashion. Both can be used to construct bounding funnels, but only the one shown in blue is admissible for regular funnel construction.

Informally, the outer funnel, for which $\forall t \in I_i^j,\ \mathcal{F}_i^j(t) \subseteq \mathcal{F}_i^{\mathrm{O},j}(t)$ holds, is chosen such that the trajectories of any initial position in $\mathcal{F}_i^j$ will not leave $\mathcal{F}_i^{\mathrm{O},j}$. This modification is necessary due to the possibly non-monotonic convergence. Consequently, if the actual initial state of the system is inside $\mathcal{F}_i^j(t_0)$, the initial state of the timed automaton corresponds to the associated outer funnel $\mathcal{F}_i^{\mathrm{O},j}$. A *switching transition* (see Section 2.4) in a bounding-funnel system has the form: $(\mathcal{F}_i^j, v) \to (\mathcal{F}_k^{\mathrm{O},l}, v')$ whenever $v'(c_g) = v(c_g)$, $v'(c_h) = 0$, $v(c_t) \in I_i^j$, $v'(c_t) \in I_k^l$, and $\mathcal{F}_i^j(v(c_t)) \subseteq \mathcal{F}_k^l(v'(c_t))$, where $\mathcal{F}_k^{\mathrm{O},l}$ denotes the bounding funnel associated with $\mathcal{F}_k^l$. In some cases (for example with fixed size inner and outer funnels), there exists a minimal duration that implies convergence from the outer to the inner funnel, i.e. a constant $h_i^{\mathrm{O},j\to j}$ such that $\mathcal{F}_i^j$ $h_i^{\mathrm{O},j\to j}$-absorbs $\mathcal{F}_i^{\mathrm{O},j}$. To put the concept of bounding funnels in perspective, a regular funnel is a bounding funnel with $\mathcal{F}_i^{\mathrm{O},j}(t) = \mathcal{F}_i^j(t)$, $\forall t \in I_i^j$, and the absorption time $h_i^{\mathrm{O},j\to j}$ is equal to zero.

Conjecturing the Properties

As stated above, formally proving the convergence and the weak positive invariance for general nonlinear systems is a complex problem. Therefore we replace the formal guarantees by conjectures based on, for example, numerical simulations. This allows to use general optimization methods to simultaneously find a control law and suitable outer/inner funnel shapes in the sense that the outer funnel is as small as possible while achieving a good convergence time $h_i^{\mathrm{O},j\to j}$. To define the conjectures, sufficiently many initial points in $\mathcal{F}_i^j$ can be numerically evaluated, and the convergence time $h_i^{j\to k}$ is defined as an upper bound of the maximal time needed to arrive and stay inside $\mathcal{F}_i^k$. The outer funnel can be taken as an ellipsoid with minimized volume under the constraint that (2.13) must hold.

This loss of guarantees may at first seem to be a very serious drawback, as obtaining certified behaviours is one of the main objectives of this work. Nevertheless, we argue that performing formal synthesis with such conjectured properties of the control laws can lead to interesting results. Indeed, after a controller has been synthesized with our approach, if an execution fails to verify the specification, we know that it can only be because at least one conjecture does not hold and therefore one or more properties of the bounding funnel are violated. We can even raise flags during execution to pinpoint the faulty bounding funnel or even the violated conjecture itself. This structure, where the logic of the controller is proven, but some "atomic" properties are only conjectured, is similar to formally verified cryptographic protocols, where the security depends on how reliable some cryptographic primitives are. It helps keeping safety issues localized, and therefore it makes it easier to improve the global behaviour with confidence by performing isolated tests of the validity of each funnel. Moreover, formally proven funnels are true funnels only in the mathematical model, and therefore, as far are as runs on the real system are concerned, they are in fact conjectured as well.

To further highlight the interest of using funnels with conjectured properties, reconsider the case of LQR funnels for LTI-systems. In this case, the funnels are "real" funnels and all properties are formally proven. Now let us take a closer look at the absorption transition. The time constant is $h^{i\to k}$ is derived based on the minimal convergence rate γ, which is equal to (for LTI-systems $\dot{\boldsymbol{x}} = A.\boldsymbol{x}$ and quadratic Lyapunov functions of the form $V(\boldsymbol{x}) = \boldsymbol{x}^{\mathrm{T}}.P.\boldsymbol{x}$) the maximal eigenvalue of ${C_i}^{\mathrm{T}}.(A^{\mathrm{T}}.P + P.A^{\mathrm{T}}).C_i$, with $C_i = \mathrm{chol}(P)^{-1}$. However, in general the eigenvalues do not coincide. Therefore the average convergence rate over any trajectory portion is (considerably) higher than it can be guaranteed, as shown in Figure 2.15. Therefore it is interesting to conjecture a smaller absorption time.

Another interesting application for bounding funnels with conjectured properties is the case of disturbed dynamics as shortly mentioned in section 2.5.1. In order to prove monotonic convergence for such a system, the disturbance has to be bounded, baring the usage of normal distributions as noise source. By using bounding funnels, this is possible by demanding for instance that 95 percent of the possible executions converge in a given time. Such bounds can then be computed using tools from stochastic control.

2.7.2 Reach-Avoid Problem for a Modified Dubins' car

We use the above introduced bounding funnel concept to perform path planning for a modified Dubins' car. A Dubins' car is a simplified model of an automobile that evolves on a 2D plane, which is frequently used in the context of path-planning for automobiles, see for instance Scheuer and Fraichard [1997] or Macharet et al. [2011].

Dynamical Model and Control Law

The state of the Dubins' car is defined by its position (denoted by $\boldsymbol{p}$) and its heading (denoted θ_p). The position is relative to the global coordinate frame and the heading is given as the angle between the global $\boldsymbol{e}_{g,x}$-axis and the local $\boldsymbol{e}_{c,x}$-axis of the car. The current linear velocity of the car, denoted v_p, always

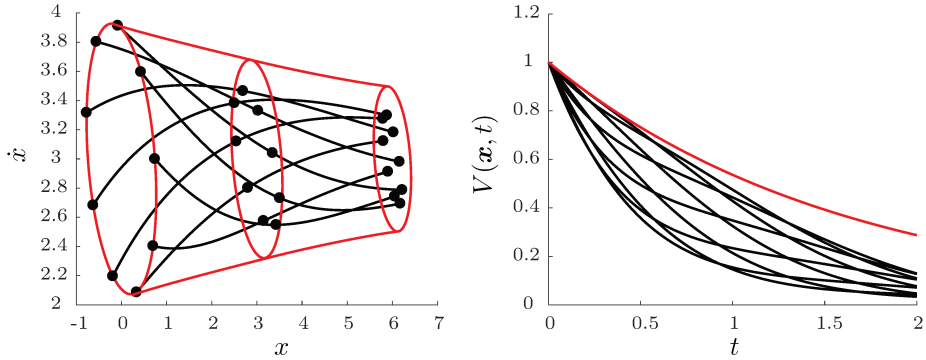


Figure 2.15 – This image showcases why it might by interesting to conjecture convergence time even in the case of LTI-systems. The system is defined by $A = \begin{bmatrix} 0 & 1 \\ -1. & -0.1 \end{bmatrix}$, $B = \begin{bmatrix} 0 & 1 \end{bmatrix}^{\mathrm{T}}$ controlled via the LQR feedback matrix resulting from $Q = Id$ and $R = Id$. Here the reference trajectory is given as the constant velocity trajectory $x(t) = 3t$. On the left the resulting system trajectories for different initial positions (shown in black) distributed on the boundary of the converging funnel (red lines) is shown. The convergence of the funnel is chosen as the minimally guaranteed convergence rate. On the right the corresponding Lyapunov values of the trajectories (black) and the funnel (red) are shown. As one can see all trajectories converge significantly faster, as the eigenvalues of $C_i^{\mathrm{T}}.(A^{\mathrm{T}}.P + P.A).C_i$ are -0.62 and -2.1. Therefore conjecturing a higher γ can be advantageous and a valid option if a minimal dwell time in the funnel is normally given.

points in the current direction of $\boldsymbol{e}_{c,x}$, so we get

$$\dot{\boldsymbol{p}} = \begin{pmatrix} \cos(\theta_p) \\ \sin(\theta_p) \end{pmatrix} v_p.$$

In this example we control directly the velocity v_p as well as the turning rate $\omega_p = \dot{\theta}_p$, but both control inputs must be continuous and bounded. The state space of the system is the concatenation of its position with respect to global frame and the heading:

$$\boldsymbol{x} = \begin{pmatrix} \boldsymbol{p} \\ \theta_p \end{pmatrix}.$$

The dynamics of the system is

$$\dot{\boldsymbol{x}} = \begin{pmatrix} \dot{\boldsymbol{p}} \\ \dot{\theta}_p \end{pmatrix} = \begin{pmatrix} v_p \cos(\theta_p) \\ v_p \sin(\theta_p) \\ \omega_p \end{pmatrix}.$$

We impose positive upper and lower bounds on the current velocity as well as bounds on the curvature of the resulting trajectory (corresponding to the turning cycle of the car), so that the control law introduced afterwards always has to satisfy

$$0 < v_m \leq v_p \leq v_M \tag{2.14a}$$

$$-c_M \leq {}^{\omega_p}/_{v_p} \leq c_M. \tag{2.14b}$$

To create a (conjectured) funnel we must first define reference trajectories and a control law. For the reference trajectory we use a continuously differentiable curve defined on an interval $I \subseteq \mathbb{R}^+$ denoted by

$$\boldsymbol{x}_{\mathrm{ref}}(t) = \begin{pmatrix} \boldsymbol{r}(t) \\ \theta_{\mathrm{ref}}(t) \end{pmatrix}$$

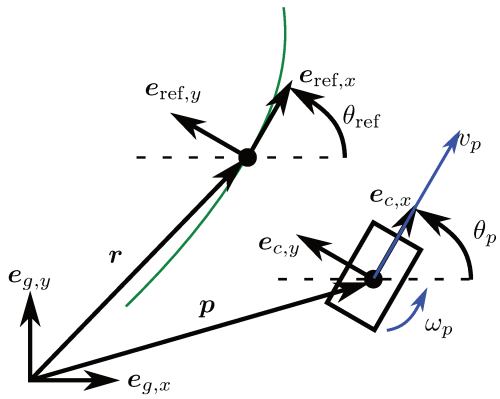


Figure 2.16 – Modified Dubins' car with controlled inputs v_p and ω_p. A possible reference trajectory and the current normal and tangent directions is shown in green and indexed by $_{\mathrm{ref}}$.

satisfying the conditions (2.14). In order to make sure that not only the curve is satisfying these constraints, but that there exists also some neighbourhood around the trajectory which can be driven towards it with a suitable control law, a margin between the absolute limits of the system and maximal values on the reference trajectory should exist. In addition the curve has to be admissible for the considered system, so it must hold that:

$$\forall t \in I : \dot{\boldsymbol{r}}(t) = \begin{pmatrix} \cos(\theta_r(t)) \\ \sin(\theta_r(t)) \end{pmatrix} v_r(t).$$

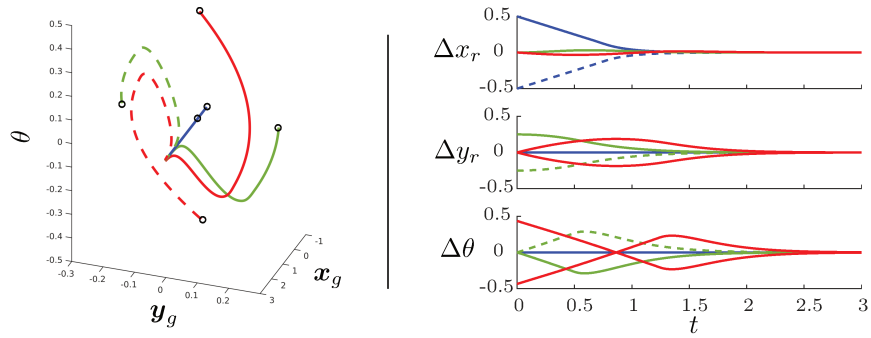
Every such curve can be used as a reference. The frame attached to the reference point is indexed by $_{\mathrm{ref}}$. The angle between the global $\boldsymbol{e}_{g,x}$ and the local $\boldsymbol{e}_{\mathrm{ref},x}$ axes (see Figure 2.16) is denoted θ_{ref}.

This nonlinear, nonholonomic dynamical system requires relatively complex control laws in order to ensure convergence. We propose the following scheme:

$$\begin{pmatrix} v_p \\ \omega_p \end{pmatrix} = \begin{pmatrix} v_r \\ \omega_r \end{pmatrix} - \begin{pmatrix} \alpha\, \Delta x_c \\ \beta(\Delta\theta + \zeta \tanh(\eta\, \Delta y_r)) \end{pmatrix} \tag{2.15}$$

where α, β and η denote parameters in $\mathbb{R}^+$, ζ is a parameter in $]0, \pi/2]$, Δx_c the projection of $\boldsymbol{p} - \boldsymbol{r}$ onto the $\boldsymbol{e}_{c,x}$-axis, Δy_r the projection of $\boldsymbol{p} - \boldsymbol{r}$ onto the $\boldsymbol{e}_{\mathrm{ref},y}$-axis and $\Delta\theta = \theta_p - \theta_r$. The resulting values are then saturated to respect the constraints in (2.14) (for example if $v_p > v_M$, $v_p = v_M$; if $\omega_p/v_p > c_M$, $\omega_p = c_M v_p$).

In this control law the term $\zeta \tanh(\eta\, \Delta y_r)$ is introduced to cope with the error in the orthogonal direction to the motion (Δy_c), which is not directly controllable due to the nonholonomy. So in order to correct an offset in the $\boldsymbol{e}_{c,y}$ direction, an error in the direction has to be introduced first. By choosing ζ large, preference is given to quickly correct this offset by allowing for higher errors in θ. We verify empirically the convergence properties of this control law: see Figure 2.17. Note that any other control law for the Dubins' car such as the ones proposed in Macharet et al. [2011] relying on different feedback modes and a finite state machine to switch between them or the approach based on sliding mode control proposed in Soueres et al. [2001] could be used as well. The reason why we did not choose one of these is due to the increased implementation complexity and the easy interpretation of the control law given in (2.15). Additionally, for both approaches cited above, the conjecture is harder to test numerically, as the control law has discontinuities (either by changing the state/feedback mode or by crossing the sliding surface), which makes it harder to draw general assumptions on the behaviour based on a finite number of testing points.



solid/dashed $e = 0.5/-0.5$: $\boldsymbol{p}(0) = \boldsymbol{r}(0) + [e\ 0]^{\mathrm{T}}$; $\Delta\theta(0) = 0$
solid/dashed $e = 0.25/-0.25$: $\boldsymbol{p}(0) = \boldsymbol{r}(0) + [0\ e]^{\mathrm{T}}$; $\Delta\theta(0) = 0$
solid/dashed $e = 45°/-45°$: $\boldsymbol{p}(0) = \boldsymbol{r}(0)$; $\Delta\theta(0) = e$

Figure 2.17 – Trajectories of the system for initial offsets in only one dimension and a reference trajectory of the form $\boldsymbol{r}(t) = [t,\, 0]^{\mathrm{T}}$, $\theta_{\mathrm{ref}}(t) = 0$. An initial offset only in the $\boldsymbol{e}_{\mathrm{ref},x}$-direction is corrected without inducing an error in the other components, since this direction is directly controllable. An initial error in the $\boldsymbol{e}_{\mathrm{ref},y}$-direction induces an error in the heading in order to be corrected and vice versa.

Optimizing the Funnel Shape and Computing the Tuples

For the bounding funnels we keep the ellipsoidal shaped funnels introduced in section 2.5 due to their computational advantages and extend them with the introduction of outer funnels:

$$\mathcal{F}_i^j(t) = \left\{ \boldsymbol{x}_{\mathrm{ref}} + \Delta\boldsymbol{x} \mid V_i^j(\Delta\boldsymbol{x}) \leq \alpha_i^j \right\} \tag{2.16}$$

$$\mathcal{F}_i^{\mathrm{O},j}(t) = \left\{ \boldsymbol{x}_{\mathrm{ref}} + \Delta\boldsymbol{x} \mid V_i^j(\Delta\boldsymbol{x}) \leq \alpha_i^{\mathrm{O},j} \right\} \tag{2.17}$$

where

$$V_i^j(\Delta\boldsymbol{x}) = \begin{bmatrix} \Delta\boldsymbol{r} \\ \Delta\theta \end{bmatrix}^{\mathrm{T}} . P_i^j . \begin{bmatrix} \Delta\boldsymbol{r} \\ \Delta\theta \end{bmatrix} \tag{2.18}$$

is a quadratic function defined by the symmetric and positive matrix P_i^j and $\alpha_i^{\mathrm{O},j} \geq \alpha_i^j \in \mathbb{R}^+$ are constants defining the size of the funnel. Here the outer funnels are chosen to be a scaled version of the inner funnel, but it is perfectly possible to chose any other (symmetric and positive) matrix $P_i^{\mathrm{O},j}$ to define the outer funnel. By restricting the size of the funnels in order to ensure that $\Delta\theta$ can never be smaller than $-\pi$ or larger than π, no special care has to be taken to correctly represent the "angular nature" of $\Delta\theta$, and it can be treated as if it were a regular scalar.

As pointed out above, the convergence time is approximated using numerical simulations and depends on the (parametrization of the) control law and the matrix P_i^j defining the funnel shape. As the dynamics of the system are invariant under a change of reference, i.e. it always behaves the same with respect to its own frame of reference, we use the same matrix for all funnels (causing them to have the same shape), by setting $P_i^j = P_k^l = P$ (when given with respect to the reference trajectory frame) to construct the larger funnels. To facilitate switching from and to funnels constructed around reference trajectories having different directions (different θ_{ref}), the smaller funnels constructed around the same reference trajectory are rotated (in the $\boldsymbol{e}_{c,x}$-$\boldsymbol{e}_{c,y}$-plane) and scaled versions of the larger funnels. In order to find a suitable ellipsoid and the corresponding control law parameters, the following optimization is performed: we fix *a priori* a diagonal matrix $D_L = \mathrm{diag}(\begin{bmatrix} 0.4^2 & 0.4^2 & (^{80\pi}/_{180})^2 \end{bmatrix})$ which is suited to englobe smaller funnels constructed around reference trajectories with a difference in θ_{ref} of up to 60°. This diagonal matrix is

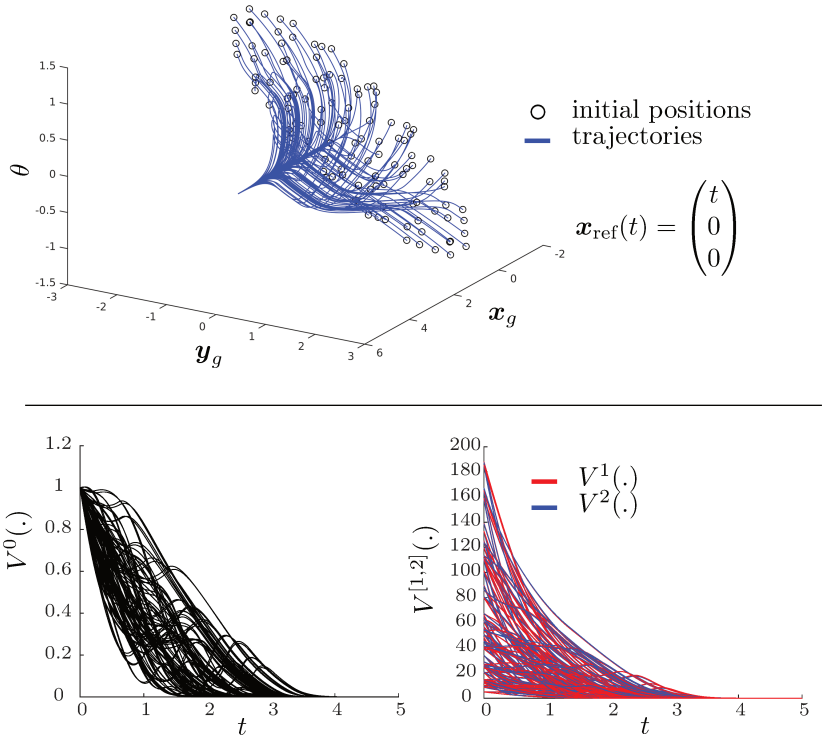


Figure 2.18 – On the left, the trajectories for initial states distributed on the surface of the optimized funnel shape $\mathcal{F}^0$ are shown. The control parameters are $\alpha = 4.43$, $\beta = 7.94$, $\eta = 2.94$ and $\zeta = 4.57$. The dynamics induced by these parameters are denoted $f(.)$. The second image depicts the evolution of $V^0(\Delta\boldsymbol{x})$ with the large funnel $\mathcal{F}^0$ being defined as $V^0(\Delta\boldsymbol{r}, \Delta\theta_r) \leq \alpha_i^0 = 1.0$. The maximal value encountered is 1.0, so the associated outer funnel $\mathcal{F}^{\text{O},0}$ can be chosen equal to $\mathcal{F}^0$. Note that even-though $\mathcal{F}^0(t) = \mathcal{F}^{\text{O},0}(t)$ the convergence is highly non-monotonic. The third image shows the evolution of V^1 (red) and V^2 (blue). After 3.4 time units all states have converged to the small funnels $\mathcal{F}^1$ and $\mathcal{F}^2$.

rotated during optimization (parametrized via three Euler angles) in order to minimize the convergence time to the two smaller funnels.

The optimization, relying on gradient descent methods to find a local optimum, resulted in a minimal convergence time of 3.4 time units for the optimized funnel shape and control law parameters as shown in Figure 2.18.

These results conjecture the absorption time between a larger and two smaller funnels as well as the equality of the outer and inner funnel (for the larger funnel). These bounding funnels can be constructed around any reference trajectory of the form

$$\boldsymbol{x}_{\text{ref}}(t) = \begin{pmatrix} x_0 \\ y_0 \\ \theta_0 \end{pmatrix} + \begin{pmatrix} \cos(\alpha) \\ \sin(\alpha) \\ 0 \end{pmatrix}$$

with $x_0, y_0 \in \mathbb{R}$, $\theta_0, \alpha \in [0, 2\pi[$, so any straight line trajectory with unit velocity.

Path Planning for the Dubins' car on a Funnel System

The objective is to perform (timed) path planning for the Dubins' car. That is to find a timed sequence of transitions between reference trajectories that brings the system from an initial region $\Omega_0 = \mathcal{F}_{i0}^{j0}(t_0)$ to a final region $\Omega_1 = \mathcal{F}_{i1}^{j1}(t_1)$. To achieve this, we need to construct a suitable system of reference trajectories, around which we can then define the (conjectured) funnels. By requiring the model checker to supply the fastest trace (i.e. a sequence with minimum time elapsed on the global clock c_g), we expect the kind of solutions as depicted in Figure 2.19 on the left for different sets of regions.

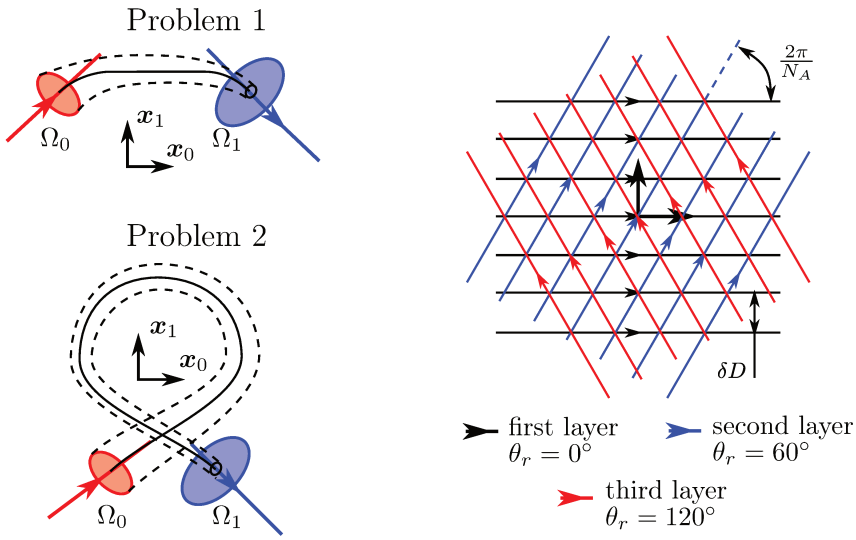


Figure 2.19 – On the left: Depiction of two instances of the problem and an "optimal" or desirable reference trajectory (solid black line). The dashed black line shows the qualitative evolution of funnel taking Ω_0 to Ω_1. In Problem 2 a reference trajectory leading directly from Ω_0 to Ω_1 would violate the curvature or velocity bounds, forcing the loop-like reference trajectory. On the right: depiction of the first three layers of reference trajectories for $\theta_r^{i,j} = \alpha_j \in [0°, 60°, 120°]$. Such a funnel system is likely to generate the kind of desired movement.

The reference trajectories used to construct the funnel system for the examples shown form a regular grid: The first layer is composed of $2N_D + 1$ trajectories with $\boldsymbol{e}_{\text{ref},x}$ parallel to $\boldsymbol{e}_{g,x}$ of the form

$$-N_D \leq i \leq N_D : \ \boldsymbol{x}_{\text{ref}}^i(t) = \begin{pmatrix} 0 \\ i\,\delta D \\ 0 \end{pmatrix} + \begin{pmatrix} tv_r - N_D\,\delta D \\ 0 \\ 0 \end{pmatrix}$$

defined on the interval $I^i = [0, {}^{(2N_D\,\delta D)}/_{v_r}]$. So δD defines the distance between two neighbouring trajectories and by fixing N_D one can determined the size of the region covered by the funnel system.

The other layers are formed by rotating the first layer around the θ axis, considering a 3D Cartesian representation of the state space. We use N_A such layers, each of the trajectories having the form

$$-N_D \leq i \leq N_D,\, 0 \leq j \leq N_A - 1 : \ \boldsymbol{x}_{\text{ref}}^{i,j}(t) = R_\theta(\alpha_j).\left(\begin{pmatrix} 0 \\ i\,\delta D \\ \alpha_j \end{pmatrix} + \begin{pmatrix} tv_r - N_D\,\delta D \\ 0 \\ 0 \end{pmatrix}\right)$$

with $\alpha_j = {}^{(2\pi j)}/_{N_A}$, $1 \leq j \leq N_A$ and $R_\theta(\alpha_j)$ denoting the rotation matrix corresponding to a rotation of angle α_j around the θ-axis. By fixing N_A, one can determine the angular offset between two neighbouring

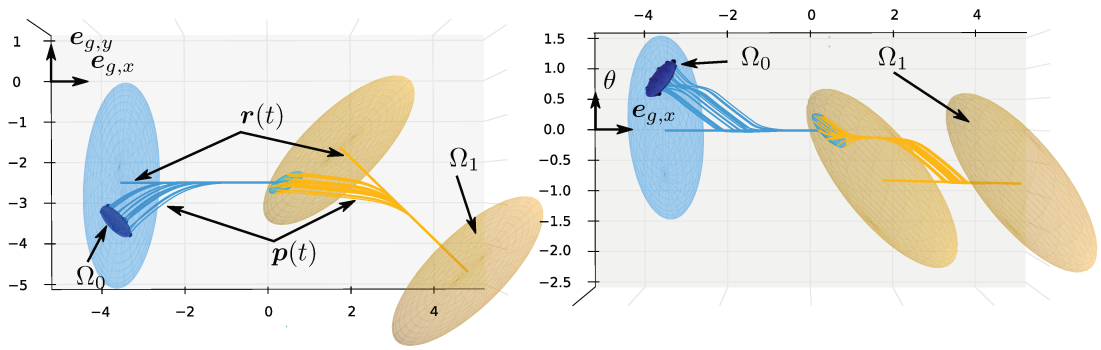


Figure 2.20 – Time-optimal solution found for problem 1. At every switching transition from $\mathcal{F}_{i,j}(t)$ to $\mathcal{F}^{\mathrm{O}}_{k,l}(t')$, we distribute states over the surface of $\mathcal{F}_{i,j}(t')$ and show two different projections of their trajectories $\boldsymbol{x}(t)$ until the next switching transition.

layers of parallel trajectories. Naturally we have that the angular difference between $\boldsymbol{x}^{i,0}_{\mathrm{ref}}$ and $\boldsymbol{x}^{i,N_A-1}_{\mathrm{ref}}$ is the same as the difference between $\boldsymbol{x}^{i,0}_{\mathrm{ref}}$ and $\boldsymbol{x}^{i,1}_{\mathrm{ref}}$.

On each of these reference trajectories three funnels of different sizes are defined. The funnels defined on the reference $\boldsymbol{x}^{i,j}_{\mathrm{ref}}$ are denoted $\mathcal{F}^0_{i,j}$ (the 'large' funnel), $\mathcal{F}^1_{i,j}$ (the *small* funnel connecting to the layer $j+1$), $\mathcal{F}^2_{i,j}$ (the *small* funnel connecting to the layer $j-1$) and $\mathcal{F}^{\mathrm{O,x}}_{i,j}$ (the associated outer funnels). Their respective sizes are chosen such that transitions exist from each small funnel $F^{1/2}_{i,j}$ to its direct neighbours in the same layer $\mathcal{F}^0_{i\pm1,j}$ (on the parallel reference trajectories) and to all the large funnels in the layer above $\mathcal{F}^1_{i,j} \to \mathcal{F}^{\mathrm{O},0}_{k,j+1}$ and below $\mathcal{F}^2_{i,j} \to \mathcal{F}^{\mathrm{O},0}_{k,j-1}$. As no switching transition *to* the small funnels exists, the associated outer funnels are not important in this case. Note that due to the angular nature of the third coordinate, there exist also transition between the reference trajectories $\boldsymbol{x}^{i,0}_{\mathrm{ref}}$ and $\boldsymbol{x}^{i,N_A-1}_{\mathrm{ref}}$.

In this example, we chose $N_D = 12$ and $N_A = 6$ resulting in a total of $25 \cdot 7 \cdot 3 = 525$ funnels. Due to the symmetry of the funnel system the number of available transitions can be approximated: on any reference trajectory, only the small funnels have outgoing transitions. Each of the two small funnels $F^{1/2}_{i,j}$ is connected to the large funnels $F^0_{i\pm1,j}$ on each of the 125 possible transition points. Furthermore there is an average of six transitions between a small funnel $F^{1/2}_{i,j}$ and any of the large funnels in the layer above ($F^0_{k,j+1}$) or below ($F^0_{k,j-1}$). So in total the automaton has $25 \cdot 7 \cdot 2(2 \cdot 125 + 25 \cdot 6) = 140.000$ transitions between 525 states.

This funnel system allows to conveniently switch the heading direction and specific funnels needed to attain a certain direction can easily be added if needed.

In the two examples presented, we consider that the initial region is centred around $\theta = 45°$ and the desired final region is centred around $\theta = -45°$. The decisive difference between the two problems is the distance (in $\boldsymbol{e}_{g,x}$-direction) between the regions as shown in Figure 2.19 on the left.

The results obtained using the funnel system described above are shown in Figure 2.20 and Figure 2.21. The generated reference trajectories are qualitatively similar to the optimal ones shown in Figure 2.19. The resulting system trajectories satisfy the specifications and are time-optimal (for the funnel system considered, not for the general case).

As shown in this example, bounding funnels (with conjectured properties) are a promising method to perform certified planning for general nonlinear systems. The resulting time-optimal trajectories for the Dubins' car resemble the classical optimal Dubins' trajectories, so trajectories of constant velocity with bounded curvature (Dubins [1957]). But in contrast to these trajectories, the here presented solution guarantees a stabilizable neighbourhood of the trajectory and allows to impose additional timing or logical constraints.

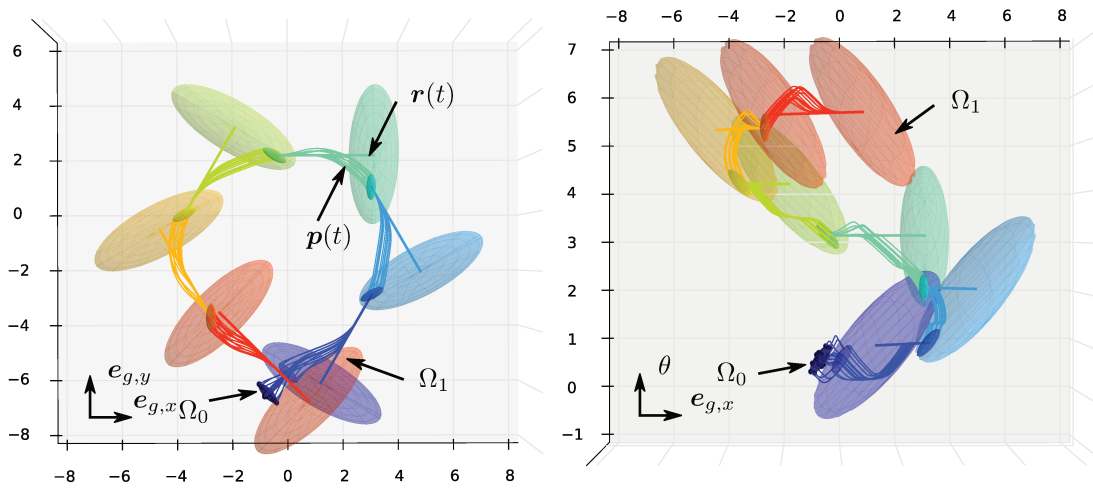


Figure 2.21 – Time-optimal solution found for problem 2.

The advantage of the proposed approach to conjecture the properties lies not only in the ability to treat nonlinear systems, but also in the possibility to adapt the funnel shape with respect to the needed convergence time without the additional constraint of monotonic convergence.

2.8 Conclusion and Future Work

In this chapter a novel timed-automata abstraction for switched dynamical systems relying on (control) funnels, i.e. time-varying regions of positive invariance is presented. Commonly used verification tools like Uppaal can be used to solve Reach-Avoid problems on the so generated abstractions and more complicated tasks can be solved by verifying the product automaton of the automaton representing the controlled dynamical system and another one representing additional tasks and constraints. These tasks and constraints can, in contrast to other approaches for similar problems existing in the literature, have quantitative timing constraints. This ability is highlighted within the pick-and-place example, necessitating the fulfilment of strict timing constraints in order to achieve task completion. Moreover it was shown how such abstractions can be used within the context of timed games to automatically synthesize reactive control strategies from given specifications.

In order to enlarge the class of dynamical systems amenable to this type of formal reasoning, bounding funnels with conjectured properties are introduced. These funnels can be generated for dynamical systems for which automated stability proofs, necessary to establish the positive invariance, are beyond the state of the art. They allow to conveniently interface numerical optimization procedures and formal reasoning as shown in the example performing path-planning for the Dubins' car.

To address more complex dynamical systems in the sense of a larger state space and to reduce the computational complexity of this approach, several interesting avenues remain to be investigated. In this work, the funnels have been created *a priori* and then abstracted to a timed automaton. This is feasible for the "small" examples, but gets quickly tedious for more complicated problems. Moreover the complexity of the Reach-Avoid problem is directly related to the number of discrete states and transitions between them. Therefore the number of funnels and their degree of connectivity is a good indicator for the complexity of the verification process. We therefore investigate ways to automatically generate a suitable but "small" funnel system directly from the task specification and dynamical system at hand. To go even further, it would be interesting to generate new funnels on the fly while verifying the automaton, that is to directly interface the verification and funnel construction process and to investigate the arising

decidability issues.

Another line of work related to the presented timed automata abstraction concerns the representation of general nonlinear systems. In section 2.7 we present bounding funnels with conjectured properties to treat such systems, however the associated numerical verification was only addressed within the examples, as no generic method can be given. Even though it is possible to conjecture the funnels based only on numerical evaluations, any method that increases the confidence in these conjectures can be of interest. Moreover empirically testing dynamical systems with a higher dimensional state space is also a challenging task, as the number of points to verify tends to grow exponentially with the system dimension. Therefore in chapter 3 a way to compute funnels for systems with polynomial vector fields is presented. Moreover it is shown how it can be used to approximate general nonlinear dynamical systems around predefined reference trajectories. By coupling the formal proof of positive invariance for the polynomial approximation of the system together with numerical evaluations of the true nonlinear dynamics, we might be able to obtain conjectures of high quality and low conservativeness.

Chapter 3

Stability of Dynamical Systems

In this chapter we are interested in the stability, or more precisely stabilizability, of dynamical systems. We present sufficient conditions for a polynomial system to be stabilizable on a subset of the state space based on Lyapunov theory and ideas from optimal control. The so derived conditions can be efficiently verified using semidefinite programming, a subclass of convex programming. The derived conditions directly prove stabilizabilty and therefore avoid the search for an explicit control law. The resulting optimization problems are therefore convex and computationally less demanding. Furthermore it is shown how this approach can be used to find an inner approximation of the true zone of stabilizability and how to construct time-varying regions of stabilizability along given reference trajectories.

3.1 Introduction

Executing robotic tasks in the presence of safety or timing constraints in a robust fashion requires not only that the reference trajectory satisfies these constraints but also that there exists a region around this trajectory which is guaranteed to converge towards it and also respects the constraints. Given a dynamic model of the robotic system, guaranteeing convergence on regions of the configuration space is usually done with stability certificates, which are computationally generated formal proofs based on Lyapunov theory or contraction analysis. However obtaining such stability certificates for nonlinear systems, such as walking or flying robots, or even simpler systems such as the Acrobot (Majumdar et al. [2013b]), that induce such regions is notoriously difficult and remains a challenging problem despite the enormous progress made in recent years using a variety of different approaches. These approaches include, but are not limited to, outer approximation via occupation measures (Henrion and Korda [2014]), counter-example guided synthesis (Ravanbakhsh and Sankaranarayanan [2016]) and sum-of-squares (SoS) approaches (Majumdar et al. [2013b]; Jarvis-Wloszek et al. [2003]) and more tractable relaxations of SoS-approaches (Ahmadi and Majumdar [2014]). The most comparable to the approach presented in this chapter are the methods mentioned last, based on approximating the system as polynomial using a truncated Taylor expansion and prove the convergence of the approximated system with respect to a polynomial, often quadratic, Lyapunov function using SoS optimization.

When the system has control inputs, what is required are not certificates of stability, often provided as a pair control law plus Lyapunov function, but certificates of stabilizabilty. Such certificates prove that for bounded control inputs, there always exists at least one input that brings the system back to its

reference. This is necessary to properly model robots, since their actuators can only provide a limited amount of effort (e.g. joint actuators are usually limited in torque), which makes obtaining certificates more difficult, especially for SoS techniques as shown later on. Moreover these certificates must be constructive and yield ways to compute such control inputs. Therefore in this chapter an approach to prove exponential stabilizability of a controlled polynomial system under input constraints with respect to quadratic Lyapunov function candidates is presented. This approach is based on two principles: state-space partitioning and convexification. This leads to a formulation that inherently yields certificates of *local* stabilizability and takes into account the boundedness of the input in a very natural way.

In the remainder of this chapter it is shown how to partition the state-space into subsets defined by the optimal control input and how this relates to proving stabilizability for a given region and system (section 3.5). This is done after briefly discussing the necessary concepts and tools in section 3.2. In section 3.8 the applied method to prove non-positiveness of multivariate polynomials based on an extension of Reformulation and Linearisation Techniques (RLT) is presented and finally in section 3.10 numerical results for the Acrobot, a torque controlled simple pendulum and controlled polynomial vector fields are presented.

The contributions detailed in this chapter can be summed up as follows.

Contributions

- Deriving a state-space partitioning based on convergence optimal control input
- Derive sufficient conditions for stabilizability on the subsets forming the partition
- Solving the arising optimization problem for polynomial dynamics based on an extension of the Reformulation and Linearisation Technique
- Showcasing the effectiveness of the proposed approach on a set of well-known test-cases

The material presented in this section was (in parts) published in Schlehuber-Caissier and Perrin [2018].

3.2 Theoretical Background

In this section the theoretical background necessary to prove stability for nonlinear systems, with an accent on the special case of polynomial systems and quadratic Lyapunov functions, is briefly recapitulated. For a more in-depth discussion the interested reader is referred to, for instance Khalil [1996].

3.2.1 Convex Optimization and Semidefinite Programming

To make it short, from all well-defined optimization problems, the set of convex problems is the only one which can be efficiently solved, in the sense of finding the global optimum and the minimizing variables. Or to put it more formally, the set of convex problems which have a conic form, such as linear programs (LP), quadratic programs (QP), second order cone programs (SOCP) or semidefinite programs (SDP) can be solved to arbitrary precision in polynomial time using interior-point methods, see Nesterov and Nemirovskii [1994]; Boyd and Vandenberghe [2004]. The class of optimization problems on which we will rely throughout this chapter is semidefinite programming, the most general class of the above cited, and which received enormous attention in the last two decades due to significant advances in the theory of convex optimization but also due to the appearance of publicly available high quality solvers.

They are optimization problems over the cone of symmetric positive semidefinite (psd) matrices denoted $\mathbb{S}^n_+$, so all matrices X for which we have $X = X^{\mathrm{T}}$ and $\lambda(X) \geq \mathbf{0}$, meaning that all eigenvalues of X are non-negative, denoted $X \succeq \mathbf{0}$.
A SDP in its standard form is given as

$$\begin{aligned} \text{minimize} \quad & \operatorname{tr}(C.X) && (3.1a)\\ \text{subject to} \quad & \operatorname{tr}(A_i.X) = b_i,\ \forall i && (3.1b)\\ & X \succeq 0. && (3.1c) \end{aligned}$$

The objective is a linear function of the decision variable $X \in \mathbb{S}^n$, so the set of all symmetric matrices of size $n \times n$, and the weighting matrix $C \in \mathbb{S}^n$. The optimization is subject to a non-negativity (equation (3.1c)) and a set of linear constraints (equation (3.1b)). Note that $\operatorname{tr}(Y.X)$, with $Y, X \in \mathbb{S}$, is the general real-valued linear function on $\mathbb{S}$, see Boyd and Vandenberghe [2004] section. 4.6. Note that a SDP, due to the self-duality of $\mathbb{S}^n_+$, can equivalently be written as (can be written in its dual form as)

$$\begin{aligned} \text{minimize} \quad & \boldsymbol{c}^{\mathrm{T}}.\boldsymbol{x} && (3.2a)\\ \text{subject to} \quad & \sum_i \boldsymbol{x}_i F_i + G \preceq \mathbf{0} && (3.2b)\\ & A\boldsymbol{x} = \boldsymbol{b}. && (3.2c) \end{aligned}$$

with F_i, $G \in \mathbb{S}^n$, $A \in \mathbb{R}^{p\times n}$ and $\boldsymbol{c} \in \mathbb{R}^n$, which is often a more convenient notation.

Semidefinite programs are very versatile as they include Linear Programming (LP), Quadratic Programming (QP) and Second-Order-Cone Programming (SOCP) as special cases, resulting in specific structures of the constraint and objective matrices and have many practical applications. Semidefinite programs can be efficiently solved, as the cone $\mathbb{S}_+$ is self-dual and therefore an SDP is a special case of cone programming. The most broadly utilized algorithm to solve SDPs is the interior point method (see for instance Sturm [1999]) for small and medium sized problems or methods based on the alternating directions method which is (partly) amenable to efficient GPU implementations suitable for large problems, see O'Donoghue et al. [2016].

3.2.2 Lyapunov Stability

Lyapunov stability theory dates back to end of the 19th century when it was first published by the Russian mathematician Aleksandr Mikhailovich Lyapunov (Lyapunov [1892]), but its importance, with respect

to control theory, was largely undiscovered until the works of Nikolaĭ Gurevich Chetaev (Chetaev [1961]) and Joseph La Salle (LaSalle [1976]; La Salle and Lefschetz [2012]) picking up on Lyapunov's discoveries and expanding the theory, see Parks [1992].

Much of the popularity of Lyapunov theory comes from what is called Lyapunov's second method, also simply known as Lyapunov stability criterion. This approach had a deep impact onto the control community from around 1960 on. It provides conditions for an autonomous system to be asymptotically/exponentially stable, which can be checked in practice for many systems. It is also a crucial building block for the method presented in the rest of the chapter and is therefore presented in greater detail here.

Lyapunov's Second Method The autonomous system $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$ with $\boldsymbol{x}, \dot{\boldsymbol{x}} \in \mathbb{R}^n$ and a continuous function $V(\boldsymbol{x})\colon \mathbb{R}^n \to \mathbb{R}$ prove that the origin is a stable equilibrium if

$$V(\boldsymbol{x}) = 0 \text{ iff } \boldsymbol{x} = 0 \tag{3.3a}$$

$$V(\boldsymbol{x}) > 0 \text{ iff } \boldsymbol{x} \neq 0 \tag{3.3b}$$

$$\forall \boldsymbol{x} \in \mathbb{R}^n \setminus \{\boldsymbol{0}\}: \ \dot{V}(\boldsymbol{x}) = \langle \frac{\partial}{\partial \boldsymbol{x}} V(\boldsymbol{x}), \frac{\mathrm{d}}{\mathrm{d}\, t} \boldsymbol{x} \rangle = \nabla_{\boldsymbol{x}} V(\boldsymbol{x}).f(\boldsymbol{x}) \leq 0 \tag{3.3c}$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product and $\nabla_{\boldsymbol{x}}$ is defined in the usual way as the (row-) vector of partial derivatives with respect to $\boldsymbol{x}$. Even though it is sufficient for Lyapunov theory that $V(\boldsymbol{x})$ is continuous, we will generally assume the stronger condition that it is also differentiable.

One can interpret the above conditions in the following way: $V(\boldsymbol{x})$ corresponds to a potential function (see Figure 3.1), as it is everywhere positive except at the origin and its value is not allowed to increase over time along any trajectory of the system. This ensures that for a trajectory with initial position $\boldsymbol{x}_0$, $V(\boldsymbol{x}(t)) \leq V(\boldsymbol{x}_0)$ holds for all $t \in \mathbb{R}_+$ and therefore the equilibrium point at the origin is *stable*. An equivalent notion of stability can be given as

$$\forall \epsilon > 0, \ \exists \delta = \delta(\epsilon) > 0, \text{ such that } \|\boldsymbol{x}_0\|_2 < \epsilon \implies \|\boldsymbol{x}(t)\|_2 < \delta \text{ holds for all } t \geq 0. \tag{3.4}$$

In order to ensure that the stability definition in (3.4) implies the one given in (3.3) and vice-versa in the entire space, and not only locally around the equilibrium point, $V(\boldsymbol{x})$ has to be radially unbounded, meaning that $\|\boldsymbol{x}\| \to \infty \Rightarrow V(\boldsymbol{x}) \to \infty$ for any norm $\|\cdot\|$.

By changing (3.3c) to be

$$\forall \boldsymbol{x} \in \mathbb{R}^n \setminus \{\boldsymbol{0}\}: \ \dot{V}(\boldsymbol{x}) = \nabla_{\boldsymbol{x}} V(\boldsymbol{x}).f(\boldsymbol{x}) < 0 \tag{3.5}$$

ones obtains a proof that the origin is *asymptotically* stable, as the value of $V(\boldsymbol{x})$ has to decrease along the trajectory at each moment on every trajectory. In order to prove *exponential* stability with convergence rate $\gamma \geq 0$ (3.3c) is replaced by

$$\forall \boldsymbol{x} \in \mathbb{R}^n \setminus \{\boldsymbol{0}\}: \ \dot{V}(\boldsymbol{x}) = \nabla_{\boldsymbol{x}} V(\boldsymbol{x}).f(\boldsymbol{x}) < -\gamma V(\boldsymbol{x}). \tag{3.6}$$

Indeed, if (3.6) holds, for the trajectory with the initial position $\boldsymbol{x}_0$ at $t = 0$, $V(\boldsymbol{x}(t)) < V(\boldsymbol{x}_0)e^{-\gamma t}$ is true for all $t \in \mathbb{R}_+$. This result is very useful as it gives a decreasing upper bound for the $V(\boldsymbol{x}(t))$ and will be used extensively.

Till here it was shown how to use Lyapunov's second method to prove the global (asymptotic / exponential) stability of the origin, in practice however it is often the case that the system dynamics possess only local stability. In order to restrict the proof to local stability, one has to introduce a suitable subset of the state space Ω_0 on which one seeks to prove stability. A natural choice is to define Ω_0 as a sublevel-set of the Lyapunov function candidate $V(\boldsymbol{x})$, $\Omega_0 = \{\boldsymbol{x} | V(\boldsymbol{x}) \leq \alpha_0\}$. The autonomous system is

then (asymptotically / exponentially) stable for Ω_0 if equations (3.3) (using (3.5)/(3.6)) hold $\forall \boldsymbol{x} \in \Omega_0$. Note that the constraints on $V(\boldsymbol{x})$ for asymptotical/exponential stability (eqs. 3.5/3.6) ensure that $V(\boldsymbol{x})$ has no local minima in Ω_0, as the gradient would vanish at these minima. This guarantees that all sets $\Omega_\alpha = \{\boldsymbol{x} | V(\boldsymbol{x}) \leq \alpha\}$ for $\alpha \in]0, \alpha_0]$ are connected, which is not necessarily the case when proving stability (using (3.3c)).

Another, sometimes easier to grasp, interpretation of Lyapunov's second method is geometric: Given the dynamical system $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$ and a region of the state space defined as the sublevel-set of the Lyapunov function $\Omega_0 = \{\boldsymbol{x} | V(\boldsymbol{x}) \leq \alpha_0\}$, the vector $\dot{\boldsymbol{x}}$ has to point to the interior of $\Omega = \{\boldsymbol{x} | V(\boldsymbol{x}) \leq \alpha\}$ at every point $\boldsymbol{x}$ on the boundary of Ω ($\forall \boldsymbol{x} \in \partial\Omega$) for all $\alpha \in [0, \alpha_0]$, see Figure 3.1.

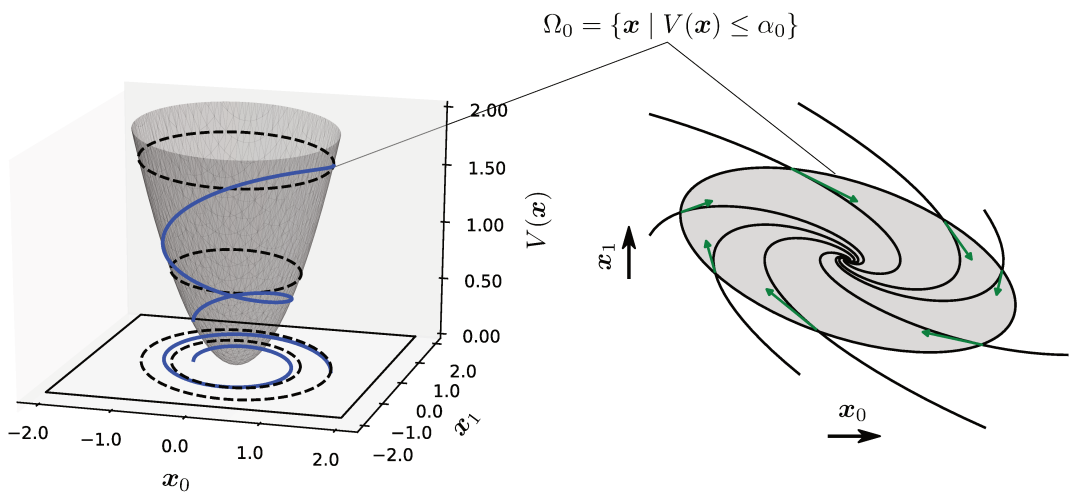


Figure 3.1 – The image on the left (Adopted from Ahmadi [2008]), depicts the interpretation of the Lyapunov function as potential, which decreases along the trajectory if the system is asymptotically stable. On the right the geometric interpretation is depicted. The velocity vector $\dot{\boldsymbol{x}}$ for the point on each trajectory $\boldsymbol{x}_\Omega$ for which $V(\boldsymbol{x}_\Omega) = \alpha_0$ is depicted by the green arrows.

Control Lyapunov Functions Control Lyapunov functions (CLF) are a useful extension to Lyapunov theory, see for instance Blondel et al. [2012], as they enlarge the class of systems considered from autonomous to controlled nonlinear systems. So the system dynamics can now be written as

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}) \tag{3.7}$$

where $\boldsymbol{u} \in \mathcal{U} \subseteq \mathbb{R}^m$ is the m-dimensional control input vector, that may be restrained to some set $\mathcal{U}$ respecting possible input constraints.

In the case of control Lyapunov functions, the constraints on $V(\boldsymbol{x})$ (equations (3.3a) and (3.3b)) are not modified, however equation (3.3c) becomes

$$\forall \boldsymbol{x},\ \exists \boldsymbol{u} \in \mathcal{U}\colon\ \dot{V}(\boldsymbol{x}) = \nabla_{\boldsymbol{x}} V(\boldsymbol{x}).f(\boldsymbol{x}, \boldsymbol{u}) \leq 0. \tag{3.8}$$

This constraint can be trivially adjusted for the case of asymptotic / exponential stabilizability and its local formulation. Moreover, in the uncontrolled case the origin has to be an equilibrium point, or $f(\mathbf{0}) = \mathbf{0}$. In the controlled case this constrained is also weakened to

$$\exists \boldsymbol{u}_0 \in \mathcal{U}\colon\ f(\mathbf{0}, \boldsymbol{u}_0) = \mathbf{0}. \tag{3.9}$$

The informal description of the above constraints is, that for every point in the considered region, there exists a valid control input which ensures that the value of the potential function does not increase, or, in the spirit of the geometric interpretation, ensures that the vector $f(\boldsymbol{x}, \boldsymbol{u})$ points into the interior of the region, see Figure 3.2.

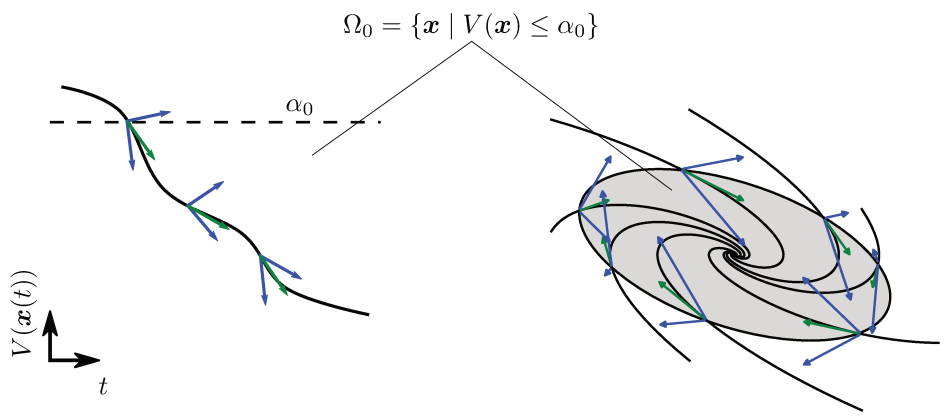


Figure 3.2 – Again the image on the left, depicts the interpretation of the Lyapunov function as potential, however now the derivative at each point (green arrows) could have taken multiple values corresponding to the possible $\boldsymbol{u} \in \mathcal{U}$ (limit cases depicted by the blue arrows) and can be interpreted as differential inclusion. On the right, the geometric interpretation is shown. The velocity vector $\dot{\boldsymbol{x}}$ for the point on the trajectory $\boldsymbol{x}_\Omega$ for which $V(\boldsymbol{x}_\Omega) = \alpha_0$ is depicted by the green arrows and again the limit cases of the possible range of velocity vectors as a function of the chosen control input is depicted by the blue arrows.

In order to clarify notations, the terms invariant region, region of attraction (RoA) and region of stabilizability (RoS) are now formally introduced.

- We say that a region $\Omega \subset \mathbb{R}^n$ is an invariant for the dynamical system $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$ iff from $\boldsymbol{x}_0 \in \Omega$ it follows that $\boldsymbol{x}_t \in \Omega$ forall $t \geq 0$.

- We say that a region $\Omega \subseteq \mathbb{R}^n$ is a region of attraction for the dynamical system $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$ with the equilibrium point $\boldsymbol{x}^* \in \Omega$ iff from $\boldsymbol{x}_0 \in \Omega$ it follows that $\lim_{t\to\infty} \boldsymbol{x} \to \boldsymbol{x}^*$, or equivalently Ω is asymptotically stable.

- We say that a region $\Omega \subseteq \mathbb{R}^n$ is a region of stabilizability for the dynamical system $\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u})$ with the equilibrium pair $(\boldsymbol{x}^*, \boldsymbol{u}^*) \in (\Omega, \mathcal{U})$ iff for every $\boldsymbol{x}_0 \in \Omega$ there exists a time-dependent control signal $\boldsymbol{u}(t) \in \mathcal{U}$ respecting the control input constraints driving the state to the equilibrium $\boldsymbol{x}^*$ in finite time.

For practical reasons, the regions used to prove invariance / stability / stabilizability in this work are defined as sublevel-sets of quadratic Lyapunov function candidates, so $\Omega = \left\{\boldsymbol{x} | V(\boldsymbol{x}) = \boldsymbol{x}^{\mathrm{T}}.P.\boldsymbol{x} = \|\boldsymbol{x}\|_P^2 \leq \alpha\right\}$ and the corresponding criteria with respect to the Lyapunov's second method become

- (invariant region) $\forall \boldsymbol{x} \in \partial\Omega\colon\ \nabla_{\boldsymbol{x}} V(\boldsymbol{x}).f(\boldsymbol{x}) \leq 0$

- (region of attraction) $\forall \boldsymbol{x} \in \Omega \setminus \{\mathbf{0}\}: \ \nabla_{\boldsymbol{x}} V(\boldsymbol{x}).f(\boldsymbol{x}) \leq 0$

- (region of stabilizability) $\forall \boldsymbol{x} \in \Omega \setminus \{\mathbf{0}\}, \exists \boldsymbol{u} \in \mathcal{U}: \ \nabla_{\boldsymbol{x}} V(\boldsymbol{x}).f(\boldsymbol{x}, \boldsymbol{u}) \leq 0$.

3.2.3 Contraction Analysis

Contraction analysis was introduced by Winfried Lohmiller and Jean-Jacques Slotine in the '90s (Lohmiller and Slotine [1998]) and since then gained a lot of interest in the control research community as it provides a different approach for proving the stability of nonlinear systems. In contrast to Lyapunov's stability criterion, which seeks to prove stability for a known equilibrium point, contraction theory considers the evolution of the distance between neighbouring trajectories. The main idea is that if the distance between any two trajectories decreases over time (with respect to some metric), then they will converge to the same trajectory and under additional conditions to an equilibrium point. By extension, as any point (in the stable region) can be taken as initial position for the trajectory, all points will converge. The location of the equilibrium does therefore not have to be known in advance and the distance between two trajectories can be defined as a virtual displacement. To put it formally without going into the details:

Given the dynamical autonomous system $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$, the virtual dynamics are defined as

$$\delta\dot{\boldsymbol{x}} = \frac{\partial}{\partial \boldsymbol{x}} f(\boldsymbol{x}).\delta\boldsymbol{x} = F(\boldsymbol{x}).\delta\boldsymbol{x} \tag{3.10}$$

where $\delta\boldsymbol{x}$ is the virtual displacement and $\delta\dot{\boldsymbol{x}}$ is the virtual velocity. By introducing the state-dependent (non-flat) metric $M(\boldsymbol{x})$, the distance of two points given as the virtual displacement $\delta\boldsymbol{x}$ with respect to $M(\boldsymbol{x})$ is

$$d(\delta\boldsymbol{x}) = \delta\boldsymbol{x}^{\mathrm{T}}.M(\boldsymbol{x}).\delta\boldsymbol{x}. \tag{3.11}$$

The derivative with respect to time of d considering the system dynamics is

$$\dot{d}(\delta\boldsymbol{x}) = \delta\boldsymbol{x}^{\mathrm{T}}.\left(F(\boldsymbol{x})^{\mathrm{T}}.M(\boldsymbol{x}) + M(\boldsymbol{x}).F(\boldsymbol{x})\right).\delta\boldsymbol{x}. \tag{3.12}$$

As stated above, the dynamical system is stable if the distance between neighbouring trajectories decreases over time. This is the case if $\dot{d}(\delta\boldsymbol{x}) < 0$ which can be relaxed to

$$F(\boldsymbol{x})^{\mathrm{T}}.M(\boldsymbol{x}) + M(\boldsymbol{x}).F(\boldsymbol{x}) \prec \mathbf{0} \tag{3.13}$$

as shown in section 3.2.4.

Note that in order for $M(\boldsymbol{x})$ to be a metric, one has to ensure that

$$\forall \boldsymbol{x}: M(\boldsymbol{x}) = M(\boldsymbol{x})^{\mathrm{T}} \tag{3.14a}$$

$$\forall \boldsymbol{x}: M(\boldsymbol{x}) \succ \mathbf{0}. \tag{3.14b}$$

Stability proofs based on contraction analysis can also be modified to prove exponential convergence or be restricted to a local proof in a similar fashion as presented for Lyapunov's second method.

3.2.4 Positive Polynomials and Hilbert's 17th Problem

The question of proving the positivity of polynomials and whether all positive polynomials can be represented as a sum-of-squares is a long standing research topic and also known as Hilbert's 17th problem, see for instance Henrion and Garulli [2005] and Marshall [2008]. In this section the main results useful to the rest of this thesis are briefly summarized and the connections to semidefinite programming are pointed out, see Boyd and Vandenberghe [2004] and Parrilo [2000].

The following standard notation is adopted: Let

$$\boldsymbol{m}_n^k = 1, \boldsymbol{x}_0, \boldsymbol{x}_1, \cdots, \boldsymbol{x}_{n-1}, \boldsymbol{x}_0\boldsymbol{x}_1, \cdots, \boldsymbol{x}_0\boldsymbol{x}_{n-1}, \boldsymbol{x}_1\boldsymbol{x}_2, \cdots, \boldsymbol{x}_0^k, \cdots, \boldsymbol{x}_{n-1}^k \tag{3.15}$$

be the vector of monomials used as the standard basis for real valued multivariate polynomials $p\colon \boldsymbol{x} \in \mathbb{R}^n \to p(\boldsymbol{x}) \in \mathbb{R}$ of degree k in n variables, which can then be written as $p(\boldsymbol{x}) = \boldsymbol{c}^{\mathrm{T}}.\boldsymbol{m}_n^k$, where $\boldsymbol{c} \in \mathbb{R}^{s_m(n,k)}$ denotes the coefficient vector. The length of the vector $\boldsymbol{m}_n^k$ is denoted $s_m(n,k)$ and is given by the binomial $\binom{n+k}{n}$. A monomial m of degree k in n variables is conveniently denoted as $m = \boldsymbol{x}^{\boldsymbol{\beta}} = \prod_{i=0}^{n-1} \boldsymbol{x}[i]^{\boldsymbol{\beta}[i]}$ with $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{\beta} \in \mathbb{N}_+^n$ and $\sum_i \boldsymbol{\beta}[i] = |\boldsymbol{\beta}| = k$.

Proving the positivity of a polynomial on a region $\Omega \subseteq \mathbb{R}^n$ is shown to be NP-hard for $k \geq 4$ even in the case $\Omega = \mathbb{R}^n$ (see Ahmadi [2012]) and is therefore not computationally tractable.

On the other hand, there are sum-of-squares (SoS) polynomials, so polynomials $p_{sos}(\boldsymbol{x})$ of degree $2k$ which can be written as

$$p_{sos}(\boldsymbol{x}) = \sum_j (\tilde{p}_j)^2 \tag{3.16}$$

where $\tilde{p}_j$ are polynomials with degree k or less. Obviously such polynomials are everywhere non-negative[1]. Moreover, if a polynomial is SoS, it can be equivalently written as

$$p_{sos}(\boldsymbol{x}) = \sum_j \left\| \tilde{\boldsymbol{c}}_j^{\mathrm{T}}.\boldsymbol{m}_n^k \right\|_2^2 = \sum_j {\boldsymbol{m}_n^k}^{\mathrm{T}}.\tilde{\boldsymbol{c}}_j.\tilde{\boldsymbol{c}}_j^{\mathrm{T}}.\boldsymbol{m}_n^k = {\boldsymbol{m}_n^k}^{\mathrm{T}}.\left(\sum_j M_j\right).\boldsymbol{m}_n^k = {\boldsymbol{m}_n^k}^{\mathrm{T}}.M.\boldsymbol{m}_n^k \tag{3.17}$$

with the $\tilde{\boldsymbol{c}}_j$ being the coefficient vector (with respect to the standard base given in (3.15)) of $\tilde{p}_j$ and $M \in \mathbb{S}_+^{s_m(n,k)}$ as it is the sum of the symmetric rank-one matrices M_j. This condition is sufficient and necessary, so a polynomial of degree $2k$ is SoS if and only if there exists a matrix $M \in \mathbb{S}_+^{s_m(n,k)}$ such that $p(\boldsymbol{x}) = {\boldsymbol{m}_n^k}^{\mathrm{T}}.M.\boldsymbol{m}_n^k$. The advantage of using SoS-polynomials for proving non-negativity stems from the fact that the constraint $M \in \mathbb{S}_+^{s_m(n,k)}$ or equivalently $M \succeq \mathbf{0}$ is an LMI-constraint and can therefore be efficiently solved in polynomial time using semidefinite programming.

The question that naturally arises is whether all non-negative polynomials can be represented as sum-of-squares. As it was stated above that proving non-negativity of polynomials of degree greater or equal to four is NP-hard and SoS-polynomials have polynomial time algorithms, the answer has to be no. Indeed there exist non-negative polynomials that are not SoS, like the Motzkin polynomials (Motzkin [1967]), but the set of SoS-polynomials is sufficiently *large* compared to the set of non-negative polynomials to be useful in practice.

3.2.5 Application to Linear and Polynomial Systems and Feedback Controller Design

The above presented methods provide sufficient conditions for a dynamical system to be stable. These constraints are mostly positivity and non-negativity constraints, which are difficult to handle for general nonlinear systems. However there are certain subclasses for which these problems are well-studied and computationally tractable, which are notably the cases of linear and the polynomial dynamics. It is also interesting to take the conditions and go one step further by using them to design feedback control laws guaranteed to stabilise the system. In this section we will briefly introduce common strategies and their links to recent developments in convex optimization. For an in-depth discussion the reader is referred to, for instance, Boyd et al. [1994]; Chesi [2010].

[1] The distinction between positive and non-negative polynomials is for numerical solutions obsolete.

Controlled Linear Dynamics Reconsider the unconstrained controlled linear dynamics

$$\dot{\boldsymbol{x}} = A.\boldsymbol{x} + B.\boldsymbol{u}. \tag{3.18}$$

By fixing a linear feedback control law, one obtains $\boldsymbol{u}(\boldsymbol{x}) = -K.\boldsymbol{x}$. From classical control theory we know that the system is asymptotically stable if the matrix $A' = A - B.K$ is Hurwitz, meaning that the real part of all eigenvalue is negative. This condition can also be formulated as an equivalent convex feasibility problem:

$$\text{exists } P \tag{3.19a}$$
$$\text{subject to } P \succ \mathbf{0} \tag{3.19b}$$
$$P.A' + A'^{\mathrm{T}}.P \preceq \mathbf{0} \tag{3.19c}$$

derived from Lyapunov's second method with $V(\boldsymbol{x}) = \|\boldsymbol{x}\|_P^2$. Constraints (3.3a) and (3.3b) are trivially fulfilled as constraint (3.19b) imposes $P \in \mathbb{S}^n_{++}$. The constraint (3.3c) can be written as

$$\begin{aligned}\forall \boldsymbol{x} \neq 0\colon\ \dot{V}(\boldsymbol{x}) &= \nabla_{\boldsymbol{x}} V(\boldsymbol{x}).\dot{\boldsymbol{x}} \leq 0\\ &= \boldsymbol{x}^{\mathrm{T}}.P.\dot{\boldsymbol{x}} + \dot{\boldsymbol{x}}^{\mathrm{T}}.P.\boldsymbol{x} \leq 0\\ &= \boldsymbol{x}^{\mathrm{T}}.P.A'.\boldsymbol{x} + \boldsymbol{x}^{\mathrm{T}}.A'^{\mathrm{T}}.P.\boldsymbol{x} \leq 0\\ &= \boldsymbol{x}^{\mathrm{T}}.(P.A' + A'^{\mathrm{T}}.P).\boldsymbol{x} \leq 0.\end{aligned}$$

The derived condition $\boldsymbol{x}^{\mathrm{T}}.(P.A' + A'^{\mathrm{T}}.P).\boldsymbol{x} \leq 0$ is true if and only if $(P.A' + A'^{\mathrm{T}}.P) \in \mathbb{S}^n_-$ which is equivalent to the constraint (3.19c).

To illustrate the advantage of this reformulation of the stability criterion into a convex optimization problem, consider the problem of designing a linear feedback control law. Using the Hurwitz criterion, one has to find the roots of the characteristic polynomial of $A - B.K$ and find the scalar values K_{ij} that satisfy the Hurwitz criterion, a nontrivial task. On the other hand, by using the above feasibility problem one gets

$$\text{exists } P,\, K \tag{3.20a}$$
$$\text{subject to } P \succ \mathbf{0} \tag{3.20b}$$
$$P.(A - B.K) + (A - B.K)^{\mathrm{T}}.P \preceq \mathbf{0} \tag{3.20c}$$

which is no longer a convex problem due to the multiplication of the decision variables P and K.

Multiplying (3.20c) left and right with P^{-1} and by introducing the new variables $Y = P^{-1}$ and $\tilde{K} = -K.Y$ one obtains

$$\text{exists } Y,\, \tilde{K} \tag{3.21a}$$
$$\text{subject to } Y \succ \mathbf{0} \tag{3.21b}$$
$$Y.A + A^{\mathrm{T}}.Y + B.\tilde{K} + \tilde{K}^{\mathrm{T}}.B^{\mathrm{T}} \preceq \mathbf{0} \tag{3.21c}$$

which is convex in Y and $\tilde{K}$. The solution to the problem (3.20) can be extracted from the solution to (3.21). This demonstrates the usefulness of LMI constraints in control applications.

Controlled Polynomials Dynamics The second class of dynamics, which is used extensively throughout the rest of this thesis, are polynomial dynamics due to their ability to approximate well general nonlinear dynamics and in particular rigid body dynamics. Moreover, if the considered dynamical system and Lyapunov function candidate are polynomial, the resulting constraint (3.3c) will also be polynomial. In this case, the in general computationally intractable constraints (3.3), can be relaxed to demanding

that the polynomial must be sum-of-squares, allowing the use of efficient optimization techniques while still providing good bounds for the original problem.

In this work we are mostly concerned with quadratic Lyapunov functions due to their importance for practical applications and third degree polynomial dynamics since they approximate well the trigonometric terms appearing in the rigid body dynamics equations. Moreover (3.3c) is the multiplication of the gradient of the Lyapunov function and the dynamics equation which corresponds to a fourth degree polynomial in this case.

Consider the nonlinear system dynamics

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}) \qquad \text{autonomous case} \tag{3.22a}$$

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}) + g(\boldsymbol{x}).\boldsymbol{u} \quad \text{controlled case} \tag{3.22b}$$

where each element $f_i(\boldsymbol{x})$ of the vector valued function $f(\boldsymbol{x})\colon \mathbb{R}^n \to \mathbb{R}^n$ is a multivariate polynomial of maximal degree k in n variables, similarly each element $g_{i,j}(\boldsymbol{x})$ of the input dynamics $g\colon \mathbb{R}^n \to \mathbb{R}n \times m$ is a multivariate polynomial of maximal degree l in n variables. The Lyapunov function candidate is given as a quadratic polynomial

$$V(\boldsymbol{x}) = \boldsymbol{x}^{\mathrm{T}}.P.\boldsymbol{x}. \tag{3.23}$$

In this case, the constraints 3.3 proving stability for the autonomous dynamical system can be relaxed to the feasibility problem

$$\text{exists } P \tag{3.24a}$$

$$\text{subject to } P \succ \mathbf{0} \tag{3.24b}$$

$$\dot{V}(\boldsymbol{x}) = \boldsymbol{x}^{\mathrm{T}}.P.f(\boldsymbol{x}) + f(\boldsymbol{x})^{\mathrm{T}}.P.\boldsymbol{x} \ \text{ is SoS} \tag{3.24c}$$

or to make the connection to SDP more obvious

$$\text{exists } P \tag{3.25a}$$

$$\text{subject to } P \succ \mathbf{0} \tag{3.25b}$$

$$-M \succeq \mathbf{0} \tag{3.25c}$$

$$\dot{V}(\boldsymbol{x}) = \boldsymbol{x}^{\mathrm{T}}.P.f(\boldsymbol{x}) + f(\boldsymbol{x})^{\mathrm{T}}.P.\boldsymbol{x} = \boldsymbol{m}_n^{2\,\mathrm{T}}.M.\boldsymbol{m}_n^2. \tag{3.25d}$$

Where (3.25b) is equivalent to the constraints (3.3a) and (3.3b). The constraint (3.25c) guarantees that the polynomial $\boldsymbol{m}_n^{2\,\mathrm{T}}.M.\boldsymbol{m}_n^2$ is SoS and therefore $-\boldsymbol{m}_n^{2\,\mathrm{T}}.M.\boldsymbol{m}_n^2$ is non-positive on $\mathbb{R}^n$, while the equality constraints (3.25d) guarantee that the system dynamics are taken into account correctly.

In the controlled case, the control input has to be defined. Consider using a polynomial feedback control law of maximal degree $3 - l$ denoted $\boldsymbol{u} = -k(\boldsymbol{x})\colon \mathbb{R}^n \to \mathbb{R}^m$, then the input dynamics become $-g(\boldsymbol{x}).k(\boldsymbol{x})\colon \mathbb{R}^n \to \mathbb{R}^n$ with each element being a polynomial of degree three (the same as the system dynamics). Then the feasibility problem for proving stability becomes

$$\text{exists } P,\ k \tag{3.26a}$$

$$\text{subject to } P \succ \mathbf{0} \tag{3.26b}$$

$$-M \succeq \mathbf{0} \tag{3.26c}$$

$$\dot{V}(\boldsymbol{x}) = \boldsymbol{x}^{\mathrm{T}}.P.(f(\boldsymbol{x}) - g(\boldsymbol{x}).k(\boldsymbol{x})) + (f(\boldsymbol{x}) - g(\boldsymbol{x}).k(\boldsymbol{x}))^{\mathrm{T}}.P.\boldsymbol{x} = \boldsymbol{m}_n^{2\,\mathrm{T}}.M.\boldsymbol{m}_n^2. \tag{3.26d}$$

Note that now the feasibility problem is no longer convex due to the multiplication of the decision variables P and k in (3.26d). One could seek to use a similar change of variables as done in the linear case, however this approach makes it impossible to add additional constraints on the control law $k(\boldsymbol{x})$ or to consider local stability. To solve this problem, two-step algorithms are usually applied, first fixing the Lyapunov function candidate and searching for the control law $k(\boldsymbol{x})$. In the second step the control law from the last-step is fixed and one searches for suitable Lyapunov function candidate by searching for P, as detailed in section 3.4.

Comparing Lyaponov's Second Method and Contraction Analysis Contraction analysis is an appealing way for proving stability as one is not required to have explicit knowledge about the equilibrium point or path. Also, at first glance, it seems to be computationally lighter as the constraint ensuring stability (equation (3.13)) reads

$$F(\boldsymbol{x})^{\mathrm{T}}.M(\boldsymbol{x}) + M(\boldsymbol{x}).F(\boldsymbol{x}) \preceq \mathbf{0}$$

with $F(\boldsymbol{x})$ being the Jacobian of the system dynamics $f(\boldsymbol{x})$ whereas one obtains

$$\boldsymbol{x}^{\mathrm{T}}.P.f(\boldsymbol{x}) + f(\boldsymbol{x})^{\mathrm{T}}.P.\boldsymbol{x} \leq 0$$

for Lyapunov's stability criterion. If one uses the flat metric P, so $\forall \boldsymbol{x}\colon M(\boldsymbol{x}) = P$, the highest degree appearing in the contraction analysis criterion is the degree of f minus one, whereas it is degree of f plus one in Lyapunov's criterion. As the computational cost is directly related to the length of the vector of monomials $s_m(n,k) = \binom{n+k}{n}$ with k being the highest degree of any monomial, this can make a crucial difference. This computational advantage however only holds for the case of flat metrics. The next toy example shows that contraction analysis with flat metrics is less expressive, in the sense that the region for which stability can be proven is never larger and sometimes smaller, than Lyapunov's criterion for quadratic candidate functions.

Consider the unidimensional autonomous system with third order polynomial dynamics

$$\dot{x} = -x + x^3 \tag{3.27}$$

and the Lyapunov function candidate

$$V(x) = x^2 \tag{3.28}$$

and the corresponding metric $M(x) = 1$.
The Lyapunov stability criterion then becomes

$$\dot{V}(x) = 2x(-x + x^3) = 2(x^4 - x^2) \leq 0$$

indicating that $f(\boldsymbol{x})$ is locally stable for $x \in [-1,\, 1]$, which coincides with the true region of attraction (RoA). Using contraction analysis on the other hand, one obtains

$$F(x) = \frac{\mathrm{d}}{\mathrm{d}\,x} f(x) = -1 + 3x^2$$

and the criterion becomes

$$2(-1 + 3x^2) < 0$$

indicating that $f(x)$ is locally contracting for $x \in \left[-\sqrt{1/3},\, \sqrt{1/3}\right]$ which is significantly smaller than the true region of attraction, see Figure 3.3.

Theorem 3.1. *The largest region of attraction derived from Lyapunov's stability criterion*

$$\Omega_L = \left\{\boldsymbol{x} | (\boldsymbol{x} - \boldsymbol{x}^*)^T.P.(\boldsymbol{x} - \boldsymbol{x}^*) \leq \alpha,\, (\boldsymbol{x} - \boldsymbol{x}^*)^T.P.f(\boldsymbol{x}) + f(\boldsymbol{x})^T.P.(\boldsymbol{x} - \boldsymbol{x}^*) \leq 0\right\}$$

containing the equilibrium point $\boldsymbol{x}^*$ is always at least containing the region of contraction (RoC) derived from contraction analysis Ω_C using the flat metric $M(\boldsymbol{x}) = P$ containing $\boldsymbol{x}^*$, so $\Omega_C \subseteq \Omega_L$

Proof. Contraction analysis proves that the distance between any two neighbouring trajectories within the region of contraction monotonically decreases over time with respect to the metric $M(\boldsymbol{x})$. However if the metric $M(\boldsymbol{x})$ is flat, the requirement that the trajectories are neighbouring can be dropped as the

geodesic between two points is the straight line for flat metrics. Choosing one of the two trajectories to be the equilibrium, one directly obtains that all trajectories monotonically converge towards $\boldsymbol{x}^*$ with respect to $\|\boldsymbol{x}-\boldsymbol{x}^*\|_{M(\boldsymbol{x})=P}$ which is equivalent to Lyapunov condition $\dot{V}(\boldsymbol{x}) = \frac{\mathrm{d}}{\mathrm{d}\,t}(\boldsymbol{x}-\boldsymbol{x}^*)^{\mathrm{T}}.P.(\boldsymbol{x}-\boldsymbol{x}^*) \leq 0$.

However the converse it not necessarily true. From

$$\forall \boldsymbol{x} \in \Omega_L\colon\ \dot{V}(\boldsymbol{x}) = \frac{\mathrm{d}}{\mathrm{d}\,t}(\boldsymbol{x}-\boldsymbol{x}^*)^{\mathrm{T}}.P.(\boldsymbol{x}-\boldsymbol{x}^*) \leq 0$$

one cannot deduce contraction criterion

$$\forall \boldsymbol{x}_1,\, \boldsymbol{x}_2 \in \Omega_L, \delta\boldsymbol{x} = \boldsymbol{x}_1 - \boldsymbol{x}_2\colon\ \delta\boldsymbol{x}^{\mathrm{T}}.P.\delta\dot{\boldsymbol{x}} + \delta\dot{\boldsymbol{x}}^{\mathrm{T}}.P.\delta\boldsymbol{x} \leq 0$$

without further knowledge of $f(\boldsymbol{x})$ and therefore $\Omega_C \subseteq \Omega_L$. □

Now one could ask about non-flat metrics and the resulting difference between RoA and RoC. It is difficult to draw general conclusions for the case of non-flat metrics, especially in higher dimension, therefore let us simply continue with the running example and consider some state-dependent metric $M(x)$. In order to ensure that $M(x)$ is indeed a metric, it has to be integrable and everywhere positive. The criterion becomes

$$2M(x)(-1+3x^2) < 0$$

and one can see that in the unidimensional case, the complexity of $M(x)$ has no influence on the RoC, which will always be $x \in \left[-\sqrt{1/3},\ \sqrt{1/3}\right]$ independently of the metric considered.

As discussed above, the non-positivity constraint in Lyapunov's criterion can be relaxed to a SoS-condition, which provides reasonably good bounds while being computationally efficient. The constraint of $M(\boldsymbol{x}) \succ \boldsymbol{0}$ is in this general form also intractable, but convex relaxation can be found, see for instance Gatermann and Parrilo [2004]. These relaxations introduce auxiliary variables which, together with the increased maximal degree due to the metric itself, absorb the computational advantage. Due to these findings the presented method is based on Lyapunov theory.

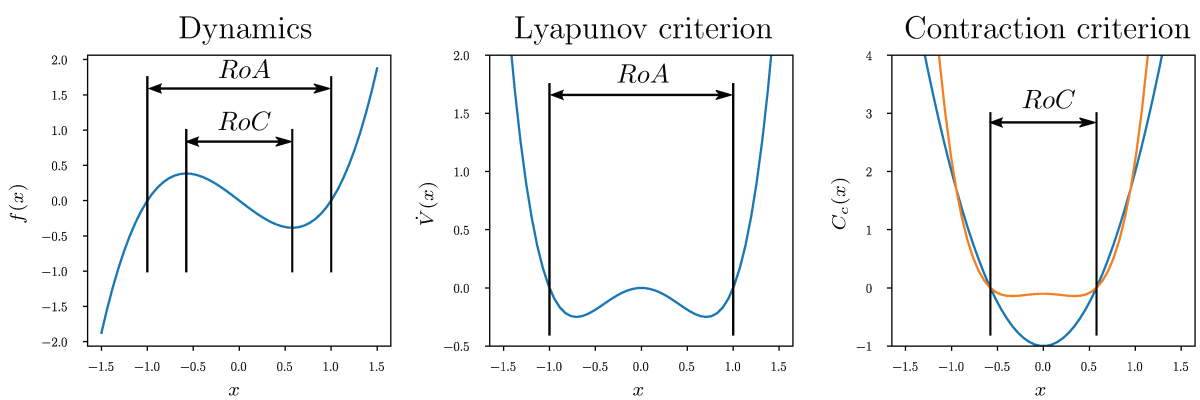


Figure 3.3 – Comparison of RoC and RoA for a unidimensional dynamical system. The left image shows the systems dynamics and the obtained RoA (Lyapunov criterion) and RoC (contraction analysis). The image in the middle shows the derivative of the Lyapunov function $\dot{V}(x)$ and the RoA, so all x for which $\dot{V}(x) \leq 0$ around $x^* = 0$. The right image shows the value of the contraction metric $C_c(x) = M(x).F(x) + F(x)^{\mathrm{T}}.M$ for the flat metric $M(x) = 1$ (blue line) and the metric $M(x) = 0.1 + x^2$. In both cases the RoC defined as $C_c(x) \leq 0$ is identical.

3.3 Problem Statement

We propose a new method to find an inner approximation of the true region of stabilizability (RoS) for polynomial control affine systems by scaling a given quadratic Lyapunov function candidate

$$V(\boldsymbol{x}) = \boldsymbol{x}^{\mathrm{T}}.P.\boldsymbol{x} = \|\boldsymbol{x}\|_P^2 \ \text{, with } P \in \mathbb{S}_{++}^n. \tag{3.29}$$

More precisely, we consider systems of the form

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}) + B.\boldsymbol{u},\ \boldsymbol{u} \in \mathcal{U} \tag{3.30}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ denotes a point in the state-space, $f(.)\colon \mathbb{R}^n \to \mathbb{R}^n$ represents the polynomial system dynamics, $B \in \mathbb{R}^{n\times m}$ is a constant input matrix defining the linear input dynamics, $\boldsymbol{u} \in \mathbb{R}^m$ denotes the control input vector and $\mathcal{U}$ is the set of admissible control inputs. We suppose that B has full column rank and that each control input $\boldsymbol{u}[i]$, is bounded, and that the constraints are independent of the other control inputs, $\boldsymbol{u}^-[i] \leq \boldsymbol{u}[i] \leq \boldsymbol{u}^+[i]$ so that

$$\mathcal{U} = \left\{\boldsymbol{u} | \boldsymbol{u}^- \leq \boldsymbol{u} \leq \boldsymbol{u}^+\right\}. \tag{3.31}$$

This type of input constraints, also called box-constraint is typical for torque controlled articulated robots, which are our primary target as far as applications are concerned, but also occurs frequently in other applications. The problem treated in this section is to find an as large as possible sublevel-set of the quadratic Lyapunov function candidate $V(\boldsymbol{x})$, denoted $\Omega = \{\boldsymbol{x}|V(\boldsymbol{x}) \leq \alpha\} \subset \mathbb{R}^n$ for which exists an admissible control input that makes this set exponentially stable, or more formally

$$\begin{aligned} \text{prove} \quad & \forall \boldsymbol{x} \in \Omega :\ \exists \boldsymbol{u} \\ \text{subject to} \quad & \boldsymbol{u}^- \leq \boldsymbol{u} \leq \boldsymbol{u}^+ \\ & \langle \nabla_{\boldsymbol{x}} V, f(\boldsymbol{x}) + B.\boldsymbol{u}\rangle \leq -\gamma.V(\boldsymbol{x}) \end{aligned}$$

where $\langle .,. \rangle$ denotes the usual scalar product, $\nabla_{\boldsymbol{x}}$ denotes the gradient with respect to $\boldsymbol{x}$ and $\gamma \geq 0$ is called the convergence rate. The limit case $\gamma = 0$ is equivalent to the set Ω, and each subset $\Omega' = \{\boldsymbol{x}|V(\boldsymbol{x}) \leq \alpha' < \alpha\}$, being invariant. To avoid confusion with other approaches, note that our approach does not modify the given Lyapunov function candidate $V(.)$ but seeks to enlarge the subset Ω by enlarging α. This approach is reasonable for dynamical systems for which a *good* Lyapunov function candidate can be found by other means. The method to compute these *good* candidates used within this work is presented in section 3.9.

To avoid ambiguity with other definitions and to recall some properties from section 3.2.2. We call a function $V(\boldsymbol{x})$ a Lyapunov function candidate if it is differentiable, radially unbounded and everywhere strictly positive, except at the origin where it evaluates to zero. A function $V(\boldsymbol{x})$ is called a Lyapunov function for the dynamical system $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$ if it is a Lyapunov function candidate and its derivative along any trajectory of the system is everywhere non-positive except at the origin where it is zero. Moreover, a Lyapunov function proves exponential stability if its derivative along any trajectory is everywhere smaller than its current value multiplied with a negative factor except at the origin where it is zero.

Due to their outstanding practical importance stemming from the ease of inclusion and intersection testing (see section 2.5.2) as well as their inherent suitability for second order systems, we restrain ourselves to quadratic Lyapunov functions of the form (3.29). The conditions for $V(\boldsymbol{x})$ being a Lyapunov function candidate are met if $P \in \mathbb{S}_{++}^n$. Within this section, it is, without loss of generality, supposed that the origin is an equilibrium for the dynamical system. Moreover, to ease notation, it is assumed that the Lyapunov function candidate is time-independent, the necessary adaptations to time-dependent Lyapunov functions are straight-forward and presented in detail in section 3.7.

3.4 Related Work

The problem of proving stability for dynamical systems is a long standing problem that was first encountered by physicists and only much later found its place in engineering sciences, namely control theory. Naturally, in a control theoretic environment one is not satisfied by drawing conclusions about the natural stability of autonomous systems, but it is much more interesting to derive criteria for the stability or stabilizability of controlled dynamical systems.

Many of the approaches developed to prove stability are based on contraction analysis or Lyapunov's stability criterion. As the proposed approach is based on Lyapunov theory, we will mainly present existing approaches that also rely on Lyapunov theory with other approaches discussed briefly afterwards.

3.4.1 Approaches Involving Lyapunov Theory on SoS-Techniques

Recall the definition of exponential stability with rate γ in the Lyapunov sense for a bounded region: given a Lyapunov function candidate $V(\boldsymbol{x})$ and a region of the state space Ω defined as sublevel-set of the Lyapunov function, proving exponential convergence for the dynamical system $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$ amounts to finding a certificate of non-positivity for $\dot{V}(\boldsymbol{x})$ plus a convergence term valid within Ω. So one has to prove that

$$\forall \boldsymbol{x} \in \Omega \setminus \{\boldsymbol{0}\} : \ \dot{V}(\boldsymbol{x}) = \langle \nabla_{\boldsymbol{x}} V(\boldsymbol{x}), f(\boldsymbol{x}) \rangle + \gamma . V(\boldsymbol{x}) \leq 0 \tag{3.32}$$

or equivalently

$$\max_{\boldsymbol{x} \in \Omega \setminus \{\boldsymbol{0}\}} \langle \nabla_{\boldsymbol{x}} V(\boldsymbol{x}), f(\boldsymbol{x}) \rangle + \gamma . V(\boldsymbol{x}) \leq 0 \ . \tag{3.33}$$

There exists no generic way to solve this optimization problem for general nonlinear systems and Lyapunov functions and therefore one has to either rely on *hand-made* solutions for specific cases or restrict the class of functions considered. In recent years enormous progress has been made by restraining both, the dynamics and the Lyapunov function candidate to be polynomial in $\boldsymbol{x}$ as showcased in Jarvis-Wloszek et al. [2003], Majumdar et al. [2013b], Ahmadi and Majumdar [2014] or Aylward et al. [2008] and Singh et al. [2017] for contraction analysis. These advancements have been made possible by the appearence and progress on solvers for SDPs like Sturm [1999], ApS [2017] or Andersen et al. [2013].

The approach presented in Majumdar et al. [2013b] is the closest to the method proposed in this chapter and is therefore detailed in the following. Note that Majumdar et al. [2013b] explicitly aims at proving convergence towards a trajectory, so time-dependent Lyapunov functions. To ease notations and concepts, this method is here presented in its time-independent version.

By restricting the Lyapunov function and the dynamical systems to polynomials in $\boldsymbol{x}$, the time-derivative of the Lyapunov function $\dot{V}(\boldsymbol{x})$ also becomes a polynomial of degree $\deg(f(\boldsymbol{x}))+\deg(V(\boldsymbol{x}))-1$ in $\boldsymbol{x}$. Therefore the question $\dot{V}(\boldsymbol{x}) \leq -\gamma V(\boldsymbol{x})$ can be relaxed to the SDP feasibility problem stated in (3.25). This problem however proves global exponential convergence of the autonomous system with respect to the obtained Lyapunov function. To be useful for solving realistic problems, the approach needs to take into account the control inputs and be restricted to prove local stability. So the considered dynamical system becomes $\dot{\boldsymbol{x}} = f(\boldsymbol{x}) + g(\boldsymbol{x}).\boldsymbol{u}$ where each element $f(\boldsymbol{x})[i]$ and $g(\boldsymbol{x})[i,j]$ are polynomials in $\boldsymbol{x}$ and, without loss of generality, $f(\boldsymbol{0}) = \boldsymbol{0}$. To achieve this, a polynomial control law $\boldsymbol{u} = -K(\boldsymbol{x}) \colon \mathbb{R}^n \to \mathbb{R}^n$

and multiplier terms are introduced and the local stability criterion on $\Omega = \{\boldsymbol{x} | V(\boldsymbol{x}) \leq \alpha\}$ becomes

$$
\begin{aligned}
\text{exists} \quad & V(\boldsymbol{x}),\, K(\boldsymbol{x}),\, L(\boldsymbol{x}), M_i(\boldsymbol{x}) && \text{(3.34a)}\\
\text{subject to} \quad & V(\boldsymbol{x}) - \epsilon \boldsymbol{x}^{\mathrm{T}}.\boldsymbol{x} \ \text{ is SoS} && \text{(3.34b)}\\
& L(\boldsymbol{x}) \ \text{ is SoS} && \text{(3.34c)}\\
& \forall i,\ M_i(\boldsymbol{x}) \ \text{ is SoS} && \text{(3.34d)}\\
& -\langle \nabla_{\boldsymbol{x}} V(\boldsymbol{x}),\, f(\boldsymbol{x}) - g(\boldsymbol{x}).K(\boldsymbol{x})\rangle - \gamma V(\boldsymbol{x}) + L(\boldsymbol{x})(V(\boldsymbol{x}) - \alpha) \text{ is SoS} && \text{(3.34e)}\\
& \forall i,\ \boldsymbol{u}^{+}[i] - K(\boldsymbol{x})[i] + M_i(\boldsymbol{x})(V(\boldsymbol{x}) - \alpha) \text{ is SoS} && \text{(3.34f)}\\
& \forall i,\ K(\boldsymbol{x})[i] - \boldsymbol{u}^{-}[i] + M_i(\boldsymbol{x})(V(\boldsymbol{x}) - \alpha) \text{ is SoS} && \text{(3.34g)}\\
& V(\mathbf{1}) = 1. && \text{(3.34h)}
\end{aligned}
$$

The constraints (3.34b), (3.34c) and (3.34d) ensure that $V(\boldsymbol{x})$ is indeed a Lyapunov function candidate (by setting $\epsilon > 0$) and that the multiplier terms $L(\boldsymbol{x})$ and $M_i(\boldsymbol{x})$ are non-negative. Constraint (3.34e) ensures the exponential convergence with minimal rate γ of the dynamical system for the control law $K(\boldsymbol{x})$ with respect to $V(\boldsymbol{x})$ in Ω. This is ensured as $L(\boldsymbol{x})$ is everywhere non-negative and $V(\boldsymbol{x}) - \alpha$ is negative within Ω and non-negative in the complement of Ω, $\bar{\Omega}$. This means that outside of Ω, the positivity of $L(\boldsymbol{x})(V(\boldsymbol{x}) - \alpha)$ can outweigh the possible negativeness of the other terms, resulting in an overall SoS expression. However inside of Ω this approach increases the conservativeness, especially farther away from the boundary of Ω. The constraints (3.34f) and (3.34g) guarantee that $\boldsymbol{u}^{-} \leq \boldsymbol{u} = K(\boldsymbol{x}) \leq \boldsymbol{u}^{+}$ in the same way as (3.34e). The last constraint (3.34h) is a normalization constraint, necessary as one could always increase α dividing all coefficients defining the polynomial $V(\boldsymbol{x})$ by some scalar, resulting in an unbounded problem.

As one can see, the input dynamics and the polynomial control law are multiplied in (3.34e), possibly increasing the maximal occurring degree. In order to be able to ensure global non-negativeness, or rather SoS-representability of the whole expression, the degree of the multipliers has to be sufficiently large. So, for instance in the constraint (3.34e), $\deg(L(\boldsymbol{x})(V(\boldsymbol{x}) - \alpha))$ has to be at least $\deg(-\langle \nabla_{\boldsymbol{x}} V(\boldsymbol{x}),\, f(\boldsymbol{x}) - g(\boldsymbol{x}).K(\boldsymbol{x})\rangle - \gamma V(\boldsymbol{x})) + 2$ increasing the overall complexity of the approach. Moreover, in the constraints (3.34a), (3.34b) and (3.34c), the decision variables (the coefficients of) $V(\boldsymbol{x})$, $K(\boldsymbol{x})$, $L(\boldsymbol{x})$ and $M_i(\boldsymbol{x})$ are multiplied causing the problem to be non-convex.

Therefore in Majumdar et al. [2013b] a three step algorithm is proposed, which already occurs in a slightly different version not taking into account input constraints in Jarvis-Wloszek et al. [2003], that seeks to find an as large as possible inner approximation of the region of attraction defined as a sublevel-set of a Lyapunov function and provides the associated polynomial feedback control law.

Algorithm 1 RoA computation as in Majumdar et al. [2013b]

1: Parameter $\gamma > 0$, $\epsilon > 0$
2: Initialize $V(\boldsymbol{x})$, α using LQR
3: converged$=false$
4: **while** not converged **do**
5: **Step 1 :** Solve feasibility problem by searching for $K(\boldsymbol{x})$, $L(\boldsymbol{x})$ and $M_i(\boldsymbol{x})$ while fixing $V(\boldsymbol{x})$ and α
6: **Step 2 :** Maximize α by searching for $K(\boldsymbol{x})$ and α while fixing $V(\boldsymbol{x})$, $L(\boldsymbol{x})$ and $M_i(\boldsymbol{x})$
7: **Step 3 :** Maximize α by searching for $V(\boldsymbol{x})$ and α while fixing $K(\boldsymbol{x})$, $L(\boldsymbol{x})$ and $M_i(\boldsymbol{x})$
8: **Step 4 :** converged $=$ $\mathsf{isConverged}(V(\boldsymbol{x}), \alpha)$
9: **end while**

In **Step 1** feasible multipliers and control laws are found for the current region size α and Lyapunov function $V(\boldsymbol{x})$. In **Step 2** the current region is enlarged by maximizing α. As the current solution is feasible, the resulting α^* has to be at least as large as the α from the last iteration. However the increase

possible depends largely on the current multiplier terms which are fixed in this step and there is no way to generate them during **STEP 1** in such a way that "large" increases of α during **STEP 2** can be ensured. Finally in **STEP 3** the Lyapunov function itself is updated, again while seeking to maximize α. In this step the quality of the normalization constraints to represent the volume of the region is crucial. Overall this algorithm is a method to solve the initial non-convex problem (3.34) by breaking it into convex subproblems and iteratively improve the found solution. Therefore the final result depends on the quality of the initial guess of $V(\boldsymbol{x})$ and α, and suitable candidates are often found relying on the linearised system and LQR-techniques or they are user-provided input.

In the following some additional remarks on Algorithm 1 are listed.

Remark 3.1. The conservativeness induced by the constraints (3.34f) and (3.34g) regarding the control input can be reduced by explicitly taken into account the saturation, which is presented in Tedrake et al. [2010b] and also applied in Majumdar et al. [2013b]. This approach increases the overall complexity and also involves multiplier terms.

Remark 3.2. By dropping the constraints (3.34c) and (3.34d) from the problem (3.34) and setting γ to zero, Ω is an invariant instead of an exponentially converging set and the input constraints are only ensured on the boundary of Ω. This is interesting as it significantly decreases the computational complexity and possibly even the conservativeness at the cost of obtaining a weaker certificate.

Remark 3.3. **Step 3** of the algorithm depends on the normalization constraint (3.34h) since the problem would be unbounded otherwise. As the goal is to maximize the RoA one needs to specify according to which measure one wants to maximize. A natural choice would be the volume. The volume of a region of the state space defined as a sublevel-set of the SoS expression $V(\boldsymbol{x})$ is however not convex with respect to the coefficients of the polynomial. Therefore simpler normalization constraint needs to be found. Here the constraint $V(\mathbf{1}) = 1$ is chosen, which is a somewhat bad measure for the volume, even for quadratic Lyapunov functions and which tends to bias the optimization as we will see in section 3.10. In Jarvis-Wloszek et al. [2003], this problem is resolved by demanding that the sublevel-set Ω_1 defined by $V(\boldsymbol{x}) \leq 1$ comprises a predefined region, providing a lower bound for the volume of the sub-level set.

As stated before, we restrict ourselves for the moment being to polynomial control affine systems of the form $\dot{\boldsymbol{x}} = f(\boldsymbol{x}) + B.\boldsymbol{u}$ and quadratic Lyapunov functions $V(\boldsymbol{x}) = \boldsymbol{x}^{\mathrm{T}}.P.\boldsymbol{x}$ with $P \in \mathbb{S}_{++}$, which is simply a special case for the approaches described above.

3.4.2 Approaches Involving Contraction Analysis and LMIs

Applying contraction theory for answering the question of local stability on Ω of controlled polynomial systems, as done in Manchester and Slotine [2017] and Singh et al. [2017], the constraint ensuring exponential convergence becomes

$$(A_{\boldsymbol{x}} + B_{\boldsymbol{x}}.K_{\boldsymbol{x}})^{\mathrm{T}}.M_{\boldsymbol{x}} + M_{\boldsymbol{x}}.(A_{\boldsymbol{x}} + B_{\boldsymbol{x}}.K_{\boldsymbol{x}}) \leq -2\gamma M_{\boldsymbol{x}} \tag{3.35}$$

with $A_{\boldsymbol{x}}$ and $B_{\boldsymbol{x}}$ being the Jacobians of the system and input dynamics, $K_{\boldsymbol{x}}$ denoting the differential control law and M_x representing the state-dependent metric. Each element $M_{\boldsymbol{x}}[i,j] = M_{\boldsymbol{x}}[j,i]$ is a multivariate polynomial in $\boldsymbol{x}$ of predefined degree. Additionally one has to ensure that $M_{\boldsymbol{x}}$ is indeed a metric on Ω. This can be done via sum-of-squares constraints and LMIs as shown in Aylward et al. [2008]. Once $K_{\boldsymbol{x}}$ and $M_{\boldsymbol{x}}$ found, the actual control law is obtained by integrating the differential control law along the trajectory for some initial value (for the system state and the control law).

As shown in Singh et al. [2017], box-constraints, so that the resulting control input has to suffices $\boldsymbol{u}^- \leq \boldsymbol{u} \leq \boldsymbol{u}^+$, can be imposed onto the the control law, however making the problem significantly more complex and inducing conservativeness.

3.5 State-Space Partitioning Based On Optimal Control Input

In the last section it was briefly discussed how SoS-techniques tackle the problem of proving local stability of dynamical systems with polynomial system and input dynamics. In this section the arising optimization problem will be viewed from a different point of view and a new approach based on ideas from optimal control and Lyapunov theory is presented. The resulting optimization problem can be resolved by polynomial programming as shown in the following sections.

3.5.1 Stabilizability As Min-Max problem

The question whether the controlled dynamical system $\dot{\boldsymbol{x}} = f(\boldsymbol{x}) + B.\boldsymbol{u}$ is exponentially stabilizable for $\boldsymbol{u} \in \mathcal{U}$ with respect to the Lyapunov function $V(\boldsymbol{x})$ within a region defined as a sublevel-set $V(\boldsymbol{x}) \leq \alpha$ comes down to finding a certificate for

$$\forall \boldsymbol{x} \in \Omega \setminus \{\boldsymbol{0}\}\,,\, \exists \boldsymbol{u} \in \mathcal{U}\colon\ \nabla_{\boldsymbol{x}} V(\boldsymbol{x}).(f(\boldsymbol{x}) + B.\boldsymbol{u}) \leq -\gamma V(\boldsymbol{x})$$

or equivalently

$$\max_{\boldsymbol{x} \in \Omega \setminus \{\boldsymbol{0}\}} \ \min_{\boldsymbol{u} \in \mathcal{U}} \ \nabla_{\boldsymbol{x}} V(\boldsymbol{x}).(f(\boldsymbol{x}) + B.\boldsymbol{u}) + \gamma V(\boldsymbol{x}) \leq 0. \tag{3.36}$$

So the above optimization problem asks, if for all points in the region there exists an admissible control input that ensures exponential convergence. Due to the interweaving of optimizations this is a very difficult optimization problem which cannot be efficiently solved even if $V(\boldsymbol{x})$ and $f(\boldsymbol{x})$ are polynomial. This is due to the structural properties of the optimization problem on one and to the lack of good available software packages on the other, see for instance Fang and Wu [1996].

The approach taken by the works cited in the last section to tackle this problem is to introduce a polynomial (differential) control law $\boldsymbol{u} = K(\boldsymbol{x})$, transforming the controlled system into a closed loop autonomous system. This simplifies the optimization problem as the inner minimization is dropped, but also increases the number of variables and adds conservativeness. Due to the predefined structure of the control law, it cannot result in the optimal control input with respect to convergence for each point in the general case.

3.5.2 State-Space Partitioning

Instead of introducing auxiliary variables representing a polynomial control law, our approach relies on input optimal state space partitioning.

Lemma 3.2. Optimal Input Partition
For polynomial control affine systems and quadratic Lyapunov candidate functions, the state space can be partitioned into 2^m subsets $\mathcal{H}_{i\in[0,2^m-1]}$ and an associated optimal control input with respect to convergence $\boldsymbol{u}_i^*$ can be defined. Moreover this optimal control input takes on exclusively values from $\boldsymbol{u}^+$ and $\boldsymbol{u}^-$.

In order to prove this claim, reconsider the quadratic Lyapunov function candidate $V(\boldsymbol{x}) = \boldsymbol{x}^{\mathrm{T}}.P.\boldsymbol{x}$ and the polynomial control affine system $\dot{\boldsymbol{x}} = f(\boldsymbol{x}) + B.\boldsymbol{u}$. Then the time derivative of V is given as

$$\begin{aligned}\dot{V}(\boldsymbol{x}) &= \langle \nabla_{\boldsymbol{x}} V(\boldsymbol{x}), \dot{\boldsymbol{x}} \rangle \\ &= 2\boldsymbol{x}^{\mathrm{T}}.P.(f(\boldsymbol{x}) + B.\boldsymbol{u})\ .\end{aligned} \tag{3.37}$$

This indicates that the derivative can be separated into two parts: An uncontrollable part resulting from the system dynamics $2\boldsymbol{x}^{\mathrm{T}}.P.f(\boldsymbol{x})$ denoted $\dot{V}_f$ and an input dependent part $2\boldsymbol{x}^{\mathrm{T}}.P.B.\boldsymbol{u}$ denoted $\dot{V}_u$. In order to obtain the desired partitioning of the state-space, $\dot{V}_u$ is explicitly written as sum

$$\dot{V}_u = \sum_j \boldsymbol{x}^{\mathrm{T}}.P.B[:,j].\boldsymbol{u}[j]. \tag{3.38}$$

So the input dependent part of the derivative can be written as the sum of each control input element $\boldsymbol{u}[j]$ multiplied by the scalar $\boldsymbol{x}^{\mathrm{T}}.P.B[:,j]$. By denoting $\boldsymbol{n}_j = P.B[:,j]$ one obtains $\boldsymbol{x}^{\mathrm{T}}.\boldsymbol{n}_j$, which is simply the minimal directed distance scaled by $\|\boldsymbol{n}_j\|_2$ between the point considered and a separating hyperplane passing through the origin with normal vector $\boldsymbol{n}_j$ denoted $\mathcal{P}_j$. The problem (3.36) can be rewritten as

$$\max_{\boldsymbol{x}\in\Omega\setminus\boldsymbol{0}} \nabla_{\boldsymbol{x}}V(\boldsymbol{x}).f(\boldsymbol{x}) + \gamma V(\boldsymbol{x}) + \min_{\boldsymbol{u}\in\mathcal{U}} \left(\sum_{j=0}^{m-1} (\boldsymbol{x}^{\mathrm{T}}.\boldsymbol{n}_j)\boldsymbol{u}[j]\right) \leq 0. \tag{3.39}$$

The box-constraint defining $\mathcal{U}$ (see (3.31)), ensuring that $\boldsymbol{u}[i]$ is independent of $\boldsymbol{u}[j]$ if $i \neq j$, implies the following equivalence:

$$\min_{\boldsymbol{u}\in\mathcal{U}} \left(\sum_{j=0}^{m-1} (\boldsymbol{x}^{\mathrm{T}}.\boldsymbol{n}_j)\boldsymbol{u}[j]\right) = \sum_{j=0}^{m-1} \min_{\boldsymbol{u}^-[j]\leq\boldsymbol{u}[j]\leq\boldsymbol{u}^+[j]} \left((\boldsymbol{x}^{\mathrm{T}}.\boldsymbol{n}_j)\boldsymbol{u}[j]\right)$$

as the minimum over a sum of independent terms is the sum of the minima of each term. Finally one can rewrite (3.36) in the simpler form

$$\max_{\boldsymbol{x}\in\Omega\setminus\boldsymbol{0}} \nabla_{\boldsymbol{x}}V(\boldsymbol{x}).f(\boldsymbol{x}) + \gamma V(\boldsymbol{x}) + \sum_{j=0}^{m-1} \min_{\boldsymbol{u}^-[j]\leq\boldsymbol{u}[j]\leq\boldsymbol{u}^+[j]} \left((\boldsymbol{x}^{\mathrm{T}}.\boldsymbol{n}_j)\boldsymbol{u}[j]\right) \leq 0. \tag{3.40}$$

Therefore demanding the system to converge as fast as possible for a fixed point $\boldsymbol{x}$ with respect to the given Lyapunov function is equal to minimizing $\dot{V}_u$, which in turn is equal to minimizing each term in the sum in (3.38) as one can deduce from (3.40).

To achieve this minimization, the j-th control input has to be chosen as small as possible if $\boldsymbol{x}^{\mathrm{T}}.\boldsymbol{n}_j > 0$ ($\boldsymbol{x}$ lies in the upper half-space of $\mathcal{P}_j$) and as large as possible if $\boldsymbol{x}^{\mathrm{T}}.\boldsymbol{n}_j < 0$ ($\boldsymbol{x}$ lies in the lower half-space of $\mathcal{P}_j$) to obtain the minimal value of each summand defining $\dot{V}_u$. The corresponding optimal control input in the sense of instantaneous convergence of the system is

$$\boldsymbol{u}^*(\boldsymbol{x})[j] = \begin{cases} \boldsymbol{u}^+[j] \text{ if } \boldsymbol{x}^{\mathrm{T}}.\boldsymbol{n}_j < 0 \\ \boldsymbol{u}^-[j] \text{ if } \boldsymbol{x}^{\mathrm{T}}.\boldsymbol{n}_j > 0 \\ u_{P_j} \text{ else} \end{cases} \tag{3.41}$$

where, in order to remove the ambiguity, any input u_{P_j}, satisfying $\boldsymbol{u}^-[j] \leq u_{P_j} \leq \boldsymbol{u}^+[j]$, can be chosen if the current state belongs to the hyperplane. The undefined character of u_{P_j} does not pose a problem for proving stability since its contribution to $\dot{V}_u$ is 0 independently of the value of u_{P_j} as $\boldsymbol{x}^{\mathrm{T}}.\boldsymbol{n}_j = 0$. As one can see this control law partitions the state-space into two open half-spaces for each of the m control inputs. Denoting $\mathcal{H}_i$ for $i \in [0, 2^m - 1]$ the unbounded convex polytope defined as the intersection of m upper or lower half-spaces generated by the hyperplanes $\mathcal{P}_{j\in[0,m-1]}$ we get

$$\mathcal{H}_i = \left\{\boldsymbol{x} | \forall j \in [0, m-1]\ c_j^i \boldsymbol{x}^{\mathrm{T}}.\boldsymbol{n}_j \leq 0\right\} \tag{3.42}$$

where $c_j^i \in \{-1, 1\}$, see Figure 3.4, is a switch to determine whether the upper or lower half-space of the j-th hyperplane is used. Note that in (3.42) the strict inequalities of (3.41) are replaced with inequalities. This poses no problem as the control input on the hyperplane can be chosen arbitrarily. One can easily see that for each such polytope $\mathcal{H}_i$ the optimal control input $\boldsymbol{u}$, denoted $\boldsymbol{u}_i^*$, is independent of $\boldsymbol{x}$. Since there exist m such hyperplanes, the state-space is partitioned into 2^m polytopes with different optimal inputs. Note that each of these polytopes has non-empty interior if the matrix P is positive definite and B has full column rank (i.e. rank m). The definiteness of P is always given in order to ensure that $V(\boldsymbol{x}) = \boldsymbol{x}^{\mathrm{T}}.P.\boldsymbol{x}$ is a Lyapunov function candidate and in general, for dynamical systems, B is of rank m. If the rank condition on B is violated, it means that there exist at least two inputs which are linearly

dependent. One can then proceed to replace these two or more inputs by fewer linearly independent inputs resulting in the equivalent input dynamics matrix B' and the corresponding control vector $\boldsymbol{u}'$ and input constraints $\boldsymbol{u}'^{+}$ and $\boldsymbol{u}'^{-}$. Then one can define the equivalent dynamical system $\dot{\boldsymbol{x}} = f(\boldsymbol{x}) + B'.\boldsymbol{u}'$ subject to $\boldsymbol{u}'^{-} \leq \boldsymbol{u}' \leq \boldsymbol{u}'^{-}$ and return to the general case. This approach is based on the ideas developed in Longchamp [1980] for bilinear systems.

To better visualize the approach of state-space partitioning, the running example of the torque controlled pendulum is introduced and preliminary results are presented.

Consider the dynamical system

$$\dot{\boldsymbol{x}} = \begin{pmatrix} \theta \\ \dot{\theta} \end{pmatrix} = f(x) + B.u = \begin{pmatrix} \dot{\theta} \\ -\omega^2 \sin(\theta) \end{pmatrix} + \begin{pmatrix} 0 \\ c_\tau \end{pmatrix} .u \tag{3.43}$$

representing a torque controlled simple pendulum with $\theta = 0$ corresponding to the stable, "hanging" position. Further ω denotes the natural frequency of the system and c_τ the normalized torque. The set of admissible control inputs is $\mathcal{U} = [u^-, u^+]$. Consider the Lyapunov function candidate $V(\boldsymbol{x}) = \boldsymbol{x}^{\mathrm{T}}.Id.\boldsymbol{x}$, corresponding to convergence with respect to the squared l_2-norm. Since the pendulum is a single input system $B = B[:, 0]$, there exists only one separating hyperplane defined by the normal vector $\boldsymbol{n} = Id.B = c_\tau \boldsymbol{e}_{\dot{\theta}}$ as indicated by the green arrow in Figure 3.4. We therefore obtain the optimal control inputs associated to the partitioning of the state space formed by the two subsets $\mathcal{H}_0$ and $\mathcal{H}_1$ as $u_0^* = u^-$ and $u_1^* = u^+$. From both half-spaces the system states converge to the hyperplane when the states belong to the true region of stabilizability when applying the optimal control law (3.41), otherwise they diverge. The black circle show the largest RoS, the largest α, such that the set $\Omega = \left\{\boldsymbol{x} | \boldsymbol{x}^{\mathrm{T}}.Id.\boldsymbol{x} \leq \alpha\right\}$ is a subset of the true region of stabilizabilty. Once the state attains the separating hyperplane, switching between u^- and u^+ occurs, at possibly infinite frequency, in order to maintain the state on the hyperplane. This bears resemblances to sliding-mode control which will be discussed in section 3.6.

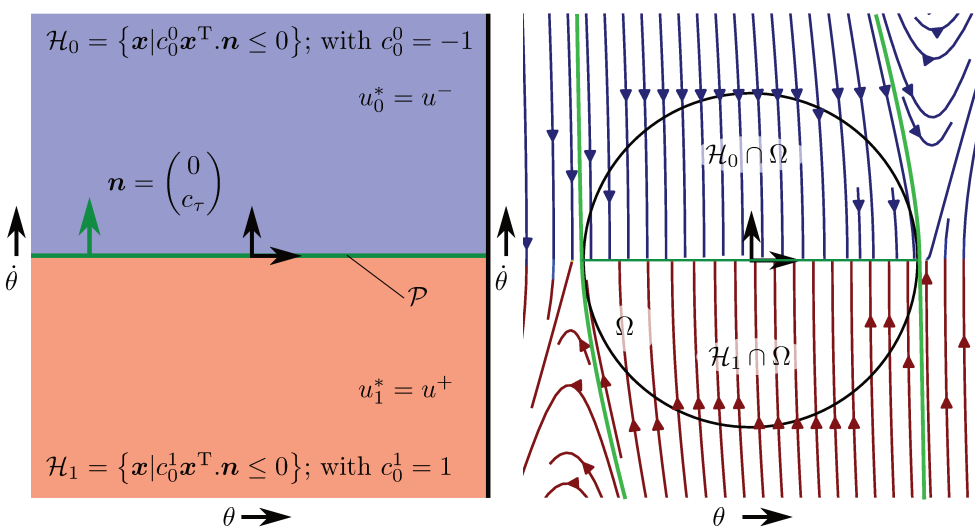


Figure 3.4 – The resulting partitioning for the torque controlled pendulum and the Lyapunov function $V(\boldsymbol{x}) = \boldsymbol{x}^{\mathrm{T}}.Id.\boldsymbol{x}$ is shown on the left, the resulting streamline plot when using the optimal control law is shown on the right. The green lines indicate the border of the true region of stabilizability for this Lyapunov function with a convergence rate γ equal to zero. The black circle is the largest inner approximation defined as a sublevel-set of $V(\boldsymbol{x})$.

This approach allows us to obtain 2^m subsets of the state space $\mathcal{H}_i$, the dynamics conditioned by the optimal control input $\dot{\boldsymbol{x}} = f(\boldsymbol{x}) + B.\boldsymbol{u}_i^*$ and the best obtainable derivative of the Lyapunov function

candidate

$$\dot{V}_i^*(x) = \langle \nabla_{\boldsymbol{x}} V(\boldsymbol{x}),\, \dot{\boldsymbol{x}}^* \rangle = 2.\boldsymbol{x}^{\mathrm{T}}.P.(f(\boldsymbol{x}) + B.\boldsymbol{u}_i^*). \tag{3.44}$$

Now the min-max problem (3.36) for proving stability can be reformulated. Since we determined the optimal input with respect to instantaneous convergence for each $\mathcal{H}_i$ we can drop the inner minimization by checking each intersection of the partition with the sublevel-set of V considered. So the inner minimization

$$\min_{\boldsymbol{u}\in\mathcal{U}} \langle \nabla_{\boldsymbol{x}} V(\boldsymbol{x}), f(\boldsymbol{x}) + g(\boldsymbol{x}).\boldsymbol{u} \rangle \tag{3.45}$$

becomes

$$\boldsymbol{x} \in \mathcal{H}_i :\ 2.\boldsymbol{x}^{\mathrm{T}}.P.(f(\boldsymbol{x}) + B.\boldsymbol{u}_i^*) \tag{3.46}$$

and therefore

$$\forall i : \max_{\boldsymbol{x}\in(\Omega\cap\mathcal{H}_i)\setminus\{\boldsymbol{0}\}} \dot{V}_i^*(\boldsymbol{x}) = 2.\boldsymbol{x}^{\mathrm{T}}.P.(f(\boldsymbol{x}) + B.\boldsymbol{u}_i^*) \leq -\gamma.\boldsymbol{x}^{\mathrm{T}}.P.\boldsymbol{x} \tag{3.47}$$

is equivalent to the initial problem (3.36) and represents a proof of exponential convergence. It also proves, constructively, that Ω is a region of exponential stabilizability with guaranteed minimal convergence rate γ.

Note the conceptual difference. While the works cited in section 3.4 prove stability by introducing a (polynomial) control law $\boldsymbol{u} = K(\boldsymbol{x})$ and then enlarge the region of attraction with respect to the closed loop system, the proposed partitioning of the state space in contrast directly takes advantage of the specific problem structure and reasons directly on the region of stabilizability. This yields two advantages. First the conservativeness resulting from enforcing the constraint that the control law has to generate admissible control inputs everywhere in the considered region is avoided. Secondly the resulting optimization has less decision variables as no control law or additional multipliers have to be introduced, resulting in a possibly simpler optimization problem.

The ultimate goal is to maximize the size or better volume of Ω, which depends only on one parameter, α, as we do not seek to modify $V(\boldsymbol{x})$ defining the shape of the sublevel-set. In section 3.7, we use a dichotomic search to quickly find a large value for α. This requires us to efficiently check the validity of (3.47). In section 3.8 we show how this can be done by using relaxations to deal with the non-convex polynomial expressions arising in (3.47).

3.5.3 State Space Partitioning for Perturbed Systems

The above presented state space partitioning can readily be extended to the case of control and perturbation affine polynomial systems of the form

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}) + B.\boldsymbol{u} + B_\omega.\boldsymbol{\omega} \tag{3.48}$$

with $B_\omega \in \mathbb{R}^{n\times l}$ representing the perturbation channels and $\boldsymbol{\omega} \in \mathbb{R}^l$ represents the perturbations. In order for this approach to be directly applicable to this case, we consider that the perturbations are also box-constrained, so $\boldsymbol{\omega}^- \leq \boldsymbol{\omega} \leq \boldsymbol{\omega}^+$ holds element-wise.

In this case we can see the perturbations much like the control inputs, however with the "goal" to make the system diverge from the equilibrium point. Therefore we can compute a separation hyperplane for each perturbation $\boldsymbol{\omega}_j$ called $\boldsymbol{n}_{\omega,j}$ as $P.B_{\boldsymbol{\omega}}[:,j]$ and define the *divergence* optimal perturbation as

$$\boldsymbol{\omega}^*(\boldsymbol{x})[j] = \begin{cases} \boldsymbol{\omega}^+[j] \text{ if } \boldsymbol{x}^{\mathrm{T}}.\boldsymbol{n}_{\omega,j} > 0 \\ \boldsymbol{\omega}^-[j] \text{ if } \boldsymbol{x}^{\mathrm{T}}.\boldsymbol{n}_{\omega,j} < 0 \\ \boldsymbol{\omega}_{P_j} \text{ else} \end{cases} \tag{3.49}$$

which is exactly the same as (3.49) except that the half-space for which the minimal or maximal perturbation is used are exchanged. Now the extended control law $\hat{\boldsymbol{u}}$ can defined as

$$\hat{\boldsymbol{u}}^* = \begin{pmatrix} \boldsymbol{u}^* \\ \boldsymbol{\omega}^* \end{pmatrix} \tag{3.50}$$

with the extended input matrix

$$\hat{B} = \begin{bmatrix} B & B_\omega \end{bmatrix} \tag{3.51}$$

and return to the equivalent control affine polynomial dynamics under the optimal control law

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}) + \hat{B}.\hat{\boldsymbol{u}}^* \tag{3.52}$$

for which we can apply the above method resulting in 2^{m+l} polytopes.

The resulting partitioning for the pendulum example with the perturbations defined by ω and $B_\omega = \begin{bmatrix} 1, 1 \end{bmatrix}^{\mathrm{T}}$ is shown in Figure 3.5.

This approach however is very conservative, as for each instant the worst possible perturbation is assumed, instead of it being drawn from some distribution.

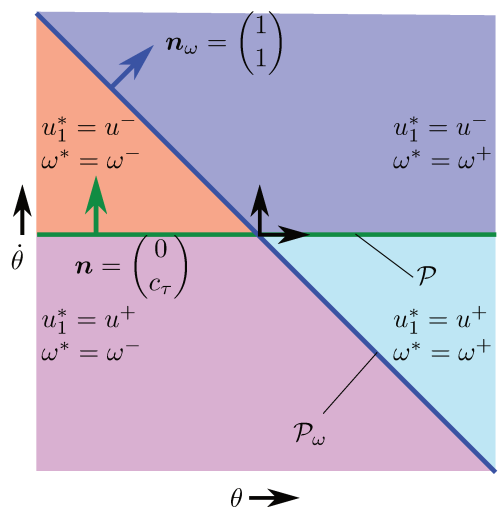


Figure 3.5 – The resulting partitioning for the torque controlled pendulum and the Lyapunov function $V(\boldsymbol{x}) = \boldsymbol{x}^{\mathrm{T}}.Id.\boldsymbol{x}$ for the perturbed case.

3.6 Resulting Closed Loop Dynamics and Links to Sliding Mode and QP-Control

In this section we point out links between the optimal control law with respect to instantaneous convergence (3.41) and first order sliding mode control. Since this control mode can induce chattering and premature wear out due to the high, possibly infinite, switching frequency on the sliding surface it is not suitable for real applications. We therefore introduce a quadratic programming (QP) control law that results in continuous control trajectories and provides the same certificates. To illustrate the approach and the resulting dynamics, the torque controlled simple pendulum, see (3.43), and the Acrobot (Spong [1995]), are used as a showcase.

3.6.1 Sliding Mode Control

Sliding mode control was introduced in the 1970s, see for instance Emel'yanov and Utkin [1964] for one of the early works, and is concerned with proving stability of nonlinear control systems of the general form $\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u})$. The main idea of sliding mode control is the following: in a first step a sliding manifold is created on which the system behaves in the desired the way, e.g. the system is asymptotically stable. Then in a second step, control laws are designed that drive the state towards the sliding manifold within some neighbourhood around it. The advantage lies in the fact that on the sliding manifold, the order of the system is effectively decreased, as the switching between the control laws will keep the state on the manifold making convergence proofs possibly easier.

To put it formally for a simple case, consider the nonlinear control system $\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u})$, the sliding manifold implicitly defined as $\Omega = h(\boldsymbol{x}) = 0$ with $h\colon \mathbb{R}^n \to \mathbb{R}$ being $\mathcal{C}^1$, the reduced order system on the sliding manifold $\dot{\tilde{\boldsymbol{x}}} = \tilde{f}(\tilde{\boldsymbol{x}})$ and the control laws $k_0(\boldsymbol{x})$ and $k_1(\boldsymbol{x})$. Then, in order to prove global stability of the system, one has to prove that

$$\forall \boldsymbol{x} \in \Omega\colon \quad \lim_{t\to\infty} \tilde{\boldsymbol{x}} = \boldsymbol{0} \tag{3.53a}$$

$$\forall \boldsymbol{x} \in U(\Omega) \cap \{\boldsymbol{x} | h(\boldsymbol{x}) > 0\}: \ \langle \nabla_{\boldsymbol{x}} h(\boldsymbol{x}),\, f(\boldsymbol{x}, k_0(\boldsymbol{x}))\rangle < 0 \tag{3.53b}$$

$$\forall \boldsymbol{x} \in U(\Omega) \cap \{\boldsymbol{x} | h(\boldsymbol{x}) < 0\}: \ \langle \nabla_{\boldsymbol{x}} h(\boldsymbol{x}),\, f(\boldsymbol{x}, k_1(\boldsymbol{x}))\rangle > 0 \tag{3.53c}$$

$$\forall \boldsymbol{x},\, \exists T_r < \infty\colon \ \forall t > T_r \ h(\boldsymbol{x}) = 0 \tag{3.53d}$$

where $U(\Omega)$ denotes some neighbourhood of Ω and the last condition, known as finite reaching time condition, ensures that all states reach the sliding manifold in finite time. There also exist local versions of the above conditions but these are usually tailored for the dynamical system at hand, see for instance Wang et al. [2004].

This gives the system a variable control structure, as the resulting dynamical system can be defined as

$$\dot{\boldsymbol{x}} = \begin{cases} f(\boldsymbol{x}, k_0(\boldsymbol{x})) & \text{if } h(\boldsymbol{x}) > 0 \\ f(\boldsymbol{x}, k_1(\boldsymbol{x})) & \text{else if } h(\boldsymbol{x}) < 0 \\ f(\boldsymbol{x}, \boldsymbol{0}) & \text{else} \end{cases}. \tag{3.54}$$

The advantages of sliding mode control are that one can choose control laws with high gains providing good perturbation rejection while ensuring a smooth overall behaviour on the sliding manifold by their design. For real applications however the high (infinite) frequency switching between the control laws can cause chattering for second-order systems and is generally not desirable.

Till here the main concept of sliding mode control was briefly introduced. For a more complete introduction to the topic and approaches to deal with the high frequency switching see, among others, Edwards and Spurgeon [1998] or Pisano and Usai [2011].

By defining $h(\boldsymbol{x}) = \boldsymbol{x}^{\mathrm{T}}.\boldsymbol{n}$, $k_0(\boldsymbol{x}) = u^-$ and $k_1(\boldsymbol{x}) = u^+$ the structure given above corresponds to the optimal control law derived in the last section for the torque controlled pendulum and given Lyapunov

function. Therefore one might draw the conclusion that the separating hyperplanes $\mathcal{P}_j$ defined by the corresponding normal vector $\boldsymbol{n}_j$ correspond to sliding manifolds defined by $h_j(\boldsymbol{x}) = \boldsymbol{x}^{\mathrm{T}}.\boldsymbol{n}_j = 0$. However this is not true. For the running example of the torque controlled pendulum, the separating hyperplane is indeed a (local) sliding manifold, but this does not hold in general. The stability proof based on Lyapunov's second method demands that all states converge towards the origin with respect to the Lyapunov function $V(\boldsymbol{x})$. The condition that all states within some neighbourhood of the manifold have to converge towards it does not follow from this constraint. Indeed one can perfectly imagine a dynamical system and a Lyapunov function candidate for which $\boldsymbol{x}^{\mathrm{T}}.\boldsymbol{n}_j$ increases along a trajectory after crossing $\mathcal{P}_j$ while also converging with respect to $V(\boldsymbol{x})$. Even though we have just shown that our approach does not directly match sliding mode control, the concerns about the possibly infinite switching frequency on the separating hyperplane do directly transfer to our approach. To be more precise, the infinite switching frequency poses problems from a practical and theoretical point of view. It induces chattering and premature wear out of the system, especially for second order models such as robotic systems. On the other hand it also raises theoretical concerns as the solution to such a system is ill-defined due to the zeno effect.

3.6.2 From Sliding Mode Control to a Continuous Control Law

To showcase the resulting partition in Figure 3.4, the dynamical system represents a torque controlled pendulum and the Lyapunov function candidate $V(\boldsymbol{x}) = \boldsymbol{x}^{\mathrm{T}}.I_d.\boldsymbol{x}$ is imposed. Since the pendulum is a single input system, there exists only one separating hyperplane defined by the normal vector $\boldsymbol{n} = I_d.B = c_\tau \boldsymbol{e}_{\dot{\theta}}$. We obtain the optimal control inputs associated to the partitioning of the state space formed by two subsets $\mathcal{H}_0$ and $\mathcal{H}_1$ as $u_0^* = u^-$ and $u_1^* = u^+$. As already stated above, the system states converge to the hyperplane if they belong to the true region of stabilizability for the considered Lyapunov function. Once this surface obtained, the system is in sliding mode, meaning that the switching between the u^- and u^+ occurs at high, possibly infinite frequency, and the state is maintained on the hyperplane. This corresponds to first order sliding mode control and the separating hyperplane is the sliding surface for this system. Note that this is merely an observation for the dynamical system at hand and not something that can be deduced from the proof of stabilizability, as pointed out above. However this observation holds for both dynamical systems used as showcases here, the already introduced torque controlled pendulum and the Acrobot which we will introduce now. The Acrobot is a planar underactuated 2R robot. Its base joint is passive, only the joint between the first and second segment is actuated. Its dynamics, written in the usual convention, are given as

$$S.\boldsymbol{\tau} = M_{\boldsymbol{q}}.\ddot{\boldsymbol{q}} + C_{\boldsymbol{q},\dot{\boldsymbol{q}}} + \boldsymbol{g}_{\boldsymbol{q}} \tag{3.55}$$

where $M_{\boldsymbol{q}} \in \mathbb{S}^2_{++}$ is the mass matrix, $C_{\boldsymbol{q},\dot{\boldsymbol{q}}}$ is the vector containing the nonlinear forces (Coriolis and centrifugal forces), $\boldsymbol{g}_{\boldsymbol{q}}$ denotes the gravity forces and $S = \begin{pmatrix} 0 & 1 \end{pmatrix}^{\mathrm{T}}$ being the input selection matrix. The indices $_{\boldsymbol{q}}$ and $_{\dot{\boldsymbol{q}}}$ denote the dependencies to the configuration $\boldsymbol{q} = \begin{pmatrix} \theta_0 & \theta_1 \end{pmatrix}^{\mathrm{T}}$ and its derivative $\dot{\boldsymbol{q}}$.
Now we can bring this system into the standard form $\dot{\boldsymbol{x}} = f^{nl}(\boldsymbol{x}) + g^{nl}(\boldsymbol{x}).\boldsymbol{u}$ by defining

$$\boldsymbol{x} = \begin{pmatrix} \boldsymbol{q} \\ \dot{\boldsymbol{q}} \end{pmatrix} = \begin{pmatrix} \theta_0 \\ \theta_1 \\ \omega_0 \\ \omega_1 \end{pmatrix} \qquad \boldsymbol{u} = \boldsymbol{\tau}$$

$$f^{nl}(\boldsymbol{x}) = \begin{pmatrix} \omega_0 \\ \omega_1 \\ -M_{\boldsymbol{q}}^{-1}.\left(C_{\boldsymbol{q},\dot{\boldsymbol{q}}} + \boldsymbol{g}_{\boldsymbol{q}}\right) \end{pmatrix} \qquad g^{nl}(\boldsymbol{x}) = \begin{pmatrix} \boldsymbol{0} \\ M_{\boldsymbol{q}}^{-1} \end{pmatrix}.S \qquad .$$

These equations, as indicated by ^{nl}, correspond to the fully nonlinear system. As the proposed approach is restricted to control affine polynomial systems, we use the truncated (multivariate) Taylor

expansion, usually up to degree 3, with respect to the equilibrium point $(\boldsymbol{x}^*, \boldsymbol{u}^*)$ for the system dynamics and the value of $g^{nl}(\boldsymbol{x}^*)$ as (linear) approximation for the input dynamics. So the dynamical system used within the stabilizability proof becomes

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}) + B.\boldsymbol{u} \tag{3.56a}$$

$$f[i](\boldsymbol{x}) = \sum_{|\boldsymbol{\beta}|=0}^{3} \frac{(\boldsymbol{x}-\boldsymbol{x}^*)^{\boldsymbol{\beta}}}{|\boldsymbol{\beta}|!} \frac{\partial^{|\boldsymbol{\beta}|}}{\partial \boldsymbol{x}^{\boldsymbol{\beta}}} \left(f^{nl}[i]\right)(\boldsymbol{x}^*) \tag{3.56b}$$

$$B = g^{nl}(\boldsymbol{x}^*) \tag{3.56c}$$

using the notations for monomials introduced in section 3.2.4. Note that the approximation of the original nonlinear system using linearisation and truncated Taylor expansion is not conservative in the sense that it is not guaranteed that the RoS found for (3.56a) is strictly contained inside the RoS of the nonlinear system. This can be ensured by considerations involving the (local) Lipschitz constant of the error dynamics or by the approach proposed by Chesi [2009], directly reasoning on the worst-case remainder of the truncated Taylor extension. In this work, as in Tedrake et al. [2010b], Singh et al. [2017] or Majumdar et al. [2013b], we do not ensure this conservativeness but rely on the supposition that the rigid-body dynamics of robotic systems can be well approximated by third order Taylor expansion, as shown empirically in section 3.10. Moreover, as the proposed approach relies on efficiently solving nonconvex polynomial programming over subsets of the state space, one can seek to refine the model by first partitioning each subset of the state space into smaller set, then use the input optimal partitioning to check convergence on each subset. This way one can linearise around the center of each subset reducing the largest occuring error between the polynomial approximation and the true nonlinear system.

QP-Control for a Continuous Control Law

As pointed out above, first order sliding control raises concerns from both theoretical and practical point of view. To remedy these problems we propose a QP-based control law and prove, using Berge's Maximum Theorem (Berge [1970]), that it guarantees exponential convergence on $\Omega \setminus \{\mathbf{0}\}$ and results in continuous control trajectories if (3.47) holds.

The proposed QP-control is

$$\underset{\boldsymbol{u}}{\text{minimize}}\ h(\boldsymbol{x}, \boldsymbol{u}) = \boldsymbol{u}^{\mathrm{T}}.Q.\boldsymbol{u} + 2\boldsymbol{x}^{\mathrm{T}}.P.B.\boldsymbol{u} \tag{3.57a}$$

$$\text{subject to } \boldsymbol{u}^- \leq \boldsymbol{u} \leq \boldsymbol{u}^+ \tag{3.57b}$$

$$2\boldsymbol{x}^{\mathrm{T}}.P.B.\boldsymbol{u} \leq -2\boldsymbol{x}^{\mathrm{T}}.P.f(\boldsymbol{x}) - \gamma.\boldsymbol{x}^{\mathrm{T}}.P.\boldsymbol{x} \tag{3.57c}$$

where $2\boldsymbol{x}^{\mathrm{T}}.P.B.\boldsymbol{u}$ is the input dependent part of the derivative of the Lyapunov function $\dot{V}_u(\boldsymbol{x})$, $Q \in \mathbb{S}_{++}$ is user chosen, typically diagonal, and reflects the cost of the control effort. The first constraint represents the boundedness of the control input and the second constraint ensures the demanded exponential convergence. Note that this optimization problem is guaranteed to have a solution for all $\boldsymbol{x} \in \Omega \setminus \{\mathbf{0}\}$ if (3.47) holds. Moreover, the form of the objective function (with $Q \in \mathbb{S}_{++}$), together with the convexity of the constraints, ensures uniqueness of solutions and makes the problem amenable to convex quadratic programming.

In order to prove continuity of the resulting control trajectory, we employ Berge's Maximum Theorem, which, combined with the uniqueness of solutions, guarantees continuous control trajectories. Berge's Maximum Theorem provides conditions to ensure the continuity (properly speaking the upper hemicontinuity) of the set of control inputs respecting the constraints and minimizing the objective function. Since this set collapses to a singleton $\boldsymbol{u}^*$ (the unique solution) in the case of convex quadratic programming, it ensures continuity. The fulfilled conditions are that the objective function $h(\boldsymbol{x}, \boldsymbol{u})$ is jointly continuous in $\boldsymbol{x}$ and $\boldsymbol{u}$ and that the set of inputs respecting the constraints is compact and hemicontinuous (upper and

lower) with respect to $\boldsymbol{x}$. This condition is trivially fulfilled as (3.57b) is independent of $\boldsymbol{x}$ and (3.57c) is a jointly smooth function in $\boldsymbol{x}$ and $\boldsymbol{u}$.

The influence of the regularization matrix Q on the control effort and dynamics is shown in Figure 3.6.

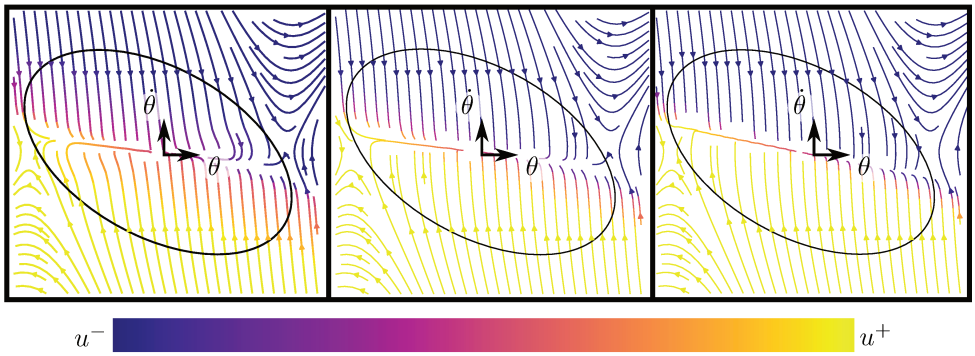


Figure 3.6 – In these figures the resulting dynamics, of the control affine polynomial approximation, for the inverse pendulum using the proposed QP-control are shown. The convergence rate $\gamma = 0.5$ is the same for all figures. The regularization term Q is takes on the values 2, 1 and 0.5 from the left to the right image. As one can see, a high regularization induces a smoother control law (illustrated by a smoother transition in the colours). As the regularization term diminishes, to resulting control law approaches more and more the optimal (discontinuous) control law u^*, which maximizes instantaneous convergence. For points outside the ellipsoid (Ω) the QP-problem is not always feasible (when the point considered is outside the true RoS) in which case u^* is used for plotting.

Remark 3.4. We will see in section 3.9 that $V(\boldsymbol{x})$ (thus P) can be chosen such that the control inputs are not only continuous but also the slopes have reasonable absolute values.

Remark 3.5. The optimization objective is arbitrary (as long as Berge's Maximum Theorem hypotheses hold) and the decisive constraint assuring convergence is a simple linear constraint on the control input, which can be easily added to existing optimization based controllers.

3.6.3 Comparison with Sliding Mode Control

In Spong [1995] a generic way to obtain stable sliding-mode control laws for a class of underactuated second-order systems, including the Acrobot, is proposed. Even though this approach also yields a hyperplane (not equivalent to ours) used to switch the sign of (a part of) the input, the resulting behaviour and properties are very distinct. The proof of convergence of the sliding mode control is performed in two steps: I) Prove that the system converges to the sliding surface (the hyperplane) II) Prove that all points on the hyperplane converge to the origin. Using such an approach, it is not obvious how to deal with bounded inputs or specify an invariant region. Since trajectories are allowed to stray very far from the origin while converging towards the sliding surface, it increases the risk of leaving the true RoA that takes into account the boundedness of the input, see Figure 3.7.

3.7 Extension to Time-Varying Case and Implementation

Up to here it was shown how we can partition the state space based on the optimal input with respect to instantaneous convergence in the case of time-independent quadratic Lyapunov functions and control affine polynomial systems. In this section we discuss how to adapt this approach to construct *large*

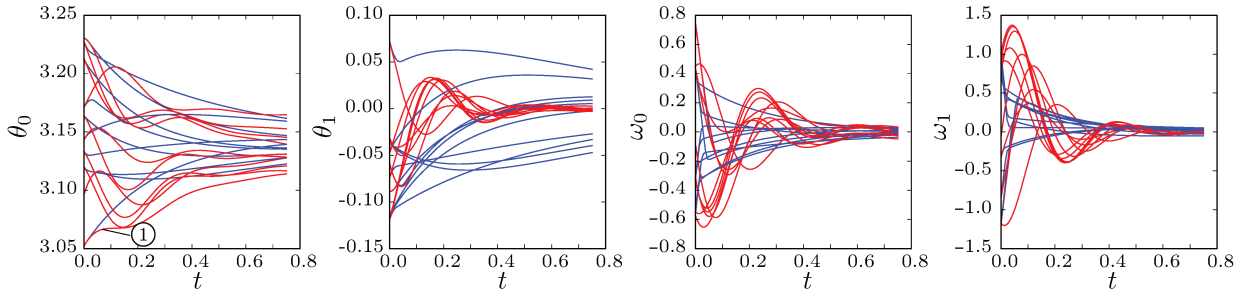


Figure 3.7 – In these images, the resulting trajectories from applying the QP-based control law (3.57) (blue lines) and the sliding mode control from Spong [1995] (red lines) are compared with the initial states randomly distributed close to the boundary of the largest RoS found using Algorithm 2. One can see that the trajectories for the sliding mode control law tend to converge faster towards the equilibrium but with greater oscillations. Due to the proof of stability based on the sliding mode conditions, no quadratic Lyapunov function (with respect to $\boldsymbol{x}$) can be found for the closed loop system. Moreover, since input constraints are not taken into account, the trajectory marked with 1 is almost leaving the RoA.

funnels, defined as time-varying region of exponential stbilizability, along reference trajectories. This imposes time-dependent Lyapunov functions and necessitates an iterative algorithm to construct the funnel. The reference trajectories are suppossed to be user-supplied as this method does not seek to perform path planning, but to establish large regions of stabilizability. The reference trajectories are supposed to have the form $(\boldsymbol{x}^r(t), \boldsymbol{u}^r(t))$ defined on $t \in I = [0, T]$ with $\boldsymbol{x}^r(t)$ being differentiable and $\forall t \in I\colon\ \frac{\mathrm{d}}{\mathrm{d}\,t}\boldsymbol{x}^r(t) = f^{nl}(\boldsymbol{x}^r_t) + g^{nl}(\boldsymbol{x}^r_t).\boldsymbol{u}^r(t)$.

3.7.1 Time-Varying Lyapunov Functions and Nonlinear Dynamics

To construct a time-varying region of stabilizability (funnel) denoted Ω_t, the ellipsoidal regions resulting from the quadratic Lyapunov functions have to be centred on the reference trajectory, so we have

$$\Omega_t = \left\{\boldsymbol{x} | V(t, \boldsymbol{x}) = (\boldsymbol{x} - \boldsymbol{x}^r_t)^{\mathrm{T}}.P_t.(\boldsymbol{x} - \boldsymbol{x}^r_t) \leq \alpha_t\right\}. \tag{3.58}$$

Therefore the time-derivative of the stability condition becomes

$$\dot{V}(t, \boldsymbol{x}) \leq \dot{\alpha}_t \tag{3.59a}$$

$$\dot{V}(t, \boldsymbol{x}) = (\dot{\boldsymbol{x}} - \dot{\boldsymbol{x}}^r_t)^{\mathrm{T}}.P_t.(\boldsymbol{x} - \boldsymbol{x}^r_t) + (\boldsymbol{x} - \boldsymbol{x}^r_t)^{\mathrm{T}}.P_t.(\dot{\boldsymbol{x}} - \dot{\boldsymbol{x}}^r_t) + (\boldsymbol{x} - \boldsymbol{x}^r_t)^{\mathrm{T}}.\dot{P}_t.(\boldsymbol{x} - \boldsymbol{x}^r_t). \tag{3.59b}$$

As our primary target of applications are articulated robotic systems, we also have to deal with the fully nonlinear dynamics $\dot{\boldsymbol{x}} = f^{nl}(\boldsymbol{x}) + g^{nl}(\boldsymbol{x}).\boldsymbol{u}(t)$.

We use the following approach: for checking exponential stabilizability at time t for a fixed P_t and α_t, the fully nonlinear condition according to (3.47) becomes

$$\forall i : \max_{\boldsymbol{x} \in (\Omega_t \cap \tilde{\mathcal{H}}_i) \setminus \{\boldsymbol{0}\}} \dot{V}^{nl*}_i(t, \boldsymbol{x}) = 2.\delta\boldsymbol{x}^{\mathrm{T}}.P.(f^{nl}(\boldsymbol{x}) + g^{nl}(\boldsymbol{x}).\tilde{\boldsymbol{u}}^*_i - \dot{\boldsymbol{x}}^r_t) + \delta\boldsymbol{x}^{\mathrm{T}}.\dot{P}_t.\delta\boldsymbol{x} \leq -\gamma.\delta\boldsymbol{x}^{\mathrm{T}}.P.\delta\boldsymbol{x} \tag{3.60}$$

with $\delta\boldsymbol{x} = \boldsymbol{x} - \boldsymbol{x}^r_t$ and $\tilde{\boldsymbol{u}}^*_i$ denoting the optimal control input with respect to the nonlinear input dynamics and $\tilde{\mathcal{H}}_i$ being the associated set in the partition of the state space (which is no longer a polytope).

This condition however is computationally not tractable due to the fully nonlinear dynamics and the higher "complexity" of the corresponding subsets defining the partitioning of the state space. We therefore replace them with their truncated Taylor expansion around the current reference point $\boldsymbol{x}^r_t$,

$f(\delta\boldsymbol{x}) \approx f^{nl}(\boldsymbol{x})$ and $g(\delta\boldsymbol{x}) \approx g^{nl}(\boldsymbol{x})$ respectively. Note that $\boldsymbol{x}_t^r$ is also the center of Ω_t and is therefore the natural choice. Now the condition is written

$$\forall i : \max_{\boldsymbol{x}\in(\Omega_t\cap\tilde{\mathcal{H}}_i)\setminus\{\boldsymbol{0}\}} \dot{V}_i^*(t,\boldsymbol{x}) = 2.\delta\boldsymbol{x}^{\mathrm{T}}.P.(f(\delta\boldsymbol{x}) + g(\delta\boldsymbol{x}).\tilde{\boldsymbol{u}}_i^* - \dot{\boldsymbol{x}}_t^r) + \delta\boldsymbol{x}^{\mathrm{T}}.\dot{P}_t.\delta\boldsymbol{x} \leq -\gamma.\delta\boldsymbol{x}^{\mathrm{T}}.P.\delta\boldsymbol{x}. \quad (3.61)$$

In this condition all terms are polynomial, and therefore efficiently solvable, but the condition on which the optimal state space partitioning is based, that the system is affine with respect to the control inputs, is violated, as we have polynomial input dynamics. Therefore the unbounded polytopes $\mathcal{H}_i$ in the control affine case now become semi-algebraic sets defined by polynomial inequalities of the form

$$\tilde{\mathcal{H}}_i = \left\{\boldsymbol{x} | c_i^j \delta\boldsymbol{x}^{\mathrm{T}}.P.g(\delta\boldsymbol{x})[:,j] \leq 0\right\} \quad (3.62)$$

inducing the associated optimal control input $\tilde{\boldsymbol{u}}^*$. These sets are much more general than the polytopes, resulting in constraints harder to express and which are not suitable for the proposed approach to prove non-positiveness presented in section 3.8. To remedy this problem, we compute the optimal control input $\boldsymbol{u}^*$ and the partitioning $\mathcal{H}_i$ based on the linearised input dynamics $B = g^{nl}(\boldsymbol{x}_t^r)$. The condition then becomes

$$\forall i : \max_{\boldsymbol{x}\in(\Omega_t\cap\mathcal{H}_i)\setminus\{\boldsymbol{0}\}} \dot{V}_i^*(t,\boldsymbol{x}) = 2.\delta\boldsymbol{x}^{\mathrm{T}}.P.(f(\delta\boldsymbol{x}) + g(\delta\boldsymbol{x}).\boldsymbol{u}_i^*) + \delta\boldsymbol{x}^{\mathrm{T}}.\dot{P}_t.\delta\boldsymbol{x} + \gamma.\delta\boldsymbol{x}^{\mathrm{T}}.P.\delta\boldsymbol{x} \leq 0 \quad (3.63)$$

which is equivalent to proving non-positiveness of a polynomial. Here the "relative" character of the current system state with respect to the current reference point is explicitly denoted using the δ prefix. In order to keep notations short, the δ will be dropped if it is clear from context that $\boldsymbol{x}$ is relative to $\boldsymbol{x}_t^r$. In this case we also redefine $f(\boldsymbol{x}) = f(\delta\boldsymbol{x}) - \dot{\boldsymbol{x}}_t^r$.

Obviously, the optimal control input $\boldsymbol{u}^*$ is only optimal with respect to the affine input dynamics, not for the polynomial input dynamics. This sub-optimality is in general *small* for the considered systems due to two reasons. First, for polynomial expressions derived from second-order systems, the nonlinear terms of $f(\boldsymbol{x})$ are usually smaller than the linear terms within the RoS. This results in the absolute error of the dynamics induced by using the sub-optimal control law to be small and moreover, the region where $\boldsymbol{u}^*$ is different from $\tilde{\boldsymbol{u}}^*$ is *close* to the separating hyperplane $\mathcal{P}_j$. As we have seen that the contribution of the control input $\boldsymbol{u}[j]$ to $\dot{V}_u$ depends on the distance between the state and the hyperplane the induced error in the controlled part of the derivative $\dot{V}_u$ is the product of two supposedly *small* values. Therefore the difference between the largest RoS provable by (3.61) and (3.63) is probably negligible (within this use-case).

3.7.2 Funnel Construction via Retro-Propagation

The initial problem that we are seeking to answer is to construct an as large as possible funnel around a reference trajectory for which we can ensure that all states inside of it will remain inside of it or even converge exponentially towards the reference trajectory. Due to this property a funnel can be used to drive all states into a goal region, see the application of funnels in chapter 2.

We have to embed the proofs of convergence with respect to time-dependent Lyapunov functions and polynomial systems into an algorithm that constructs these funnels. It is theoretically possible to treat time *just like* another dimension and develop conditions that proof stability for all $t \in [T_0, T_1]$. But this approach suffers from multiple drawbacks. Approximating the evolution of the dynamical system along a reference trajectory with respect to time is likely to cause larger errors as the change of state along the reference trajectory from T_0 to T_1 might be larger than the difference within Ω_t for a fixed t. Secondly, due to the increased dimension, the optimization problem is computationally more costly. Finally, in order to use the proposed state space partitioning, the evolution of the optimal control law $\boldsymbol{u}^*$ would also have to be approximated, leading to additional errors and increased maximal degree of the condition. Instead condition (3.63) is ensured on finitely many points distributed along the reference trajectory similar to the approaches in Tedrake et al. [2010b] and Majumdar et al. [2013b].

The iterative algorithm used in this work to construct the funnel for a given trajectory and dynamical system is given in Algorithm 2. This algorithm relies on two important sub-procedures: the retro-

Algorithm 2 Funnel construction via retro-propagation

Input $\Omega_T = \left\{ \boldsymbol{x} | \, \|\boldsymbol{x} - \boldsymbol{x}_T^r\|_{P_T} \leq \alpha_T \right\}$
Parameter $N_{steps} \in \mathbb{N}^+$, $N_{inter} \in \mathbb{N}^+$, $\gamma \in \mathbb{R}^+$
Output $\left(P^k, \alpha^k\right)_{0 \leq k \leq N_{steps}}$
$dT = \frac{T}{N_{steps}}$
$P^k \leftarrow P_T$
$\alpha^k \leftarrow \alpha_T$
$\boldsymbol{T} \leftarrow \textsc{linspace}(0, T, N_{steps})$
for k from N_{steps} to 1 by -1 **do**
    P^{k-1}, $\alpha^{k-1} \leftarrow \textsc{Retro-Prop}(P^k, \alpha^k, dT)$
    $\alpha^{k-1} \leftarrow \textsc{DichotomicSearch}(P^{k-1}, P^k, \alpha^k, \boldsymbol{T}[k-1], \boldsymbol{T}[k], N_{inter})$
end for
Return $\left(P^k, \alpha^k\right)_{0 \leq k \leq N_{steps}}$

propagation of a region Ω along a reference trajectory for a given system and the maximization of the volume by performing a dichotomic search.

The retro-propagation is a crucial step in the algorithm as it is the only step in which the shape of the Lyapunov function (the P_t) is modified. Therefore it is vital that the resulting shape is compatible with the dynamical system. The method used for the examples shown in section 3.10 is based on the linear quadratic regulator (LQR) and presented in detail in section 3.9.

In order to maximize the volume of Ω, we seek to maximize α^{k-1} at each step of the iteration. As the sufficient conditions for stabilizability presented above necessitate to fix Ω, a dichotomic search is used to quickly converge towards the largest admissible α^{k-1}. In contrast to the approach in Majumdar et al. [2013b], in which the stability condition is only check at each "base" point, we also check the stabilizability condition on N_{inter} intermediate points, equidistantly distributed between $\boldsymbol{T}[k-1]$ and $\boldsymbol{T}[k]$. This means the stabilizability condition checked within the procedure DichotomicSearch becomes

$$\begin{aligned} &\forall t_l \in \textsc{linspace}(T_0 = \boldsymbol{T}[k-1], T_1 = \boldsymbol{T}[k], N_{inter} + 2), \, \forall i : \\ &\qquad \max_{\boldsymbol{x} \in (\Omega_{t_l} \cap \mathcal{H}_i^l) \setminus \{\boldsymbol{0}\}} 2.\delta\boldsymbol{x}^{\mathrm{T}}.P_{t_l}.(f^l(\delta\boldsymbol{x}) + g^l(\delta\boldsymbol{x}).\boldsymbol{u}_i^{l*} - \dot{\boldsymbol{x}}_{t_l}^r) + \delta\boldsymbol{x}^{\mathrm{T}}.\dot{P}_{t_l}.\delta\boldsymbol{x} + \gamma.\delta\boldsymbol{x}^{\mathrm{T}}.P_{t_l}.\delta\boldsymbol{x} \leq 0 \end{aligned} \tag{3.64}$$

with $\delta\boldsymbol{x} = \boldsymbol{x} - \boldsymbol{x}_{t_l}^r$ and $f^l(.)$ and $g^l(.)$ being the truncated Taylor expansion of the nonlinear dynamical system at $\boldsymbol{x}_{t_l}^r$ as shown in Figure 3.8. The partitioning of the state space into the polytopes $\mathcal{H}_i^l$ and the corresponding optimal control input $\boldsymbol{u}_i^{l*}$ are derived based on P_{t_l} and linear input dynamics $B^l = g^{nl}(\boldsymbol{x}_{t_l}^r)$.

Special care has to be taken to properly interpolate between the start and end region as linear interpolation can result in undesired behaviour, as shown in section 3.9.4.

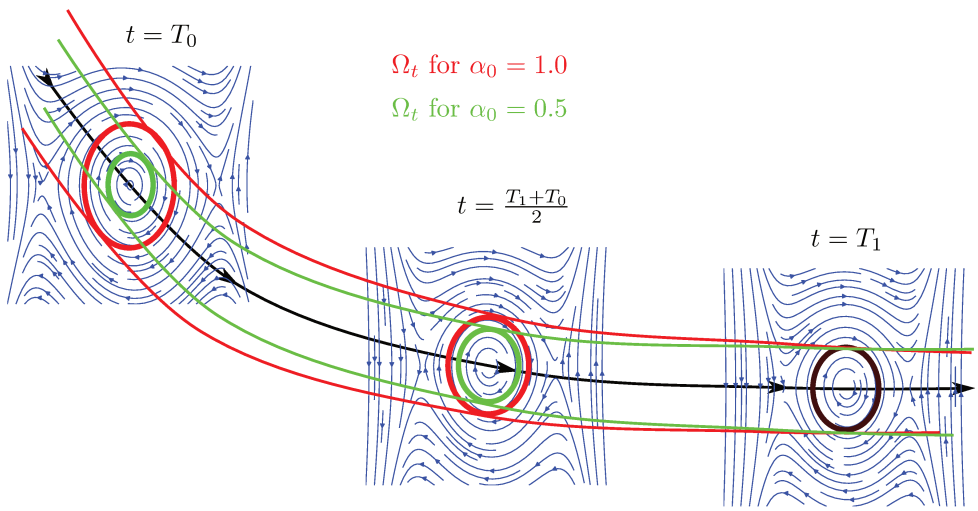


Figure 3.8 – Qualitative depiction of the dichotomic search and the checked criterion for $N_{inter} = 1$. The reference trajectory is depicted by the black line. With N_{inter} being one, the stabilizability criterion is checked at T_0, T_1 and $T_0{+}T_1/2$, for which the differential closed loop dynamics using the optimal control law is depicted by the streamlines. The final zone Ω_{T_1} is fixed, but modifying α_{T_0} scales the resulting funnel around the reference trajectory, as shown by the green and red ellipsoids.

3.8 Certificates for Non-Positiveness

3.8.1 Underestimators Based on Reformulation-Linearisation-Techniques

Up to now it was detailed how to partition the state space into input optimal subsets and derive an associated polynomial expression (3.47), whose non-positiveness on the respective subset is a partial proof of exponential stabilizability. In this section we show how to efficiently treat this problem using a modified version of the well-known Reformulation and Linearisation Technique (RLT) first introduced in Sherali and Adams [1990] and Sherali and Tuncbilek [1992] for zero-one programming problems, extended to continuous variables in Sherali and Tuncbilek [1995] and further enhanced with semidefinite constraints in Lovász and Schrijver [1991] and Sherali and Fraticelli [2002].

In order to informally introduce the basic idea behind RLT consider the general quadratic objective subjected to a set of linear constraints $\mathcal{C}_1$

$$\underset{\boldsymbol{x}}{\text{minimize}}\ \boldsymbol{x}^{\mathrm{T}}.Q.\boldsymbol{x} + L.\boldsymbol{x} \tag{3.65a}$$

$$\text{subject to } \boldsymbol{x} \models \mathcal{C}_1 \tag{3.65b}$$

with $Q^{\mathrm{T}} = Q$, $Q \in \mathbb{R}^{n\times n}$ and the decision variable $\boldsymbol{x} \in \mathbb{R}^n$. Recall that by $\boldsymbol{x} \models \mathcal{C}_1$ we denote that $\boldsymbol{x}$ has to satisfy some set of linear constraints and is therefore equivalent to $A.\boldsymbol{x} \leq \boldsymbol{b}$. This notation is very handy when dealing with general polynomial constraints of the form $\forall i\colon g_i(\boldsymbol{x}) \leq 0$ with the highest occurring degree being k, as they can be written $\boldsymbol{x} \models \mathcal{C}_{\leq k}$. Note that the symmetry of Q is for mere convenience and does not restrain the generality of the problem, as the value of the pure quadratic form $\boldsymbol{x}^{\mathrm{T}}.Q.\boldsymbol{x}$ only depends on the symmetric part of the matrix Q.

This problem is non-convex and NP-hard if $Q \notin \mathbb{S}_+$ (see Pardalos and Vavasis [1991]) and therefore cannot be efficiently solved. The Relaxation and Linearisation Technique copes with this problem by replacing all nonlinears terms in the optimization problem with new variables. This gives rise to a set of additional new variables, here denoted X and the objective function becomes linear in the set $\boldsymbol{x} \cup X$. This linearisation comes at the cost of the new linear objective being unbounded, due to the lack of constraints

on X. For quadratic programming this set of new variables has to replace all terms quadratic in $\boldsymbol{x}$ and can be conveniently written as a symmetric matrix variable X, where the new variable X_{ij} linearises the term $\boldsymbol{x}_i\boldsymbol{x}_j$. To ease notations we introduce the linearisation operator $\mapsto_{\text{Lin}}$, which replaces all nonlinear terms in the expression on the left hand side with their corresponding newly introduced variable, so

$$\boldsymbol{x}_i\boldsymbol{x}_j \mapsto_{\text{Lin}} X_{ij} \tag{3.66}$$

which can also be applied to vector or matrix expressions by applying it to each element, for instance

$$\begin{aligned}
&\boldsymbol{x}.\boldsymbol{x}^{\mathrm{T}} \mapsto_{\text{Lin}} X \\
&\boldsymbol{x}^{\mathrm{T}}.\boldsymbol{x} \mapsto_{\text{Lin}} \operatorname{tr}(X)\,.
\end{aligned}$$

In the running example, one therefore gets

$$\boldsymbol{x}^{\mathrm{T}}.Q.\boldsymbol{x} + L.\boldsymbol{x} \mapsto_{\text{Lin}} \operatorname{tr}(X.Q) + L.\boldsymbol{x}$$

and the problem becomes

$$\underset{\boldsymbol{x},X}{\text{minimize}}\ \operatorname{tr}(X.Q) + L.\boldsymbol{x} \tag{3.67a}$$

$$\text{subject to } \boldsymbol{x} \models \mathcal{C}_1 \tag{3.67b}$$

In order to ensure that the gap between the solution of the original and the linearised problem is small and remains conservative, in the sense that the linearised problem is an underestimator for the original problem, one has to construct valid constraints on X. This can be done by taking products of constraints on $\boldsymbol{x}$. For instance, reconsider the set of linear constraints $\mathcal{C}_1 = \cup_{k\in[0,K-1]} C_1^k$. By multiplying two constraints C_1^m, C_1^n one obtains a valid constraint with terms up to order 2 in $\boldsymbol{x}$, then again one can replace the nonlinear (quadratic in this case) terms by the corresponding variables in X and thereby a linear constraint in $\boldsymbol{x} \cup X$ is obtained. For short, $C_1^m \otimes C_1^n = C_2^{m,n} \mapsto_{\text{Lin}} \widetilde{C}_X^{m,n}$, where $\otimes$ denotes the product of constraints. For instance consider the following example

$$C_1^m = a\boldsymbol{x}_i + b\boldsymbol{x}_j \geq 0 \tag{3.68a}$$

$$C_1^n = \boldsymbol{x}_i - c \geq 0 \tag{3.68b}$$

then

$$C_1^m \otimes C_1^n = C_2^{m,n} = a\boldsymbol{x}_i^2 + b\boldsymbol{x}_i\boldsymbol{x}_j - ac\boldsymbol{x}_i - bc\boldsymbol{x}_j \geq 0 \tag{3.68c}$$

and finally

$$C_{\leq 2}^{m,n} \mapsto_{\text{Lin}} \widetilde{C}_{\leq X}^{m,n} = aX_{ii} + bX_{ij} - ac\boldsymbol{x}_i - bc\boldsymbol{x}_j \geq 0 \tag{3.68d}$$

with $\widetilde{C}_{\leq X}$ being linear in $\boldsymbol{x} \cup X$, as denoted by $_{\leq X}$.

By an abuse of notation, we will denote by $\otimes$ also the product of constraint sets, enumerating all possible combinations, so $\mathcal{C}_1 \otimes \mathcal{C}_1 = \cup_{i\in[0,K-1]} \cup_{j\in[i,K-1]} C_1^i \otimes C_1^j$. We denote by $\mathcal{C}_i$ a constraint set were all appearing monomials are of degree i or less and $\widetilde{\mathcal{C}}_{\leq X}$ denotes the corresponding linearised version, a set of linear constraints with variables in $\boldsymbol{x} \cup X$. Forming such product constraints ensures that the new variables are bounded while guaranteeing that the linearised objective function remains an underestimator of the original objective function, since the set of admissible values in $\boldsymbol{x} \cup X$ is strictly larger than the set of admissible values of the original problem[2]. This corresponds to the standard RLT and was originally proposed in Sherali and Adams [1990].

In Lovász and Schrijver [1991] an improvement to the standard approach is given by adding semidefinite constraints, reducing significantly the gap between the solution of the relaxed and original problem.

[2]For more details see Sherali and Tuncbilek [1995].

To introduce these constraints reconsider the nonconvex part of the objective function $\boldsymbol{x}^{\mathrm{T}}.Q.\boldsymbol{x}$ and its linearisation $\mathrm{tr}(X.Q)$. It is clear that the minimum of the original objective is obtained if $X = \boldsymbol{x}.\boldsymbol{x}^{\mathrm{T}}$ holds, as the problem

$$\begin{aligned}
&\underset{\boldsymbol{x},X}{\text{minimize}}\ \mathrm{tr}\,(X.Q) + L.\boldsymbol{x} && \text{(3.69a)}\\
&\text{subject to } \boldsymbol{x} \models \mathcal{C}_1 && \text{(3.69b)}\\
&\qquad\qquad X = \boldsymbol{x}.\boldsymbol{x}^{\mathrm{T}} && \text{(3.69c)}
\end{aligned}$$

is equivalent to the the original problem (3.65). The only difference is, that the non-convexity of the objective function was translated into the non-convex constraint (3.69c). So as we cannot impose this constraint in a convex program, suitable relaxations have to be found. And indeed by transforming the problem into an SDP, we can impose the constraint $X \succeq \boldsymbol{x}.\boldsymbol{x}^{\mathrm{T}}$ as an linear matrix inequality (LMI) via the Schur-complement by adding the constraint

$$\begin{bmatrix} 1 & \boldsymbol{x}^{\mathrm{T}} \\ \boldsymbol{x} & X \end{bmatrix} \succeq \mathbf{0}$$

which is amenable to semidefinite programming. The optimisation resulting from applying the enhanced RLT on (3.65) is given as

$$\begin{aligned}
&\underset{\boldsymbol{x},X}{\text{minimize}}\ \mathrm{tr}(X.Q) + L.\boldsymbol{x}\\
&\text{subject to } \boldsymbol{x} \models \mathcal{C}_1\\
&\qquad\qquad \boldsymbol{x}, X \models \widetilde{\mathcal{C}}_{\leq X}\\
&\qquad\qquad \begin{bmatrix} 1 & \boldsymbol{x}^{\mathrm{T}} \\ \boldsymbol{x} & X \end{bmatrix} \succeq \mathbf{0}
\end{aligned}$$

where $\widetilde{\mathcal{C}}_{\leq X}$ denotes the linearisation of the set of constraints $\mathcal{C}_1 \otimes \mathcal{C}_1 = \mathcal{C}_2 \mapsto_{\mathrm{Lin}} \widetilde{\mathcal{C}}_{\leq X}$.

This relaxation also conserves the property that the linearised objective function is an underestimator for the original problem as the set of admissible values is strictly enlarged, as $\left\{X | X = \boldsymbol{x}.\boldsymbol{x}^{\mathrm{T}}\right\} \subset \left\{X | X \succeq \boldsymbol{x}.\boldsymbol{x}^{\mathrm{T}}\right\}$.

This approach was successfully applied to nonconvex quadratic optimisation problems (Kezurer et al. [2015]) or within branch and bound algorithms for general nonlinear programming (Sherali and Tuncbilek [1992]) and is further demonstrated in the following example.

Consider the following unidimensional problem

$$\begin{aligned}
&\underset{x}{\text{minimize}}\quad -x^2 + x && \text{(3.70a)}\\
&\text{subject to}\quad x + 1 \geq 0 && \text{(3.70b)}\\
&\qquad\qquad -x + 1 \geq 0 && \text{(3.70c)}
\end{aligned}$$

which is nonconvex. The set of linear constraints $\mathcal{C}_1$ is $\{x + 1 \geq 0, -x + 1 \geq 0\}$. By introducing the new variable X linearising x^2, we can compute the set of linearised product constraints $\widetilde{\mathcal{C}}_{\leq X}$ to be $\{X + 2x + 1 \geq 0,\ -X - 2x + 1 \geq 0,\ X - 2x + 1 \geq 0\}$. Moreover, as we have $-1 \leq x \leq 1$, we also obtain the "natural" constraints $0 \leq X$ and $X \leq 1$, which can be added to $\widetilde{\mathcal{C}}_{\leq X}$. So the linearised SDP reads

$$\begin{aligned}
&\underset{x}{\text{minimize}}\quad -X + x && \text{(3.71a)}\\
&\text{subject to}\quad x \models \mathcal{C}_1 && \text{(3.71b)}\\
&\qquad\qquad x \cup X \models \widetilde{\mathcal{C}}_{\leq X} && \text{(3.71c)}\\
&\qquad\qquad X \succeq \boldsymbol{x}.\boldsymbol{x}^{\mathrm{T}}. && \text{(3.71d)}
\end{aligned}$$

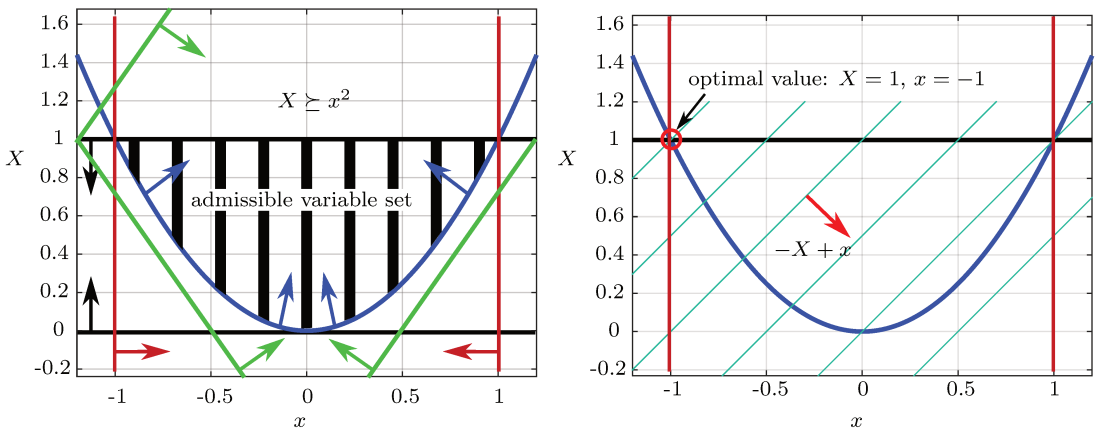


Figure 3.9 – On the left the admissible set is depicted. The original constraints on x, $\mathcal{C}_1$ are shown in red. The product constraints are shown in green, whereas black corresponds to the natural constraint. Finally, in blue, the LMI constraint $X \succeq x^2$. Note that in this case the product constraints are redundant, as the LMI-constraint and the natural constraint would be sufficient. This however is not true in general. On the right the gradient of the objective functions and some level-sets are depicted as well as the minimizing variables.

The constraint set and the sublevel-set of the objective function are shown in Figure 3.9. As one can see, the minimizer of the linearised problem is $X = 1$ and $x = -1$. As in this case the inequality $X \succeq x^2$ holds strictly, this also corresponds to true optimum and x is a minimizer for the non-convex problem. In this case there is no gap between the convex underestimator and the original problem. Note that another, yet equivalent way, to test whether there is no gap is to check the rank of the LMI. If

$$\operatorname{rank}\left(\begin{bmatrix} 1 & \boldsymbol{x}^{\mathrm{T}} \\ \boldsymbol{x} & X \end{bmatrix}\right) = 1 \tag{3.72}$$

holds, then there is no gap.

3.8.2 Reformulation-Linearisation-Techniques for Polynomial Programming

Below, we apply our method to polynomial dynamics of degree 3, and therefore the polynomial expressions in (3.47) are of degree 4. So, the above presented enhanced RLT has to be modified in order to treat such polynomials. As seen above, a new set of variables, denoted X, is created to linearise quadratic terms in $\boldsymbol{x}$ which we write as $\boldsymbol{x}.\boldsymbol{x}^{\mathrm{T}} \mapsto_{\mathrm{Lin}} X$. We denote $\boldsymbol{z}$ the column vector composed of all quadratic terms in $\boldsymbol{x}$ and $\operatorname{vec}(X)$ the column vector where the upper triangle of X is stored, for $\boldsymbol{x} \in \mathbb{R}^n$

$$\operatorname{vec}(X) = \begin{pmatrix} X[0,:] & X[1,1:] & \cdots & X[n-2,n-2:] & X[n-1,n-1] \end{pmatrix}^{\mathrm{T}}. \tag{3.73}$$

So we have $\boldsymbol{z} \mapsto_{\mathrm{Lin}} \operatorname{vec}(X)$ and one can construct the following matrix

$$\begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{z} \end{pmatrix} \cdot \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{z} \end{pmatrix}^{\mathrm{T}} \mapsto_{\mathrm{Lin}} \begin{pmatrix} \boldsymbol{x} \\ \operatorname{vec}(X) \end{pmatrix} \cdot \begin{pmatrix} \boldsymbol{x} \\ \operatorname{vec}(X) \end{pmatrix}^{\mathrm{T}} \mapsto_{\mathrm{Lin}} \begin{bmatrix} X & Y \\ Y^{\mathrm{T}} & Z \end{bmatrix}$$

where the symmetric matrix X linearises all quadratic terms, $\boldsymbol{x}.\operatorname{vec}(X)^{\mathrm{T}} \mapsto_{\mathrm{Lin}} Y$ linearises all cubic terms and $\operatorname{vec}(X).\operatorname{vec}(X)^{\mathrm{T}} \mapsto_{\mathrm{Lin}} Z$ linearises all quadratic terms in X and therefore corresponds to quartic terms

in $\boldsymbol{x}$. Again, imposing the constraint

$$\begin{bmatrix} X & Y \\ Y^{\mathrm{T}} & Z \end{bmatrix} = \begin{pmatrix} \boldsymbol{x} \\ \mathrm{vec}(X) \end{pmatrix} \cdot \begin{pmatrix} \boldsymbol{x} \\ \mathrm{vec}(X) \end{pmatrix}^{\mathrm{T}}$$

is not impossible due to its non-convexity and is therefore relaxed to

$$\begin{bmatrix} X & Y \\ Y^{\mathrm{T}} & Z \end{bmatrix} \succeq \begin{pmatrix} \boldsymbol{x} \\ \mathrm{vec}(X) \end{pmatrix} \cdot \begin{pmatrix} \boldsymbol{x} \\ \mathrm{vec}(X) \end{pmatrix}^{\mathrm{T}}$$

Finally, by applying the Schur-complement a valid LMI-constraint is generated for the linearisation of all monomials of degree up to 4

$$\begin{bmatrix} 1 & \boldsymbol{x}^{\mathrm{T}} & \mathrm{vec}(X)^{\mathrm{T}} \\ \boldsymbol{x} & X & Y \\ \mathrm{vec}(X) & Y^{\mathrm{T}} & Z \end{bmatrix} \succeq \mathbf{0} \ . \tag{3.74}$$

Note that not all elements in Y or Z are unique, as for instance the linearisation of $\boldsymbol{x}_i X_{jj}$ is equivalent to the linearisation of $\boldsymbol{x}_j X_{ij}$ and that Z is symmetric.

To further illustrate the approach, consider the case $\boldsymbol{x} \in \mathbb{R}^2$. Then we have

$$\boldsymbol{x}.\boldsymbol{x}^{\mathrm{T}} = \begin{bmatrix} \boldsymbol{x}_0^2 & \boldsymbol{x}_0\boldsymbol{x}_1 \\ \boldsymbol{x}_1\boldsymbol{x}_0 & \boldsymbol{x}_1^2 \end{bmatrix} \mapsto_{\mathrm{Lin}} X = \begin{bmatrix} X_{00} & X_{01} \\ X_{01} & X_{11} \end{bmatrix} \tag{3.75a}$$

$$\mathrm{vec}(X) = \begin{pmatrix} X_{00} & X_{01} & X_{11} \end{pmatrix}^{\mathrm{T}} \tag{3.75b}$$

$$\boldsymbol{x}.\mathrm{vec}(X)^{\mathrm{T}} = \begin{bmatrix} \boldsymbol{x}_0 X_{00} & \boldsymbol{x}_0 X_{01} & \boldsymbol{x}_0 X_{11} \\ \boldsymbol{x}_1 X_{00} & \boldsymbol{x}_1 X_{01} & \boldsymbol{x}_1 X_{11} \end{bmatrix} \mapsto_{\mathrm{Lin}} Y = \begin{bmatrix} Y_{00} & Y_{01} & Y_{02} \\ Y_{01} & Y_{02} & Y_{13} \end{bmatrix} \tag{3.75c}$$

$$\mathrm{vec}(X).\mathrm{vec}(X)^{\mathrm{T}} = \begin{bmatrix} X_{00}X_{00} & X_{00}X_{01} & X_{00}X_{11} \\ X_{01}X_{00} & X_{01}X_{01} & X_{01}X_{11} \\ X_{11}X_{00} & X_{11}X_{01} & X_{11}X_{11} \end{bmatrix} \mapsto_{\mathrm{Lin}} \begin{bmatrix} Z_{00} & Z_{01} & Z_{02} \\ Z_{01} & Z_{02} & Z_{12} \\ Z_{02} & Z_{12} & Z_{22} \end{bmatrix} \tag{3.75d}$$

In order to ensure that the linearised objective gives tight bounds, valid constraints on X, Y and Z have to be constructed based on the original constraint sets $\mathcal{C}_1$, $\mathcal{C}_2$, $\mathcal{C}_3$ and $\mathcal{C}_4$. Valid constraints can in this case be obtained if the degree of the resulting constraint is less or equal to 4. Some examples of valid constraint sets are $\mathcal{C}_1 \otimes \mathcal{C}_1 \otimes \mathcal{C}_1 = \mathcal{C}_{\leq 3} \mapsto_{\mathrm{Lin}} \widetilde{\mathcal{C}}_{\leq Y}$, $\mathcal{C}_1 \otimes \mathcal{C}_1 \otimes \mathcal{C}_{\leq 2} = \mathcal{C}_{\leq 4} \mapsto_{\mathrm{Lin}} \widetilde{\mathcal{C}}_{\leq Z}$, where $\widetilde{\mathcal{C}}_{\leq Y}/\widetilde{\mathcal{C}}_{\leq Z}$ is a set of linear constraints in $\boldsymbol{x} \cup X \cup Y/\boldsymbol{x} \cup X \cup Y \cup Z$. Using this approach, which is inspired by the ideas developed in Sherali et al. [2012], we can underestimate the original objective function of degree 4

$$\underset{\boldsymbol{x}}{\text{minimize}} \ \begin{pmatrix} \boldsymbol{x}^{\mathrm{T}} & \boldsymbol{z}^{\mathrm{T}} \end{pmatrix} .Q. \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{z} \end{pmatrix} + L.\boldsymbol{x} \tag{3.76a}$$

$$\text{subject to } \boldsymbol{x} \models \mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3, \mathcal{C}_4 \tag{3.76b}$$

with

$$\underset{\boldsymbol{x},X,Y,Z}{\text{minimize}} \ \mathrm{tr}\left(\begin{bmatrix} X & Y \\ Y^{\mathrm{T}} & Z \end{bmatrix} .Q \right) + L.\boldsymbol{x} \tag{3.77a}$$

$$\text{subject to } \boldsymbol{x} \models \mathcal{C}_1 \tag{3.77b}$$

$$\boldsymbol{x},\ X,\ Y,\ Z \models \widetilde{\mathcal{C}}_{\leq X},\ \widetilde{\mathcal{C}}_{\leq Y},\ \widetilde{\mathcal{C}}_{\leq Z} \tag{3.77c}$$

$$\begin{bmatrix} 1 & \boldsymbol{x}^{\mathrm{T}} & \mathrm{vec}(X)^{\mathrm{T}} \\ \boldsymbol{x} & X & Y \\ \mathrm{vec}(X) & Y^{\mathrm{T}} & Z \end{bmatrix} \succeq \mathbf{0} \tag{3.77d}$$

where $\widetilde{\mathcal{C}}_{\leq X}$, $\widetilde{\mathcal{C}}_{\leq Y}$ and $\widetilde{\mathcal{C}}_{\leq Z}$ are the sets of all obtainable constraints constructed via multiplication of $\mathcal{C}_{\leq 1}$, $\mathcal{C}_{\leq 2}$ and $\mathcal{C}_{\leq 3}$ plus the linearisation of $\mathcal{C}_2$, $\mathcal{C}_3$, $\mathcal{C}_4$, the naturally arising constraints on the new variables and their respective products.

Remark 3.6. The number of variables is this optimisation problem is equal to the number of monomials of degree up to 4 in $\boldsymbol{x}$. The approach presented in Majumdar et al. [2013b], or in general all approaches relying on introducing a polynomial control law, need a similar number of variables to represent just one component of the polynomial control law (depending on the degree of the control law) plus the variables necessary to represent the multiplier functions which have to be of superior degree in general. Therefore the number of decision variables and the size of the LMI constraints are significantly larger. Moreover it is not clearly stated in [Majumdar et al., 2013b] whether polynomial or linear input dynamics are used within stability criterion. Using polynomial input dynamics is expensive for SoS-based techniques as the degree of the input dynamics and the controller add up, significantly increasing the complexity of the optimization problem. For our case on the other hand, polynomial input dynamics *only* introduce a usually small sub-optimality of the control law. Therefore this only increases conservativeness, but not the computational complexity.

Remark 3.7. The number of linear constraints in the sets $\mathcal{C}_{\leq 1/\leq 2/\leq 3/\leq 4}$ and (even more so) $\widetilde{\mathcal{C}}_{\leq X/\leq Y/\leq Z}$ grows rapidly with the size of $\boldsymbol{x}$. However linear constraints are computationally cheap and additionally there exist methods to limit number of constraints. These methods are based on the redundancy often occurring, when taking all products possible between the constraint sets.

Remark 3.8. The above presented approach to construct enhanced RLT representation of nonconvex polynomial problems is by no means confined to polynomials of degree 4. By reapplying the method above k times one can linearise optimisation problems containing monomials of degree up to 2^k. Again, all the new variables have to be bounded forming the natural and product constraints.

Remark 3.9. The decision variables in the resulting semidefinite program are associated with remarkably sparse matrices in the LMI constraints. For instance, the sdp matrices associated to the decision variables in the LMI constraint (3.77d) for $\boldsymbol{x} \in \mathbb{R}^2$ are of size 5×5 but have at most 4 non-zero entries. This structure can probably be exploited to solve the problem even faster with a tailored solver.

3.8.3 Application

In this section it is shown how to apply the above introduced method for proving stabilizability and some implementation details are discussed.

RLT for Proving Stabilizabilty

Above is shown how to construct a linear underestimator in a higher dimensional space ($\boldsymbol{x} \cup X \cup Y \cup Z$) for a (nonconvex) polynomial function of order 4 in $\boldsymbol{x}$ and how to construct constraint sets for the new variables ($X \cup Y \cup Z$) such that the underestimation usually results in reasonably tight bounds. Now we will shortly discuss how to apply this approach to prove stabilizability of a controlled system.

According to (3.47) we can prove the stabilizability of a polynomial control affine system on a sublevel-set Ω of a quadratic Lyapunov function candidate $V(\boldsymbol{x}) = \boldsymbol{x}^{\mathrm{T}}.P.\boldsymbol{x}$ by assuring the negativity of each of the 2^m (nonconvex) terms $2\boldsymbol{x}^{\mathrm{T}}.P.(f(\boldsymbol{x}) + B.\boldsymbol{u}_i^*)$ on $\Omega \cap \mathcal{H}_i \setminus \{\boldsymbol{0}\}$. For each $i \in [0, 2^m - 1]$, this can be written as

$$\underset{\boldsymbol{x}}{\text{minimize}} \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{z} \end{pmatrix}^{\mathrm{T}}.Q_i.\begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{z} \end{pmatrix} + L_i.\begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{z} \end{pmatrix} \tag{3.78a}$$

$$\text{subject to } \mathcal{C}_1 = \left\{\cup_{j\in[0,\,m-1]} c_j^i \boldsymbol{x}^{\mathrm{T}}.\boldsymbol{n}_j \leq 0\right\} \tag{3.78b}$$

$$\mathcal{C}_2 = \left\{\boldsymbol{x}^{\mathrm{T}}.P.\boldsymbol{x} \leq \alpha\right\} \tag{3.78c}$$

where $Q_i[i,j]$ and $L_i[j]$ can be uniquely identified with the terms of $2\boldsymbol{x}^{\mathrm{T}}.P.(f(\boldsymbol{x})+B.\boldsymbol{u}_i^*)+\boldsymbol{x}^{\mathrm{T}}.(\gamma P+\dot{P}).\boldsymbol{x}$ with $f(\boldsymbol{x}) \approx f^{nl}(\boldsymbol{x})$ being the truncated Taylor expansion of the differential nonlinear system dynamics. $\mathcal{C}_2$ represents the quadratic constraint confining $\boldsymbol{x}$ to the sublevel-set Ω and $\mathcal{C}_1$ is the set of linear constraints restricting $\boldsymbol{x}$ to the i-th polytope of the optimal input partitioning. This shows how to bring the

resulting optimal convergence for each of the 2^m input optimal subsets into the form of (3.76). If the minimal objective value of the optimization problem (3.78) is larger than 0 for each subset $\mathcal{H}_i \cap \Omega$, then Ω is exponentially stabilisable.

In order to reduce the difference between the system used within the proof and the full nonlinear dynamical system, we proposed to derive the optimal input and the partitioning with respect to the linear input dynamics, but use the truncated Taylor expansion of the input dynamics with the stabilizability condition (3.63). This is straight forward by identifying Q_i and L_i with the terms of $2\boldsymbol{x}^{\mathrm{T}}.P.(f(\boldsymbol{x}) + g(\boldsymbol{x}).\boldsymbol{u}_i^*) + \boldsymbol{x}^{\mathrm{T}}.(\gamma P + \dot{P}).\boldsymbol{x}$ with $g(\boldsymbol{x}) \approx g^{nl}(\boldsymbol{x})$ being the truncated Taylor expansion of the nonlinear input system dynamics.

Implementation with Bounding Box and Hypersphere

Even though the proposed enhanced RLT approach can be directly applied to solve the optimization problem (3.78), this formulation does not provide very tight bounds. Indeed, the *conditioning* of the problem, meaning the ratio between largest absolute value of each decision variable, can have a significant influence on the size of the gap. It turns out that these methods work best when the problem is bounded such that the conditioning approaches 1, which is naturally the case for the original use cases of 0-1-programming.

Therefore instead of solving directly (3.78), a linear coordinate transformation taking the ellipsoidal region Ω to the unit hypersphere defined as

$$C = \mathrm{chol}\left(\frac{P}{\alpha}\right) \tag{3.79a}$$

$$\hat{\boldsymbol{x}} = C.\boldsymbol{x}. \tag{3.79b}$$

is applied. Then, by substituting $\boldsymbol{x}$ with $C^{-1}.\hat{\boldsymbol{x}}$ in the polynomial expressions $f(\boldsymbol{x})$ and $g(\boldsymbol{x})$, the optimization problem becomes

$$\underset{\hat{\boldsymbol{x}}}{\text{minimize}} \begin{pmatrix}\hat{\boldsymbol{x}}\\ \hat{\boldsymbol{z}}\end{pmatrix}^{\mathrm{T}}.\hat{Q}.\begin{pmatrix}\hat{\boldsymbol{x}}\\ \hat{\boldsymbol{z}}\end{pmatrix} + \hat{L}_i.\begin{pmatrix}\hat{\boldsymbol{x}}\\ \hat{\boldsymbol{z}}\end{pmatrix} \tag{3.80a}$$

$$\text{subject to } \mathcal{C}_1 = \left\{\cup_{j\in[0,\,m-1]} c_j^i \hat{\boldsymbol{x}}^{\mathrm{T}}.\hat{\boldsymbol{n}}_j \leq 0\right\} \tag{3.80b}$$

$$\mathcal{C}_2 = \left\{\hat{\boldsymbol{x}}^{\mathrm{T}}.Id.\hat{\boldsymbol{x}} \leq 1\right\}. \tag{3.80c}$$

Now we can apply the enhanced RLT approach obtaining the linearised problem (3.77) in $\hat{\boldsymbol{x}}$-variables and take a closer look at the construction of admissible constraints. The technique is demonstrated for a two-dimensional problem with one input, but it is valid for any number of dimensions.

The set of linear constraints $\mathcal{C}_1$ of the original problem consists only of the constraints defining the polytope $\mathcal{H}_i$. But as we know that the problem is bounded to the unit hypersphere independently of considered dynamics or polytope, additional constraints can be constructed, some of them possibly redundant, see Figure 3.10.

For the equivalent problem on the hypersphere, the natural linear constraints of the form

$$\forall i\colon\ \hat{\boldsymbol{x}}_i \geq -1 \tag{3.81a}$$

$$\forall i\colon\ -\hat{\boldsymbol{x}}_i \geq -1 \tag{3.81b}$$

are obviously admissible and can be added to $\mathcal{C}_1$. In fact, valid linear constraints on $\hat{x}$ can be deduced from every hyperplane tangent to the unit hypersphere. The problem is, that there are infinitely many of them and there is no generic way to select a suitable finite subset of them, given the concrete problem.

In the next step, we can search for natural constraints for the new variables in X, Y and Z. As, due to $\hat{\boldsymbol{x}}^{\mathrm{T}}.\hat{\boldsymbol{x}} \leq 1$, all monomials in $\hat{\boldsymbol{x}}$, denoted $\boldsymbol{m}_n^4$, have to be bounded too and so is their respective

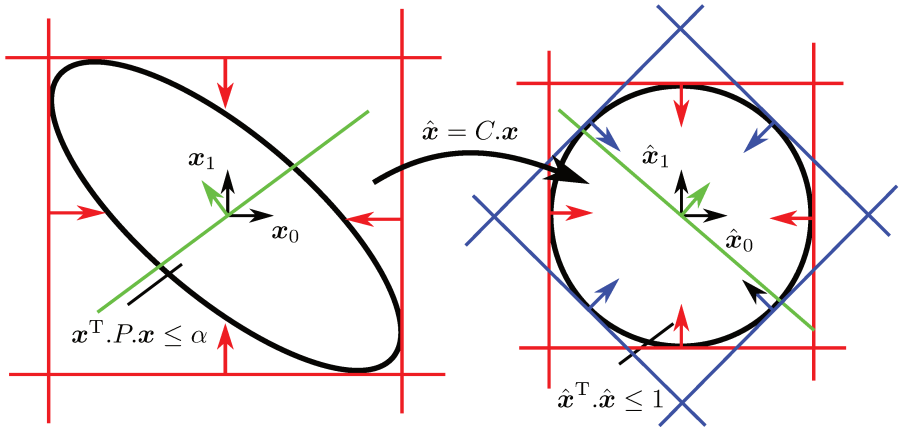


Figure 3.10 – On both images, the region Ω is indicated by the black ellipsoid or circle. The constraint defining the polytope is shown in green. The natural constraints defined by the minimal and maximal values for $\boldsymbol{x}_i$ or $\hat{\boldsymbol{x}}_i$ within Ω are shown in red. The admissible constraints for arbitrary weighted sum of $\hat{\boldsymbol{x}}_0$, $\hat{\boldsymbol{x}}_1$, so $\boldsymbol{\beta}_0\hat{\boldsymbol{x}}_0 + \boldsymbol{\beta}_1\hat{\boldsymbol{x}}_1 \leq c$ for some $\boldsymbol{\beta}_0$, $\boldsymbol{\beta}_1 \in \mathbb{R}^n$ and $c \geq 0$ can be computed, but as there exist infinitely many of them and there is no systematic way to choose from them, they are not used. The generally larger gap for the original problem, the corresponding bounding constraints are shown on the left, is possibly linked to the fact that Ω covers a smaller portion of the natural bounding box.

linearisation. By denoting the minimal and maximal value of the monomial $\boldsymbol{m}_n^4[k]$ attainable in the unit hypersphere $\boldsymbol{m}_n^{4,-}[k]/\boldsymbol{m}_n^{4,+}[k]$, we can add the constraints

$$\boldsymbol{m}_n^4[k] \leq \boldsymbol{m}_n^{4,+}[k] \mapsto_{\mathrm{Lin}} v \leq \boldsymbol{m}_n^{4,+}[k] \tag{3.82a}$$

$$\boldsymbol{m}_n^4[k] \geq \boldsymbol{m}_n^{4,-}[k] \mapsto_{\mathrm{Lin}} v \geq \boldsymbol{m}_n^{4,-}[k] \tag{3.82b}$$

$$\tag{3.82c}$$

for some variable v in X,Y or Z to the respective constraint set $\widetilde{\mathcal{C}}_{\leq X}$, $\widetilde{\mathcal{C}}_{\leq Y}$ or $\widetilde{\mathcal{C}}_{\leq Z}$. Now all natural constraints on the linearised problem have been constructed, but before proceeding to form all admissible product constraints between $\mathcal{C}1$, $\widetilde{\mathcal{C}}_{\leq X}$ and $\widetilde{\mathcal{C}}_{\leq Y}$, the relaxation of the unit hypersphere condition $\hat{\boldsymbol{x}}^{\mathrm{T}}.\hat{\boldsymbol{x}} \leq 1 \mapsto_{\mathrm{Lin}} \mathrm{tr}\,(X) \leq 1$ has to be added to $\widetilde{\mathcal{C}}_{\leq X}$.

3.8.4 Connections to the Theory of Moments

The theory of moments is so to speak the dual problem of the non-negativity of polynomials on a compact, semi-algebraic set K. In this section the theory of moments is briefly revisited with a focus on polynomial optimization. For a detailed introduction into this topic see, among others, Henrion and Garulli [2005], Lasserre [2001] and Lasserre [2009].

Consider the problem of finding the minimum of a real-valued polynomial $p(\boldsymbol{x})\colon \mathbb{R}^n \to \mathbb{R}$ on a compact, semialgebraic set K, so a set defined by polynomial constraints $K = \{\boldsymbol{x} | \forall i\colon\ g_i(\boldsymbol{x}) \geq 0\}$:

$$p^* = \min_{\boldsymbol{x} \in K} p(\boldsymbol{x}). \tag{3.83}$$

One can show that this is equivalent to the dual problem

$$p^* = \min_{\mu \in \mathcal{P}(K)} \int p(\boldsymbol{x})\mu(\boldsymbol{x}) \tag{3.84}$$

with $\mathcal{P}(K)$ being the space of finite Borel measures defined on K. This problem as such is not tractable as the space of Borel measures is infinite, however if the polynomial is of degree m in n variables, then the criterion becomes linear $\boldsymbol{a}^{\mathrm{T}}.\boldsymbol{y}$ on the finite collection of moments $\{\boldsymbol{y}_{\boldsymbol{\alpha}}\}$, up to order m of the probability measure μ (Lasserre [2001]) defined as

$$\boldsymbol{y}_{\boldsymbol{\alpha}} = \int \boldsymbol{x}^{\boldsymbol{\alpha}} d\mu. \tag{3.85}$$

So the optimization problem is transformed into a problem about the variables $\boldsymbol{y}_\alpha$ and how to constrain the support of μ to K via suitable constraints on the $\boldsymbol{y}_\alpha$ in order to reduce the gap between the original problem and its relaxation using the finite moment series.

In Lasserre [2001] it is shown that this can be done using LMIs: a sufficient but not necessary condition that the $\{\boldsymbol{y}_{\boldsymbol{\alpha}}\}$ are the moments of a measure is that the so-called moment matrix is psd. This matrix, even though derived very differently, has the same structure as the matrix used as LMI constraint (3.74) in the enhanced RLT. The moment matrix constructed from the sequence of moments $\boldsymbol{y}$ for the multivariate polynomial of degree $2k$, will be denoted $M_k(\boldsymbol{y})$. So every new variable in $X \cup Y \cup Z$ can be uniquely identified with a corresponding moment $\boldsymbol{y}_{\boldsymbol{\alpha}}$. Moreover, as we know which variable linearises which monomial, so for instance the monomial $\boldsymbol{x}_0\boldsymbol{x}_1 = \boldsymbol{x}^{\boldsymbol{\alpha}}$ with $\boldsymbol{\alpha} = \begin{bmatrix}11\end{bmatrix}$ is linearised by the variable X_{01} in enhanced RLT and identified with the moment $\boldsymbol{y}_{11}$ and we have

$$M_2(\boldsymbol{y}) = \begin{bmatrix} 1 & \boldsymbol{x}^{\mathrm{T}} & \mathrm{vec}(X)^{\mathrm{T}} \\ \boldsymbol{x} & X & Y \\ \mathrm{vec}(X) & Y^{\mathrm{T}} & Z \end{bmatrix}. \tag{3.86}$$

If the moment matrix is psd, the moments correspond to an actual measure, but it is not ensured that this measure is finite. This corresponds to the situation in RLT were the new variables have been introduced, but no admissible constraints to bound them have been added to the optimization problem. In Lasserre [2001] it is shown that this too can be done via LMI's. Consider the constraint $g(\boldsymbol{x}) \geq 0$ with degree $2l$. This can be enforced by constructing a "constraint moment matrix" $M_{k-l}(g\boldsymbol{y})$ (see Lasserre [2001] for details) and impose $M_{k-l}(g\boldsymbol{y}) \succeq 0$ as an additional LMI constraint to the optimization problem.

The main result of Lasserre [2001] is, that by increasing the maximal degree $2k$ represented in the moment and constraint moment matrices $M_k(\boldsymbol{y})$ and $M_{k-l}(g\boldsymbol{y})$, the gap between the relaxation and the original problem vanishes. This is always the case when allowing $k \to \infty$, but is often the case for finite values of k.

This provides a different approach to solving the resulting non-convex polynomial optimization arising in the stabilizability proofs. The larger amount of linear variables is replaced by a smaller number of LMI constraints, however it is not clear if the method proposed by Lasserre achieves tighter bounds for a *reasonable* order of the relaxation. To further investigate and potentially improve the relaxation of Lasserre by adding the *most significant* linear constraints obtained by RLT is a possible avenue for future work.

3.9 Computing and Propagating Suitable Lyapunov Functions

In section 3.5 it was shown how state-space partitioning and convexification can be used to obtain certificates of stabilizability for polynomial dynamics and a quadratic Lyapunov function candidate $V(\boldsymbol{x})$ on a given sublevel-set $\Omega = \{\boldsymbol{x} | V(\boldsymbol{x}) = \|\boldsymbol{x}\|_P \leq \alpha\}$. However it was neither discussed how to obtain a suitable Lyapunov candidate function nor how to propagate it along a given reference trajectory for a given dynamical system with input constraints. In contrast to the works cited in section 3.4 we do not seek to modify the shape of the Lyapunov function candidate due to the reasons mentioned beforehand, but rely on the similarity between the behaviour of the original system and the behaviour of its linearisation in

the neighbourhood of a stabilizable reference point. The approach presented here, like the one presented in Tedrake et al. [2010b], is based on LQR techniques and time-dependent linearization, differs however in certain important points, like the scaling of the feedback gain matrix.

3.9.1 Time-Dependent Linearisation

Reconsider the nonlinear system dynamics $\dot{\boldsymbol{x}} = f(\boldsymbol{x}) + g(\boldsymbol{x}).\boldsymbol{u}$ and the reference trajectory $(\boldsymbol{x}^r(t), \boldsymbol{u}^r(t))$ defined for $t \in [T_0, T_1]$. By definition the reference trajectory satisfies[3]

$$\begin{aligned} \forall t \in [T_0, T_1] : \frac{\mathrm{d}}{\mathrm{d}\,t}\boldsymbol{x}_t^r &= f(\boldsymbol{x}_t^r) + g(\boldsymbol{x}_t^r).\boldsymbol{u}_t^r \\ \boldsymbol{u}^- + \epsilon \leq \boldsymbol{u}_t^r &\leq \boldsymbol{u}^+ - \epsilon \end{aligned}$$

for some predefined control margin $\epsilon > 0$ ensuring that each point on the reference trajectory is an equilibrium point for the system in the deviation variables $\delta_{\boldsymbol{x}}$ and $\delta_{\boldsymbol{u}}$

$$\dot{\delta}_{\boldsymbol{x}} = f(\boldsymbol{x}_t^r + \delta_{\boldsymbol{x}}) + g(\boldsymbol{x}_t^r + \delta_{\boldsymbol{x}}).(\boldsymbol{u}_t^r + \delta_{\boldsymbol{u}}) \ . \tag{3.87}$$

Moreover, as $\epsilon > 0$ provides a margin between the true input constraints and the reference control input $\boldsymbol{u}^r$, there exists a stabilizable neighbourhood for each point on the reference trajectory.

We can then define the time-dependent (Jacobian) linearisation as

$$\dot{\delta}_{\boldsymbol{x}} = A_t.\delta_{\boldsymbol{x}} + B_t.\delta_{\boldsymbol{u}} \tag{3.88}$$

with

$$\begin{aligned} A_t &= \frac{\partial}{\partial \boldsymbol{x}} f(\boldsymbol{x}_t^r) \\ B_t &= g(\boldsymbol{x}_t^r) \ . \end{aligned}$$

Next we have to consider the control input. Consider the time-dependent saturated linear feedback control law $\tilde{K}(t, \boldsymbol{x}) : [T_0, T_1] \times \mathbb{R}^n \mapsto \mathbb{R}^m$ defined as

$$\tilde{K}(t, \boldsymbol{x})[i] = \begin{cases} \boldsymbol{u}^+[i], & \text{if } \boldsymbol{u}_t^r[i] - K_t[i,:].\boldsymbol{x} > \boldsymbol{u}^+[i] \\ \boldsymbol{u}_t^r[i] - K_t[i,:].\boldsymbol{x} & \text{else if } \boldsymbol{u}_t^r[i] - K(t)[i,:].\boldsymbol{x} > \boldsymbol{u}^-[i] \\ \boldsymbol{u}^-[i], & \text{else} \end{cases} \tag{3.89}$$

and the resulting closed-loop system

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}) + g(\boldsymbol{x}).\tilde{K}(t, \boldsymbol{x}) \tag{3.90}$$

and its time-dependent linearisation at the reference point

$$\dot{\delta}_{\boldsymbol{x}} = A_t.\delta_{\boldsymbol{x}} - B_t.K_t.\delta_{\boldsymbol{x}} = (A_t - B_t.K_t).\delta_{\boldsymbol{x}} = \tilde{A}_t.\delta_{\boldsymbol{x}}. \tag{3.91}$$

Conjecture 3.1. *The time-dependent linearisation defined eq. 3.90 is a reasonably good approximation of the nonlinear system in a "large" region around the reference point.*

The nonlinear systems of interest here are polynomial functions, often approximating the second order rigid body dynamics of a robot using Taylor expansion. Therefore the system dynamics f and input dynamics g are smooth functions which asymptotically behave like the linearisation around the equilibrium. The only discontinuity arises from the input constraints (the linearisation of $\tilde{K}$). Therefore as long Ω does not intersect with the zone that saturates the control law $\tilde{K}$, the linearisation approximates the real dynamics reasonably well.

[3]Note that in order to ease notation, explicit time-dependency is indicated by the index $_t$, so $a(t) \equiv a_t$. Also, in the case that the time-dependency is clear from context, as for instance in the case of the current state of the system $\boldsymbol{x}$, it is dropped.

3.9.2 Computing Lyapunov Function Candidates Based on LQR-Techniques

In the above we have established a time-dependent linearisation of the nonlinear dynamics along a reference trajectory. Now we can modify classical optimal control approaches to compute suitable Lyapunov function candidates. A broadly utilized control law synthesis method for linear systems yielding a quadratic Lyapunov function as byproduct is called linear quadratic regulator (LQR) in its finite or infinite horizon version. Before adapting and applying this method to our case, let us briefly revisit the theory behind it, see also Lunze [2013].

LQR falls into the category of unconstrained optimal control. Given a (controllable) linear time-invariant system

$$\dot{\boldsymbol{x}} = A.\boldsymbol{x} + B.\boldsymbol{u} \tag{3.92}$$

with the usual system $A \in \mathbb{R}^{n\times n}$ and input dynamics $B \in \mathbb{R}^{n\times m}$ matrices, LQR seeks to compute a time-dependent linear feedback matrix K_t that minimizes the cost functional[4]

$$J(\boldsymbol{u}, \boldsymbol{x}_0) = \frac{1}{2}\boldsymbol{x}(T)^{\mathrm{T}}.S.\boldsymbol{x}(T) + \int_0^T \boldsymbol{x}^{\mathrm{T}}.Q.\boldsymbol{x} + \boldsymbol{u}^{\mathrm{T}}.R.\boldsymbol{u}\, dt\ . \tag{3.93}$$

Where $S \in \mathbb{S}^n_{++}$ is the terminal cost, so a penalty for the distance between the origin and the attained position for $t = T$, $\boldsymbol{x}(T)$. The matrix $Q \in \mathbb{S}^n_{+}$ determines the state cost and finally $R \in \mathbb{S}^m_{++}$ defines the cost of the control input. All of these matrices are frequently chosen to be diagonal and roughly speaking large eigenvalues of Q favour fast convergence whereas large eigenvalues of R favour smaller control inputs. However, no *optimal* way for choosing Q and R exists, but they need to be adapted for the specific dynamical system and application at hand.

Without going into the details, the minimization problem can be solved, for instance, using a dynamic programming approach, which yields a time-dependent cost function, denoted $V(t) = \boldsymbol{x}^{\mathrm{T}}.P(t).\boldsymbol{x}$ with $P(t) \in \mathbb{S}^n_{++}$. The evolution of P with respect to time is called the differential Riccati equation:

$$\dot{P}(t) = -P_t.A - A^{\mathrm{T}}.P_t + P_t.B.R^{-1}.B^{\mathrm{T}}.P_t - Q \tag{3.94}$$

with the boundary condition $P(T) = S$. The associated optimal linear feedback matrix is

$$K_t^* = R^{-1}.B^{\mathrm{T}}.P_t. \tag{3.95}$$

Theorem 3.3. *The above defined cost function $V(t) = \boldsymbol{x}^T.P_t.\boldsymbol{x}$ is a Lyapunov function for the linear dynamics defined in equation (3.92) for $\boldsymbol{u} = -K_t^*.\boldsymbol{x}$ with a guaranteed convergence exponent γ.*

Proof. Reconsider the (Lie) derivative of the Lyapunov function with respect to the closed loop system dynamics

$$\dot{V}(t) = \boldsymbol{x}^{\mathrm{T}}.P_t.(A - B.K_t).\boldsymbol{x} + \boldsymbol{x}^{\mathrm{T}}.(A^{\mathrm{T}} - K_t^{\mathrm{T}}B^{\mathrm{T}}).P_t.\boldsymbol{x} + \boldsymbol{x}^{\mathrm{T}}.\dot{P}_t.\boldsymbol{x}$$

When using the optimal feedback controller K_t^* and by substituting the corresponding terms in the Riccati differential equation (3.94) we get

$$\dot{V}(t) = \boldsymbol{x}^{\mathrm{T}}.(-Q - P_t.B.R^{-1}.B^{\mathrm{T}}.P_t).\boldsymbol{x} \leq \lambda_{\max}(-Q - P_t.B.R^{-1}.B^{\mathrm{T}}.P_t)\boldsymbol{x}^{\mathrm{T}}.\boldsymbol{x} = -\gamma.\boldsymbol{x}^{\mathrm{T}}.\boldsymbol{x} \leq 0$$

where $\lambda_{\max}(A)$ denotes the largest eigenvalue of the square matrix A. The inequalities hold since $Q \in \mathbb{S}^n_{+}$, $P_t \in \mathbb{S}^n_{++}$ and $R \in \mathbb{S}^n_{++}$. Moreover the cone of psd-matries is self-dual, therefore $R^{-1} \in \mathbb{S}^n_{++}$ and $Y^{\mathrm{T}}.X.Y \succeq 0$ if $X \succeq 0$ for any matrix Y of compatible size. Therefore $-Q - P_t.B.R^{-1}.B^{\mathrm{T}}.P_t \preceq 0$ and the guaranteed minimal convergence exponent γ is equal to the negation of the largest eigenvalue which has to be non-positive.

□

[4]There exist also versions of LQR adding the mixed cost $\boldsymbol{x}^{\mathrm{T}}.N.\boldsymbol{u}$, which could be equally applied in this scheme.

3.9.3 Adaption to the Constrained Time-Depending Case

In section 3.9.1 it was shown how to compute a suitable (time-dependent) Lyapunov function candidate for a nonlinear system using its linearisation and standard LQR-techniques. Now this computation has to be adapted in order to take into account the time-dependency of the linearisation and the input constraints. In the literature exist methods to incorporate (linear) input and state constraints to the problem of minimizing the quadratic cost defined in (3.93) with respect to a LTI system. However, the resulting minimization problem is frequently reformulated into a MPC-formulation, like Scokaert and Rawlings [1996] or Johansen et al. [2000]. These approaches however only consider the input (and state) constraints, not the time-dependency of the linearisation. This problem is tackled by another line of work mostly known under the name State-Dependent Ricatti equation, see Cimen [2008] or Erdem [2001]. These approaches however are either computationally intensive (MPC-based formulations), or not easily adaptable to our use case and the correspond implementations are not publicly available.

As we do not care about the optimality of the resulting control law, but only seek to find suitable Lyapunov function candidates for the nonlinear system by using its linearisation, the method is based on the following reasoning:

- The system matrices A and B can be time-dependent within the differential Riccati equation
- Control input saturation can be avoided by rescaling the optimal feedback controller K_t^*.

So given a time-dependent linearisation A_t, B_t, a reference trajectory $(\boldsymbol{x}_t^r, \boldsymbol{u}_t^r)$ defined on $t \in [0, T]$, the cost matrices Q and R, the final zone $\Omega(T) = \left\{\boldsymbol{x} | V(T, \boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{x}^r(T)\|_{P(T)}^2 \leq 1\right\}$ can be retro-propagated using

$$P(T_i) = P(T) + \int_T^{T_i} \dot{P}(t)\, dt \tag{3.96}$$

with

$$\dot{P}(t) = -P_t.A_t - {A_t}^{\mathrm{T}}.P_t + \frac{1}{2}P_t.B.\hat{K}_t^* + \frac{1}{2}\hat{K}_t^{*\mathrm{T}}.B^{\mathrm{T}}.P_t - Q \tag{3.97}$$

where $\hat{K}_t^*$ is the scaled optimal (in the LQR-sense) feedback controller.

So with $K_t^* = R^{-1}.{B_t}^{\mathrm{T}}.P_t$ the scaling factor α_{K^*} is computed as

$$\begin{aligned}
&\Delta\boldsymbol{u} = \min\ \left(\boldsymbol{u}^+ - \boldsymbol{u}_t^r,\ \boldsymbol{u}_t^r - \boldsymbol{u}^-\right)\\
&C_t = \mathrm{chol}(P_t)\\
&K_c = K_t^*.C_t^{-1}\\
&\alpha_{K^*} = \min\left(\min_i\ \frac{\Delta\boldsymbol{u}\,[i]}{\|K_c\,[i,:]\|_2},\ 1.\right)
\end{aligned}$$

and finally define $\hat{K}_t^* = \alpha_{K^*}K_t^*$ which ensures that

$$\forall \boldsymbol{x} \in \Omega_t :\ \boldsymbol{u}^- \leq \boldsymbol{u}_t^r - \hat{K}_t^*.(\boldsymbol{x} - \boldsymbol{x}_t^r) \leq \boldsymbol{u}^+ \tag{3.98}$$

with $\Omega_t = \left\{\boldsymbol{x} |\ \|\boldsymbol{x} - \boldsymbol{x}_t^r\|_{P_t}^2 \leq 1\right\}$.

Using (3.96) we can compute a suitable time-dependent Lyapunov function for the time-dependent linear system which provides a good Lyapunov function candidate for the nonlinear system. Note that the initial formulation and parametrization are adopted from LQR-theory, but since we are only interested in computing suitable Lyapunov function candidates for the nonlinear system, one can look at the equations (3.96) and (3.97) as regularized dynamics defining an approximative evolution of the backwards reachable set, taking into account the linearised system dynamics.

3.9.4 Examples and interpolation

In order to assess the improvements brought about by the changes to LQR listed above and to showcase the results of this approach, reconsider the dynamics of the torque controlled pendulum with $\boldsymbol{x} = \begin{pmatrix}\theta & \omega\end{pmatrix}^{\mathrm{T}}$. In Figure 3.11 the resulting RoS using Algorithm 2 for two different initial regions are compared. The reference trajectory is given as

$$\begin{aligned}
&\boldsymbol{x}^r(t) = \begin{pmatrix}\frac{160}{180}\pi & 0\end{pmatrix}\\
&\dot{\boldsymbol{x}}^r(t) = \boldsymbol{0}\\
&\boldsymbol{u}^r(t) \text{ such that } f(\boldsymbol{x}^r) + g(\boldsymbol{x}).\boldsymbol{u}^r(t) = \boldsymbol{0}.
\end{aligned}$$

The first initial region is computed using the standard LQR approach (RoS LQR), the second region (RoS VAR) is obtained by retro-propagating the final zone (a small sphere centred at $\dot{\boldsymbol{x}}^r(0)$)

$$\Omega(T_f) = \left\{\boldsymbol{x} | V(T_f, \boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{x}^r(T_f)\|^2_{P(T_f)=100Id} \leq 1\right\}$$

according to equations (3.96) and (3.97) until a steady-state is reached ($\dot{P}_t \approx 0$) resulting in an initial zone

$$\Omega(0) = \left\{\boldsymbol{x} | \, \|\boldsymbol{x} - \boldsymbol{x}^r(0)\|^2_{P(0)} \leq 1\right\}.$$

Due to the constant reference trajectory, the matrices A_t and B_t of the linearisation are constant (time independent) and the only difference to the usual differential Riccati equation is the scaling of the feedback gain matrix.

The second example is closer to the way the proposed approach is used in Algorithm 2 by retro-propagating a zone along a trajectory during a specified period, again for the torque controlled pendulum. In Figure 3.12, the results of using the time-dependent linearisation and the scaled feedback gain matrix are compared to the results obtained using the differential Riccati equation in the setting

$$\begin{aligned}
&t \in [0., T = 0.075]\\
&\boldsymbol{x}_0 = \begin{pmatrix}\pi\\ 2.5\end{pmatrix}\\
&\boldsymbol{u}^r_t = \begin{pmatrix}2.\end{pmatrix}\\
&\boldsymbol{x}^r_t = \boldsymbol{x}_0 + \int_0^t f(\boldsymbol{x}^r_t) + g(\boldsymbol{x}^r_t).\boldsymbol{u}^r_t \, d\tau.
\end{aligned}$$

In the case of the standard differential Riccati equation, the linearisation of the system around the final point $\boldsymbol{x}^r_T$ is used to compute the matrices A and B. As one can see, the modifications lead to a better estimate of the evolution of the stabilizable region.

In the first example, the final zone is retro-progated according to the ode defined in (3.96) and (3.97) until a steady-state is reached. In contrast, Algorithm 2 which constructs the funnel for the time-dependent polynomial system, alternates between two steps. First the above introduced method to retro-propagate a given zone $\Omega(T_i)$ from T_i to T_{i-1} taking into account the linearised system is used. Then in the second step the volume of the funnel for the given shape is maximized while guaranteeing stabilizability. Therefore the initial zone for this step has the parametrized form $\Omega(T_{i-1}) = \left\{\boldsymbol{x} | \left\|\boldsymbol{x} - \boldsymbol{x}^r_{T_{i-1}}\right\|^2_{P_{T_{i-1}}} \leq \alpha_{i-1}\right\}$ and Algorithm 2 seeks to maximize α_{i-1} using a line search approach.

As the evolution of P_t is nonlinear, storing and modifying P_t directly is not possible. Therefore only the final zone and the shape of the initial zone are stored and in order to check convergence on intermediate points with $T_{i-1} \leq t_j < T_i$ and to compute the time-derivative of P_t and α_{i-1} one has to rely on appropriate interpolation methods.

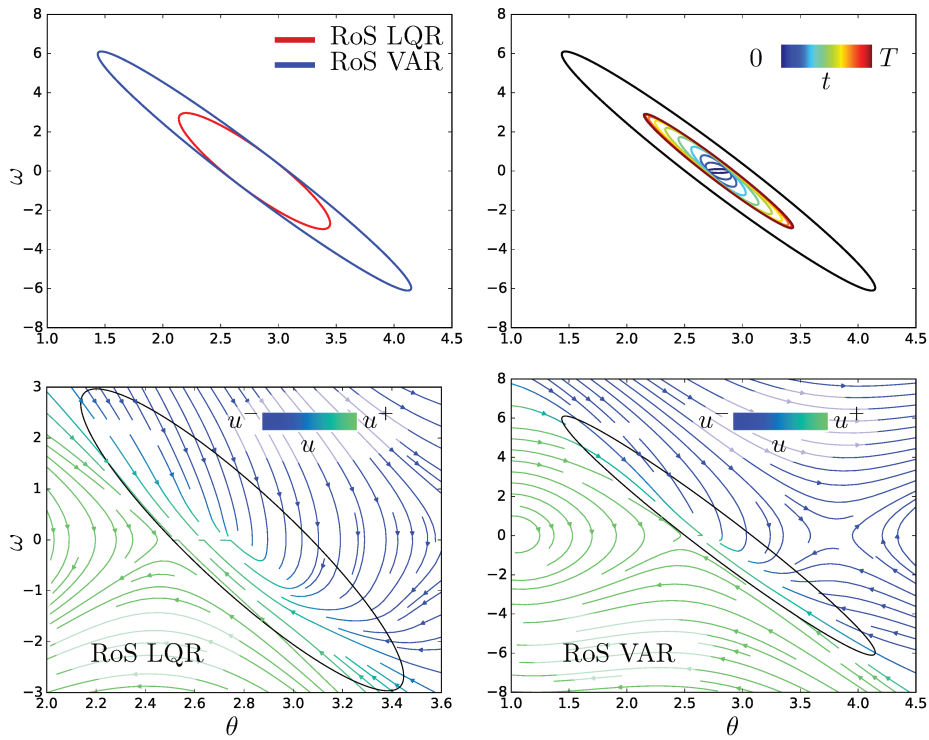


Figure 3.11 – In the top left figure the resulting RoS for the shapes resulting from standard LQR, P_{LQR} (RoS LQR) and the proposed approached P_{VAR} (RoS VAR) are depicted for the equilibrium point $\boldsymbol{x}^* = (160, 0.)^{\mathrm{T}}$,$u^* = 1.68$. In both cases the costs are defined as $Q = Id$ and $R = \left[1\right]$. Note that the eigenvectors of both matrices almost coincide, but the eigenvalues differ due to the scaling of the feedback gain matrix, resulting in a twice as large volume of RoS VAR compared to RoS LQR. The top right images shows the evolution of the zone $\Omega_t = \left\{\|\boldsymbol{x} - \boldsymbol{x}_t^r\|_{P_t} \leq 1.\right\}$ with initial condition $P_T = 100.Id$ and T chosen such that $\dot{P}_0 \approx \mathbf{0}$ (steady-state). The black ellipsoid corresponds to RoS VAR. In the bottom row (RoS LQR left, RoS VAR right) the resulting vector fields (using the QP-controller proposed in section 3.6.2 with $\epsilon_u = 1.0.$) are shown and the colouring of the streamlines corresponds to the control effort.

A very similar problem occurs in Majumdar et al. [2013b]. In this work, multiple intermediate points defined by T_i, $\boldsymbol{x}_i$, P_i and α_i are distributed along the trajectory and a linear parametrization is assumed, resulting in

$$P(t_j) = \frac{t_j - T_{i-1}}{T_i - T_{i-1}} P(T_i) + \frac{T_i - t_j}{T_i - T_{i-1}} P(T_{i-1}) \tag{3.99}$$

and

$$\dot{P}(t_j) = \frac{P(T_i) - P(T_{i-1})}{T_i - T_{i-1}} \tag{3.100}$$

when using the right-handed limit as derivative. As Majumdar et al. [2013b] also seek to modify the shape of funnel (in other words, the P_i's are in the decision variables), $P(t_j)$ has to be defined as a weighted sum of the P_i's or the resulting problem would no longer be convex. In contrast, Algorithm 2 provides

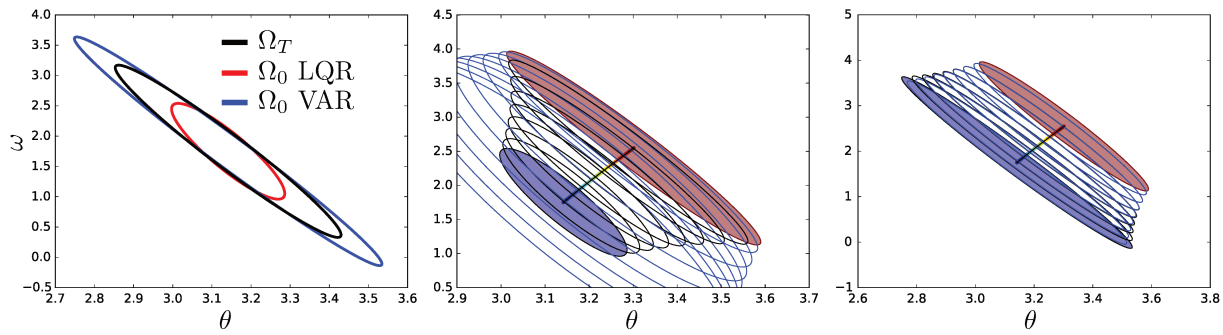


Figure 3.12 – In the left figure the resulting RoS for the shapes resulting from standard LQR (Ω_0 LQR) and the proposed approached Ω_0 VAR are depicted alongside with the target region Ω_T, translated so that all centres coincide. The target region is computed using infinite horizon LQR using the same cost matrices used for retro-propagation. In both cases the costs are defined as $Q = Id$ and $R = \begin{bmatrix}1\end{bmatrix}$. The middle image depicts the evolution of the stabilizable region Ω_t LQR. The black ellipsoids represent the interpolations (using the Cholesky interpolation presented in the next paragraph) of the actual stabilizable region, whereas the blue ellipsoids represent the evolution of Ω_T described by the Riccati equation. The actual region is significantly smaller, as the retro-propagation based on the Riccati equation does not take into account the control input limits. In the image on the right, the black ellipsoids represent once more the interpolations of the actual stabilizable region, whereas the blue ellipsoids show the evolution of Ω_T using the proposed approached. One can see that the difference between the blue and black ellipsoids is significantly reduced, implying a more suitable retro-propagating of the proposed method.

a certificate of stabilizability for fixed P_{i-1}, P_i and α_i and performs a line search to maximize α_{i-1}. Therefore it does not impose the necessity of $\alpha(t_j)$ and $P(t_j)$ to be affine in $\{\alpha_i\}_i$ and $\{P_i\}_i$ respectively.

We therefore propose to perform an interpolation based on the Cholesky decomposition of the P_i's and illustrate the advantages of this approach bby comparing the obtained overall results for the torque controlled pendulum.

Cholesky Interpolation Given P_0, $P_1 \in \mathbb{S}^n_{++}$ and the corresponding times T_0, $T_1 \in \mathbb{R}_+$, $T_1 > T_0$ the Cholesky interpolation for time t in the interval $I = [T_0, T_1[$ is given as

$$C_0 = \text{chol}(P_0),\ C_1 = \text{chol}(P_1) \tag{3.101}$$

$$C(t) = \frac{t - T_0}{T_1 - T_0} C_1 + \frac{T_1 - t}{T_1 - T_0} C_0 \tag{3.102}$$

$$\dot{C}(t) = \frac{C_1 - C_0}{T_1 - T_0} \tag{3.103}$$

$$P(t) = C(t)^{\mathrm{T}}.C(t) \tag{3.104}$$

$$\dot{P}(t) = \dot{C}(t)^{\mathrm{T}}.C(t) + C(t)^{\mathrm{T}}.\dot{C}(t)\ . \tag{3.105}$$

As seen in Figure 3.13, this nonlinear (with respect to P_0 and P_1) interpolation method rotates and scales the initial zone Ω_0 gradually, whereas the linear parametrization of (3.99) contracts and expands zone over time, often resulting in an increased control effort.

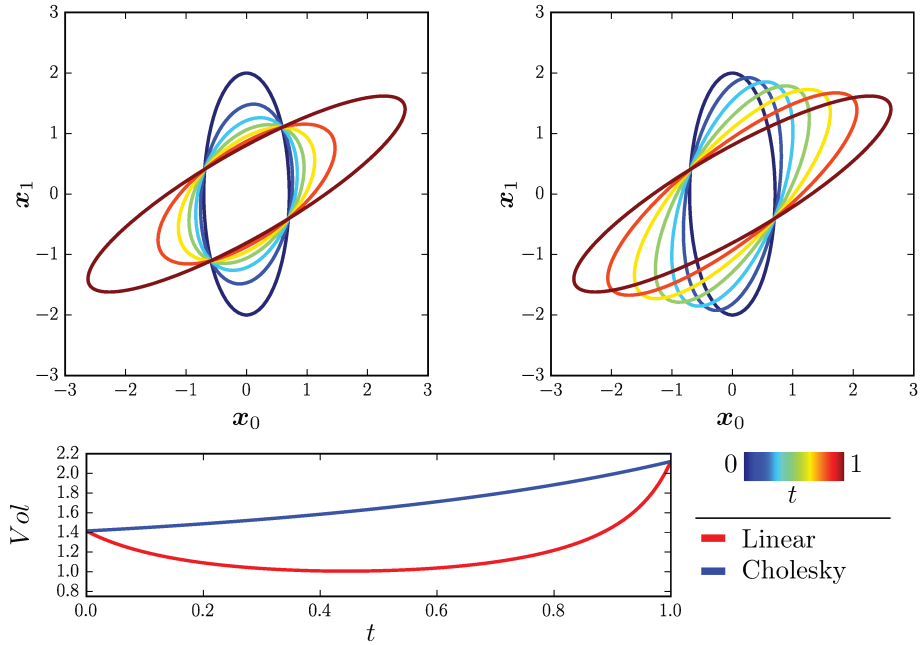


Figure 3.13 – Comparison of linear and Cholesky interpolation for zones defined as sublevel-sets of quadratic functions centred at the origin defined by $\Omega_t = \left\{ \boldsymbol{x} | \, \|\boldsymbol{x}\|_{P_t}^2 \leq 1 \right\}$. In this example $T_0 = 0$, $T_1 = 1$, $P_0 = \begin{bmatrix} 2. & 0. \\ 0. & 0.25 \end{bmatrix}$ and $P_1 = \begin{bmatrix} 0.85 & -0.82 \\ -0.82 & 1.53 \end{bmatrix}$. The evolution of Ω_t is depicted in the upper row for linear interpolation (left image) and the proposed Cholesky interpolation (right image). The evolution of the volume of Ω_t over time is depicted in the lower row. The volume of intermediate zones for linear interpolation is significantly smaller than the volume of the final or initial zone. As these zones have to be invariant for the dynamical system, this can indicate the necessity of higher control efforts. Cholesky interpolation on the other hand provides a *more natural* evolution of the zone and its volume. Note that in the linear case $\forall t \in [0, 1] : \; \Omega_t \subset \Omega_0 \cup \Omega_1$, which is not true for Cholesky interpolation.

3.10 Examples and Numerical Results

The above presented approach based on state space partitioning and underestimation of real-valued polynomials is implemented in python using cvxopt (Andersen et al.) and compared to the results obtained using the drake toolbox (Tedrake and Team [2016]) for Matlab™ relying on a very similar approach to the one presented in Majumdar et al. [2013b] and other examples found in the literature. The largest region of attraction for the unstable position of a torque controlled simple pendulum and the Acrobot, a 2 DoF underactuated robotic arm as defined in (3.55) with the physical constants taken from Tedrake and Team [2016], are presented and compared. Additionally, a time varying region of stabilizability ("funnel") is presented for a swing-up trajectory for the pendulum and the Acrobot using reference trajectories generated with OMPL (Şucan et al. [2012]) and KPIECE1 (Şucan and Kavraki [2012]).

3.10.1 Simple Pendulum

The first example provided is a torque controlled simple pendulum, modelled as a point mass on a massless beam with viscous friction in the hinge joint, for which we want to approximate the region of stabilizability of the upright position. The numerical values of the system are taken from the drake toolbox as mass equal to 1kg, the beam length is 0.5m and the damping coefficient is $0.1^{\mathrm{Nm}}/_{\mathrm{s}}$. Note that for this example, the system dynamics are nonlinear, whereas the input dynamics are affine. This means that the state-space partitioning and the resulting control law are truly optimal.

Comparison of Methods Generating Lyapunov Candidates

Before comparing the method presented in this chapter with a state-of-the-art SoS-technique, the resulting RoS for three different approaches to compute the shape of the Lyapunov function are compared:

a) Computing $V(\boldsymbol{x})$ based on the linearisation of the system around the equilibrium point $\boldsymbol{x}^*$ and the standard LQR method (RoS LQR).

b) Retro-propagating the singleton $\boldsymbol{x}^*$ (approximated by a small sphere) according to (3.97) until the steady-state is reached (RoS VAR).

c) Constructing a funnel for the trajectory $\boldsymbol{x}_t^r = \boldsymbol{x}^*$, $\boldsymbol{u}_t^r = \boldsymbol{u}^*$ with the final zone Ω_T being a small sphere centred at the equilibrium with T chosen such that steady-state is reached (RoS Funnel).

In the last two approaches attaining steady-state means that $\dot{P}_t \approx 0$, that is the funnel shape does no longer change. The results are shown in Figure 3.14. Each of the methods yields ellipsoids of comparable surface but slightly different shapes. This indicates that all three methods yield suitable Lyapunov function candidates for the polynomial approximation of the nonlinear system by relying on the linearisation around the equilibrium point.

Comparison with drake toolbox

Next we compare the RoA for the upright position obtained with the drake toolbox and the RoS for the upright position obtained with the approach presented in this chapter and the Lyapunov function candidates described above. Our approach provides in this example significantly larger regions of stabilizability than the iterative SoS-approach, even-though it does not modify the shape of the region. It is worth noting that in Majumdar et al. [2013b] the ellipsoids are normalized by the condition $V(\mathbf{1}) = 1$, where $\mathbf{1}$ is the vector of all ones. Even though this normalization does not introduce any conservativeness in the sense that a class of functions is excluded from the optimization, it introduces a bias since it is not equivalent to normalizing the ellipsoids by their volume. The ellipsoids in the top right image of Figure 3.15 obtained with the proposed method (red, green and blue ellipsoid) have similar cost values for $\mathbf{1}$, but have an about seven times larger surface than the one obtained using drake (black ellipsoid). One can see that the resulting closed-loop dynamics are relatively similar for the chosen regularization value Q. Even more interesting, the generated RoA SoS could be scaled without changing the control law and still be an invariant set, however the conservativeness introduced by the multiplier terms seems to inhibit this. It is also worth noting that the deduced control input and dynamics at the critical point (marked by the bright green ellipsoid) are also equivalent for the two approaches. A last remark on a qualitative difference between the generated sets: while the set RoA SoS is an *invariant* set according to Majumdar et al. [2013b], the regions of stabilizability obtained with the proposed approach are a *exponentially converging* sets. This is worth noting since this change does not impact the runtime of our approach but has a significant impact on SoS approaches due to the additional SoS constraints necessary on the multiplier terms.

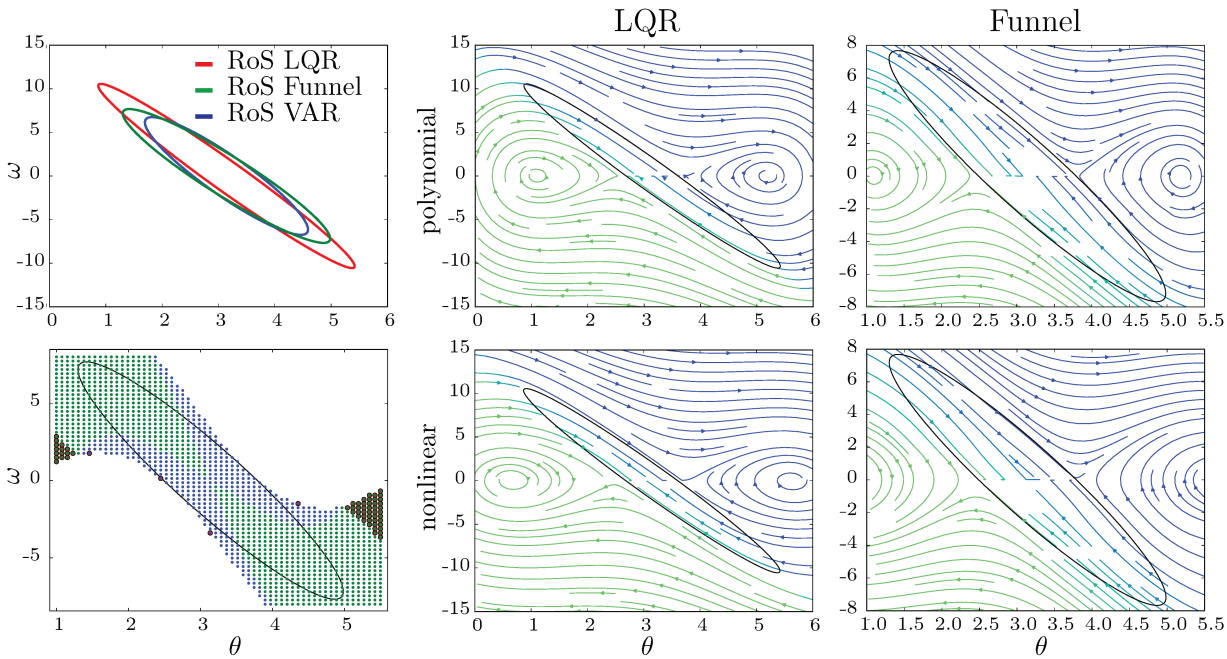


Figure 3.14 – On the top left the different regions of stabilizability are compared. The largest surface (4.99) is obtained for the funnel approach (RoS Funnel), whereas the standard LQR (RoS LQR, 4.86) outperforms the retro-propagation (RoS VAR, 4.04), where in all cases $Q = Id_n$ and $R = 5Id_m$ was used. It is important to notice that despite the different areas obtained, they all significantly outperform the results obtained with drake toolbox as shown in Figure 3.15. On the right the resulting streamlines using the QP-based control law is shown for RoS Funnel and RoS LQR. As one can see the polynomial approximation of the dynamical system (upper row) approximates well the fully nonlinear dynamics (lower row). In fact for this case the approximation is conservative, that is RoS Funnel and LQR are also stabilizable regions for the original dynamics. In the lower left image a quantitative impression of the QP-based control law is given. Small red circles indicate that the control input constraint is active, whereas blue dots indicate the activation of the constraint ensuring convergence. Green dots indicate that no constraint is active, whereas the absence of a dot indicates that the problem is infeasible, meaning that convergence cannot be ensured.

Swing-Up Funnel

The last example shows a funnel generated around a swing-up reference trajectory for the pendulum and is presented in Figure 3.16. The input constraints do not allow the system to directly attain the unstable upright position. Therefore several "pumping" movements, corresponding to the oscillations of the reference trajectory in the lower right image of Figure 3.16, have to be performed to accumulate energy. Generating a funnel in such a case is interesting for two reason. Firstly the funnel can be used for collision detection and certified planning under the assumption that the mathematical description of the true system is correct. Secondly one can define a time-independent control law in the following way: given a funnel described by the time-dependent Lyapunov function $V(t, \boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{x}_t^r\|_{P_t}$ and the current state of the system $\boldsymbol{x}$, then one can find the most suitable reference point as

$$t_c = \min_t V(t, \boldsymbol{x}), \tag{3.106}$$

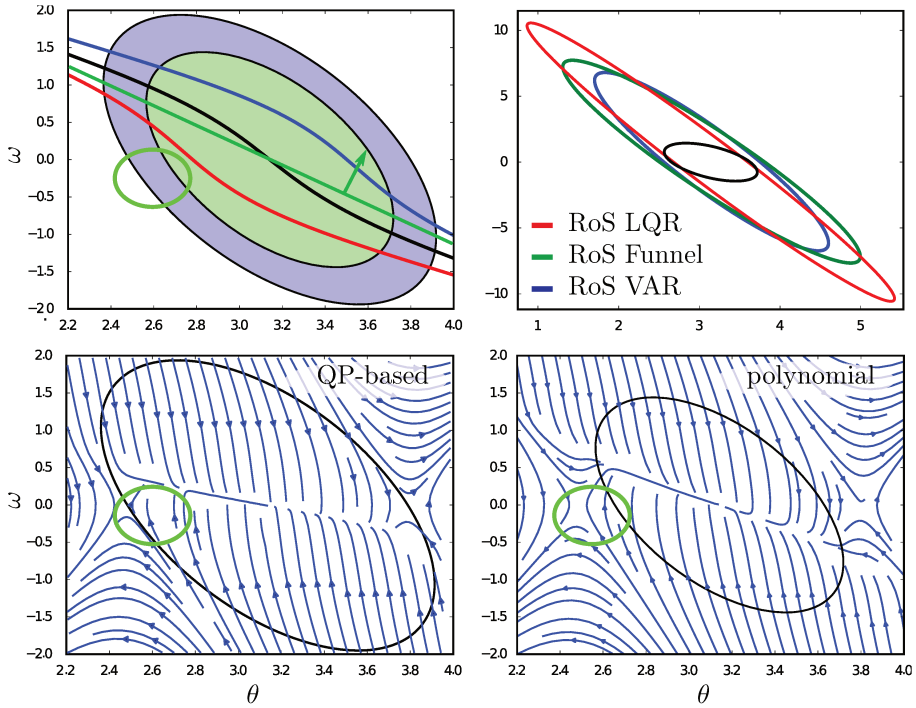


Figure 3.15 – Figure top left: Green ellipsoid: Region of attraction obtained with the drake toolbox for a cubic controller denoted RoA SoS; Blue ellipsoid obtained with our approach (denoted RoS Opt) initialised with the Lyapunov function of RoA SoS. Green line: Separating hyperplane and defining normal vector; red, black and blue line: Set of points where the cubic control law (from drake toolbox) attains $\boldsymbol{u}^+$, 0 and $\boldsymbol{u}^-$. This means that below the red curve and above the blue curve, the control inputs of the optimal control law $\boldsymbol{u}^*$ and that of the saturated polynomial control law from drake toolbox are equivalent. Figure top right: Region of stabilizability obtained with our approach for the different initialisations presented in Figure 3.14 (RoS LQR in red, RoS VAR in blue and RoS Funnel in green). Bottom row: Streamlines obtained when applying the generated control law. Left: Regularized QP-control with $\gamma = 0.001$ and $Q = [0.5]$ based on our approach. Right: Cubic feedback control law saturated to meet input constraints. For both plots the Taylor expansion up to degree 3 of the simple pendulum is used as system dynamics.

so as the time-point for which the associated cost is minimal, as it is shown in Tedrake et al. [2010b]. Then the control input according to this time-point can be computed. Moreover, if $V(t_c, \boldsymbol{x}) \leq \alpha_t$, then the state of the system will converge towards the (time-dependent) trajectory and therefore remain inside the funnel. By defining this time-independent control law, one can also see the advantage of generating "large" funnels. In Tedrake et al. [2010b] not just one funnel, but a whole tree of interconnect funnels is constructed until the whole state space is covered. Generating larger funnels is in this case directly linked to a smaller number of funnels necessary and therefore reduces offline and online computation time.

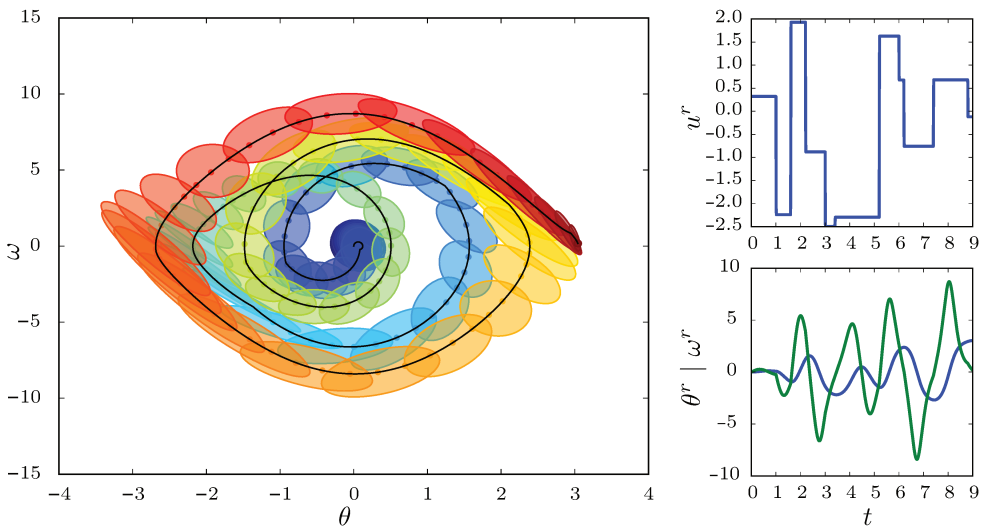


Figure 3.16 – The reference trajectory is shown on the right. The input constraints of the system are $-3. \leq u \leq 3$ and are respected with a relatively large margin by the reference control input shown in the upper right image. The lower right image shows the evolution of θ^r (blue) and ω^r (green). On the left the resulting funnel for $\gamma = 2.$ is presented. Note that a convergence rate of 2 is fairly large in the sense that it implies quick convergence, but the algorithm is still able to find a *large* funnel. One can easily see the influence of the time-depending linearisation which induces significant changes of the funnel shape.

3.10.2 Acrobot

The underactuated two link robot called Acrobot does not belong to the class of control affine systems since its mass matrix is state-dependent and we therefore obtain nonlinear input and system dynamics $\dot{\boldsymbol{x}} = f^{nl}(\boldsymbol{x}) + g^{nl}(\boldsymbol{x}).\boldsymbol{u}$. Therefore we seek to quantify the suboptimalty induced by the polynomial input before presenting results for stabilizing the unstable, upright position and a swing-up trajectory.

Quantifying the Suboptimality

In order to get a quantitative impression of the order of magnitude of the error induced by considering the linearised input dynamics to derive the input partitioning instead of the polynomial one, in Figure 3.17 the separating hyperplane and the separating hypersurface defined as

$$\mathcal{P}^{poly} = \left\{\boldsymbol{x} | \boldsymbol{x}^{\mathrm{T}}.P.g(\boldsymbol{x}) = 0\right\} \tag{3.107}$$

are compared.

The intersection of the hyperplane / hypersurface with five different hyperplanes parallel to the θ_0-θ_1 plane are plotted. These planes are chosen such that they quantitatively cover the RoS.

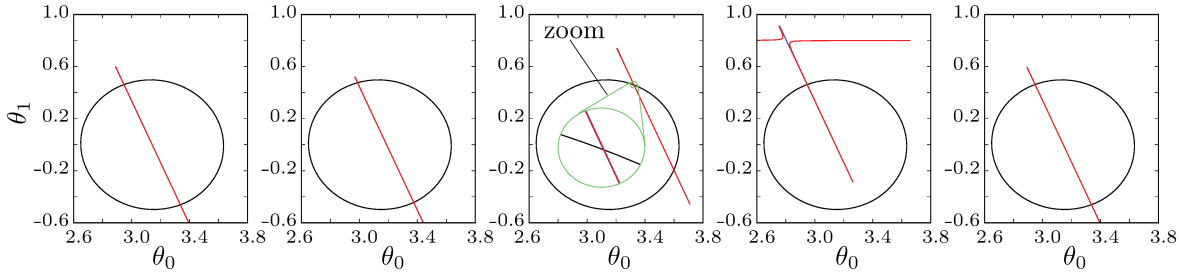


Figure 3.17 – In this figure the intersection of the separation hyperplane (blue) and the hypersurface (red) with the θ_0-θ_1-plane is shown. The difference between them within the region Ω is marginal in this example, empirically justifying the use of the optimal control and the associated polytopes even for nonlinear or polynomial input dynamics.

Stabilising the upright position

Again we compare the region of attraction found for the upright position under input constraints for the drake toolbox and the region of stabilizability. Due to numerical issues solving the resulting semidefinite optimization problem when searching for a cubic controller using the drake toolbox, only a linear feedback controller is used. In Figure 3.18 the projections of the two regions on different planes (the θ_0/θ_1-plane, θ_0/ω_0 and the θ_1/ω_1-plane) are shown. Here again, the volume of the ellipsoid obtained with the proposed approach (RoS Funnel) is significantly larger (about 25 times) than the volume of RoA SoS, but the value of the Lyapunov functions for normalization constraint $V(\mathbf{1})$ are very similar ($20.\,10^{-3}$ to $22.\,10^{-3}$).

Funnel around a Swing-Up Trajectory

As final example we seek to construct an as large as possible time-varying exponentially converging region, around a given reference trajectory. The reference trajectory describes a swing up motion, i.e. a motion taking the Acrobot from the stable "hanging" position to the unstable upright position. We want to compute a funnel that takes as many states as possible, measured by the volume of the ellipsoids defining the funnel, to a small ellipsoid centred on the upright position. To achieve this, we first distribute N sample points along the trajectory, for which we will actually prove convergence as described in section 3.7.2, similar to the method in Majumdar et al. [2013b]. It has proven to be better, in the sense of achieving larger volumes, to distribute the points for which convergence is verified equidistantly on the curve $\boldsymbol{x}_t^r$ and not equidistantly along the time, as equidistant distribution with respect to time can lead to long curve segments in the high-velocity parts of the trajectories. These longer curve segments often induce larger changes in the shape of the regions, complicating interpolation.

The projection of the resulting funnel onto different planes is shown in Figure 3.19. For this example we prove convergence on 95 points ($N_{steps} = 95$) and 2 intermediate steps for each interval ($N_{inter} = 2$). Each retro-propagation step takes about 12 seconds, so the entire funnel is computed in about 20 minutes on a desktop pc with an Intel(R) Core(TM) i7-3770 @ 3.4GHz processor.

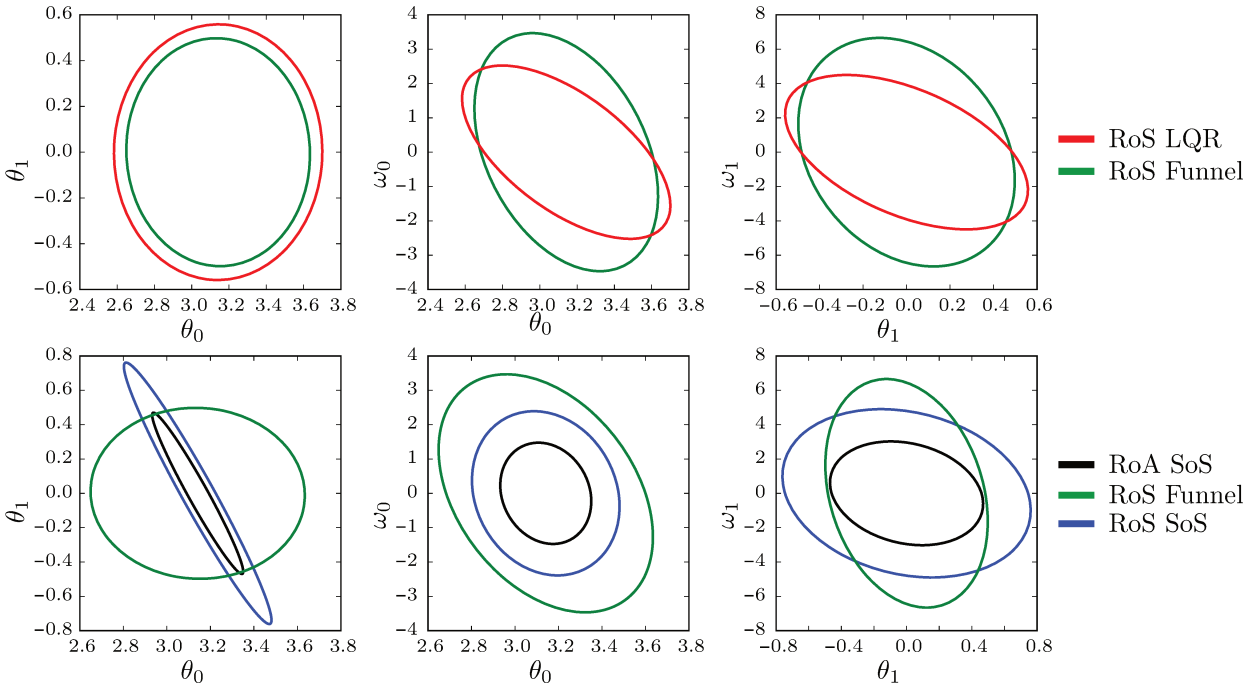


Figure 3.18 – Top row: different projections of the RoS found using the proposed approach initialised with standard LQR is shown in red (RoS LQR), the steady-state region for the funnel constructed around the reference point relying on Algorithm 2 is shown in green (RoS Funnel). In this case the standard LQR initialisation outperforms the funnel approach by achieving a hypervolume of 0.053 compared to 0.04. For both approaches the cost matrices are given as $Q = \text{diag}([10, 10, 1, 1])$ and $R = 0.01 * Id_m$. It is worth noting that, probably due to the underactuation of system, the volume of the region is more sensitive to the parameters Q and R than for the pendulum. In the bottom row the resulting RoS are compared to the RoA found using Drake toolbox and a linear control law (RoA SoS, black). The RoS SoS, that is the proposed approach initialised with RoA SoS, is shown in blue. As for the torque controlled pendulum, the region RoA SoS can be enlarged without changing its shape. The shapes of RoS Funnel and and RoS SoS differ significantly with RoS Funnel having a larger volume, empirically validating the effectiveness of Algorithm 2.

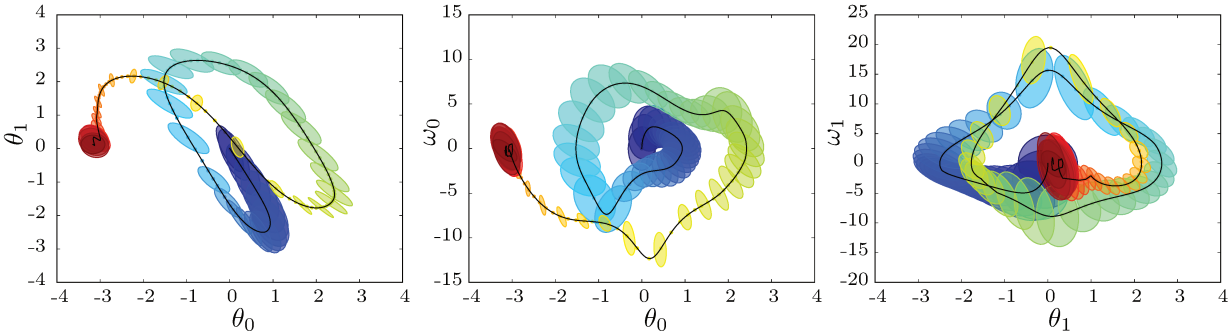


Figure 3.19 – Projection of the funnel (from left to right) onto the θ_0/θ_1-plane, θ_0/ω_0-plane and θ_1/ω_1-plane. Due to the under-actuation of the system, a small neighbourhood around the origin has to be excluded to prove exponential convergence, so we have $\dot{V}(\boldsymbol{x},t) \leq -\gamma V(\boldsymbol{x},t)$ on $\Omega' = \{\boldsymbol{x} | \epsilon\alpha(t) \leq V(\boldsymbol{x},t) \leq \alpha(t)\}$ with $\epsilon = 0.05$ and $\gamma = 0.1$ for this example. The reference input is confined between -10 and 10 and we set the minimal/maximal input to $-20/20$, other numerical values defining the system are taken from the drake toolbox.

3.10.3 Controlled Polynomial System

To showcase the versatility of the proposed approached and to show that it can be successfully applied to polynomial systems that do not correspond to Taylor expansions, we consider the two-dimensional system proposed in Jarvis-Wloszek et al. [2003]:

$$\dot{\boldsymbol{x}} = \begin{pmatrix} 0 \\ -\boldsymbol{x}_0 + \frac{1}{6}\boldsymbol{x}_0^3 \end{pmatrix} + \begin{pmatrix} 1 \\ -1 \end{pmatrix} u \tag{3.108}$$

with $u \in \mathbb{R}$. As one can see, this dynamical system falls into the class of control affine polynomial systems, for which the derived state-space partitioning is optimal. Secondly we see that the control input is unbounded, so $u^+ = \infty$ and $u^- = -\infty$. This is not compatible with the proposed approach, as the derivative of the Lyapunov function would always be equal to plus or minus infinity. However we can solve the problem of finding the largest RoS for increasingly large absolute values for artificially introduced upper and lower bound of the input value. If the system is only locally stabilizable, the RoS of the usual form $\Omega = \left\{\boldsymbol{x} | \boldsymbol{x}^{\mathrm{T}}.P.\boldsymbol{x} \leq \alpha\right\}$ will monotonously converge towards α^*, with α^* being the largest value for which Ω is still strictly contained in the true region of stabilizability. In the case of global stabilizability, α^* is infinite, and the series will diverge.

By taking a closer look at the dynamical system (3.108), we see that, as u is unbounded, the system can always be dominated by the control input which is state-independent. Or, in other words, for all states within a ball of radius r centred at the origin one can find an u^M such that the dynamics can be simplified to

$$\dot{\boldsymbol{x}}' \approx \begin{pmatrix} 1 \\ -1 \end{pmatrix} u^*. \tag{3.109}$$

This approximated system is clearly globally stabilizable for any quadratic Lyapunov function candidate and a convergence rate $\gamma = 0$ when applying the optimal control input.

This corresponds to the findings using our approach as shown in Figure 3.20. These findings highlight that reasoning directly on optimal control inputs and local stabilizability can yield significant advantages over approaches enforcing locality using multipliers and iterations as they can get stuck in local minima. This is not the case for the presented algorithm, however at the cost of fixing the shape of the Lyapunov function.

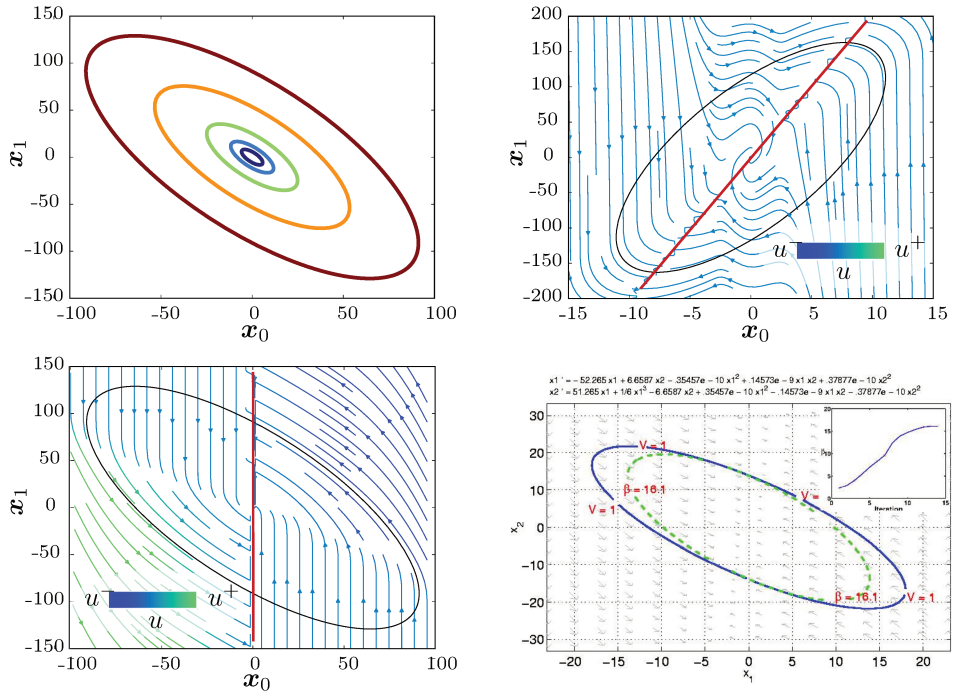


Figure 3.20 – On the top left image the sequence of RoS obtained for a monotonically increasing sequence of the artificially introduced input constraints is shown. Starting from blue to red, u^M takes on the values $[10, 100, 1000, 10000, 50000]$ and the constraint becomes $-u^M \leq u \leq u^M$. The Lyapunov function is taken as the one found in Jarvis-Wloszek et al. [2003], shown on the bottom right image (image taken from Jarvis-Wloszek et al. [2003]), using an iterative approach indicated by the blue ellipsoid. The green dashed ellipsoid is the initial guess for the iteration. In the bottom left image the resulting streamlines for $u^M = 50000$ are shown and the color correspond to the control input, where again the QP-based control is used. The red line corresponds to the separating hyperplane. The top right image shows the obtainable RoS for $u^M = 50000$ when using the Lyapunov function derived from standard LQR with $Q = Id_2$ and $R = [1/u^M]$. As we can see the RoS based on LQR has a larger area than the RoA found in Jarvis-Wloszek et al. [2003], indicating once again that computing Lyapunov function candidates based on the linearised system is a good option.

3.11 Conclusion and Outlook

In this chapter a new approach to prove stabilizability for polynomial systems has proposed. It is based on ideas from optimal control and Lyapunov theory and results in an input optimal state space partitioning. The proof of stabilizability is then reformulated as an optimization problem on each of the subsets forming the partition and it was shown how to efficiently solve them by constructing a linear underestimator.

The effectiveness of the proposed approach has been showcased on typical examples from the literature outperforming state-of-the-art sum-of-squares techniques in terms of the volume of the region of attraction compared to the region of stabilizability.

To increase the scalability of the approach, it is interesting to investigate the possibility of adding the LMI constraints derived from the theory of moments. By constructing these constraints for a "small" maximal degree in the relaxation and adding the "most significant" linear constraints derived with RLT one can possibly obtain tight bounds at a reduced computational complexity. Moreover the size and number of the LMI constraints scales better to higher dimensions than the number of linear constraints.

This will allow to treat robotic systems actual used in manufacturing like manipulators with 6 or 7 degrees of freedom in a very efficient manner.

Finally we are interested in enlarging the modelling capabilities of the approach by taking into account perturbations represented by distributions and model uncertainties. Even though the formal the character of the funnels (no states can leave it) must be weakened in this case to a statistical statement (95% of the executions will not leave the funnel), this is an important step to take as many real-life applications are subject to non-negligible disturbances, which can be approximated well by (Gaussian) distributions.

Chapter 4

Learning Globally Asymptotically Stable Vector Fields

In this chapter we investigate methods to learn globally asymptotically stable vector fields and apply them within a learning from demonstration framework. To successfully learn nonlinear vector fields providing such guarantees, we propose a novel approach based on diffeomorphic transformations. Indeed such transformations allow us to increase the expressiveness of "simple" globally asymptotically stable vector fields while conserving the convergence property. The diffeomorphic transformations used within this chapter are based on locally weighted translations, which can, in contrast to state-of-the-art diffeomorphisms based on flows, be evaluated extremely quickly and are therefore suitable for real-time controller implementations.

4.1 Introduction

As seen in previous chapters, the combined notion of stability and safety essentially require to know in advance bounds on the evolution of the system, such that these bounds remain under control and exclude states that are considered unsafe. Moreover, these bounds must be valid at all time. A weaker version of these properties consist in focusing only in the asymptotic behaviour of the robot. For instance, having the guarantee that it will eventually reach its target. When stronger guarantees are not available, it can be interesting to try to enforce such weaker properties.

To this end, two approaches for learning globally asymptotically stable (GAS) vector fields are presented. Such vector fields provide guarantees on the asymptotic behaviour of the system. More specifically, they guarantee that a) there exists an unique equilibrium point and b) this equilibrium point is a global attractor. That is, all states will be driven to and will remain inside an ϵ small neighbourhood of it in finite time. These properties can be useful in different settings.

One might observe that, by denoting the state of the system with $\boldsymbol{x} \in \mathbb{R}^d$, the global attractor with $\boldsymbol{x}^* \in \mathbb{R}^d$ and the GAS vector field $f\colon \mathbb{R}^d \to \mathbb{R}^d$, saying that "all states will be driven to and remain inside an ϵ small neighbourhood of the global attractor in finite time" is somewhat equivalent to the LTL specification "eventually $\boldsymbol{x} = \boldsymbol{x}^*$" when the system evolves according to $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$.

The ability to translate such atomic LTL specification into a desired velocity for a dynamical system

can be exploited to perform control strategy synthesis as done in Kress-Gazit et al. [2007]. Here the "high-level" control strategy for (a fragment of) LTL specifications are synthesized relying on "low-level" properties provided by the controller developed in Conner et al. [2003].

The here proposed methods for learning a vector field can provide similar "low-level" guarantees, but, as the vector field is deduced from the given demonstrations, it is likely to be such that it can be easily followed by the controlled system given the demonstrations are suitable. This is an interesting property as it reduces the gap between the mathematical model used for verification and synthesis and the real world system.

On the other hand, by interpreting the learned vector field as a velocity field in the joint or task space of a robotic manipulator, learning a globally asymptotically stable vector field can be interpreted as a type of learning from demonstration under stability guarantees. As all trajectories generated by the learned vector field converge to a unique point, such vector or velocity fields naturally correspond to grabbing or reaching motions, as they also converge to a unique position and we focus on this use case within this chapter.

Learning from Demonstration Programming robots to perform specific tasks is a very challenging problem which can usually only be performed by trained and experienced persons ("experts"), even for very simple tasks, including reaching and grabbing motions. This is due to multiple reasons, such as the nonlinear mapping between the joint and task space via the geometric model of the robot, the often occurring redundancy of joints, the avoidance of self-collision or the dynamic feasibility of the trajectory. This poses an essential problem especially for the growing field of robots deployed in flexible manufacturing chains typically found in small or medium sized business or the possibility of relying on robots to provide services within regular households in the future. Such robots have to come with a built-in mechanism that allows the owner or the production workers to conveniently *teach* the robot how to successfully carry out a new task.

The learning from demonstration paradigm provides a possible solution to this problem. It allows the user the teach the robot by providing successful examples of task completion, relying for instance on kinesthetic training. This is a very natural way for humans to teach, as it is comparable to teaching movements to children, and moreover alleviates many of the above cited problems of motion planning and trajectory generation. It inherently takes the geometric model and the problem of self-collision into account. Also movements demonstrated by humans are dynamically feasible, in the sense that they do not exceed the maximal effort the actuators can provide, for usual robotic systems. Moreover, the "programming" of the robot does not involve coding be it in general purpose languages or in a provided high-level language.

These are the advantages of learning from demonstration, but they do not provide a clear path as to *how* such learning can be achieved. There exists a broad variety of approaches to tackle this problem, depending on the kind of task being demonstrated. The first developments in this sector were seeking to extract a set of (timed) way-points to be attained by the robot, see Grossman [1977] or Lozano-Perez [1983] and references therein. This i also comparable to how the learning is accomplished in the (formerly) commercial robots Baxter and Sawyer. Later on the research community focused on constructing abstract descriptions of tasks as finite state machines or decision trees allowing for symbolic reasoning, as in Segre and DeJong [1985] or DeJong and Mooney [1986].

In more recent developments, a clearer distinction is made between high-level and low-level learning (see the surveys Atkeson and Schaal [1997] or more recently Billard et al. [2008] and Argall et al. [2009]). For high-level planning Markov processes are an attractive possibility (see Konidaris et al. [2012]), but due to the necessarily discrete state and action space it is less suitable for learning low-level motions or motor control. For this scenario, recent developments have shown that representing a motion as dynamical systems (DS) and learning the parameters of the DS from demonstration yields interesting results (Schaal [2006]). In particular, expressing a motion as a dynamical system naturally increases the robustness to spatial and (if the system is autonomous) temporal perturbations. On the other hand, by introducing a dynamical system, the problem of stability naturally arises. Indeed, data-driven stochastic

approaches generally provide no guarantee concerning the stability of the resulting system. The problem of reconciling learning from demonstration while guaranteeing stability has become an active field of research, starting with Khansari-Zadeh and Billard [2010].

The contributions proposed in this chapter fall into this category. More precisely, we present novel ways to construct or learn globally asymptotically stable nonlinear dynamical systems which are able to reproduce the given demonstrations. As discussed later on in detail, global asymptotic stability is an important, but difficult to ensure, property for dynamical systems and therefore the main focus of the presented approach, next to the ability to properly reproduce the given demonstrations.

The main idea of the proposed approach is to use a diffeomorphic transformation in order to be able to guarantee the global stability of complex vector fields able to reproduce complicated motions.

The rest of this chapter is structured as follows. After a brief recapitulation of the properties of diffeomorphic transformations and their implications on dynamical systems in section 4.2, the treated problem is formally stated and existing techniques seeking to solve similar problems are reviewed in section 4.3. In section 4.4 the first approach for learning nonlinear globally asymptotically stable vector fields relying on diffeomorphic transformations, called One-Step learning is presented and evaluated. In section 4.5 an extension to the One-Step learning, called Two-Step learning is presented. Here the focus lies on interweaving the diffeomorphic matching and machine learning techniques in order to overcome several limitations of the One-Step learning. Concluding remarks and some avenues for future work are given in section 4.6.

The contributions in this chapter can be summed as follows.

Contributions

- A novel method to construct diffeomorphic transformation by composing local diffeomorphic transformations
- The extension of these transformations to multitranslations
- An efficient algorithm to construct such transformations
- Showcasing the obtainable results for the LASA-Dataset
- Extensions to multimodal and cyclic demonstrations

The material presented in this chapter was (in parts) published in Perrin and Schlehuber-Caissier [2016].

4.2 Diffeomorphic Transformations and Smooth Equivalence

In this chapter show how diffeomorphic transformations can be used within a learning from demonstration or supervised learning framework, but before doing so, the main properties and notations of diffeomorphisms are recapitulated.

A diffeomorphism is defined as an isomorphism between smooth manifolds, that is given two smooth manifolds $\mathcal{X}$ and $\mathcal{Y}$, the transformation

$$\begin{aligned} \Phi\colon\ & \mathcal{X} \to \mathcal{Y} \\ & \boldsymbol{x} \mapsto \boldsymbol{y} \end{aligned} \tag{4.1}$$

is a diffeomorphism if it is bijective, invertible and the transformation and its inverse are differentiable on $\mathcal{X}$ and $\mathcal{Y}$ respectively. Moreover, if Φ and Φ^{-1} are k-times differentiable, then the diffeomorphormism is said to be a $\mathcal{C}^k$-diffeomorphism. The demand that Φ is bijective, forces the manifolds $\mathcal{X}$ and $\mathcal{Y}$ to have the same dimension.

In this work we only use diffeomorphic transformations from $\mathbb{R}^d$ onto itself, so $\mathcal{X} = \mathcal{Y} = \mathbb{R}^d$, but we keep the distinct symbols $\mathcal{X}$ and $\mathcal{Y}$ to clarify notations. Note that in this case, the tangent space of both manifolds is trivial and also equal to $\mathbb{R}^d$, so we have $\mathcal{X} = T_{\boldsymbol{x}}\mathcal{X} = \mathcal{Y} = T_{\boldsymbol{y}}\mathcal{Y} = \mathbb{R}^d$. The usage of diffeomorphisms in the context of learning is motivated by the conservation of topological properties, such as connectedness, disjointness or convergence. The later is particularly interesting when reasoning about vector fields and dynamical systems: given two dynamical systems $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$ and $\dot{\boldsymbol{y}} = g(\boldsymbol{y})$ defined by the vector fields $f\colon \boldsymbol{x} \in \mathcal{X} \to \dot{\boldsymbol{x}} \in T_{\boldsymbol{x}}\mathcal{X}$ and $g\colon \boldsymbol{y} \in \mathcal{Y} \to \dot{\boldsymbol{y}} \in T_{\boldsymbol{y}}\mathcal{Y}$. Suppose that f is globally asymptotically stable, then one can prove the stability of g by proving that it is smoothly equivalent, or diffeomorphic, to f under some $\mathcal{C}^1$-diffeomorphism Φ.

That is if

$$\forall \boldsymbol{y}\colon\ g(\boldsymbol{y}) = J_\Phi(\Phi^{-1}(\boldsymbol{y})).f(\Phi^{-1}(\boldsymbol{y})) \tag{4.2}$$

with J_Φ denoting the Jacobian matrix of Φ, $\frac{\partial \Phi}{\partial \boldsymbol{x}}(\boldsymbol{x})$, holds. To ease notation, the explicit state dependency of the Jacobian is often dropped, when the evaluation point is clear from context. So $J_\Phi(\Phi^{-1}(\boldsymbol{y})).f(\Phi^{-1}(\boldsymbol{y}))$ is for instance typically written as $J_\Phi.f(\Phi^{-1}(\boldsymbol{y}))$.

To put the above statement formally:

Theorem 4.1. *If two DS $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$ and $\dot{\boldsymbol{y}} = g(\boldsymbol{y})$ are smoothly equivalent, then if one is globally asymptotically stable, both are.*

Proof. Let $\Phi\colon \mathcal{X} \to \mathcal{Y}$ be a diffeomorphism such that $\forall \boldsymbol{x} \in \mathcal{X} = \mathbb{R}^d$ we have $g(\Phi(\boldsymbol{x})) = J_\Phi f(\boldsymbol{x})$, which is equivalent to (4.2). For any forward orbit of $f(.)$, i.e. a trajectory $(\boldsymbol{x}(t))_{t\geq 0}$ such that $\frac{\mathrm{d}}{\mathrm{d}\,t}\boldsymbol{x}(t) = f(\boldsymbol{x}(t))$ and $\boldsymbol{x}(0) = \boldsymbol{x}_0$, let us consider its image under Φ, $(\Phi(\boldsymbol{x}(t)))_{t\geq 0}$. We get:

$$\frac{\mathrm{d}}{\mathrm{d}\,t}\Phi(\boldsymbol{x}(t)) = J_\Phi.\dot{\boldsymbol{x}}(t) = J_\Phi.f(\boldsymbol{x}(t)) = g(\Phi(\boldsymbol{x}(t))). \tag{4.3}$$

This implies that $(\Phi(\boldsymbol{x}(t)))_{t\geq 0}$ is a forward orbit of $g(.)$. More generally, any orbit $(\boldsymbol{y}(t))_{t\geq 0}$ of $g(.)$ can be written $(\Phi(\boldsymbol{x}(t)))_{t\geq 0}$, with $\boldsymbol{x}(0) = \Phi^{-1}(\boldsymbol{y}(0))$ and $(\boldsymbol{x}(t))_{t\geq 0}$ orbit of $f(.)$. If f is globally asymptotically stable, then all orbits converge towards the unique equilibrium $\boldsymbol{x}^*$, and thus all orbits $g(.)$ converge towards $\Phi(\boldsymbol{x}^*)$, which proves that $g(.)$ is globally asymptotically stable. A similar demonstration proves the converse implication. □

Using the smooth equivalence, one can make further statements about the properties of the DS. If the dynamical system $\dot{\boldsymbol{y}} = g(\boldsymbol{y})$ is smoothly equivalent to the DS $\boldsymbol{x} = f(\boldsymbol{x})$ with f being $\mathcal{C}^k$ via the C^{l+1}-diffeomorphism Φ, then we know $g(\boldsymbol{y})$ to be $\mathcal{C}^{\min(k,l)}$. This can immediately be deduced from deriving (4.2) using the product rule.

This property is used extensively in the following sections to learn globally stable nonlinear systems.

4.3 Problem Statement and Related Work

In this chapter the problem of learning a dynamical system, or better the associated vector field, from demonstrations is considered. More precisely, given a list of trajectories as the list of tuples $\left(t_{ji},\, \boldsymbol{y}_j(t_i),\, \dot{\boldsymbol{y}}_j(t_i)\right)_j$, observing each demonstration indexed by $_j$ as timed sequences of points and velocities $\boldsymbol{y}, \dot{\boldsymbol{y}} \in \mathbb{R}^d$ at given timed-points indexed by $_i$, the objective is to build a (continuous) autonomous system $\dot{\boldsymbol{y}} = g(\boldsymbol{y})$ (i.e. the vector field $g\colon \mathcal{Y} \to T_{\boldsymbol{y}}\mathcal{Y}$) that reproduces the demonstrations as closely as possible[1].

The ability to construct such DS is an important skill in imitation learning (see for example Schaal et al. [2003]), as they provide an elementary building block to achieve high-level goals. The learned

[1]The choice of denoting a point in the state-space by $\boldsymbol{y}$, instead of for instance $\boldsymbol{x}$, may seem odd at the moment, but facilitates notations in the next sections.

systems can be used as dynamical movement primitives generating goal-directed behaviours (see for instance Ijspeert et al. [2013]), from given data in a very natural way. Modelling movement primitives with DS is convenient for closed loop implementations as the current position of the system is the only input necessary, and their generalization to unseen parts of the state space provides robustness to spatial perturbations. Moreover, the choice of autonomous (i.e. time-invariant) systems, while not always suitable or preferable, is interesting in many situations as they are inherently robust to temporal perturbations.

The most common movement primitives consist of motions that converge towards a single targeted configuration. They therefore naturally correspond to globally asymptotically stable DS as their unique global attractor can be identified with the target configuration. The catch is that classical learning algorithms cannot provide guarantees regarding the global asymptotic stability of their output. They might produce DS showing divergent behaviour, spurious attractors or limit cycles, depending on the initial condition. This issue has recently been studied by, among others, by Mohammad S. Khansari-Zadeh and Aude Billard (see for instance Khansari-Zadeh and Billard [2010]) who proposed several approaches to learn globally asymptotically stable nonlinear DS.

Note that in this chapter we are not only interested in proving global stability of the learned movement, but, as we are concerned with reaching and grasping motions, also seek to guarantee that the learned motions end precisely at the targeted configuration. So we seek to proof that the targeted configuration is the only global attractor of the system, in contrast to approaches like Calinon et al. [2010] which guarantees stability (to some extent) but not that the target configuration and the global attractor coincide.

Basically the approaches of Khansari-Zadeh and others can be divided into two main groups.

The first group fixes a Lyapunov function candidate $V(\boldsymbol{y})$, often taken to be the squared euclidean distance, and trains a model under the constraint that $V(\boldsymbol{y})$ is indeed a (control) Lyapunov function and thereby assuring global asymptotic stability. For instance in Khansari-Zadeh and Billard [2011] an approach called Stable Estimator of Dynamical Systems (SEDS) is presented. Here the utilized model to represent the data is a GMM. As the GMM is a weighted sum of Gaussian components, the regression, with respect to the maximum a posteriori method (MAP), can be interpreted as a weighted sum of linear dynamics, where the weights are nonlinear functions of the position $\boldsymbol{y}$. In order to ensure global asymptotic stability, the linear dynamics induced by each component have to admit $V(\boldsymbol{y})$ as a common Lyapunov function.

The second group seeks to learn a Lyapunov function candidate (also simply called Lyapunov candidate) $V(\boldsymbol{y})$ that is highly compatible with the demonstrations in the following sense: at almost every point $\boldsymbol{y}_j(t_{ji})$, which we also denote by $\boldsymbol{y}_{ji}$ to shorten notations, the estimated or measured velocity $\dot{\boldsymbol{y}}_{ji}$ is such that its scalar product with the gradient of V is negative: $\nabla_{\boldsymbol{y}} V(\boldsymbol{y}_{ji}).\dot{\boldsymbol{y}}_{ji} < 0$. Then standard learning techniques like Gaussian Mixture Models, Locally Weighted Projection Regression (LWPR) or even neural networks can be used for unconstrained learning, resulting in a model that (hopefully) represents well the data, however without stability guarantees. Finally in the last step, the velocity deduced from the model, denoted $\dot{\tilde{\boldsymbol{y}}}$, is modified using an "online" correction signal if it violates the convergence criteria of the learned Lyapunov function. That is if the scalar product of the gradient of the Lyapunov function at $\boldsymbol{y}$ and the estimated velocity from model $\dot{\tilde{\boldsymbol{y}}}$ is positive. This correction signal should not be active or only slightly modify the learned dynamics in the neighbourhood of the demonstrations as we have $\nabla_{\boldsymbol{y}} V(\boldsymbol{y}_{ji}).\dot{\boldsymbol{y}}_{ji} < 0$ for almost all points. This, together with a "good" model of the dynamics, implies that $\dot{\tilde{\boldsymbol{y}}}_{ji} \approx \dot{\boldsymbol{y}}_{ji}$ and therefore $\nabla_{\boldsymbol{y}} V(\boldsymbol{y}_{ji}).\dot{\tilde{\boldsymbol{y}}}_{ji} < 0$ should also hold for almost all points. This approach is presented in Khansari-Zadeh and Billard [2014].

Each of these approaches has its own drawbacks. Incorporating the stability directly into the learning process as done in the first group, complicates the learning. The arising constraints that ensure stability are typically nonlinear and nonconvex, necessitating the utilisation of general nonlinear programming approaches that are susceptible to local minima causing the found optimum to be depending on the initial parameters, as for the approach presented in Khansari-Zadeh and Billard [2011]. Due to these constraints the standard method for training a GMM, the expectation maximization (EM) algorithm,

cannot be applied and instead the optimization is performed using the simplex method (Nelder-Mead-algorithm). We show later why this can be problematic. Secondly the predefined nature of the Lyapunov function candidate induces problems itself. In the case of SEDS, the chosen Lyapunov function candidate is the squared euclidean distance. This on one hand guarantees global asymptotic stability, on the other hand this also means that only movements for which the euclidean distance decreases monotonically along the trajectory are representable, significantly reducing its ability to learn complex motions.

The second group, which aims at separating the learning of the Lyapunov function from the learning of the movement has the drawback that one does not a priori know when the correction will be triggered, resulting in possibly undesired behaviour. Also learning or training Lyapunov functions itself is a complex problem. The conditions necessary for $V(\boldsymbol{y})$ to be considered a Lyapunov function candidate are that it has to be radially unbounded and everywhere strictly positive except at the origin (see section 3.2.2). This comprises a very large class of functions for which no coherent parametrization, such that the set of admissible parameters is for instance convex, can be found. Also Lyapunov function candidates are in general not stable by addition or multiplication, meaning that the sum or product of Lyapunov function candidates does not have to be a Lyapunov candidate, as local extrema might appear. Therefore one has to restrain the search to a well-defined subclass of Lyapunov function candidates, which in turn limits the expressiveness.

In Khansari-Zadeh and Billard [2014] an approach called Control Lyapunov Function based Dynamic Movements (CLF-DM) is presented, belonging to this group. In this approach the class of considered Lyapunov function candidates are weighted sums of asymmetric quadratic functions (WASQF), so

$$V(\boldsymbol{y}) = \sum_i \beta_i(\boldsymbol{y})(\boldsymbol{y} - \boldsymbol{y}_i^*)^{\mathrm{T}}.P_i.(\boldsymbol{y} - \chi_i - \boldsymbol{y}_i^*) \tag{4.4}$$

with $\boldsymbol{y}_i^*$ being the center, χ_i representing the asymmetry and P_i being the weighting matrix for the i-th component which has to be positive definite. Note the additional function $\beta_i(\boldsymbol{y})$ ensuring non-negativity as

$$\beta_i(\boldsymbol{y}) = \begin{cases} 1 \text{ if } (\boldsymbol{y} - \boldsymbol{y}_i^*)^{\mathrm{T}}.P_i.(\boldsymbol{y} - \chi_i - \boldsymbol{y}_i^*) \geq 0. \\ 0 \text{ else} \end{cases} . \tag{4.5}$$

This causes $V(\boldsymbol{y})$ to be "only" $\mathcal{C}^1$, so one time continuously differentiable. The weighting matrix does not necessarily have to be symmetric and due to χ_i adding antisymmetric matrices (matrices for which we have $A^{\mathrm{T}} = -A$) does actually influence the shapes of the level-sets.

This parametrisation facilitates the search for a suitable function as the resulting sum is guaranteed to be a Lyapunov function candidate as long all P_i are definite positive, however this comes at the cost of reducing the expressiveness. For instance all Lyapunov functions that are WASQF are necessarily compatible with the dynamical system $\dot{\boldsymbol{y}} = -\boldsymbol{y}$. Or, to have a better comparison with SEDS, there exists no WASQF Lyapunov function that is compatible with a trajectory whose norm is not monotonically decreasing with respect to the uniform norm $\|\boldsymbol{y}\|_\infty$. Note that the inverse of the above statement is not true, meaning that not all trajectories decreasing with respect to the uniform norm along the trajectory do have a compatible WASQF Lyapunov function. Additionally, each asymmetric quadratic function is convex and therefore the sublevel-set of any WASQF Lyapunov function is a convex set. The search for the best WASQF function given the demonstration is rather efficient due to the following fact: the set of Lyapunov function candidates compatible with a given DS (for instance $\dot{\boldsymbol{y}} = -\boldsymbol{y}$) is a (blunt) convex cone. This follows directly from the definition of a cone: suppose $V_0(\boldsymbol{x})$ and $V_1(\boldsymbol{x})$ are compatible with the dynamical system $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$, so $\nabla_{\boldsymbol{x}} V_0(\boldsymbol{x}).f(\boldsymbol{x}) < 0$ and $\nabla_{\boldsymbol{x}} V_1(\boldsymbol{x}).f(\boldsymbol{x}) < 0$, then $\alpha \nabla_{\boldsymbol{x}} V_0(\boldsymbol{x}).f(\boldsymbol{x}) + \beta \nabla_{\boldsymbol{x}} V_1(\boldsymbol{x}).f(\boldsymbol{x}) < 0$ for any $\alpha, \beta \in \mathbb{R}^+$. Obviously $\alpha V_0(\boldsymbol{x}) + \beta V_1(\boldsymbol{x}) \geq 0$ for all $\boldsymbol{x}$ except $\mathbf{0}$ and any $\alpha, \beta \in \mathbb{R}^+$ also holds as V_0 and V_1 are everywhere strictly positive (except at the origin). Therefore $V(\boldsymbol{x}) = \alpha V_0(\boldsymbol{x}) + \beta V_1(\boldsymbol{x})$ is a Lyapunov function for $f(\boldsymbol{x})$.

There exist also approaches that do not necessarily fall into one of the above categories. In a very recent work, Ravichandar et al. [2017] propose a method to learn globally stable dynamics based on

GMMs and contraction theory. In difference to the approaches cited so far, the optimization process seeks to find a state dependent metric $M(\boldsymbol{y})$ with respect to which the movement has to contract **and** the parameters defining the GMM (and thereby the dynamical system) simultaneously. This comes at the cost of ever increased complexity of the optimization problem due to the larger number of variables (The parametrisation of the metric and the GMM) and the increased number of constraints. The usage of contraction analysis (instead of Lyapunov's stability criterion) has the advantage that it is not only proven that all trajectories converge to the target point, but also converge towards each other (with respect to the metric). This is a very desirable property, especially for GMM representations as here "limit" cases tend to diverge, as shown in Figure 4.1. It is important to recall that this contraction property only holds between trajectories generated by the GMM and not between demonstrations and their replays. Moreover, the complexity of the movement has to be "matched" by the complexity of the metric, in the sense that for a trajectory converging with respect to the l_2-norm, the flat metric $M(\boldsymbol{y}) = Id$ is sufficient, whereas it has to be nonlinear function of the state for more complex motions. As the contraction property only holds with respect to the metric, the distance between trajectories can locally grow in an euclidean sense, somewhat weakening the advantages.

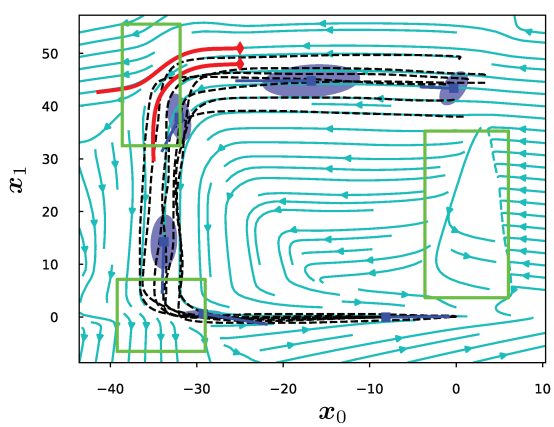


Figure 4.1 – This figure highlights the problem of unconstrained learning using GMM's.

The figure on the left showcases the problems of learning a dynamical system using unconstrained learning. It shows the given demonstrations and velocities (dashed black lines) used to train a GMM (via greedy insertion, see Verbeek et al. [2003]), and the velocity field (cyan streamlines), which is defined as the regression on the GMM by the maximum a posteriori method. The demonstrations used are taken from the "ShaprC"-set, part of the LASA-Dataset. The components of the GMM are qualitatively depicted by the blue ellipsoids, centred at the mean and whose shape corresponds to the covarianz matrix. As expected, the learned vector field is not stable. To remedy this drawback one can search for additional constraints derived from Lyapunov theory. Moreover GMM's tend to create vector fields with regions (bright green boxes) where "neighbouring" initial points generate very different trajectories (red lines). This is (partly) prohibited when relying on stability proofs derived from contraction theory.

The approach yields very good results on commonly used test-cases, which however only comprise multiple demonstrations of the *same movement*, also called unimodal data or demonstration sets. It would be interesting to see its performance for multimodal datasets, as the parametrization of the metric as polynomial (for each element of the matrix) and the necessary symmetry of the matrix favours symmetry of the metric with respect to the origin. Nonetheless this is a very interesting approach which is however, except for the results, hardly comparable to the one presented in the this chapter due to the different theoretical frameworks.

In Manschitz et al. [2018] an approach to learn nonlinear dynamics from demonstration called Mixture of Attractors (MoA) is presented. Even though it can be hardly compared to the other approaches cited so far as it learns time-dependent vector fields, it is an approach worth mentioning as it has several interesting properties. Firstly the learning process is formulated as a *convex* optimization problem and

therefore yields a globally optimal solution without depending on *good* initial guesses for the parameters or heuristics. Secondly it prevents what we term drift-error. In all of the other methods, in a form or another, a statistical model is trained on the given data (position and velocity) and regression is used to obtain an estimator for the *best* or most likely velocity, given a position. As this velocity is then integrated over time, even small, but persistent, errors can lead to large differences in the obtained trajectories. The reproduction drifts away from the demonstration, ending up in possibly unexplored regions of the state space. Formulating integral constraints or costs, that is constraints or costs that penalize the difference between the reproduction and the demonstration, for model learning the velocity is very difficult. To our knowledge MoA is the only approach directly taking into account such a cost during the learning phase, but it does not provide (even for the time-dependent case) any kind of convergence guarantee. In the same article they also show a modified version of their approach to construct a time-independent vector field, but the construction method is prone to produce limit cycles instead of convergent behaviour.

In Neumann and Steil [2015] the, to our knowledge, first approach to use diffeomorphic transformations in the context of learning from demonstration is proposed. The approach can be briefly summarized into the three following steps: First, much like the second group above, one seeks to obtain a highly compatible Lyapunov function $V_{\mathcal{Y}}(\boldsymbol{y})$ for the given dataset. In the second step, a (smooth) diffeomorphic transformation

$$\begin{aligned} \Phi\colon\ & \mathcal{Y} \to \mathcal{X} \\ & \boldsymbol{y} \mapsto \boldsymbol{x} \end{aligned} \tag{4.6}$$

is constructed such that the image of the level-sets in the demonstration space $\{\boldsymbol{y} | V_{\mathcal{Y}}(\boldsymbol{y}) = \alpha\}$ are hyperspheres in the control space $\left\{\boldsymbol{x} | \boldsymbol{x}^{\mathrm{T}}.\boldsymbol{x} = \alpha\right\}$ (see Figure 4.2), with their radius equal to the value of the Lyapunov function. Note that here the diffeomorphism is constructed as transformation from the demonstration space to the control space, whereas within the approach presented in this chapter it is constructed from the control space to the demonstration space. This is worth noting not because it is an actual difference, but in order to avoid confusion regarding the meaning of Φ and Φ^{-1}.

As $V_{\mathcal{Y}}(\boldsymbol{y})$ is a *highly* compatible Lyapunov function for the original data, the transformed data-points $\boldsymbol{x}_{ji} = \Phi(\boldsymbol{y}_{ji})$ and $\dot{\boldsymbol{x}}_{ji} = J_{\Phi}.\dot{\boldsymbol{y}}_{ji}$, with J_{Φ} denoting the Jacobian matrix of Φ, are *highly* compatible with the simple Lyapunov function $V_{\mathcal{X}}(\boldsymbol{x}) = \boldsymbol{x}^{\mathrm{T}}.\boldsymbol{x}$, reducing the complexity of the movement and making it thus easier to learn. This approach can, for instance, be used together with SEDS (called τ-SEDS) and alleviate the constraint that the demonstration space trajectories have to converge with respect to the l_2-norm along the trajectory, as only the control space trajectories are used for training SEDS. The approach in depicted in Figure 4.2.

In order to be able to construct the diffeomorphic transformation Φ from the Lyapunov function $V_{\mathcal{Y}}(\boldsymbol{y})$ in a generic way, not all types of Lyapunov functions are admissible, depending on the construction. Without going into details, the diffeomorphic transformation presented in Neumann and Steil [2015] is given as

$$\boldsymbol{x} = \Phi(\boldsymbol{y}) = \begin{cases} \sqrt{V_{\mathcal{Y}}(\boldsymbol{y})}\frac{\boldsymbol{y}}{\|\boldsymbol{y}\|_2} & \text{if } \boldsymbol{y} \neq \boldsymbol{0} \\ \boldsymbol{0} & \text{else} \end{cases} \tag{4.7}$$

and therefore directly inherits the smoothness properties of the Lyapunov function used for construction, except at the origin. This definition of the diffeomorphism relies on the property that every ray whose initial point is located at the origin intersects only once with any level-set of the Lyapunov function. Otherwise the function would not be injective, which can be seen in Figure 4.3. This forces all level-sets to be "star"-shaped inducing, among others, the restriction that any admissible Lyapunov function has to be everywhere compatible with the dynamical system $\dot{\boldsymbol{y}} = -\boldsymbol{y}$.

In the following sections we present methods to learn globally asymptotically stable nonlinear vector fields from demonstrations, using an in some sense dual approach to τ-SEDS. Instead of inferring a diffeomorphic transformation from a compatible Lyapunov function, we seek directly for a diffeomorphic

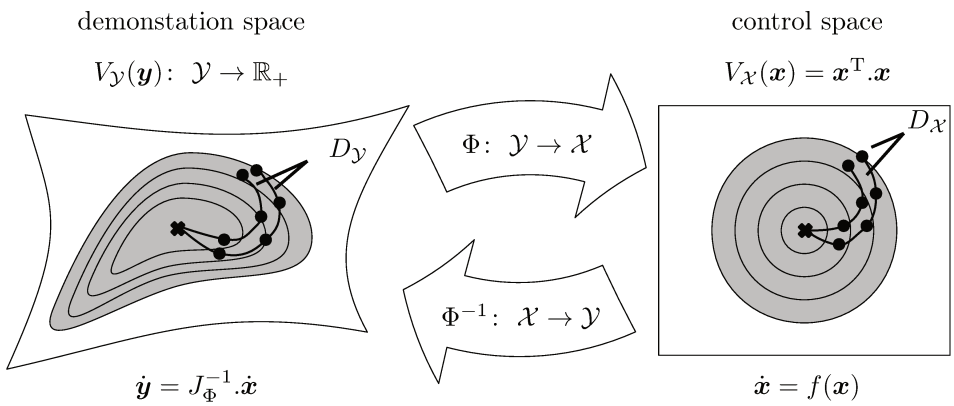


Figure 4.2 – This image depicts the approach of (and is also adapted from) Neumann and Steil [2015]. On the left, the original dataset $D_{\mathcal{Y}}$ and a suitable Lyapunov function $V_{\mathcal{Y}}$ is shown. The diffeomorphism Φ, constructed based upon $V_{\mathcal{Y}}$, maps the level-sets onto circles, shown on the right. Applying Φ onto the dataset $D_{\mathcal{Y}}$, one obtains the transformed dataset $D_{\mathcal{X}}$ which converges, in contrast to $D_{\boldsymbol{y}}$, with respect to the l_2-norm along the trajectories (almost everywhere). Then some adequate model is trained on $D_{\mathcal{X}}$, resulting in the estimation function $f(.)$ with $\dot{\boldsymbol{x}}_{ji} \approx \dot{\tilde{\boldsymbol{x}}} = f(\boldsymbol{x}_{ji})$ and ${\boldsymbol{x}_{ji}}^{\mathrm{T}}.f(\boldsymbol{x}_{ji}) < 0$. Finally the learned velocity in the demonstration space is obtained as $\dot{\tilde{\boldsymbol{y}}} = J_{\Phi}^{-1}.f(\boldsymbol{x}_{ji})$.

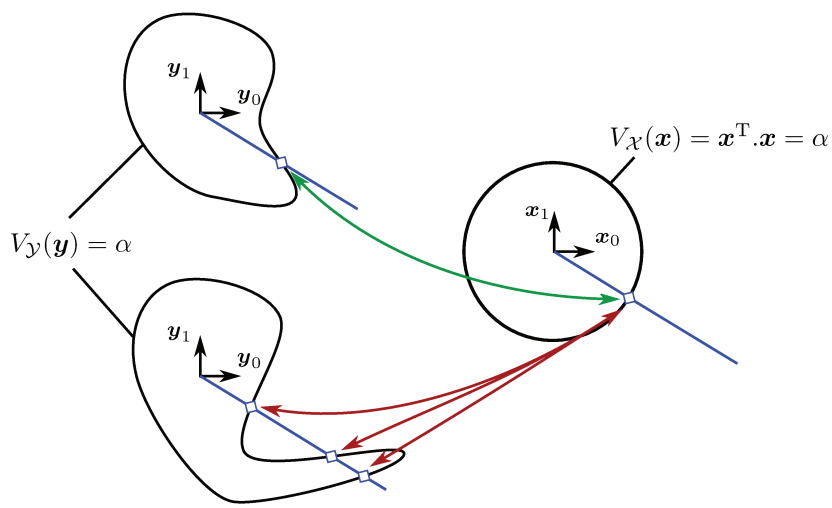


Figure 4.3 – In order for the transformation defined in (4.7) to be a diffeomorphism, each ray with its initial point at the origin may intersect only once with any level-set of the Lyapunov function. This is shown in the upper half of the image. The transformation and its inverse are defined by differentiable one-to-one maps. *More complex* level-set shapes, as shown in the lower half, cause each ray to intersect multiple times and therefore the transformation becomes a many-to-one map, which can not be diffeomorphic.

transformation and can then infer a suitable Lyapunov function based on the transformation. The main idea is to construct a diffeomorphism mapping a "simple" curve, often a straight line, in the control space $\mathcal{X}$ onto the given demonstrations (trajectories in the demonstration space $\mathcal{Y}$). Then a "simple" globally asymptotically vector field can be constructed or learned, which faithfully reproduces these simple curves.

By transposing this vector field into the demonstration space using the diffeomorphism, one obtains a "complex" vector field able to reproduce the demonstrations. As the transformation is diffeomorphic, the global asymptotic stability of the "complex" vector field is guaranteed by the stability of the "simple" one.

In section 4.4 we show how this can be done for fixed pseudo-linear control-space dynamics allowing the representation of a unimodal demonstration set for similar demonstrations. In section 4.5 we extend this approach to unlabelled multimodal demonstration sets showing a higher variance in the demonstrations relying on learned control-space dynamics.

4.4 One-Step Learning

In this section we first introduce a new algorithm for diffeomorphic matching based on smooth symmetric positive definite kernels and compare it with a state-of-the-art algorithm. Then, in the second part, we show how it can be used to map simple curves, which can be easily reproduced by DS known to be globally asymptotically stable like $\dot{\boldsymbol{x}} = -\boldsymbol{x}$ (which we call control-space dynamics) onto the given demonstrations. This gives a new way to generate Lyapunov candidates as well as globally asymptotically stable smooth autonomous systems reproducing the given demonstrations.

Contributions

- A novel method to construct diffeomorphic transformation by composing local diffeomorphic transformations
- An efficient algorithm to construct such transformations
- Showcasing the obtainable results for the LASA-Dataset
- Extensions to multimodal and cyclic demonstrations

The material presented in this section was (in parts) published in Perrin and Schlehuber-Caissier [2016].

4.4.1 Diffeomorphic Locally Weighted Translations

Diffeomorphic transformations are by definition stable by composition, that is given two $\mathcal{C}^k$-diffeomorphism from $\mathbb{R}^d$ onto itself, denoted Ψ_0 and Ψ_1, the transformation Φ defined as

$$\begin{aligned} \Phi = \Psi_1 \circ \Psi_0 \colon \ \mathcal{X} = \mathbb{R}^d &\to \mathcal{Y} = \mathbb{R}^d \\ \boldsymbol{x} &\mapsto \boldsymbol{y} = \Psi_1(\Psi_0(\boldsymbol{x})) \end{aligned} \tag{4.8}$$

is also a $\mathcal{C}^k$-diffeomorphism from $\mathbb{R}^d$ onto itself. This is a very convenient property, but before introducing the algorithm used construct the diffeomorphism we start by introducing its building blocks, the locally weighted translation (LWT).

Given a smooth (symmetric positive definite) kernel function $k_\rho(\boldsymbol{x}, \boldsymbol{x}')\colon \ \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^+$, depending on some parameter ρ, such that $\forall \boldsymbol{x},\ \forall \rho\colon \ \ k_\rho(\boldsymbol{x}, \boldsymbol{x}) = 1$ and $\forall \rho\colon \ k_\rho(\boldsymbol{x}, \boldsymbol{x}') \to 0$ when $\|\boldsymbol{x}' - \boldsymbol{x}\|_2 \to \infty$, given a "translation" $\boldsymbol{v} \in \mathbb{R}^d$ and a center $\boldsymbol{c} \in \mathbb{R}^d$, we consider the following locally weighted translation:

$$\Psi_{\rho,\boldsymbol{c},\boldsymbol{v}} = \boldsymbol{x} + k_\rho(\boldsymbol{x}, \boldsymbol{c})\boldsymbol{v}. \tag{4.9}$$

The so defined transformation is obviously not always a diffeomorphic, so one has to came up with adequate restrictions and ways to proof their validity.

Theorem 4.2. *If*

$$\forall\,(\boldsymbol{x},\boldsymbol{x}')\in\mathbb{R}^d\times\mathbb{R}^d\colon\ \frac{\partial k_\rho}{\partial \boldsymbol{x}}(\boldsymbol{x},\boldsymbol{x}').\boldsymbol{v}>-1 \tag{4.10}$$

then $\Psi_{\rho,\boldsymbol{c},\boldsymbol{v}}$ *is a* $\mathcal{C}^\infty$*-diffeomorphism.*

Proof. For a given $\boldsymbol{y}\in\mathbb{R}^d$, let us try to find $\boldsymbol{x}\in\mathbb{R}^d$ such that $\Psi_{\rho,\boldsymbol{c},\boldsymbol{v}}(\boldsymbol{x})=\boldsymbol{y}$. This can be rewritten $\boldsymbol{x}=\boldsymbol{y}-k_\rho(\boldsymbol{x},\boldsymbol{c})\boldsymbol{v}$, so we know that $\boldsymbol{x}$ must be of the form $\boldsymbol{y}+r\boldsymbol{v}$ for some scalar r. The equation becomes $\Psi_{\rho,\boldsymbol{c},\boldsymbol{v}}(\boldsymbol{y}+r\boldsymbol{v})=\boldsymbol{y}$, i.e.: $r\boldsymbol{v}+k_\rho(\boldsymbol{y}+r\boldsymbol{v},\boldsymbol{c})\boldsymbol{v}=\boldsymbol{0}$. If $\boldsymbol{v}=\boldsymbol{0}$, $\Psi_{\rho,\boldsymbol{c},\boldsymbol{v}}$ is the identity (and a smooth diffeomorphism), and $\boldsymbol{x}=\boldsymbol{y}$. Otherwise, solving $\Psi_{\rho,\boldsymbol{c},\boldsymbol{v}}(\boldsymbol{x})=\boldsymbol{y}$ amounts to solving $r+k_\rho(\boldsymbol{y}+r\boldsymbol{v},\boldsymbol{c})=0$.

Let us define:

$$\begin{aligned} h_{\boldsymbol{y}}:\ &\mathbb{R}\to\mathbb{R}\\ &r\mapsto r+k_\rho(\boldsymbol{y}+r\boldsymbol{v},\boldsymbol{c}). \end{aligned} \tag{4.11}$$

If $\frac{\partial k_\rho}{\partial \boldsymbol{x}}(\boldsymbol{x},\boldsymbol{x}').\boldsymbol{v}\geq-1$, we get: $\forall r\in\mathbb{R}$, $\frac{\mathrm{d}}{\mathrm{d}\,r}h_{\boldsymbol{y}}>0$. Because of the absolute monotonicity of $h_{\boldsymbol{y}}$, and since $h_{\boldsymbol{y}}(r)$ tends to $-\infty$ when r tends to $-\infty$, and to $+\infty$ when r tends to $+\infty$, we deduce that there exists a unique scalar value $s_{\rho,\boldsymbol{c},\boldsymbol{v}}(\boldsymbol{y})\in\mathbb{R}$ such that $h_{\boldsymbol{y}}(s_{\rho,\boldsymbol{c},\boldsymbol{v}}(\boldsymbol{y}))=0$. It follows that the equation $\Psi_{\rho,\boldsymbol{c},\boldsymbol{v}}(\boldsymbol{x})=\boldsymbol{y}$ has a unique solution:

$$\boldsymbol{x}=\boldsymbol{y}+s_{\rho,\boldsymbol{c},\boldsymbol{v}}(\boldsymbol{x})\boldsymbol{v}.$$

We conclude that $\Psi_{\rho,\boldsymbol{c},\boldsymbol{v}}$ is invertible, and:

$$\Psi^{-1}_{\rho,\boldsymbol{c},\boldsymbol{v}}(\boldsymbol{y})=\boldsymbol{y}+s_{\rho,\boldsymbol{c},\boldsymbol{v}}(\boldsymbol{x})\boldsymbol{v}. \tag{4.12}$$

The above proves that Ψ is indeed a bijection. As Ψ has $\mathcal{C}^\infty$ smoothness, the implicit function theorem can be applied to prove that $s_{\rho,\boldsymbol{c},\boldsymbol{v}}(\boldsymbol{y})$ is smooth, and as a consequence $\Psi_{\rho,\boldsymbol{c},\boldsymbol{v}}$ is a smooth diffeomorphism. □

Gaussian Radial Basis Function

In the first part of this chapter, we rely on Gaussian Radial Basis functions (also simply called Gaussian kernels), due to their smoothness and succesfull application in many areas, including other diffeomorphic matching algorithms like Glaunes et al. [2004]. The resulting deformation is showcased in Figure 4.4.

The Gaussian Radial Basis function is defined as

$$k_\rho(\boldsymbol{x},\boldsymbol{x}')=\exp\left(-\rho^2\left\|(\boldsymbol{x}-\boldsymbol{x}')\right\|_2^2\right) \tag{4.13}$$

and therefore

$$\frac{\partial k_\rho}{\partial \boldsymbol{x}}(\boldsymbol{x},\boldsymbol{x}').\boldsymbol{v}=-2\rho^2\exp\left(-\rho^2\left\|(\boldsymbol{x}-\boldsymbol{x}')\right\|_2^2\right)(\boldsymbol{x}-\boldsymbol{x}')^{\mathrm{T}}.\boldsymbol{v}$$

with the lower bound

$$-2\rho^2\exp\left(-\rho^2\left\|(\boldsymbol{x}-\boldsymbol{x}')\right\|_2^2\right)\left\|\boldsymbol{x}-\boldsymbol{x}'\right\|_2\left\|\boldsymbol{v}\right\|_2\leq-2\rho^2\exp\left(-\rho^2\left\|(\boldsymbol{x}-\boldsymbol{x}')\right\|_2^2\right)(\boldsymbol{x}-\boldsymbol{x}')^{\mathrm{T}}.\boldsymbol{v}.$$

After replacing $\left\|\boldsymbol{x}-\boldsymbol{x}'\right\|_2$ with δx to shorten notations, we derive the above expression with respect to it to find an extrema by solving

$$-4\rho^4\delta x\exp\left(-\rho^2\delta x^2\right)\delta x\left\|\boldsymbol{v}\right\|_2-2\rho^2\exp\left(-\rho^2\delta x^2\right)\left\|\boldsymbol{v}\right\|_2=0.$$

This equation is equal to zero for $\delta x=\frac{1}{\sqrt{2}\rho}$ and considering the strictly monotonic decrease of k_ρ, one can conclude that it is a minima.

This minima then yields the lower bound

$$\frac{\partial k_\rho}{\partial \boldsymbol{x}}(\boldsymbol{x}, \boldsymbol{x}').\boldsymbol{v} \geq -\sqrt{2}\,\|\boldsymbol{v}\|_2\,\rho \exp\left(-\frac{1}{2}\right). \tag{4.14}$$

Finally, by applying Theorem 4.2, one can conclude that the locally weighted translation is a smooth diffeomorphic transformation if either $\boldsymbol{v} = \boldsymbol{0}$ or

$$\rho < \rho_{Max}(\|\boldsymbol{v}\|_2) = \frac{1}{\sqrt{2}\,\|\boldsymbol{v}\|_2} \exp\left(1/2\right). \tag{4.15}$$

This provides a tractable condition that $\Psi_{\rho,\boldsymbol{c},\boldsymbol{v}}$ is a bijection. As moreover $\Psi_{\rho,\boldsymbol{c},\boldsymbol{v}}$ is $\mathcal{C}^\infty$, one can deduce that it is indeed a $\mathcal{C}^\infty$-diffeomorphism.

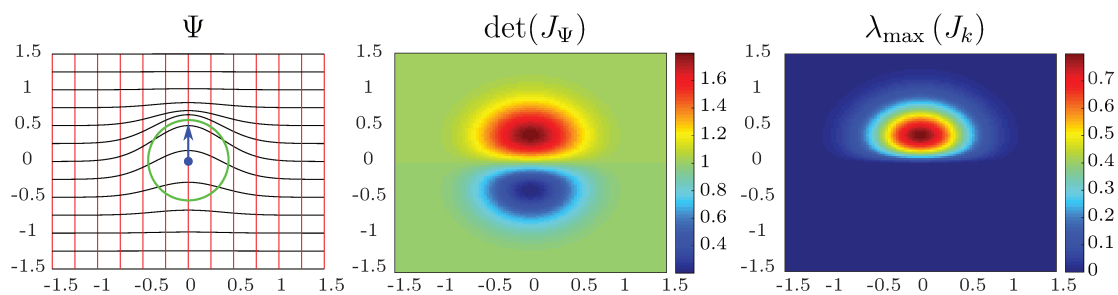


Figure 4.4 – The image on the left depicts the transformation resulting from applying one locally weighted translation in $\mathbb{R}^2$ with a Gaussian kernel centred at the origin ($\boldsymbol{c} = \boldsymbol{0}$), the translation vector defined as $\boldsymbol{v} = \begin{bmatrix}0. & 0.5\end{bmatrix}^{\mathrm{T}}$ and the coefficient ρ chosen as $0.8\rho_{\mathrm{Max}}(\boldsymbol{v}) = 1.87$ onto a regular rectangular grid. The value of the Gaussian kernel is depicted by the green circle, on which $k_\rho(\boldsymbol{x}, \boldsymbol{c}) = 0.5$. In the middle image the determinant of the Jacobian is shown. One can see that it is everywhere strictly positive. Note that values smaller than one for the determinant of the Jacobian indicate dilatation, whereas values larger than one correspond to compression. On the right, the maximal eigenvalue of $J_k = \boldsymbol{v}.\frac{\partial k_\rho}{\partial \boldsymbol{x}}(\boldsymbol{x}, \boldsymbol{c})$ is shown. As one can see the maximal value attained is 0.8 (corresponding to the definition of ρ as $0.8\rho_{\mathrm{Max}}$). On the lower half of the image the only nonzero eigenvalue is smaller than zero.

Computing Ψ^{-1}

The locally weighted transform $\Psi_{\rho,\boldsymbol{c},\boldsymbol{v}}$ is a diffeomorphic transformation if either $\boldsymbol{v} = \boldsymbol{0}$ or $\rho < \rho_{\mathrm{Max}}(\|\boldsymbol{v}\|_2)$. This only proves the existence of the inverse transformation Ψ^{-1}, but does not provide an efficient way to compute it. Due to the occurrence of $\boldsymbol{x}$ as such and in the kernel function within Φ^{-1}, no analytic formula can be given for the inverse function. However, as we have seen in section 4.4.1, finding Ψ^{-1} is equivalent to solving $r + k_\rho(\boldsymbol{x} + r\boldsymbol{v}, \boldsymbol{c}) = 0$. This is a problem of optimizing one bounded variable, as we have $-1 \leq r \leq 0$.

For the Gaussian kernel one obtains

$$h_{\boldsymbol{y}} = r + \exp\left(-\rho^2\,\|\boldsymbol{y} + r\boldsymbol{v} - \boldsymbol{c}\|_2^2\right) = 0$$

and the derivative

$$\frac{\mathrm{d}}{\mathrm{d}\,r} h_{\boldsymbol{y}} = h'_{\boldsymbol{y}} = 1 - 2\rho^2(\boldsymbol{y} + r\boldsymbol{v} - \boldsymbol{c})^{\mathrm{T}}.\boldsymbol{v} \exp\left(-\rho^2\,\|\boldsymbol{y} + r\boldsymbol{v} - \boldsymbol{c}\|_2^2\right)$$

The function $h_{\boldsymbol{y}}$ is strictly monotone and r is bounded between -1 and 0, therefore one can rely on Newton's method to efficiently compute the ϵ-close inverse of Ψ as described in Algorithm 3.

Algorithm 3 $\boldsymbol{x} = \Psi^{-1}(\boldsymbol{y})$

```
Input y, ρ, c
Parameter ϵ > 0
Initialize r = 0.
while |h_y(r)| > ϵ do
    r ← r − h_y(r) / h'_y(r)
    r ← max(−1., min(r, 0.))
end while
Return x = y + r v
```

Algorithm 3 is the usual Newton method, with the difference that the bounds are explicitly enforced to achieve faster convergence. Initializing r to zero, instead of the median of the admissible region -0.5 for example, proves to be more efficient, as the kernel is acting only locally and therefore the chances are higher that the point will not be significantly influenced by this kernel than the opposite.

4.4.2 Diffeomorphic Matching

The problem of diffeomorphic point matching poses the following problem: given two sequences of N pair-wise distinct points of the same dimension d, $X = (\boldsymbol{x}_i)_{0 \leq i \leq N-1}$ and $Y = (\boldsymbol{y}_i)_{0 \leq i \leq N-1}$, compute a diffeomorphic transformation Φ that maps each $\boldsymbol{x}_i$ onto the $\boldsymbol{y}_i$, either exactly or approximately. By pair-wise distinct points we mean that $\boldsymbol{x}_i = \boldsymbol{x}_j$ for some $i \neq j$ implies that $\boldsymbol{y}_i = \boldsymbol{y}_j$. Or, to put it more formally, by defining some cost or distance function $\mathrm{dist}(X, Y)$ between two point sequences holding the same number of points having the same dimension and by denoting $\Phi(X)$ the point sequence $(\Phi(\boldsymbol{x}_i))_i$, we seek to find the diffeomorphism that minimizes $\mathrm{dist}\,(\Phi(X), Y)$. We often call the sequence X the source sequence or curve and Y the target sequence or curve.

State-of-the-Art

Most of the research concerning diffeomorphic matching is related to medical imaging. Within this context it is often used to match different images of organs deformed under mechanical stress together. This stress is often caused by the imaging process itself, as for instance ultrasound imaging, and can therefore not be avoided. As neither an *analytical* model nor the necessary applied forces are available to compute the elastic deformation and thereby correct the image, matching such images via diffeomorphic transformations has proven to be very successful, see for instance Sotiras et al. [2013]. This is partly due to the inherent property of diffeomorphisms to conserve topologies. In the case of medical imaging, this means that organs or tissue segments can neither be dissected nor glued together by the image treatment process.

The diffeomorphic matching approaches used within the context of medical imaging can be divided into methods to match points, lines or surfaces. As we use diffeomorphic matching within the context of learning from demonstration, the distance between two successive points is an important property, as these points have corresponding time-stamps. So the distance between these points is directly linked to the current velocity on the trajectory. Therefore the only method applicable in this context is point matching. Since we have no a priori knowledge about the shapes of the curves to match, the, to the best of our knowledge, state-of-the-art techniques to solve this problem are based on the Large Deformation Diffeomorphic Metric Mapping (LDDMM) framework introduced in the seminal article by Joshi and Miller [2000]. The core idea of this approach is to work with a time-dependent vector field $v(\boldsymbol{x}, t) \in \mathbb{R}^d$ with $t \in [0, 1]$ and define a flow $\Phi(\boldsymbol{x}, t)$ via the transport equation:

$$\frac{\mathrm{d}}{\mathrm{d}\,t}\Phi(\boldsymbol{x}, t) = v(\Phi(\boldsymbol{x}, t), t) \tag{4.16}$$

with the initial condition $\Phi(\boldsymbol{x}, 0) = \boldsymbol{x}$. Under very mild regularity conditions of the vector field $v(\boldsymbol{x}, t)$, such as that the velocity has to be continuous in $\boldsymbol{x}$ and its partial derivatives with respect to $\boldsymbol{x}$ have to be finite (see Dupuis et al. [1998] for details) the resulting transformation

$$\begin{aligned} \Phi\colon\ \mathbb{R}^d \times [0,1] &\to \mathbb{R}^d \\ \boldsymbol{x}, t &\mapsto \boldsymbol{y} = \Phi(\boldsymbol{x}, t) \end{aligned} \tag{4.17}$$

is indeed a diffeomorphism. This diffeomorphism also inherits the smoothness property of the generating vector field, meaning that if the partial derivatives of $v(\boldsymbol{x}, t)$ with respect to $\boldsymbol{x}$ are $\mathcal{C}^k$, then the resulting diffeomorphism is a $\mathcal{C}^k$-diffeomorphism given by

$$\Phi(\boldsymbol{x}) = \boldsymbol{x} + \int_0^1 v(\Phi(\boldsymbol{x}, t), t)\, dt. \tag{4.18}$$

Using an appropriate Hilbert space for the vector field $v(\boldsymbol{x}, t)$ and interpreting the point sequences as a discrete distribution given as a weighted sum of Dirac measures, one obtains a variational formulation for a cost function taking into account, and therefore performing a trade-off between, the regularity of the resulting transformation and the matching of the point pairs. This problem can then be solved using gradient descent algorithms, as done in Glaunes et al. [2004]. Note that the regularity of a diffeomorphic transformation is difficult to quantify and we use this term (within the context of describing a diffeomorphism) in a very informal way. We say that a diffeomorphic transformation is regular, if the kernels have an appropriate minimal size compared to the problem and there is some well identifiable margin for it being inversible.

This formulation has several advantages like being able to efficiently trade-off between the regularity of the transformation and matching accuracy. Also this formulation allows for different types of cost functions measuring the similarity between diffeomorphic geometrical objects and not just point sequences.

Nonetheless this formulation also entrails significant disadvantages. As Φ is not given as a closed-form solution, evaluating it requires an integration which can be slighty time-comsuming. As our interest is using the diffeomorphic transformation within the closed loop controller of a robotic system, fast evaluation of Φ and Φ^{-1} is crucial. Moreover, the complexity of the approach proposed in Glaunes et al. [2004] scales badly with the number of points in the provided dataset for two reasons. Firstly, the complexity of constructing the time-dependent vector field is at least cubic in the number of data points. This is not desirable, but acceptable as it is offline computation time and therefore not crucial. The real disadvantage concerns the structure of the vector field $v(\boldsymbol{x}, t)$, as it is given as

$$v(\boldsymbol{x}, t) = \sum_i k(\boldsymbol{x}, \Phi(\boldsymbol{x}_i, t))\boldsymbol{v}_i(t). \tag{4.19}$$

This means that to each point in the sequence X a time-dependent vector field of the form $v_i(\boldsymbol{x}, t) = k(\boldsymbol{x}, \Phi(\boldsymbol{x}_i, t))\boldsymbol{v}_i(t)$ is associated, defined by the symmetric positive definite kernel k centred at the current image of $\boldsymbol{x}_i$ and the current velocity $\boldsymbol{v}_i(t)$. The total vector field is then simply the sum over the individual ones. Therefore the evaluation time of Φ and Φ^{-1} scales linearly with the number of points which is not admissible in our context.

Fast Diffeomorphic Matching

We propose a completely different approach to construct a diffeomorphic transfomation (approximatively) matching two point sequences based on the locally weighted translations presented in section 4.4.1. These translations and their inverse can be evaluated extremely quickly which, together with constant evaluation complexity with respect to the number of points in the sequences, allows it to be used within a closed loop controller alleviating some of the drawbacks of LDDMM.

The main idea is that, as each locally weighted translation defines a simple diffeomorphic transformation, many such transformations can be composed to obtain more complex ones in an iterative process. Therefore we fix a number of iterations K, and two parameters $0 < \mu < 1$ and $0 < \beta < 2$. K is defined empirically, as the number of iterations required for a good approximation depends on the intrinsic difficulty of the problem. That is it depends on the difference between the source and target sequences. μ can be interpreted as a "safety margin": strictly less than 1, it ensures that each resulting LWT cannot be arbitrarily close to being non-invertible. β is similar to a learning rate: a small value allows only small modifications at every iteration, whereas large values force larger changes and kernels.

Initially we define $Z = X$ and at every iteration Z is updated with a newly constructed LWT. The j-th iteration constructing the j-th LWT can be summed up into the following four steps, that can also be found in the pseudo-code Algorithm 4:

1. select the center $\boldsymbol{c}_j$ as the point $\boldsymbol{z}_l$ in Z that is the furthest away from the corresponding target point $\boldsymbol{y}_l$ in Y
2. select the translation $\boldsymbol{v}_j = \beta(\boldsymbol{y}_l - \boldsymbol{z}_l)$
3. construct the j-th LWT $\Psi_{\rho_j,\boldsymbol{c}_j,\boldsymbol{v}_j}$ by optimizing the size of the kernel ρ_j such that it minimizes $\operatorname{dist}\left(\Psi_{\rho_j,\boldsymbol{c}_j,\boldsymbol{v}_j}(Z), Y\right)$ subject to $0 \leq \rho_j < \rho_{\text{Max}}(\boldsymbol{v}_j)$ ensuring that it is diffeomorphic
4. perform the update $Z = \Psi_{\rho_j,\boldsymbol{c}_j,\boldsymbol{v}_j}(Z)$

The resulting (smooth) diffeomorphism Φ is the composition of all LWT constructed as described above, so

$$\Phi = \Psi_{\rho_{K-1},\boldsymbol{c}_{K-1},\boldsymbol{v}_{K-1}} \circ \Psi_{\rho_{K-2},\boldsymbol{c}_{K-2},\boldsymbol{v}_{K-2}} \circ \cdots \circ \Psi_{\rho_1,\boldsymbol{c}_1,\boldsymbol{v}_1} \circ \Psi_{\rho_0,\boldsymbol{c}_0,\boldsymbol{v}_0}. \tag{4.20}$$

Algorithm 4 Construct Φ

Input $X = (\boldsymbol{x}_i)_i$,$Y = (\boldsymbol{y}_i)_i$
Parameters $K \in \mathbb{N}^+$, $\mu \in [0, 1[$, $\beta \in]0, 2[$
Initialize $Z = X$
for $j \leftarrow 0$ to $K - 1$ **do**
    $l \leftarrow \operatorname{argmax}_i \left(\|\boldsymbol{z}_i - \boldsymbol{y}_i\|_2\right)$
    $\boldsymbol{c}_j \leftarrow \boldsymbol{z}_l$
    $\boldsymbol{v}_j \leftarrow \beta\left(\boldsymbol{y}_l - \boldsymbol{z}_l\right)$
    $\rho_j \leftarrow \operatorname*{argmin}_{\rho \in [0, \mu\rho_{\text{Max}}(\boldsymbol{v}_j)]} \operatorname{dist}\left(\Psi_{\rho,\boldsymbol{c}_j,\boldsymbol{v}_j}(Z), Y\right)$
    $Z \leftarrow \Psi_{\rho_j,\boldsymbol{c}_j,\boldsymbol{v}_j}(Z)$
end for
return $(\rho_j, \boldsymbol{c}_j, \boldsymbol{v}_j)_j$

Before presenting the results of this algorithm graphically, we would like to give some additional remarks and insights gained from experience.

Remark 4.1. Here we have presented a version of the algorithm in which the parameter μ and β are constant, but we can also make them vary from one iteration to another. By experience, matching performance for the LASA-Dataset was increased by interpolating β linearly between 0.3 for the first iteration and 0.7 for the last iteration. Varying μ has less influence and is therefore usually kept constant with values ranging from 0.5 to 0.9, where smaller values lead to increased regularity, whereas higher values might speed up convergence if the underlying trajectories (represented by the point sequence) have higher curvatures.

Remark 4.2. In line 8, the distance function denoted dist was not yet properly defined. A natural choice for it is to use the sum of the squared errors, so $\operatorname{dist}(X,Y) = \sum_i \|\boldsymbol{x}_i - \boldsymbol{y}_i\|_2^2$, however it has proven more efficient to use the largest singular value norm of $(X - Y) = [\boldsymbol{x}_i - \boldsymbol{y}_i]_i$ taken as a matrix in $\mathbb{R}^{d\times N}$. This can be explained by the fact that the largest singular value norm is a better measure for the improvement in the direction of $\boldsymbol{v}_j$.

Remark 4.3. Also line 8 of the algorithm performs a nonlinear optimization, but it depends only on one bounded real variable, so a minimum can be found very quickly and precisely. There is no guarantee that there exist no local minimum for the admissible range of ρ, however it seems to be the case in practice. We can add a fixed upper bound $\rho_M > 0$, so that all $\rho_j \in [0, \min(\rho_M, \rho_{\text{Max}}(\boldsymbol{v}_j))[$, and add a regularization term in the cost of the optimization problem of line 8, to prevent the diffeomorphism from overly deforming the space by using kernels approaching a Dirac, to get a perfect matching. For the examples later on we used the cost function $1/N \operatorname{dist}\left(\Psi_{\rho,\boldsymbol{c}_j,\boldsymbol{v}_j}(Z), Y\right) - \epsilon\rho_j^2$, with $\epsilon > 0$ being the trade-off parameter between better matching and the use of larger kernels. Simply using inputs with a dense representation (large value of N) has a similar effect (and it barely slows the algorithm down).

Remark 4.4. Nothing prevents the algorithm from getting stuck in a local minimum, so a general proof of convergence cannot be found. However, as we show in the next sections, experimental results give empirical evidence that the algorithm is efficient and converges quickly in practice, even on difficult matching problems. An exception to this are matching problems between straight lines and spirals in $2d$ or their equivalent in higher dimensions. In this case the algorithm will inevitable be stuck in a bad local minima as shown in Figure 4.5. Note that flow based methods also fail in such cases.

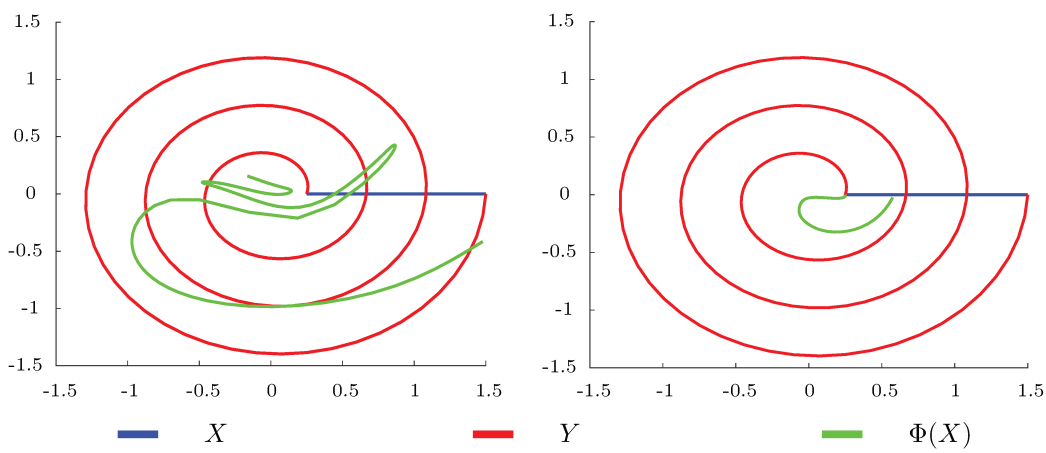


Figure 4.5 – In this figure the matching results for spiral-like trajectories are shown. The matching result for the MATCHINE software is shown on the left, the result for the proposed approach is shown the right. In both cases the source and target curves are given as a sequence of 150 points and both methods fail to find a meaningful matching, highlighting the intrinsic difficulty of such trajectories.

Remark 4.5. If taking the scaling factor β larger than 2, the algorithm will diverge for any concrete matching problem as this is equivalent to enlarge and change the sign of the maximal error at each iteration. Scaling factors larger than one are rarely useful as they correspond to an oscillatory convergence (see Figure 4.6). In practice useful values for β range between 0.2 and 0.8.

Remark 4.6. The safety margin μ is directly related to the smallest occurring eigenvalue of the Jacobian by $\lambda_{\min}(J_\Psi) = 1 - \mu$. By composing K such LWT within the diffeomorphism Φ, the worst-case safety margin that Φ is invertible is $(1-\mu)^K$. This however never happens in practice as the LWT are distributed over the point sequences and the regions of the smallest eigenvalues do usually not coincide. Nonetheless the safety margin can get small even in realistic applications as seen on the top right image of Figure 4.7.

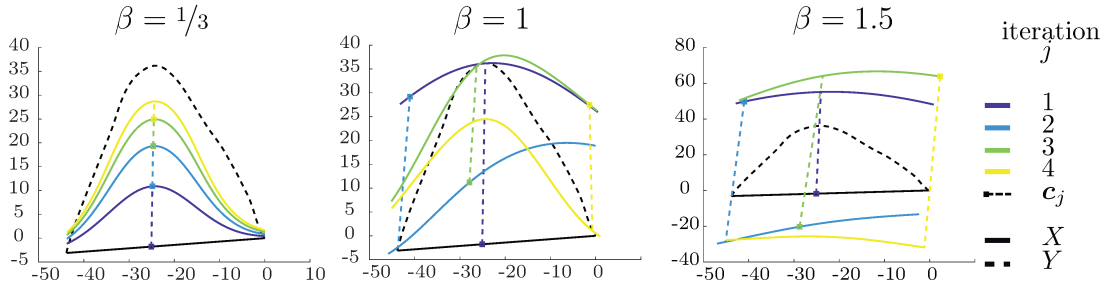


Figure 4.6 – The images show the first four iterations of Algorithm 4 for a simple matching problem and different values of β. A value of $\beta = 1.5$ causes Z to "oscillate" and the large values of $\|\boldsymbol{v}_j\|_2$ force small values for ρ and therefore the source points are mainly translated. Choosing $\beta = 1/3$ the iterated point-sequence Z converges "monotonically" towards the target sequence Y. For this setting $\beta = 1/3$ clearly induces the most desirable behaviour.

The top three gridlines, initially parallel to the $\boldsymbol{x}_0$-axis (black line top left image), almost coincide in the region marked by the blue circle. This is equivalent to the Jacobian having small eigenvalues.

Remark 4.7. As shown in Figure 4.7, the algorithm performs better when the length of the source curve (by interpreting the point sequence X as discretized curve) is shorter than the length of the target curve. This is due to the tendency to create pleats when seeking to compress regions. These pleats then generate local minima and cannot be undone by Algorithm 4.

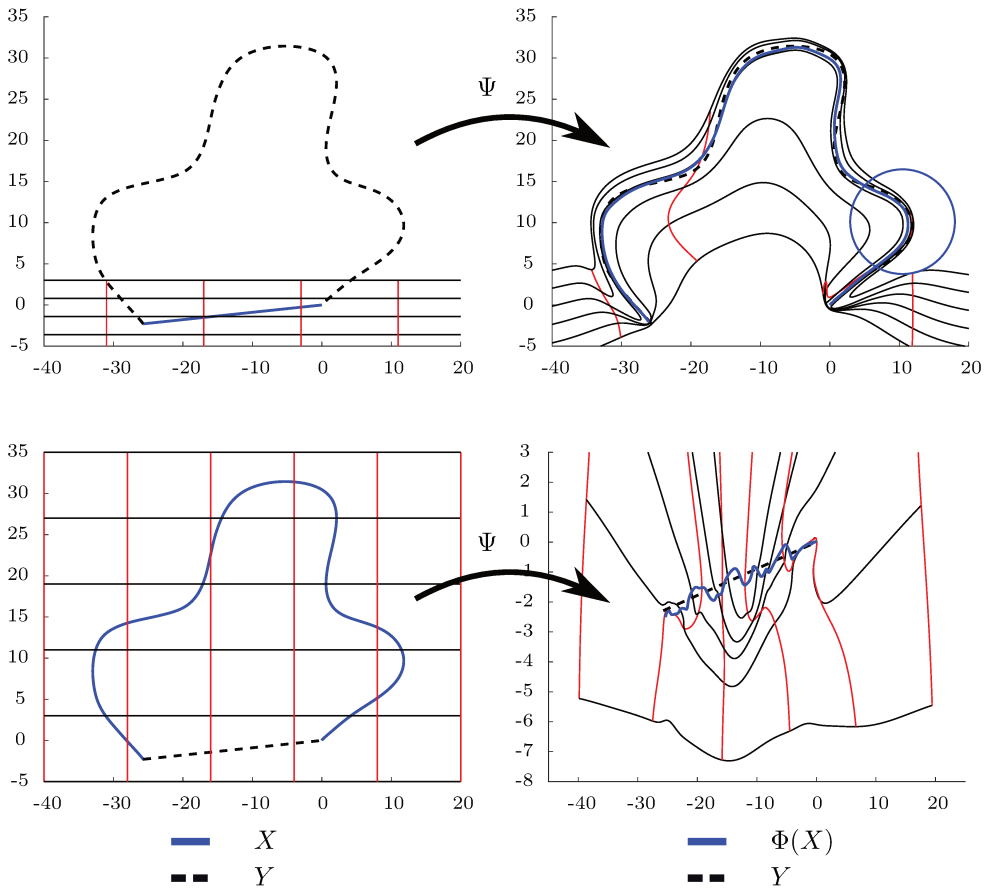


Figure 4.7 – The initial configuration is depicted in the left column, the blue line corresponds to the source sequence and the dashed black line is the target sequence. The right column depicts the image of the source sequence (blue line) and the regular grid under the transformation Φ. As one can see the image of the source curve in the top row is very smooth and matches well the target sequence. In the bottom row, the image of the source sequence shows many pleats and is very irregular, indicating that stretching the distances between the points is *easier* than compressing them. As one can see (blue circle) due to the sequential composition the minimal guaranteed safety margin to non-invertible transformations can become very small. This results in a highly deformed space.

Experimental Evaluation

We compared the proposed Algorithm 4 to a publicly available implementation for diffeomorphic matching in the LDDMM framework developed by J. Glaunès (the "MATCHINE" software Glaunes [2005])

Given a sequence of points $Y = (\boldsymbol{y}_i)_{i\in[0,N-1]}$ representing a trajectory, we set $X = (\boldsymbol{x}_i)_{i\in[0,N-1]}$ with

$$\boldsymbol{x}_i = \boldsymbol{y}_0 + \frac{i}{N}(\boldsymbol{y}_N - \boldsymbol{y}_0)$$

and applied our algorithm or the LDDMM algorithm to construct a diffeomorphism Φ such that $\Phi(X)$ and Y match. As shown in the next section, the definition of the source sequence as linear interpolation between the start and end-point of the target trajectory is not arbitrary, but is exactly our use-case. The results of the proposed algorithm for some 2D-trajectories taken from the LASA-Dataset. In Table 4.1 a comparison of the results obtained on these trajectories with our algorithm and the LDDMM algorithm is shown. Each trajectory, was represented as sequences of 20, 50 and 100 points (i.e. $N = 20$, $N = 50$, $N = 100$). For both algorithms, the same parameters were kept across all the trials, notably the number of LWTs or iterations for Algorithm 4 was set to 150. In all cases, our algorithm provided a substantial speedup. For example, with $N = 50$ (a very small number of data points for statistical learning, Φ was learned in average 58 times faster and, more importantly, evaluated 240 times faster, while the error $\text{dist}(\Phi(X), Y)$ was 2.67 times smaller. The tests were made on an Intel(R) Core(TM) i7-4700MQ @ 2.4 GHz with 4 GB of RAM, with both approaches implemented in Matlab(R).

	N	Our algorithm	LDDMM
Learning: average duration of the construction of Φ	20	0.25s	2.78s
	50	0.25s	14.5s
	100	0.26s	53.3s
Forward evaluation: average duration of computing $\Phi(\mathcal{X})$	20	3.05ms	157ms
	50	3.35ms	804ms
	100	3.72ms	3130ms
Backward evaluation: average duration of computing $\Phi^{-1}(\mathcal{X})$	20	29.8ms	145ms
	50	35ms	798ms
	100	38.5ms	3110ms
Accuracy of mapping: average value of $\text{dist}(\Phi(\mathcal{X}), Y)$	20	3.49×10^{-3}	18.2×10^{-3}
	50	8.32×10^{-3}	22.2×10^{-3}
	100	9.51×10^{-3}	22.0×10^{-3}
Generalization: average value of $\text{dist}(\Phi(\mathcal{X}_{1000}), Y_{1000})$	20	19.8×10^{-3}	20.3×10^{-3}
	50	9.51×10^{-3}	21.6×10^{-3}
	100	11.5×10^{-3}	22.3×10^{-3}

Table 4.1 – Comparison of experimental results for the 4 examples ("Sine", "Snake", "hee", "Leaf$_2$") of the LASA-Dataset. Remark: standard deviations are negligible for our algorithm: it is deterministic, and the computation times depend almost entirely on the input size (N) and on the fixed number of iterations (K). Y is obtained by subsampling from an initial recording of 1000 points: Y_{1000}. X_{1000} is the linear progression from $\boldsymbol{y}_0$ to $\boldsymbol{y}_{999}$. To get a sense of how precisely the mapping generalizes around the set of training points, we compute $\text{dist}(\Phi(X_{1000}), Y_{1000})$. We observe that for $N = 50$ and $N = 100$, our results are about twice as accurate as the ones obtained with the algorithm based on LDDMM.

4.4.3 Learning Globally Asymptotically Stable Nonlinear Dynamical Systems

In this section, we show how a diffeomorphic matching algorithm can be used to learn globally asymptotically stable DS that reproduce the demonstrated trajectories. We show how the introduction of the control space, demonstration space and a diffeomorphism between them can simplify stability proofs and allows for the reproduction of highly complex trajectories.

Remark 4.8. We only consider locally Lipschitz dynamical systems, which is compatible with the aim of learning trajectories for robotic systems. Secondly we assume, without loss of generality, that the equilibrium point of the dynamical system is at the origin, in order to ease notations.

Remark 4.9. We consider only diffeomorphisms for which we have $\Phi(\mathbf{0}) = \mathbf{0}$, or in words, diffeomorphisms that are equivalent to identity at the origin. As we suppose that all demonstrations converge to the origin, this ensures that all trajectories of the dynamical system converge to the same point and that this point coincides with the target of the demonstrations. To enforce this, an additional transformation is added to the diffeomorphism found using Algorithm 4 with $\boldsymbol{v}_K = -\Phi(\mathbf{0})$ and $\boldsymbol{c}_K = \Phi(\mathbf{0})$. Nonetheless, all proofs presented also hold for the general case, by introducing the equilibrium points $\boldsymbol{x}^*$ and $\boldsymbol{y}^*$ with $\boldsymbol{y}^* = \Phi(\boldsymbol{x}^*)$.

Recall from section 3.2.2 that a function has to be everywhere strictly positive except at the origin, be zero at the origin and be radially unbounded in order to be called Lyapunov function candidate[2]. Additionally we demand in this section that the function has no other local extrema than the origin and we denote them by $V(.)$. This Lyapunov function candidate is called a Lyapunov function for the DS $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$, if they are compatible, that is if

$$\forall \boldsymbol{x} \in \mathbb{R}^d \setminus \{\mathbf{0}\} : \ \nabla_{\boldsymbol{x}} V(\boldsymbol{x}).f(\boldsymbol{x}) < 0.$$

The demand that a Lyapunov function candidate has no other extrema makes it stronger than the usual definition, but necessary when using the Lyapunov candidate as a stepping-stone to construct the GAS dynamical system, as, due to the vanishing gradient at the extrema the compatibility is numerically not tractable.

In the last sections we have seen that Algorithm 4 is able to construct a $\mathcal{C}^\infty$-diffeomorphism matching straight line segments (represented by an equally spaced point sequence) onto a demonstrated trajectory (represented by a point sequence of the same length). We have also shown that global asymptotic stability is preserved under diffeomorphic transformations (smooth equivalence). Therefore if one could construct a globally asymptotically stable dynamical system, which we call control-space dynamics denoted as $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$, whose forward trajectory is precisely this line segment, then we can use the diffeomorphism found beforehand to deduce the globally asymptotically stable demonstration-space dynamics using the smooth equivalence. We denote this DS as $\dot{\boldsymbol{y}} = g(\boldsymbol{y})$ and its relation to the $f(\boldsymbol{x})$ is $g(\boldsymbol{y}) = J_\Phi.f(\Phi^{-1}(\boldsymbol{y}))$. If the matching is exact, then the trajectory generated by $g(\boldsymbol{y})$ starting at the initial point of the demonstrated trajectory will coincide with it. Moreover, with the diffeomorphism being $\mathcal{C}^\infty$, the demonstration-space dynamics is guaranteed to have the same smoothness as the control-space dynamics (but with different Lipschitz constants). In the following we show how to construct such control-space dynamics, deduce compatible Lyapunov functions in the demonstration space and provide preliminary results.

Control-Space Dynamics

The forward trajectories of the control-space have to correspond to the straight line segment between starting point $\boldsymbol{x}_0 = \boldsymbol{x}(0)$ and the end point $\boldsymbol{x}_T = \mathbf{0}$ of the trajectory, which is, without loss of generality, terminating at the origin after T seconds. The prototype of the control space dynamics is therefore given

[2]Note that here we are always interested in proving global asymptotic or even exponential stability and therefore the term Lyapunov function candidate is adjust accordingly

as

$$\dot{\boldsymbol{x}} = \tilde{f}(\boldsymbol{x}) = -\gamma \frac{A.\boldsymbol{x}}{\|A.\boldsymbol{x}\|_2}. \tag{4.21}$$

Defined as such, the matrix $A \in \mathbb{R}^{d\times d}$ defines the direction of the velocity and $\gamma \in \mathbb{R}^+$ defines the speed. By taking

$$\gamma = \frac{\|\boldsymbol{x}_0\|_2}{T} \tag{4.22}$$

and $A = Id_d$, we obtain a dynamical system whose trajectory from any starting point is a straight line to the origin and the time needed to attain the origin from $\boldsymbol{x}_0$ is precisely T. The origin is a global attractor, but it is not an equilibrium point, as $\lim_{\|\boldsymbol{x}\|_2 \to 0} \tilde{f}(\boldsymbol{x})$ is ill-defined. To remedy this problem we define a r-ball around the origin within which the norm of the velocity is gradually reduced, so

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}) = \begin{cases} \tilde{f}(\boldsymbol{x}) & \text{if } \|\boldsymbol{x}\|_2 > r \\ \frac{\|\boldsymbol{x}\|_2}{r} \tilde{f}(\boldsymbol{x}) & \text{else} \end{cases}. \tag{4.23}$$

Defining $f(\boldsymbol{x})$ this way, it is obviously globally asymptotically stable and $\|\boldsymbol{x}\|_2^2$ is a Lyapunov function. Also, by choosing r sufficiently small, the difference between the point sequence used to construct the diffeomorphism and the forward trajectory of $\boldsymbol{x}_0$ under f is negligible. This configuration is shown in the middle row of Figure 4.8.

In a similar fashion, we can construct control-space dynamics providing additional properties. Consider the rotation matrix R (unitary matrix in $\mathbb{R}^{d\times d}$) defined as

$$R = \begin{bmatrix} \frac{\boldsymbol{x}_0}{\|x_0\|_2} & \text{null}(\boldsymbol{x}_0{}^{\mathrm{T}}) \end{bmatrix} \tag{4.24}$$

where null denotes the base of the nullspace of a matrix. This causes $R.\boldsymbol{e}_0$ with $\boldsymbol{e}_0 = \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix}^{\mathrm{T}}$ to point into the direction of $\boldsymbol{x}_0$. Now we define the matrix A according to

$$A = R.\,\text{diag}(\begin{bmatrix} 1. & \lambda' & \cdots & \lambda' \end{bmatrix}).R^{\mathrm{T}} \tag{4.25}$$

where diag constructs a diagonal matrix from a vector. If choosing λ' strictly larger then zero, than $f(\boldsymbol{x})$ is again globally asymptotically stable with $\|\boldsymbol{x}\|_2^2$ being a Lyapunov function as (considering the case outside the r-ball around the origin, but the results also hold within):

$$\nabla_{\boldsymbol{x}} V(\boldsymbol{x}).f(\boldsymbol{x}) = \boldsymbol{x}^{\mathrm{T}}.\left(Id.\left(-\gamma \frac{A.\boldsymbol{x}}{\|A.\boldsymbol{x}\|_2}\right) + \left(-\gamma \frac{A.\boldsymbol{x}}{\|A.\boldsymbol{x}\|_2}\right)^{\mathrm{T}}.Id \right).\boldsymbol{x} < 0. \tag{4.26}$$

By multiplying with the non-negative term $\|A.\boldsymbol{x}\|_2$ ($\boldsymbol{x} \neq \boldsymbol{0}$), dividing by the strictly positive term γ and considering that A is symmetric, one obtains

$$\boldsymbol{x}^{\mathrm{T}}.(-A).\boldsymbol{x} < 0 \tag{4.27}$$

By chosing λ' strictly larger than zero, then $-A$ is symmetric definite negative and the inequality holds everywhere except at the origin as demanded.

Defining $f(\boldsymbol{x})$ this way has the advantage that by setting λ' to specific values we can influence the global behaviour of the system. Choosing λ' larger than 1, causes the system to converge quickly towards the straight line going through $\boldsymbol{x}_0$ and the origin, whereas values smaller than 1 cause the trajectories to keep the distance to the straight line and be "parallel" to it. This behaviour is also transposed into the demonstration space and therefore provides a effective tool to counter the drift error. If the matching via Φ is perfect, then larger λ' values cause the state to converge to the demonstration, whereas smaller values usually cause better generalization of the movement into the regions surrounding the demonstration, as shown in the top and bottom row of Figure 4.8.

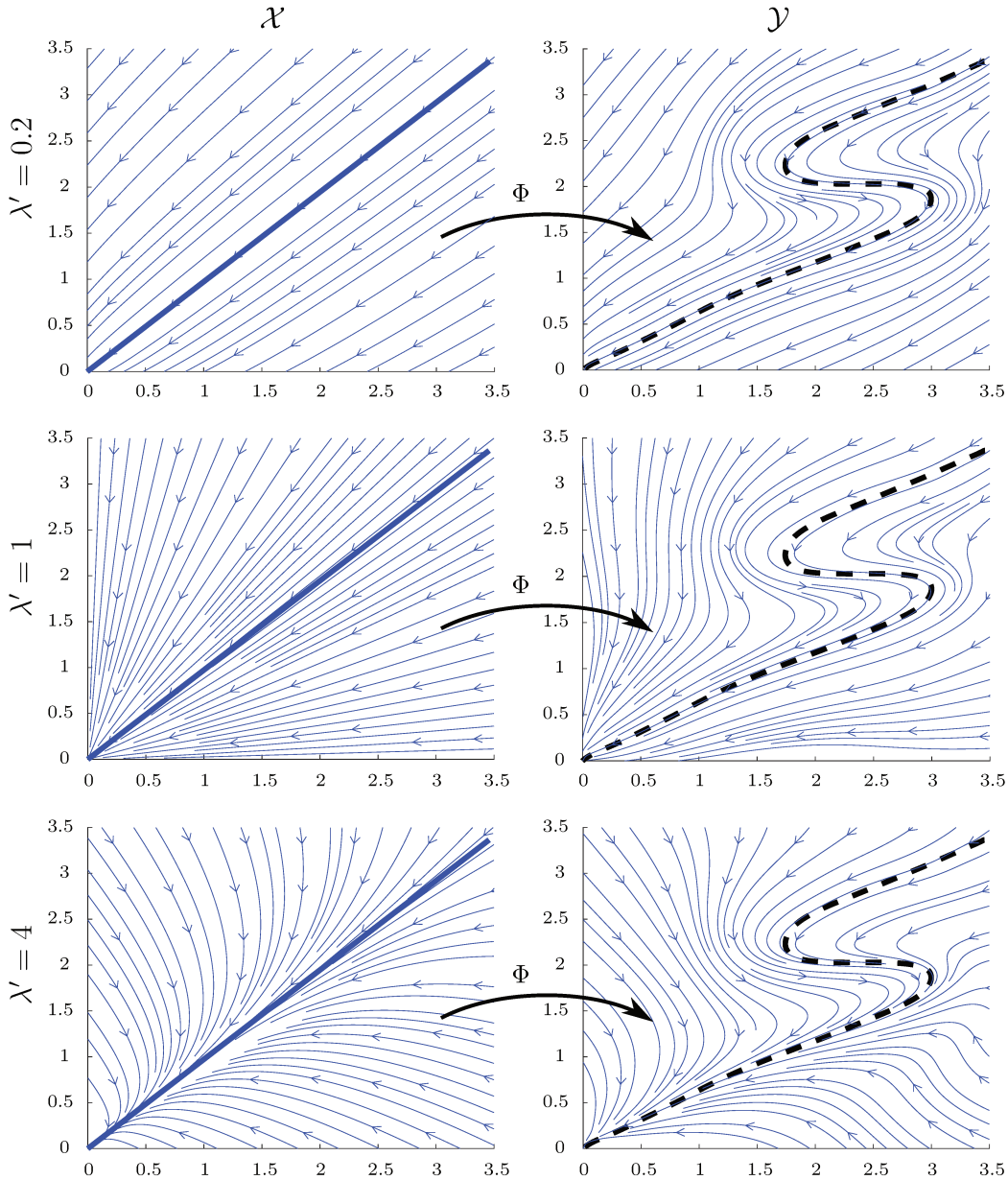


Figure 4.8 – This figure shows the influence of λ' on the control- and demonstration-space dynamics. The diffeomorphism Φ is the same for all three rows. The value for λ' increases from top to bottom and one can see how this gives priority to converging to the trajectory over converging towards the origin.

Lyapunov Functions

The prototypic control-space dynamics defined in (4.23) all admit $\|\boldsymbol{x}\|_2^2$ as Lyapunov function, we denote control-space Lyapunov function (candidates) by $V_{\mathcal{X}}(\boldsymbol{x})$. Now we show how demonstration-space Lyapunov functions, denoted $V_{\mathcal{Y}}(\boldsymbol{y})$, can be derived from the control-space Lyapunov functions and the diffeomorphism Φ and showcase their expressiveness.

Theorem 4.3. *Let $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$ and $\dot{\boldsymbol{y}} = g(\boldsymbol{y})$ be smoothly equivalent via the $\mathcal{C}^1$-diffeomorphism $\Phi\colon \boldsymbol{x} \in$*

$\mathcal{X} \mapsto \boldsymbol{y} \in \mathcal{Y}$. *If* $V_{\mathcal{X}}(\boldsymbol{x})$ *is a Lyapunov function for* $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$ *, then* $V_{\mathcal{Y}} = V_{\mathcal{X}} \circ \Phi^{-1}$ *is a Lyapunov function for* $\dot{\boldsymbol{y}} = g(\boldsymbol{y})$.

Proof. As $V_{\mathcal{X}}$ is a Lyapunov function and therefore also a Lyapunov function candidate, it can immediately be deduced that $V_{\mathcal{Y}}$ is also a Lyapunov function candidate, since Φ is a one-to-one correspondence. I.e. if there exists a $\boldsymbol{y}' \neq \mathbf{0}$ for which $V_{\mathcal{Y}}$ would be either zero or a local extrema, then one could immediately deduce that $\boldsymbol{x}' = \Phi^{-1}(\boldsymbol{y}')$ is a point for which $V_{\mathcal{X}}$ is either zero or a local extrema. As no such point can exist, $V_{\mathcal{Y}}$ is a Lyapunov function candidate.

For $V_{\mathcal{Y}}$ to be a Lyapunov function, we need to have

$$V_{\mathcal{Y}}(\boldsymbol{y}) = \nabla_{\boldsymbol{y}} V_{\mathcal{Y}}.g(\boldsymbol{y}) < 0$$

for all $\boldsymbol{y} \neq \mathbf{0}$. This however is equivalent to (chain rule)

$$\nabla_{\boldsymbol{x}} V_{\mathcal{X}}(\Phi^{-1}(\boldsymbol{y})).\nabla_{\boldsymbol{y}} \Phi^{-1}.J_{\Phi}.f(\Phi^{-1}(\boldsymbol{y})) = \nabla_{\boldsymbol{x}} V_{\mathcal{X}}(\boldsymbol{x}).f(\boldsymbol{x})$$

which we know to be strictly negative as $V_{\mathcal{X}}$ is a Lyapunov function for $f(\boldsymbol{x})$. □

This provides a convenient way to determine Lyapunov functions in the demonstration space, which are highly adapted to the given demonstration as seen in Figure 4.9.

The prototypic control-space dynamic corresponds to a scaled linear system and from basic control theory we know that linear systems are stable iff the system matrix is Hurwitz. In section 3.2 we have seen that the Hurwitz-criterion is equivalent to proving the existence of a symmetric definite positive matrix P for which $P.A + A^{\mathrm{T}}.P$ is symmetric definite negative and that the quadratic function $\boldsymbol{x}^{\mathrm{T}}.P.\boldsymbol{x}$ is indeed a Lyapunov function in this case. This holds for the linear, but also for the scaled linear system. Now the interesting question is, can we construct an *optimal* Lyapunov function for the prototypic control-space dynamics?

If we consider for the moment the linear system, then by defining optimal as having the fastest exponential convergence, so

$$P.A + A^{\mathrm{T}}.P \preceq -\gamma^* P$$

with $\gamma^* > 0$ being the largest value for which the inequality holds, then we see that

$$2P.A \preceq -\gamma^* P$$

as $A^{\mathrm{T}} = A$ and by left-multiplying with P^{-1} we get $2A \preceq -\gamma Id$.

This indicates that for symmetric system matrices, the convergence rate for quadratic Lyapunov functions is independent of the matrix P. This is an interesting fact, but also means that all sdp-matrices are *optimal* in this sense, providing no way to chose a particular one.

Therefore let's reconsider the scaled linear dynamics $f(x)$. By defining optimality of the Lyapunov function as the smallest angle between the inward pointing normal vector on the level-set (equal to $-{}^{P.\boldsymbol{x}}/_{\|P.\boldsymbol{x}\|_2}$) and $\dot{\boldsymbol{x}}$, one obtains a criterion for ranking the control-space Lyapunov functions. This criterion is very natural, as it prefers Lyapunov functions for which the velocity points steeper into the sublevel-set. This angle is the smallest (equal to zero) if these vectors point into the same direction, or in other words, if $\dot{\boldsymbol{x}}$ is perpendicular to the level-set. The direction of the scaled linear dynamics is $A.\boldsymbol{x}$, the direction of the (inward pointing) normal vector is $-P.\boldsymbol{x}$. Therefore if we identify P with $-A$ we obtain the optimal Lyapunov function (Note that it is not just the optimal **quadratic** Lyapunov function, but truly the optimal Lyapunov function among all Lyapunov function candidates). This, and the obtained demonstration-space Lyapunov functions, is showcased in Figure 4.9.

Another way to look at the resulting diffeomorphism and demonstration-space dynamics is to compare it with the diffeomorphic construction of navigation functions in Rimon and Koditschek [1991]. A navigation function in this work is basically a Lyapunov function candidate defined inside the unit-sphere, with

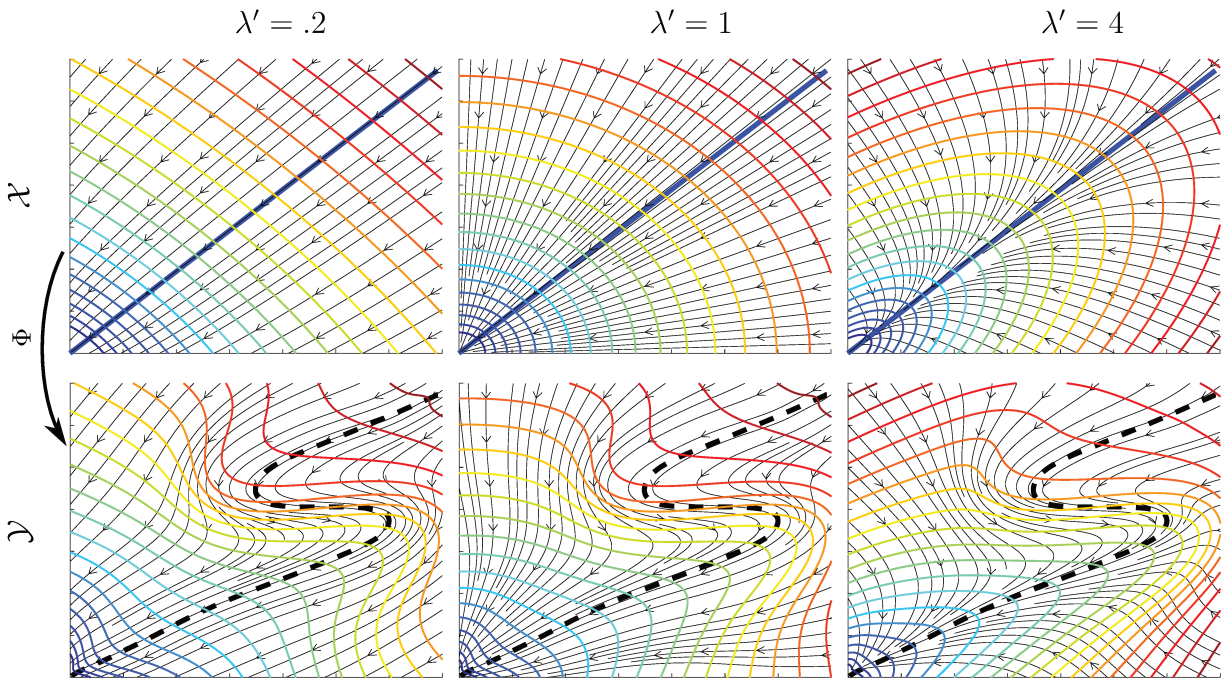


Figure 4.9 – This figure showcases the demonstration-space Lyapunov functions defined as $V_{\mathcal{Y}} = V_{\mathcal{X}} \circ \Phi^{-1}$. For all cases of λ', the optimal Lyapunov function is chosen, and the middle column corresponds to the case where the control-space Lyapunov function is $\|\boldsymbol{x}\|_2^2$ ($\lambda' = 1$). The diffeomorphism is the same as the one used in Figure 4.7. As one can see the optimality of the Lyapunov function in the control space is, to some extent, transposed into the demonstration space. However, due to the higher gradient of the diffeomorphism in comparison to the Lyapunov function, the optimality is less obvious.

the additional restriction that it takes its maximal value (arbitrarily chosen equal to 1) on the boundary of the unit-sphere and the boundary of all obstacles (all obstacles are hyperspheres strictly included inside the unit-sphere). Such a function can be analytically constructed for a *Euclidean sphere world*, see Rimon and Koditschek [1991], page 93. Then a diffeomorphic transformation is given in its analytical form to transform the sphere world into a star world, representing the real environmental. Here, similar to the constraints for τ-SEDS, only diffeomorphisms between spheres and star-shaped sets (given as level-sets of positive definite homogeneous polynomials) can be computed. If then, during execution, the velocity of the system to be navigated is chosen such that the value of the navigation function declines, one is guaranteed to avoid the obstacles and converge to the origin/target. So here too the desired properties, encoded as navigation/Lyapunov function, are conserved under a diffeomorphic transformation and the diffeomorphism helps to augment the expressiveness of the overall approach. In this work, the matching algorithm could readily replace the analytic construction of the diffeomorphism and alleviate the necessity of the target world being star-shaped. It would be sufficient if the obstacles is the target world do not intersect.

4.4.4 Results and Numerical Evaluation

In this section the results obtained by generating the demonstration-space dynamics based on the prototypic control-space dynamics (section 4.4.3) and the diffeomorphic matching (Algorithm 4) are compared to the ones obtained using τ-SEDS using WASQF Lyapunov functions. Note that the way the diffeomor-

phism is constructed in τ-SEDS, necessitates all level-sets of the Lyapunov functions to be star-shaped. WASQF function do satisfy the even stronger criteria of having convex level-sets, as one can see in the examples showing the results for the LASA-Dataset, see Figure 4.11

Results for the LASA-Dataset and Comparison with τ-SEDS

The LASA-Dataset (Khansari-Zadeh and Billard [2011]) is a publicly available database for testing and comparing different learning from demonstration approaches. It is composed of several datasets representing handwritten symbols, so 2D-trajectories. Most of them being unimodal, so showing only one symbol per dataset, which can be learned by the proposed approach in the following way: all trajectories in the LASA-Dataset are given as a timed sequence of 1000 points, so in the first step a *meta-demonstration* is created by averaging over all demonstrated trajectories as they show the same symbol. So each point in the target sequence $Y = (\boldsymbol{y}_i)_i$ is defined as

$$\boldsymbol{y} = \frac{1}{N} \sum_{j=0}^{N-1} \boldsymbol{y}_{ji} \tag{4.28}$$

where N is the number of demonstrations available, which are indexed by j. This way a representative trajectory (the meta-demonstration) is obtained. Finally we scale each dimension, such that the variance of all points is equal to one, separately for each dimension. Doing this, improves the results obtained, as the locally weighted translations used as building blocks for the diffeomorphism show a radial symmetry. This scaling can be seen as another diffeomorphic transformation, as long as the demonstrations are not degenerated, in the sense that the variance for each dimension is non-zero. Then we proceed to construct the source sequence the same way as described in section 4.4.2 as an equally spaced point sequence representing the straight line segment between the initial point of the meta-demonstration and the origin, denoted X, as shown in Figure 4.10. Finally the diffeomorphism Φ and the control-space dynamics are constructed based on Y and X. The parameter λ' is fixed beforehand and good results are typically obtained for values between 1.5 and 5.

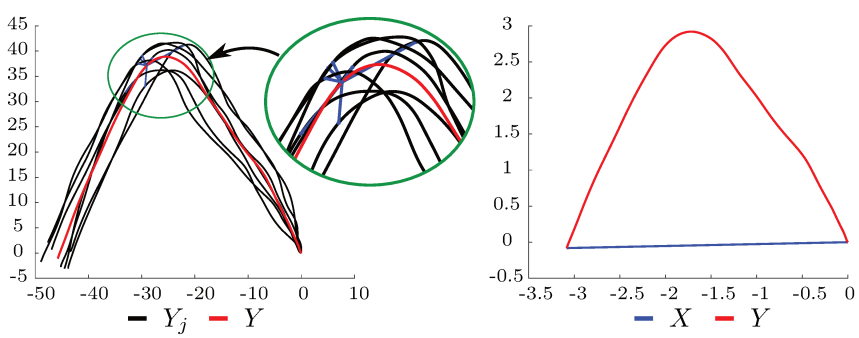


Figure 4.10 – On the left the averaging over all given demonstrations (Y_j, black lines) to obtain the meta-demonstration (Y, red line) is shown. In the zoom, all lines connecting $\boldsymbol{y}_i$ and all $\boldsymbol{y}_{ji}$ for some i are shown in blue. On the right the scaled version of Y and X used to construct the diffeomorphism are shown.

As one can see in Figure 4.11, the proposed approach allows to efficiently learn very expressive Lyapunov functions from demonstrations, without the restriction of them being compatible with some predefined dynamical system. For instance the Lyapunov functions in rows 2,4 and 6 are not compatible with $\dot{\boldsymbol{y}} = -\boldsymbol{y}$ and could therefore not be used within the τ-SEDS approach.

Another advantage is the, comparatively, high smoothness of the obtained vector field. SEDS uses Gaussian mixture models to learn the data under the additional constraints ensuring that $\|\boldsymbol{y}\|_2^2$ is a

common Lyapunov function for all components. These constraints are not compatible with the EM-algorithm, which is undoubtedly the most popular way to train GMMs, but one has to rely on gradient descent methods from general nonlinear programming. This often causes the centers of the Gaussians to be *far* away from the demonstrations, causing the weights for some kernels to approach machine precision, preventing a smooth blending of the components. This is especially striking in the 4th row.

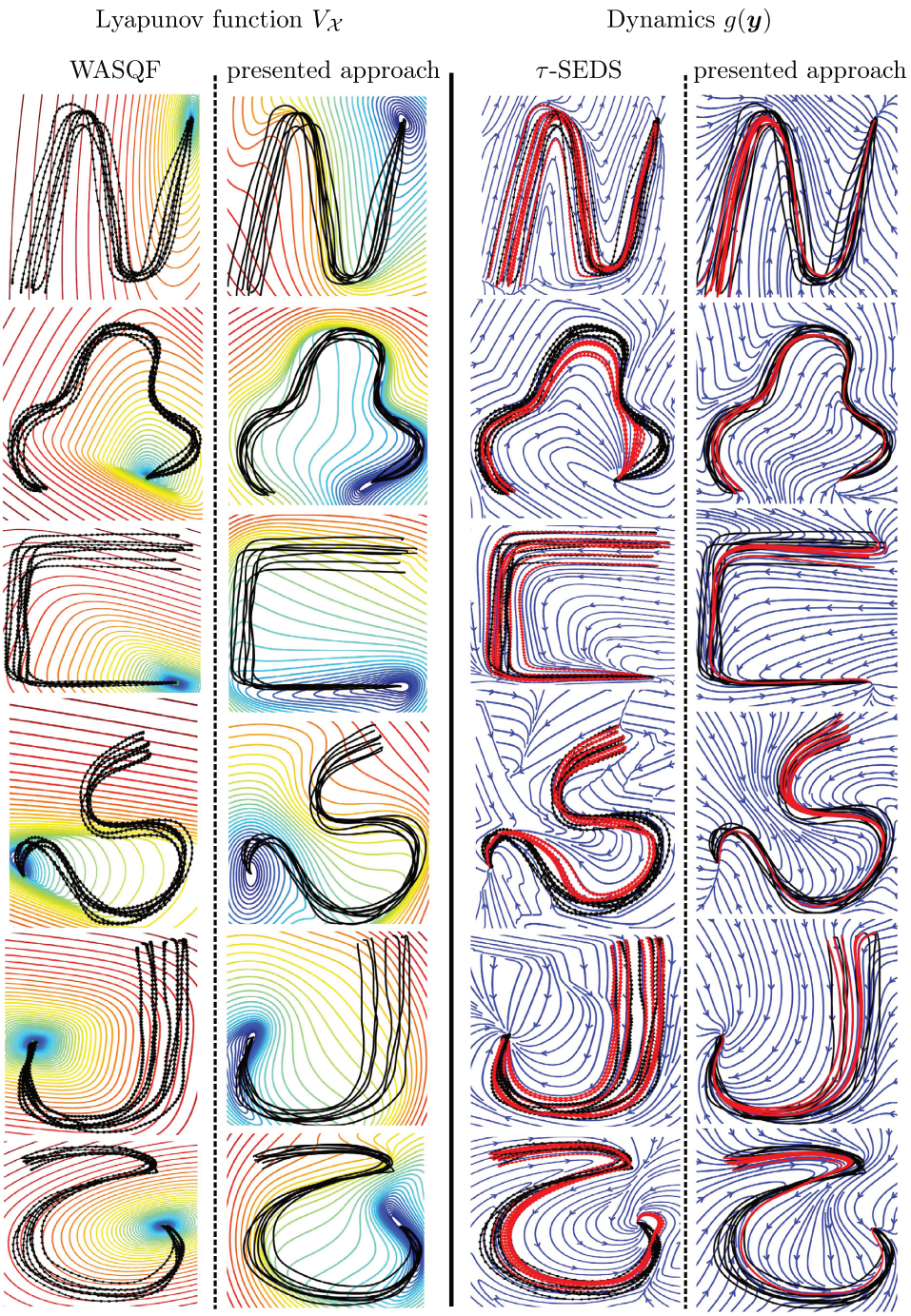


Figure 4.11 – In this figure the results for some of the demonstrations contained in the LASA-Dataset are shown and compared with the results obtained using τ-SEDS with WASQF (Images in the columns "WASQF" and "τ-SEDS" taken from Neumann and Steil [2015]). On the left the resulting demonstration-space Lyapunov functions are shown. On the right, the given demonstrations (black lines) the reproductions from the learned dynamics (red lines) and the streamlines (blue lines) are shown for both approaches.

Evaluating the Dynamics

In the last section it was shown that the learned demonstration-space dynamics is able to represent complex motions. However so far we have only seen that it is able to recreate the shape or geometry of the given trajectories, but have not yet taken a closer look at the velocity profile.

Unsurprisingly, the velocity profile for demonstrations which are very similar to the meta-demonstration are well matched. In fact if the match were perfect, the velocity profiles would inevitably match to due to the smooth equivalence via a $\mathcal{C}^\infty$-diffeomorphism. More surprisingly, the velocity profile is also "well" generalized into some neighbourhood of the meta-demonstration. A common problem is a "time-delay" occurring at the beginning of the trajectory, see for instance the trajectory indicated by the green arrow bottom left image in Figure 4.12. This is caused by a usually high deformation in that area and the trajectories starting farther away from the meta-demonstration need some transition time to attain the neighbourhood of the meta-demonstration into which the movement is well generalised. The approach reaches its limits when the variance in the initial positions is comparable to the size of the trajectories, so if the bounding box containing all initial positions is, in at least one dimension, of comparable size to the bounding box containing all demonstrations. In this case the ability of the proposed approach to generalise the movement is often not sufficient, as in the case of the "WShape" (bottom-right in Figure 4.12).

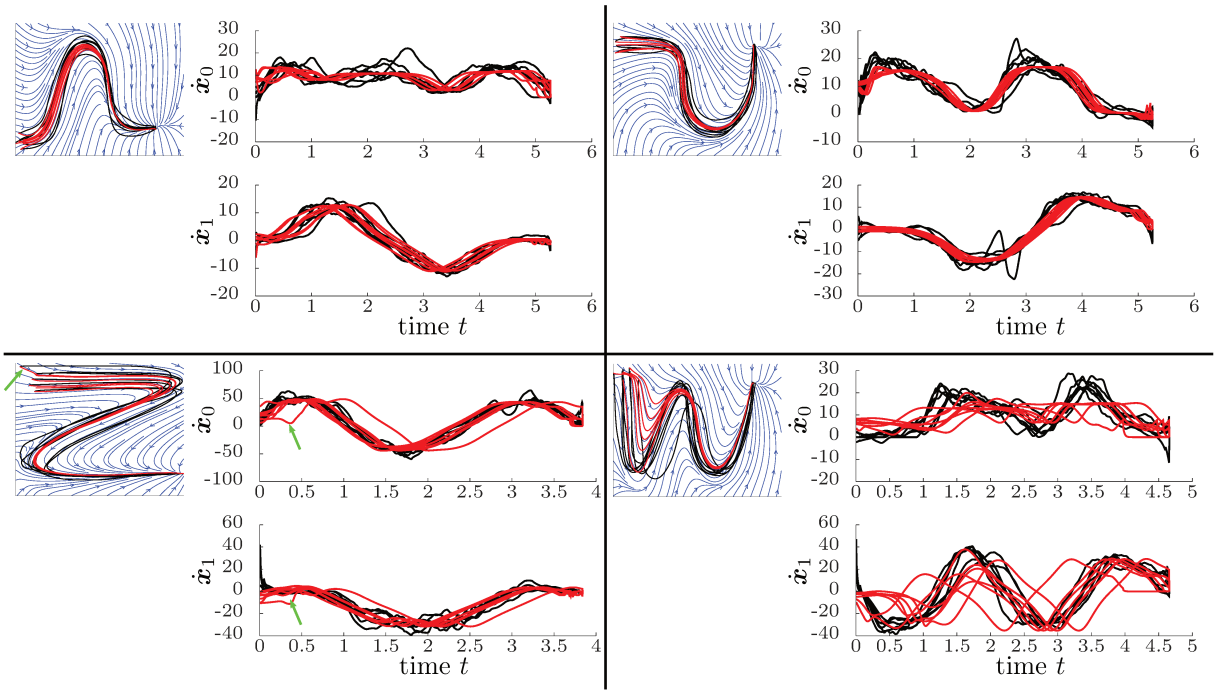


Figure 4.12 – This figure depicts the obtained velocity profiles for four different demonstration-sets. The velocity profiles are generally well matched, with the exception of the "WShape". The time-delay sometimes occuring at the beginning of the trajectory can be best observed for the "ZShape". Here the trajectory indicated by the green arrow converges first towards the meta-demonstrations and then follows it with a time-delay of about 0.5s.

Results on Multimodal Demonstrations and Cyclic Movements

To go one step further, we seek to apply the method introduced above to multimodal datasets and cyclic movements, i.e. movements converging towards a limit cycle when $t \to \infty$. Due to how the diffeomorphism and the control-space dynamics are defined, this necessitates some manual adjustments. First we take a look at the multimodal demonstrations before presenting results for the Van der Pol oscillator representing a cyclic movement.

Multimodal Demonstrations When working with multimodal demonstration sets, one can define a source and target sequence by concatenating the source and target sequences constructed in the same way as in the unimodal case for each mode present in the demonstration set. In order to be able to construct them, the only additional information necessary for each trajectory is which motion is represented by it, so the demonstrations have to be labelled. Then, once all source-target tuples $\left(X^k, Y^k\right)$ defined, we can simply concatenate them (by interpreting each point sequence as a $d \times N_k$-matrix) in order to obtain the source and target point sequences used to construct the diffeomorphism. This is possible as Algorithm 4 only seeks to match paired points and uses no other underlying asumptions on the point sequences. Therefore one can obtain a diffeomorphism for multimodal demonstration sets in a fairly straight-forward way and the result is shown in Figure 4.13.

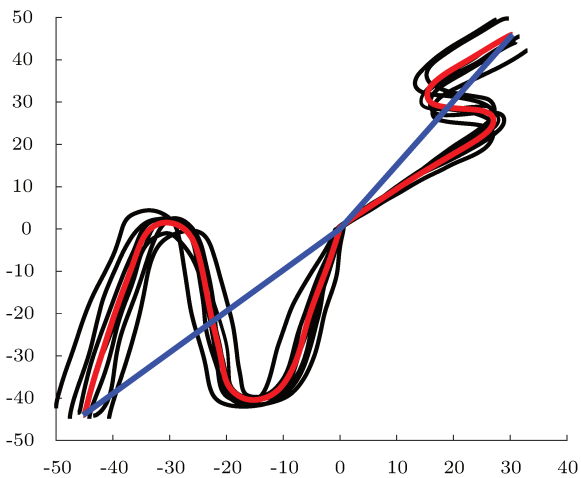


Figure 4.13 – Here a demonstration set with 2 modes or movements is shown. Each of them is represented by 7 trajectories (black lines), which are then averaged in order to obtain the two meta-demonstrations Y^0 and Y^1 (red lines) as well as the source curves X^0 and X^1 (blue lines). These curves, taken as point sequences are then concatenated in order to obtain X and Y.

A more significant problem is the definition of the control-space dynamics relying on the prototypic function. The first observation is, that, except in very specific cases, the matrix A has to be chosen as the identity matrix in order to assure that the forward trajectories from the initial point of each meta-demonstration (one for each distinct movement in the dataset) are indeed straight line segments from this point to the origin. So we can no longer modify the value of λ' to improve the global behaviour of the learned DS, which has proven to be very efficient. Secondly the velocity scaling factor γ was previously defined as distance of the initial point of the meta-demonstration divided by the average time of a demonstration $\|\boldsymbol{x}_0\|_2/T$. Now multiple $\boldsymbol{x}_0^k$ and and T^k exist, whose quotients do usually not coincide. There is no obvious way of modifying the prototypic control-space dynamics to account for this and therefore the *best* possible solution is to average over all meta-demonstrations. Therefore, in the case of multimodal demonstration sets, the velocity profiles of the movements are not well learned, but have a constant scaling factor with respect to the velocity profiles of the demonstrations. The results for 2 such datasets are shown in Figure 4.14.

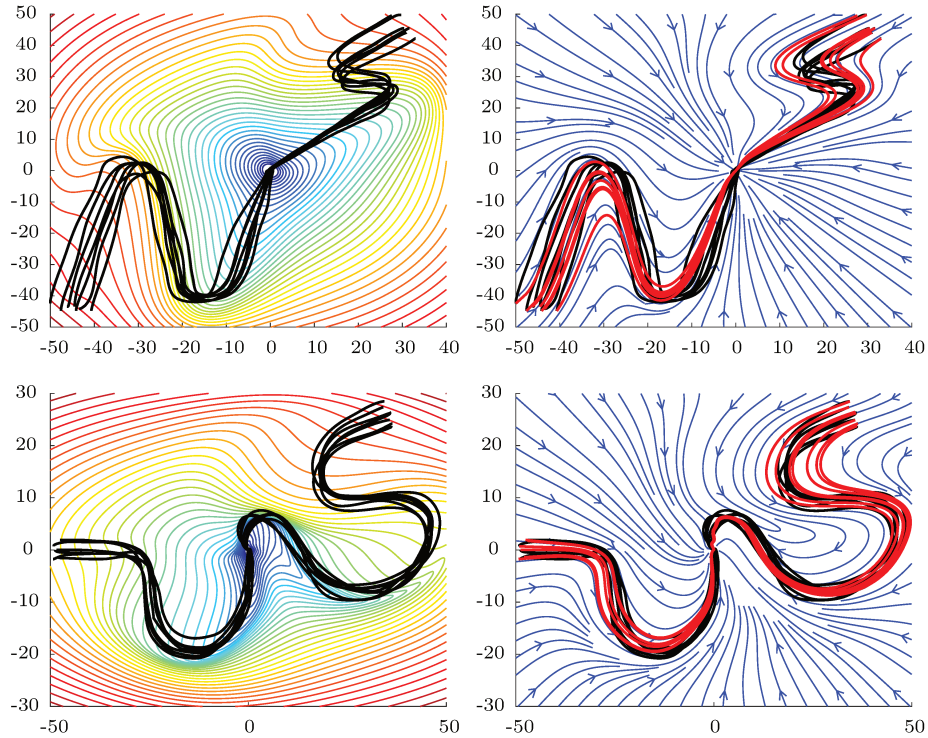


Figure 4.14 – As in Figure 4.11 level-sets for the demonstration-space Lyapunov functions and the demonstration-space dynamics are shown. Note that the reproduced trajectories (shown in red) have been integrated till convergences, due to the unique speed-factor γ, the velocity profiles cannot be respected. As one can see the restriction to $\lambda' = 1$ causes the reproduction to stray further from the demonstrations.

Cyclic Movements The source curves used to construct the diffeomorphism are defined as straight line segments, as they are, in theory, diffeomorphic to any loop free open curve, which is typically the case for trajectories representing reaching or grabbing motions. Limit cycles on the other hand are closed curves and therefore not diffeomorphic to any straight line segment, however they are diffeomorphic to any circle. Moreover we can define prototypic control-space dynamics in a similar fashion to the globally asymptotically stable case. In this section we consider only dynamical systems with 2 dimensions, but the proposed approach can be readily extended to any number of dimensions.

Consider the limit cycle of the DS $\dot{\boldsymbol{y}} = g(\boldsymbol{y})$ in the demonstration space given as timed point sequence of $N+1$ points $(\boldsymbol{y}_i, t_i)_i$ with $\boldsymbol{y}_0 = \boldsymbol{y}_N$ and $T = t_N$. We empirically set the source curve to be a circle with radius $r_s = 0.75 \min_i (\|\boldsymbol{y}_i\|_2)$ and again represent this source curve by N equally spaced points. The prototypic control space dynamics producing this source curve as forward trajectory can then be defined as

$$\dot{\tilde{\boldsymbol{x}}} = \begin{pmatrix} \dot{\theta} \\ \dot{r} \end{pmatrix} = \begin{pmatrix} \omega \\ \dot{r} \end{pmatrix} = \begin{pmatrix} \omega_0 \\ -k\,(r - r_s) \end{pmatrix} \tag{4.29}$$

in polar coordinates with $\omega_0 = {}^{2\pi}/_{T}$, $k > 0$ and the usual transformation between the polar coordinates $\tilde{\boldsymbol{x}}$ and the cartesian coordinates $\boldsymbol{x}$. Defined this way, one can easily see that the circle defined above is the limit cycle for this dynamical system. Note that the diffeomorphism from cartesian to polar coordinates has a singularity at the origin, which is however without practical significance.

We demonstrate this approach for the limit cycle of the Van der Pol oscillator, a second order system

defined as

$$\ddot{y} = \gamma(1 - y^2)\dot{y} - y \tag{4.30}$$

or equally

$$\dot{\boldsymbol{y}} = \begin{pmatrix} \boldsymbol{y}_1 \\ \gamma(1 - \boldsymbol{y}_0^2)\boldsymbol{y}_1 - \boldsymbol{y}_0 \end{pmatrix} \tag{4.31}$$

which can be shown to possess a limit cycle for $\gamma > 0$, see for instance Grimshaw [2017]. For the results presented in Figure 4.15 we used $\gamma = 1.5$ and $k = 0.75 r_s \, {}^{2\pi}/_{T}$.

Till now we have only been concerned with trajectories, which can be treated like controllable first order systems, however the Van der Pol oscillator is a second order system. Therefore two possibilities arise: either we ignore the second order nature of the system and treat it like a first order system according to (4.31). This has the advantage that the global asymptotic stability towards the limit cycle is proven, but leads to inconsistency in the following sense: as $\boldsymbol{y}_1$ corresponds to the velocity, we must have $\frac{\mathrm{d}}{\mathrm{d}\,t}\boldsymbol{y}_0 = \boldsymbol{y}_1 = g(\boldsymbol{y})[0]$ (With $g(\boldsymbol{y})$ denoting again the learned demonstration-space dynamics). This however is not guaranteed to hold for the demonstration-space dynamics arising from the prototypic control-space dynamics and the diffeomorphism. The second possibility is to redefine the demonstration-space dynamics in the following way

$$\dot{\boldsymbol{y}} = \begin{pmatrix} \boldsymbol{y}_1 \\ g(\boldsymbol{y})[1] \end{pmatrix}. \tag{4.32}$$

This means that we construct a second order system based on the second component from the learned vector field. This way, the redefined model, has the same order as the original model, but no guarantees can be given concerning stability. A trajectory visualizing both approaches is shown on the bottom right image of Figure 4.15.

As one can see, the learned limit cycle matches closely the demonstrated one, but the trajectory during the transient phase, so until the limit cycle is reached, can be significantly different. This is due to the lack of information in these regions as only the limit cycle is provided.

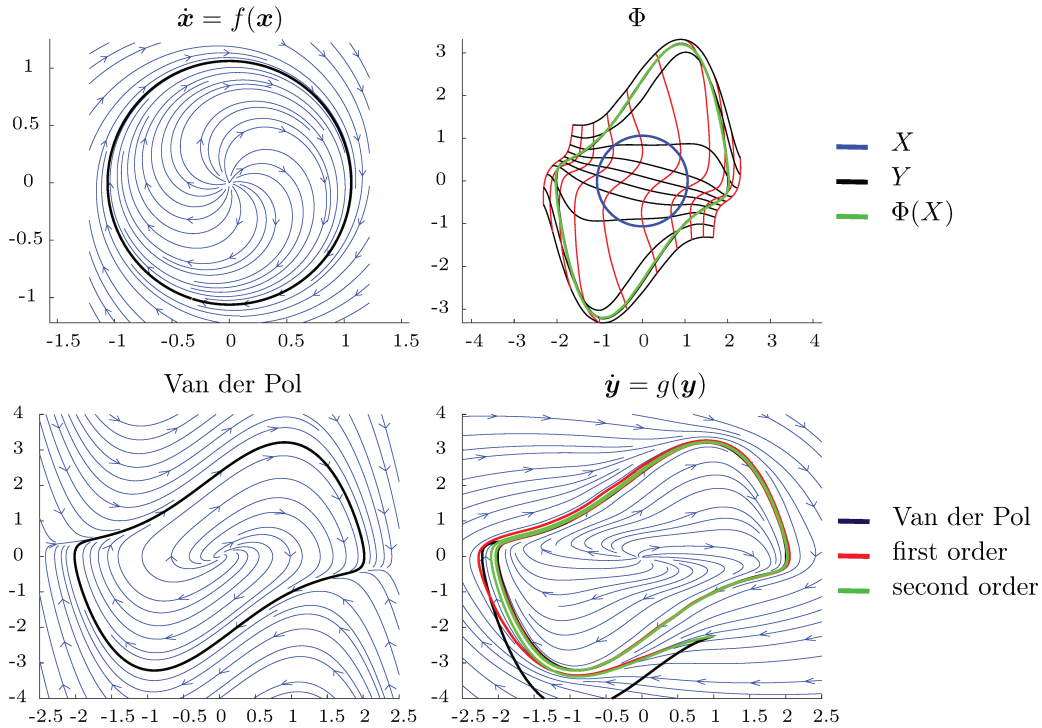


Figure 4.15 – In the top left the prototypic control-space dynamics and the resulting limit cycle is shown. All trajectories converge asymptotically towards the limit cycle except at the origin where the behaviour is undefined. In the bottom left image, the resulting streamlines for the Van der Pol oscillator are shown, again together with the limit cycle. Top right depicts the resulting diffeomorphism, by showing the source, target and image of the source sequence. Also the image of a regular grid is shown. Note that Y and $\Phi(X)$ do almost coincide. Bottom right depicts the resulting vector field (when interpreted as first order system) along with a resulting trajectory for the original system and the learned dynamical system interpreted as first and second order system, which all have the (approximately) same limit cycle.

4.5 Two-Step Learning

In the last section we have seen how to learn globally asymptotically stable nonlinear systems from demonstration, relying on a diffeomorphism constructed between a straight line segment and a meta-demonstration formed by averaging over all trajectories demonstrating the same movement. This approach, due to its inherent restrictions, works best if the demonstration set is unimodal and only one or *highly similar*[3] trajectories are given. This considerably restrains the applicability of this method to perform well with real-world data, where multiple, possibly contradicting demonstrations are available for the same movement. To overcome this and some other drawbacks detailed later on, we present a new approach which seeks to interweave statistical learning and diffeomorphic matching.

[3]The notion of similarities between demonstrations is detailed later on, and used in an informal way for now.

Contributions

- Introduction of locally weighted multitranslations
- Conception of an approximative curve matching algorithm
- Interweaving of learning and curve matching in order to learn movements from one or many demonstrations
- Implementation, experimental set-up and evaluation

4.5.1 Motivation and Problem Statement

The approach proposed in the last section composed many locally weighted translations into a diffeomorphism, based on a heuristic point matching algorithm. This algorithm performs poorly when presented with a large variance within the trajectories representing the same movement and has an reduced expressiveness if confronted with multimodal demonstrations set.

This is caused by multiple reasons. The demonstration-space speed (we say "speed" when we talk about the norm of $\dot{\boldsymbol{x}}$ or $\dot{\boldsymbol{y}}$ and velocity when we talk about the vectors and finally we say "direction" for the normalized velocity) is "encoded" in the diffeomorphism, as the norm of the prototypic control-space dynamics is state-independent (except for the small r-ball around the origin). Therefore the evolution of the speed is entirely defined by the diffeomorphism. While this works well if only one demonstration is given, this leads to problems for multiple demonstrations as shown in Figure 4.16. Especially if there exist trajectories which have a similar geometry, but different velocity profiles. Secondly, as the diffeomorphism and the control-space dynamics is built considering solely the meta-demonstration, the approach cannot deduce suitable generalizations of the movement into neighbouring regions, as the variance is lost during the averaging process. Finally, the structure of the diffeomorphism makes it hard to modify or improve a once constructed diffeomorphism if presented with new demonstrations. Therefore the drawbacks which we seek to compensate are:

(a) Each demonstration has to be labelled according to the movement it represents, its mode.

(b) The algorithm cannot handle well multiple demonstrations having a high variance for the same movement.

(c) The expressiveness is reduced for multimodal demonstration sets.

(d) The dynamics found cannot be improved or modified providing new demonstrations.

(e) The approach often results in overly deformed state-spaces.

In order to tackle these drawbacks, we propose a new approach based on the following main ideas.

Firstly we separate the learning of the speed from the direction or the normalised tangent vector of a trajectory. This has the advantage that different velocity profiles for (geometrically) similar trajectories can be better handled. Reconsider the left image of Figure 4.16. Point matching algorithms fail to find meaningful transformations in this setting, as there exists no reasonably regular diffeomorphism between the points in the source and target sequences and averaging over the demonstrations also fails to produce representative a meta-demonstration. However, if we are only interested in matching the geometry of the source curves onto the geometry of the target curves, then the problem becomes feasible. Learning the speed directly in the demonstration space is what allows us to do this. The statistical model for learning the speed averages over the different velocity profiles, whereas the transformation of the geometry is learned by the diffeomorphism based on curve matching.

Secondly we seek to learn or construct richer control-space dynamics based on the given demonstrations without loosing the GAS property in the control space. As the speed is directly learned in the

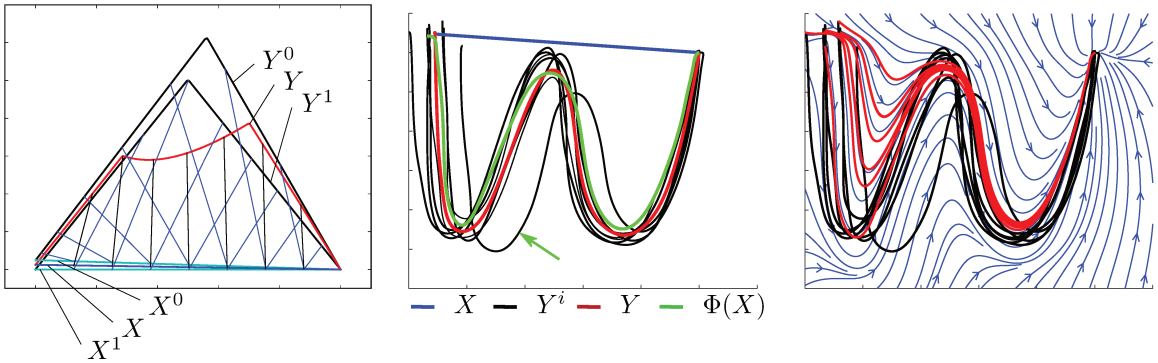


Figure 4.16 – On the left, the resulting source (blue line) and target (red line) point sequences of the averaging process proposed in section 4.4.3 when used on demonstrations with similar trajectories having different velocity profiles (black lines) is shown. As one can see, the averaged curve is not representative in this case and the averaging process is only a valid option for demonstrations having similar velocity profiles. The small blue lines going from the ith point in the source sequence to the corresponding point in the target sequence. If the velocity profiles are similar, these line were almost parallel. The middle and right image showcase the limitations of the proposed One-Step learning approach concerning the variance within the demonstration set. Even though the averaging process does produce a representative curve indicating similar velocity profiles for the demonstrations, the reproduced trajectories do not match well the demonstrations especially in the first part of the movement. This is partly caused by the large, compared to the total size of the demonstrations, distances between the initial positions of the demonstrations and the initial position of the meta-demonstration.

demonstration-space, the control-space dynamics are only concerned with learning directions and the stability is guaranteed due to Theorem 4.4. By allowing for a richer variety of control-space dynamics compared to the prototypic control-space dynamics function used within the approach presented in the last section we are able extract more information from the different demonstrations representing the same movement. This helps to avoid situations as the one shown in the middle and right image of Figure 4.16.

Each of these steps is detailed in the next sections, but we first take a look at the definitions of curve matching and why it tends to perform better when presented with multiple demonstrations compared to point matching algorithms

4.5.2 Definitions and Curve Matching

Before introducing the building blocks of the proposed diffeomorphic transformation, some definitions and notations concerning curves and an adapted version for the smooth equivalence have to be given.

Notations Concerning Curves

In the remainder of this section, several definitions concerning curves are used, which are given here[4]. A curve $\mathcal{C}$ embedded in $\mathbb{R}^d$ is defined as the set of all points generated from its parametric function $c : t \in [0, T] \mapsto c(t) \in \mathbb{R}^d$, with $c(0)/c(T)$ being the start/end point of the curve. We denote by velocity of the curve the tangent vector

$$\boldsymbol{v}_c(t) = \frac{\mathrm{d}}{\mathrm{d}\,t} c(t), \tag{4.33}$$

[4]Some definitions have already been stated before, but are repeated for clarity.

we say the speed of a curve for $\|\boldsymbol{v}_c\|_2$ and the direction of a curve for its normalised velocity, so the velocity divided by the speed. We say that a curve is in its discretized form $\mathcal{C}^{dis}$ when talking about a sorted list of $N+1$ points that correspond to the evaluation of its parametric form for some $t \in [0,T]$ resulting in $\mathcal{C}^{dis} = (\boldsymbol{x}_i = c(t_i))_{i\in[0,N]}$ with $0 \leq t_i < t_{i+1} \leq T$. We say that a curve is naturally parametrized if

$$\forall t \in [0,T] : \; s = \frac{t}{T} = \frac{\int_0^t \|\boldsymbol{v}_c(z)\|_2 \; dz}{\int_0^T \|\boldsymbol{v}_c(z)\|_2 \; dz} \; ,$$

so s corresponds to the fraction of the arc length of the curve between the start point and the point corresponding to t. Or, put in a different way, if $\|\boldsymbol{v}_c(t)\|_2 = \text{const}$ for all $t \in [0,T]$. We say that two curves $\mathcal{C}_1$ and $\mathcal{C}_2$ given in their parametric form c_1 and c_2 are *geometrically* identical if there exists a strictly monotonic function $h : [0,1] \mapsto [0,1]$ such that $\forall s \in [0,1] : \; c_1(T_1 s) = c_2(T_2 h(s))$. We say that the source and target curves are *compatible* if there exists a *reasonably regular* diffeomorphic transformation Φ such that the images of the source curves under the transformation *approximately* match the target curves. We keep this notion of reasonable regularity vague: it roughly corresponds to diffeomorphisms that produce only limited deformations. So in the middle image of Figure 4.17, the region between the two target curves would have to be strongly or unreasonably deformed to achieve a match between the source and target curves. They are therefore not compatible. We also introduce the weaker notion of *dynamic compatibility*: we say that the source and target curves are dynamically compatible if there exist parametrization functions and a transformation Φ such that the images of the parametrized source curves under Φ approximately match the target curves, see Figure 4.17. Finally the source and target curves are considered *dynamically incompatible* if no such combination of parametrization functions and a transformation exists (see Figure 4.17).

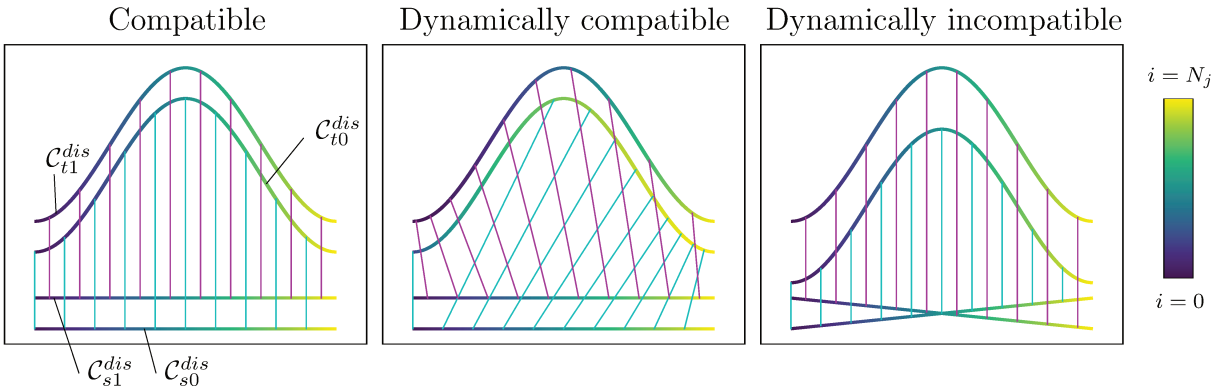


Figure 4.17 – The images show different source ($\mathcal{C}^{dis}_{s0}$, $\mathcal{C}^{dis}_{s1}$) and target ($\mathcal{C}^{dis}_{t0}$, $\mathcal{C}^{dis}_{t1}$) curves in their discretized form. The left image shows a *compatible* configuration, that is there exists a reasonably regular diffeomorphic transformation for which $\Phi(\mathcal{C}^{dis}_{s0}) \approx \mathcal{C}^{dis}_{t0}$ and $\Phi(\mathcal{C}^{dis}_{s1}) \approx \mathcal{C}^{dis}_{t1}$. The image in the middle shows a *dynamically compatible* configuration. That is, there exists no transformation taking the sources to the targets. However if we seek to match the *curves* instead of the points representing the discretized curves, we can find two parametrization functions h_0, h_1 and a transformation Φ for which $\Phi(c_{s0}(T_{s0}h_0(s))) \approx c_{t0}(T_{t1}s)$ and $\Phi(c_{s1}(T_{s1}h_1(s))) \approx c_{t1}(T_{t1}s)$ holds for all $s \in [0,1]$, where c_x is the parametric form of the curve $\mathcal{C}_x$. The image on the right depicts a *dynamically incompatible* configuration. Since the source curves do intersect and the target curves do not intersect, no diffeomorphic transformation can exist between these curves independently of the parametrization. In this case the *best* solution is to find a transformation that minimizes some error criterion.

Geometric Equivalence

In section 4.2 it was shown that if two DS $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$ and $\dot{\boldsymbol{y}} = f(\boldsymbol{y})$ are smoothly equivalent via the $\mathcal{C}^1$-diffeomorphism Φ, then both are globally asymptotically stable if one of them is. This theorem is not directly applicable in this new setting, as the speed of the trajectory is directly learned in the demonstration space. Therefore the demonstrations space dynamics is now obtained as

$$\dot{\boldsymbol{y}} = g(\boldsymbol{y}) = m(\boldsymbol{y}) J_\Phi(\Phi^{-1}(\boldsymbol{y})).f(\Phi^{-1}(\boldsymbol{y})) \tag{4.34}$$

with $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$ being the GAS control-space dynamics and $m\colon \boldsymbol{y} \in \mathbb{R}^d \to \mathbb{R}^+$ denoting the regression model for the speed which takes a point in the demonstration space and maps it to a strictly positive scaling factor. If (4.34) holds, we say f and g are geometrically equivalent. As we can see if $\forall \boldsymbol{y}\colon m(\boldsymbol{y}) = 1$ we return to the definition of smooth equivalence.

Theorem 4.4. *If two DS $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$ and $\dot{\boldsymbol{y}} = g(\boldsymbol{y})$ are geometrically equivalent via a $\mathcal{C}^1$-diffeomorphism Φ and strictly positive scaling function $m\colon \boldsymbol{y} \in \mathbb{R}^d \to \mathbb{R}^+$, then if one is globally asymptotically stable, both are.*

Proof. In Theorem 4.2 it was shown that the image of forward trajectories of $f(\boldsymbol{x})$ under Φ are forward trajectories of $g(\boldsymbol{y})$ if they are smoothly equivalent via Φ. As $f(\boldsymbol{x})$ is known to be GAS all trajectories converge and therefore all trajectories of $g(\boldsymbol{y})$ converge too.

The only difference between Theorem 4.4 and Theorem 4.1 is the scaling function $m(\boldsymbol{y})$. This scaling only affects the norm of the velocity, not the direction, as it is strictly positive. Therefore consider an (endless) forward trajectory of $f(\boldsymbol{x})$ denoted $(\boldsymbol{x}(t))_{t\geq 0}$ starting at $\boldsymbol{x}_0 = \boldsymbol{x}(0)$ and its image under diffeomorphism $(\Phi(\boldsymbol{x}(t)))_{t\geq 0}$ as well as the forward trajectory of $g(\boldsymbol{y})$ starting at $\boldsymbol{y}_0 = \Phi(\boldsymbol{x}_0)$ denoted $(\boldsymbol{y}(t))_{t\geq 0}$. Then for each point we have

$$\frac{\mathrm{d}}{\mathrm{d}\,t}\Phi\left(\boldsymbol{x}(t)\right) = J_\Phi.\dot{\boldsymbol{x}}(t) = J_\Phi.f(\boldsymbol{x}(t)) \tag{4.35}$$

which is exactly $g(\boldsymbol{y})$ divided by $m(\boldsymbol{y})$, so the tangent vector of the transformed trajectory and the tangent vector of $g(.)$ are parallel. This means that all points on the image of the forward trajectory $(\Phi(\boldsymbol{x}(t)))_{t\geq 0}$ are also on the forward trajectory $(\boldsymbol{y}(t))_{t\geq 0}$. Moreover as $m(\boldsymbol{y})$ is strictly positive, the direction of movement on the curve does not change and each point on the trajectory is attained exactly once, so for each $\boldsymbol{y}_i \in (\boldsymbol{y}(t))_{t\geq 0}$ there exists a unique t_i such that $\boldsymbol{y}_i = \boldsymbol{y}(t_i)$ and therefore for each t there exists a unique t' such that $\boldsymbol{y}(t') = \Phi(\boldsymbol{x}(t))$ holds and one can conclude that $g(\boldsymbol{y})$ is GAS if $f(\boldsymbol{x})$ is GAS.

A similar demonstration using $m'(\boldsymbol{y}) = \frac{1}{m(\boldsymbol{y})}$ proves the converse implication as the inverse of a diffeomorphism is also a diffeomorphism.

□

4.5.3 Locally Weighted Multitranslations

In section 4.4.1 locally weighted translations have been introduced, which were then composed to construct the diffeomorphism. As these transformations act only locally, many such transformations have to be composed, giving Φ a *deep* structure.

In order to give the diffeomorphism a more regular structure reducing evaluation time, we seek to group multiple translations, similar to the ones presented in section 4.4.1, into one diffeomorphic (sub-) transformation. These transformations, called locally weighted multitranslations (LWMT) have the form

$$\Psi_{(\boldsymbol{\rho}_i, \boldsymbol{c}_i, \boldsymbol{v}_i)_i}(\boldsymbol{x}) = \boldsymbol{x} + \sum_i k_{\boldsymbol{\rho}_i}(\boldsymbol{x} - \boldsymbol{c}_i)\boldsymbol{v}_i \ , \tag{4.36}$$

where $k_{\boldsymbol{\rho}}$ denotes a (symmetric positive definite) kernel function parametrized by the vector of variables $\boldsymbol{\rho}$, $\boldsymbol{c}$ is again the center of the kernel and $\boldsymbol{v}$ is the associated translation. All of them are indexed by $_i$ to indicate the different components of the multitranslation.

Gaussian Radial Basis functions have the advantage of being $\mathcal{C}^\infty$, but they are also unbounded in the sense that their value vanishes nowhere, complicating the proofs that the resulting transformation is diffeomorphic (when using them within LWMT). Therefore we propose to use the piecewise Polynomial Radial Basis functions introduced in the next section.

Polynomial Radial Basis Functions

The symmetric kernels used in this section are based on piecewise polynomial functions, so we have

$$k_\rho(\boldsymbol{x}-\boldsymbol{c}) = k_{\boldsymbol{\theta},b}(\boldsymbol{x}-\boldsymbol{c}) = \begin{cases} p_{\boldsymbol{\theta}}(\|\boldsymbol{x}-\boldsymbol{c}\|_2) & \text{if } \|\boldsymbol{x}-\boldsymbol{c}\|_2 \leq b \\ 0 & \text{else} \end{cases}. \tag{4.37}$$

The so defined kernel has therefore zero influence outside of the hypersphere centred at $\boldsymbol{c}$ with radius b. Inside the hypersphere the value corresponds to the value of an nth order piecewise continuous polynomial $p_{\boldsymbol{\theta}}\colon r \in [0, b_i] \mapsto p_i(r) \in [0,1]$, evaluated at $r = \|\boldsymbol{x}-\boldsymbol{c}\|_2$ and whose coefficients are stored in $\boldsymbol{\theta}$.

In order for the so defined function to be useful in the context of constructing diffeomorphic transformations additional restrictions have to be enforced:

- $p(0) = 1$
- $\frac{\mathrm{d}}{\mathrm{d}\,r}p(0) = 0$
- $\frac{\mathrm{d}^2}{\mathrm{d}\,r^2}p(0) = 0$
- $p(r)$ is strictly monotonically decreasing from 0 to b
- $p(b) = 0$
- $\frac{\mathrm{d}}{\mathrm{d}\,r}p(b) = 0$
- $\frac{\mathrm{d}^2}{\mathrm{d}\,r^2}p(b) = 0$
- $k_\rho(r)\colon \mathbb{R}_0^+ \to \mathbb{R}_0^+$ is $\mathcal{C}^{k\geq 1}$

These conditions ensure the smoothness of the function and that it can be used as a kernel. More specifically, for the rest of this chapter we consider fourth order piece-wise polynomial functions with p being $\mathcal{C}^2$.

Sufficient Conditions

In order for the locally weighted multitranlation to be diffeomorphic additional constraints ensuring that the resulting function is bijective have to be constructed. To this end, we show that computationally tractable conditions can be found, guaranteeing that the LWMT is injective, before showing that these conditions are also sufficient for the LWMT to be surjective.

The proof of injectivity is presented in multiple steps, with each step reducing the conservativeness of the condition at the cost of increased complexity. In a first step, consider the application

$$\begin{aligned} \Xi\colon\ & \mathcal{X} \to \mathcal{Y} \\ & \boldsymbol{x} \mapsto \boldsymbol{x} + \sum_{j=0}^{N-1} \nu_j(\boldsymbol{x}) \end{aligned} \tag{4.38}$$

with each $\nu_j\colon\ \mathcal{X} \to \mathcal{Y}$ being $\mathcal{C}^1$ and representing a position dependent translation vector.

Lemma 4.5. *The application Ξ defined in (4.38) is injective if each ν_j has a (global) Lipschitz constant K_j strictly smaller $1/N$.*

Proof. An application is injective if any element (point) in the codomain is the image of at most one element (point) in the domain. Therefore any two distinct points $\boldsymbol{x}_A, \boldsymbol{x}_B \in \mathcal{X},\ \boldsymbol{x}_A \neq \boldsymbol{x}_B$, must have distinct images $\boldsymbol{y}_A, \boldsymbol{y}_B \in \mathcal{Y}$:

$$\begin{aligned}
&\boldsymbol{y}_A \neq \boldsymbol{y}_B \\
&\Xi(\boldsymbol{x}_a) \neq \Xi(\boldsymbol{x}_b) \\
&\boldsymbol{x}_A + \sum_{j=0}^{N-1} \nu_j(\boldsymbol{x}_A) \neq \boldsymbol{x}_B + \sum_{j=0}^{N-1} \nu_j(\boldsymbol{x}_B) \\
&\sum_{j=0}^{N-1} \nu(\boldsymbol{x}_A) - \sum_{j=0}^{N-1} \nu_j(\boldsymbol{x}_B) \neq \boldsymbol{x}_B - \boldsymbol{x}_A.
\end{aligned}$$

This can be guaranteed by imposing

$$\begin{aligned}
&\left\| \sum_{j=0}^{N-1} (\nu(\boldsymbol{x}_A)) - \sum_{j=0}^{N-1} (\nu(\boldsymbol{x}_B)) \right\|_2 < \|\boldsymbol{x}_B - \boldsymbol{x}_A\|_2 \\
&\left\| \sum_{j=0}^{N-1} (\nu(\boldsymbol{x}_A) - \nu(\boldsymbol{x}_B)) \right\|_2 < \|\boldsymbol{x}_B - \boldsymbol{x}_A\|_2 .
\end{aligned}$$

Applying the triangle inequality and as we supposed ν_j to be $\mathcal{C}^1$ with a global Lipschitz constant $K_j < 1/N$ we obtain

$$\begin{aligned}
&\left\| \sum_{j=0}^{N-1} (\nu(\boldsymbol{x}_A) - \nu(\boldsymbol{x}_B)) \right\|_2 \leq \sum_{j=0}^{N-1} \|\nu(\boldsymbol{x}_A) - \nu(\boldsymbol{x}_B)\|_2 \leq \\
&\qquad \sum_{j=0}^{N-1} K_j \|\boldsymbol{x}_B - \boldsymbol{x}_A\|_2 < \sum_{j=0}^{N-1} \frac{1}{N} \|\boldsymbol{x}_B - \boldsymbol{x}_A\|_2 = \|\boldsymbol{x}_B - \boldsymbol{x}_A\|_2 \qquad (4.39)
\end{aligned}$$

□

In the case of LWMT based on Polynomial Radial Basis function the ν_j can be identified with the locally weighted translations $\boldsymbol{v}_j k_j(\|\boldsymbol{x} - \boldsymbol{c}_j\|_2)$. As $k_j(\|\boldsymbol{x} - \boldsymbol{c}_j\|_2) = 0$ if $\|\boldsymbol{x} - \boldsymbol{c}_j\|_2 > b_j$, only the case of $\boldsymbol{x}$ being inside the basis is of interest. Therefore we now have to construct a condition limiting the Lipschitz constant of $\boldsymbol{v}_j p_j(\|\boldsymbol{x} - \boldsymbol{c}_j\|_2)$ to $1/N$:

$$\left\|\boldsymbol{v}_j p_j(\|\boldsymbol{x} + \Delta\boldsymbol{x} - \boldsymbol{c}_j\|_2) - \boldsymbol{v}_j p_j(\|\boldsymbol{x} - \boldsymbol{c}_j\|_2)\right\|_2 \leq K_j \|\Delta\boldsymbol{x}\|_2 ,$$

where $\Delta\boldsymbol{x} = \boldsymbol{x}_B - \boldsymbol{x}_A$ and $\boldsymbol{x} = \boldsymbol{x}_A$. One obtains

$$\begin{aligned}
&\left\|\boldsymbol{v}_j p_j(\|\boldsymbol{x} + \Delta\boldsymbol{x} - \boldsymbol{c}_j\|_2) - \boldsymbol{v}_j p_j(\|\boldsymbol{x} - \boldsymbol{c}_j\|_2)\right\|_2 \leq \\
&\sup_{\boldsymbol{x},\epsilon} \left\| \boldsymbol{v}_j \left(p_j(\|\boldsymbol{x} - \boldsymbol{c}_j\|_2) + \left| \frac{p_j(\|\boldsymbol{x} + \epsilon^{\Delta\boldsymbol{x}}/_{\|\Delta\boldsymbol{x}\|_2} - \boldsymbol{c}_j\|_2) - p_j(\|\boldsymbol{x} - \boldsymbol{c}_j\|_2)}{\epsilon} \right| \|\Delta\boldsymbol{x}\|_2 - p_j(\|\boldsymbol{x} - \boldsymbol{c}_j\|_2) \right) \right\|_2 \leq \\
&\qquad \sup_{\boldsymbol{x}}(\left|p_j'(\|\boldsymbol{x} - \boldsymbol{c}_j\|_2)\right|) \|\Delta\boldsymbol{x}\|_2 \frac{(\boldsymbol{x} - \boldsymbol{c}_j)^{\mathrm{T}}}{\|\boldsymbol{x} - \boldsymbol{c}_j\|_2} . \boldsymbol{v}_j \leq {}^M p_j' \|\boldsymbol{v}_j\|_2 \|\Delta\boldsymbol{x}\|_2 \qquad (4.40)
\end{aligned}$$

with ${}^M p_j' = \sup_{\boldsymbol{x}}(\left|p_j'(\|\boldsymbol{x} - \boldsymbol{c}_j\|_2)\right|)$. This means that each locally weighted translation has a global Lipschitz constant of ${}^M p_j' \|\boldsymbol{v}_j\|_2$. As $\|\boldsymbol{v}_j\|_2$ is known and ${}^M p_j'$ can be analytically determined for the polynomials used in this section as a function of the size of the kernel (b_j), the condition

$$\forall j \in [0, N-1] : \ {}^M p_j' \|\boldsymbol{v}_j\|_2 \leq \frac{1}{N} - \mu \qquad (4.41)$$

with $0 < \mu < \frac{1}{N}$ provides a computationally tractable condition to ensure injectivity of the LWMT with a safety margin of μ.

The above presented condition is very conservative for multiple reasons. First, it always takes into account all locally weighted translations, even in the case that their bases do not intersect. Secondly it directly reasons about the Lipschitz constants, neglecting the possibility that the translation directions associated to each kernel can be different.

First the latter drawback is addressed. Again, we seek to proof injectivity by showing that $\boldsymbol{x}_A, \boldsymbol{x}_B \in \mathcal{X}$, $\boldsymbol{x}_A \neq \boldsymbol{x}_B$ implies that the images $\boldsymbol{y}_A$ and $\boldsymbol{y}_B$ are distinct:

$$\begin{aligned}
&\boldsymbol{y}_A \neq \boldsymbol{y}_B \\
&\sum_{j=0}^{N-1} \boldsymbol{v}_j p_j(\|\boldsymbol{x}_A - \boldsymbol{c}_j\|_2) - \sum_{j=0}^{N-1} \boldsymbol{v}_j p_j(\|\boldsymbol{x}_B - \boldsymbol{c}_j\|_2) \neq \boldsymbol{x}_B - \boldsymbol{x}_A.
\end{aligned}$$

By denoting $\boldsymbol{x} = \boldsymbol{x}_A$ and $\Delta\boldsymbol{x} = \boldsymbol{x}_B - \boldsymbol{x}_A$ this can again be ensured by

$$\left\| \sum_{j=0}^{N-1} \boldsymbol{v}_j p_j(\|\boldsymbol{x} + \Delta\boldsymbol{x} - \boldsymbol{c}_j\|_2) - \sum_{j=0}^{N-1} \boldsymbol{v}_j p_j(\|\boldsymbol{x} - \boldsymbol{c}_j\|_2) \right\|_2 < \|\Delta\boldsymbol{x}\|_2 . \tag{4.42}$$

One obtains

$$\begin{aligned}
&\left\| \sum_{j=0}^{N-1} p_j(\|\boldsymbol{x} + \Delta\boldsymbol{x} - \boldsymbol{c}_j\|_2)\boldsymbol{v}_j - \sum_{j=0}^{N-1} p_j(\|\boldsymbol{x} - \boldsymbol{c}_j\|_2)\boldsymbol{v}_j \right\|_2 = \\
&\left\| \sum_{j=0}^{N-1} \left(\left(p_j(\|\boldsymbol{x} + \Delta\boldsymbol{x} - \boldsymbol{c}_j\|_2) - p_j(\|\boldsymbol{x} - \boldsymbol{c}_j\|_2)\right) \boldsymbol{v}_j \right) \right\|_2 = \\
&\left\| \sum_{j=0}^{N-1} \left(\alpha_j \sup_{\boldsymbol{x},\epsilon} \left(\left| \frac{p_j(\|\boldsymbol{x} + \epsilon^{\Delta\boldsymbol{x}}/_{\|\Delta\boldsymbol{x}\|_2} - \boldsymbol{c}_j\|_2) p_j(\|\boldsymbol{x} - \boldsymbol{c}_j\|_2)}{\epsilon} \right| \right) \|\Delta\boldsymbol{x}\|_2 \boldsymbol{v}_j \right) \right\|_2
\end{aligned} \tag{4.43}$$

for some $\alpha_j \in [-1, 1]$ depending on $\boldsymbol{x}$, $\Delta\boldsymbol{x}$ and $\boldsymbol{c}_j$. Therefore one can write

$$\begin{aligned}
&\left\| \sum_{j=0}^{N-1} \left(\alpha_j \sup_{\boldsymbol{x},\epsilon} \left(\left| \frac{p_j(\|\boldsymbol{x} + \epsilon^{\Delta\boldsymbol{x}}/_{\|\Delta\boldsymbol{x}\|_2} - \boldsymbol{c}_j\|_2) p_j(\|\boldsymbol{x} - \boldsymbol{c}_j\|_2)}{\epsilon} \right| \right) \|\Delta\boldsymbol{x}\|_2 \boldsymbol{v}_j \right) \right\|_2 \leq \\
&\max_{\{\alpha_j\}_j} \left\| \sum_{j=0}^{N-1} \left(\alpha_j \sup_{\boldsymbol{x},\epsilon} \left(\left| \frac{p_j(\|\boldsymbol{x} + \epsilon^{\Delta\boldsymbol{x}}/_{\|\Delta\boldsymbol{x}\|_2} - \boldsymbol{c}_j\|_2) p_j(\|\boldsymbol{x} - \boldsymbol{c}_j\|_2)}{\epsilon} \right| \right) \|\Delta\boldsymbol{x}\|_2 \boldsymbol{v}_j \right) \right\|_2 = \\
&\left\| \sum_{j=0}^{N-1} \left({}^M p_j'^A \boldsymbol{v}_j \right) \right\|_2 \|\Delta\boldsymbol{x}\|_2
\end{aligned} \tag{4.44}$$

with ${}^M p_j' = \sup_{\boldsymbol{x}} \left(\left| p_j'(\|\boldsymbol{x} - \boldsymbol{c}_j\|_2) \right| \right)$ and ${}^A\boldsymbol{v}_j[i] = |\boldsymbol{v}_j[i]|$ is the element-wise absolute translation vector. The associated constraint

$$\left\| \sum_{j=0}^{N-1} \left({}^M p_j'^A \boldsymbol{v}_j \right) \right\|_2 \leq 1 - \mu \tag{4.45}$$

provides a less conservative alternative to the constraint (4.41) while providing a similar safety margin $\mu \in]0, 1[$. These constraints also have a geometrical interpretation as shown in Figure 4.18.

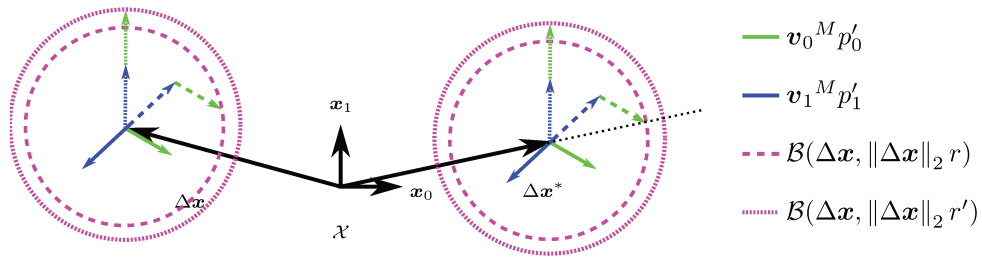


Figure 4.18 – In this figure the different constraints (4.41) and (4.45) are showcased for a two dimensional multitranslation with two components. Constraint (4.41) sums up the norm of the translations scaled by the maximal derivative of the kernels. By imposing that it has to be smaller than one, this can be reinterpreted as a closed ball centred at $\Delta\boldsymbol{x}$ and the sum corresponds to r. As r is smaller than one, the ball $\mathcal{B}(\Delta\boldsymbol{x}, \|\Delta\boldsymbol{x}\|_2\, r)$ does not contain the origin, which is equivalent to ensuring injectivity. The second constraint (4.45) can also be interpreted as a ball centred at $\Delta\boldsymbol{x}$, but here $r' = \left\| {}^M p_0'\, {}^A\boldsymbol{v}_0 + {}^M p_1'\, {}^A\boldsymbol{v}_1 \right\|_2$ takes into account the direction of the translation. Therefore the associated ball $\mathcal{B}(\Delta\boldsymbol{x}, \|\Delta\boldsymbol{x}\|_2\, r')$ is smaller, indicating that the constraint is less conservative.

Now the drawback that all locally weighted translations are taken into account, even in the case that their bases do not intersect, is addressed. Some of the conservativeness of condition (4.44) ensuring injectivity stems from the overapproximation of the directional derivative of the kernel $p_j(\|\boldsymbol{x} - \boldsymbol{c}_j\|_2)$ at point $\boldsymbol{x}$ by its supremum over $\mathcal{X}$. However, it is clear that if the bases of two locally weighted translations, so the closed ball centred on $\boldsymbol{c}_j$ with radius b_j (see (4.37)), do not intersect, it is sufficient to ensure injectivity of each locally weighted translation to ensure injectivity of the multitranslation.

For each locally weighted translation the sets

$$\Omega_j^m = \left\{ \boldsymbol{x} \mid r_j^{m-1} \leq \|\boldsymbol{x} - \boldsymbol{c}_j\|_2 < r_j^m \right\} \tag{4.46}$$

and the associated maximal norm of the directional derivative

$$ {}^m p_j' = \sup_{\boldsymbol{x} \in \Omega_j^m, \epsilon} \left(\left| \frac{p_j(\|\boldsymbol{x} + \epsilon^{\Delta\boldsymbol{x}}/_{\|\Delta\boldsymbol{x}\|_2} - \boldsymbol{c}_j\|_2) p_j(\|\boldsymbol{x} - \boldsymbol{c}_j\|_2)}{\epsilon} \right| \right) = \sup_{\boldsymbol{x} \in \Omega_j^m} \left(\left| p'(\|\boldsymbol{x} - \boldsymbol{c}_j)\|_2 \right| \right) \tag{4.47}$$

are defined for $m = [1, M]$ with $r_j^0 = 0$, $r_j^{M-1} = b_j$, $r_j^M = \infty$ and $r_j^{m-1} < r_j^m$, as shown in Figure 4.19. The particular choice of the r_j^m ensures that the sets $\left\{\Omega_j^m\right\}_m$ form a partition of the control-space for each j. Further we define

$$\bar{p}_j'(\boldsymbol{x}) = {}^m p_j' \text{ with } m \text{ such that } \boldsymbol{x} \in \Omega_j^m \tag{4.48}$$

where m is uniquely defined, as the set $\left\{\Omega_j^m\right\}_m$ is a partition of the control-space.

Using this partitioning (4.44) can be rewritten as

$$\left\| \sum_{j=0}^{N-1} \left({}^A\boldsymbol{v}_j \bar{p}_j'(\boldsymbol{x}) \|\Delta\boldsymbol{x}\|_2 \right) \right\|_2 = \left\| \sum_{j=0}^{N-1} \left({}^A\boldsymbol{v}_j \bar{p}_j'(\boldsymbol{x}) \right) \right\|_2 \|\Delta\boldsymbol{x}\|_2 \,. \tag{4.49}$$

The associated constraint

$$\left\| \sum_{j=0}^{N-1} \left({}^A\boldsymbol{v}_j \bar{p}_j'(\boldsymbol{x}) \right) \right\|_2 \leq 1. - \mu \tag{4.50}$$

therefore guarantees injectivity of the LWMT at point $\boldsymbol{x}$. In order to ensure global injectivity this condition has to hold for all $\boldsymbol{x}$. This can be done by computing the intersections of each Ω_j^m of the jth-kernel with the sets Ω_i^l of all kernels. By computing the worst-case scenario for each set, the conservativeness is decreased in comparison to (4.44). Or to put it more formally, the expression

$$\forall k \in [0, N-1]\,,\ \forall l \in [1, M]: \quad \left\| \sum_j \left({}^A\boldsymbol{v}_j \max\left({}^m p'_j \mathbb{1}(\Omega_k^l, \Omega_j^m) \right) \right) \right\|_2 \|\Delta \boldsymbol{x}\|_2 \tag{4.51}$$

with the indicator function $\mathbb{1}\colon\ S_0 \subseteq \mathbb{R}^n, S_1 \subseteq \mathbb{R}^n \to \{0, 1\}$ defined as

$$\mathbb{1}(S_0, S_1) = \begin{cases} 1 & \text{if } S_0 \cap S_1 \neq \emptyset \\ 0 & \text{else} \end{cases} \tag{4.52}$$

overapproximates (4.43). The associated condition

$$\forall k \in [0, N-1]\,,\ \forall l \in [1, M]: \quad \left\| \sum_j \left({}^A\boldsymbol{v}_j \max\left({}^m p'_j \mathbb{1}(\Omega_k^l, \Omega_j^m) \right) \right) \right\|_2 \leq 1. - \mu \tag{4.53}$$

therefore guarantees injectivity while being less conservative than (4.45). This approach is depicted in Figure 4.19 on the right.

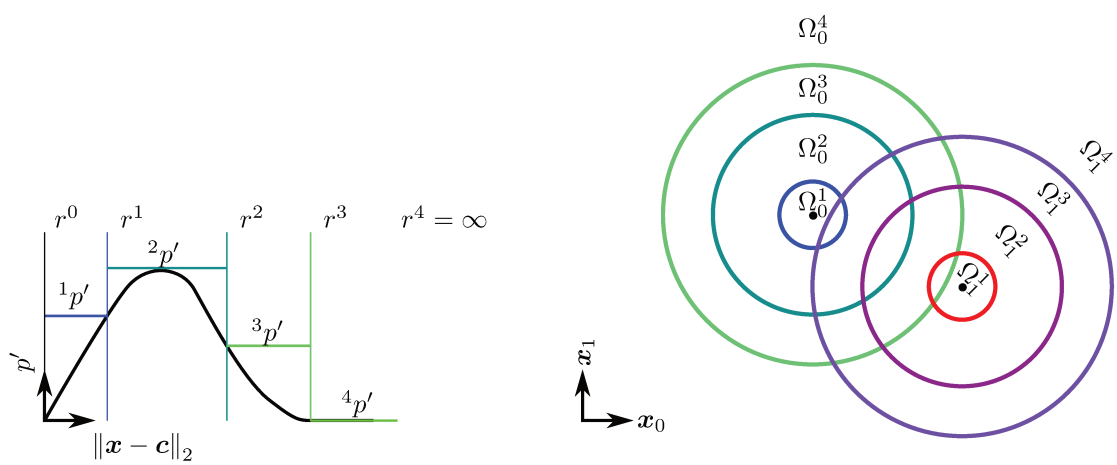


Figure 4.19 – In this figure the overapproximation of $p'(\|\boldsymbol{x} - \boldsymbol{c})\|_2$ with respect to the partitioning induced by $r^{[0,4]}$ is shown on the left. The condition (4.53) is depicted on the right.

To showcase the constraint (4.53) consider its evaluation for $k = 0$ and $l = 3$ for a multitranslation formed by two kernels of identical base, which ensures injectivity of the set Ω_0^3:

$$\left\| {}^A\boldsymbol{v}_0 \max\left({}^1p'_0 \cdot 0 + {}^2p'_0 \cdot 0 + {}^3p'_0 \cdot 1 + {}^4p'_0 \cdot 0 \right) +^A \boldsymbol{v}_1 \max\left({}^1p'_1 \cdot 1 + {}^2p'_1 \cdot 1 + {}^3p'_1 \cdot 1 + {}^4p'_1 \cdot 1 \right) \right\|_2 = \left\| {}^A\boldsymbol{v}_0 \cdot {}^3p'_0 +^A \boldsymbol{v}_1 \cdot {}^2p'_2 \right\|_2 .$$

Ω_0^3 intersect only with itself for the partition formed by the sets for the first kernel, but it intersects with all sets of the partition for the second kernel. As the largest value of $p'_1(\|\boldsymbol{x} - \boldsymbol{c}_1)\|_2$ occurs in Ω_1^2, ${}^A\boldsymbol{v}_1 \cdot {}^2p'_2$ is added to the sum.

The above proposed constraint guarantees that the LWMT is injective and globally Lipschitz continuous. Now surjectivity is proven by contradiction.

Lemma 4.6. *For a LWMT to be surjective it is sufficient that a) condition (4.53) holds b) there exists some value $R_M \in \mathbb{R}^+$ such that $\|\boldsymbol{c}_j\|_2 < R_M$ and $|b_j| < R_M$ holds for all j*

Proof. Reconsider the LWMT

$$\Psi(\boldsymbol{x}) = \boldsymbol{x} + \sum_j \boldsymbol{v}_j p_j(\|\boldsymbol{x} - \boldsymbol{c}_j\|_2)$$

respecting (4.53). All locally weighted translations have a bounded base, therefore the locally weighted multitranslation is equal to the identity mapping outside of the closed ball centred at the origin with radius $r_M = \max_j(\|\boldsymbol{c}_j\|_2 + b_j) < 2R_M$ (supposition b) denoted $\mathcal{B}_M$ (black circle in Figure 4.20). Suppose the LWMT Ψ is not surjective, then there must exist an open set in the control-space $S \subset \mathcal{Y}$ (green shape), for which

$$\forall \boldsymbol{y} \in S, \nexists \boldsymbol{x} \in \mathcal{X} \text{ such that } \boldsymbol{y} = \Psi(\boldsymbol{x}). \tag{4.54}$$

As the LWMT is equal to the identity mapping outside of $\mathcal{B}_M$, S must be inside, so $S \subset \mathcal{B}_M$. Consider two hyperplanes $\mathcal{H}_{\mathcal{X},0}$, $\mathcal{H}_{\mathcal{X},1}$ in the demonstration-space perpendicular to the first dimension $\boldsymbol{x}_0$ (so the space spanned by the dimension 1 to $d-1$) containing the points $-1.1r_M\boldsymbol{x}_0$ and $1.1r_M\boldsymbol{x}_0$ (black lines). As $\mathcal{H}_{\mathcal{X},0}$ and $\mathcal{H}_{\mathcal{X},1}$ are completely outside of $\mathcal{B}_M$, they do not change under Ψ, so $\mathcal{H}_{\mathcal{Y},i} = \Psi(\mathcal{H}_{\mathcal{X},i}) = \mathcal{H}_{\mathcal{X},i}$ for $i = 0, 1$. Now consider the hyperplane defined as

$$\mathcal{H}_{\mathcal{X},\lambda} = \mathcal{H}_{\mathcal{X},0} + 2.2\lambda r_M \boldsymbol{x}_0 \tag{4.55}$$

for $\lambda = [0, 1]$. So $\mathcal{H}_{\mathcal{X},\lambda}$ is a continuous mapping from $\mathcal{H}_{\mathcal{X},0}$ (for $\lambda = 0$) to $\mathcal{H}_{\mathcal{X},1}$ (for $\lambda = 1$). And its image under Ψ is denoted as $\mathcal{H}_{\mathcal{Y},\lambda} = \Psi(\mathcal{H}_{\mathcal{X},\lambda})$. By continuity one concludes that there must exist a value for λ, denoted λ', such that $\Psi(\mathcal{H}_{\mathcal{Y},\lambda'})$ is tangential to S, that is $\Psi(\mathcal{H}_{\mathcal{Y},\lambda'}) \cap \partial S \neq \emptyset$ and $\Psi(\mathcal{H}_{\mathcal{Y},\lambda'}) \cap S = \emptyset$ (blue line). Finally there must exist two $\epsilon > 0$ close points (red points), element of $\mathcal{H}_{\mathcal{X},\lambda'}$, with the distance of their images bounded by $K\epsilon$ (supposition a). Now consider the hyperplane $\mathcal{H}_{\mathcal{X},\lambda'+\epsilon'}$ (cyan line). The two points translated by ϵ' along $\boldsymbol{x}_0$ (purple points) are still ϵ close, but their image under Ψ can no longer be separated by at most $K\epsilon$ due to the set S which is supposed to have no preimage. Therefore the supposition that Ψ has a global Lipschitz constant is violated by these points and therefore no such set S can exist and Ψ is surjective. □

This concludes the proof that the LWMT defined in (4.36) under the constraint (4.53) is a bijection and as it is moreover in $\mathcal{C}^2$, it is indeed a $\mathcal{C}^2$-diffeomorphism.

Computing Ψ^{-1}

Before continuing on how to use such LWMT in the context of diffeomorphic curve matching, two possibilities to compute the inverse transformation are proposed. Again, as for the locally weighted translations, it is not possible to compute an analytic expression of the inverse function, and one has to rely on iterative algorithms to solve the inversion. The first algorithm relies on a simple update scheme and provides exponential convergence, however at an in theory small convergence rate. The second algorithm proposed is closer to the one used to invert the locally weighted translation (Algorithm 3), and provides a theoretically slower convergence but is nonetheless very efficient in practice.

Given the criterion (4.53) holds, the LWMT

$$\boldsymbol{y} = \Psi(\boldsymbol{x}) = \boldsymbol{x} + \sum_{i=0}^{N_l-1} k_{\rho_i}(\boldsymbol{x} - \boldsymbol{c}_i)\boldsymbol{v}_i$$

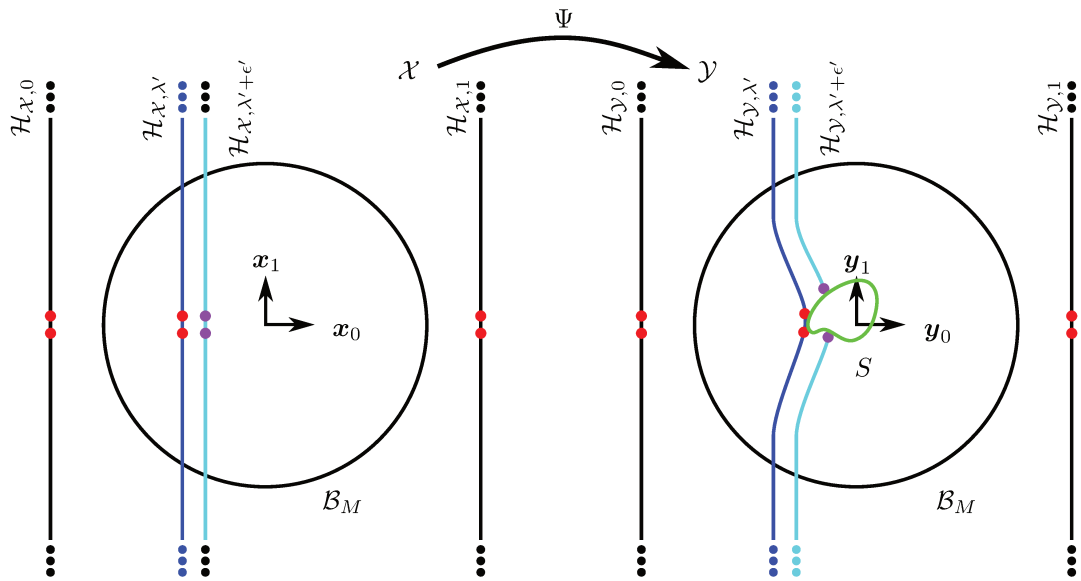


Figure 4.20 – On the left the demonstration space with the different hyperplanes (lines in $2d$) are shown. On the right their image and the set S is shown. The ϵ close points on $\mathcal{H}_{\mathcal{X},\lambda'}$ are also ϵ-close on $\mathcal{H}_{\mathcal{X},\lambda'+\epsilon'}$, but their images cannot respect the maximal distance $K\epsilon$ contradicting the supposition that Ψ is Lipschitz continuous with constant K.

is a $\mathcal{C}^2$-diffeomorphism. The geometrical reinterpretation of the constraint (4.53) tells us that the image of $\boldsymbol{x} + \delta\boldsymbol{x}$ has to lie inside of the closed ball centred at $\Psi(\boldsymbol{x}) + \Delta\boldsymbol{x}$ with radius $(1-\mu)\left\|\Delta\boldsymbol{x}\right\|_2$ with the safety factor $0 < \mu < 1$. Consider the following update scheme to compute $\boldsymbol{x}^*$ for a given $\boldsymbol{y}^*$ such that $\boldsymbol{y}^* = \Psi(\boldsymbol{x}^*)$: in iteration i, the current guess for $\boldsymbol{x}^*$, denoted $\boldsymbol{x}^i$, is updated according to

$$\boldsymbol{x}^{i+1} = \boldsymbol{x}^i + (\boldsymbol{y}^* - \Psi(\boldsymbol{x}^i)). \tag{4.56}$$

Due to the constraint (4.53) we know that $\Psi(\boldsymbol{x}^{i+1})$ lies within $\mathcal{B}\left(\boldsymbol{y}^*, (1-\mu)\left\|\boldsymbol{y}^* - \Psi(\boldsymbol{x}^i)\right\|_2\right)$. We therefore obtain

$$\left\|\boldsymbol{y}^* - \Psi(\boldsymbol{x}^{i+1})\right\|_2 \leq (1-\mu)\left\|\boldsymbol{y}^* - \Psi(\boldsymbol{x}^i)\right\|_2 \tag{4.57}$$

which indicates that the distance between $\Psi(\boldsymbol{x}^i)$ and $\boldsymbol{y}^*$ decreases exponentially with respect to the number of iterations. By initializing the algorithm with $\boldsymbol{x}^0 = \boldsymbol{y}^*$, the error is bounded by

$$\left\|\boldsymbol{y}^* - \Psi(\boldsymbol{x}^{i+1})\right\|_2 = (1-\mu)^i \left\|\boldsymbol{y}^* - \Psi(\boldsymbol{y}^*)\right\|_2. \tag{4.58}$$

Therefore the worst-case convergence can be influenced by choosing a larger μ, which is however not always desirable, as this often hinders the smooth overlapping of kernels in practical cases.

The second method relies on the multivariate vector-valued version of the Newton-Euler algorithm. We introduce the coefficient vector $\boldsymbol{r} \in \mathbb{R}^{N_l}$, where N_l is the number of translations, with $\boldsymbol{r}_j = k_j(\boldsymbol{x} - \boldsymbol{c}_j)$ the LWMT can be written as

$$\boldsymbol{y} = \boldsymbol{x} + \sum_{j=0}^{N_l-1} \boldsymbol{r}_j \boldsymbol{v}_i \tag{4.59}$$

and obviously

$$\boldsymbol{x} = \Psi^{-1}(\boldsymbol{y}) = \boldsymbol{y} - \sum_{j=0}^{N_l-1} \boldsymbol{r}_j \boldsymbol{v}_j. \tag{4.60}$$

In section 4.4.1 the problem of finding the inverse is reformulated to the problem of finding the solution of a function in one variable, which, due the monotonicity properties of the problem, could be efficiently solved using Newtons-method. Here we can define the multivariate vector-valued equivalent:

$$\begin{aligned} h_{\boldsymbol{y}} \colon \mathbb{R}^{N_l} &\to \mathbb{R}^{N_l} \\ \boldsymbol{r} &\mapsto \boldsymbol{r} - \left[k_{\rho_i}(\boldsymbol{y} - \textstyle\sum_{j=0}^{N_l-1} \boldsymbol{r}_j \boldsymbol{v}_j - \boldsymbol{c}_i)\right]_i. \end{aligned} \tag{4.61}$$

Finding an $\boldsymbol{r}^*$ for which $h_{\boldsymbol{y}}(\boldsymbol{r}^*)$ is element-wise zero is equivalent to finding the point $\boldsymbol{r}_i = k_{\rho_i}(\boldsymbol{x} - \boldsymbol{c}_i)$ which is unique and therefore the same as finding Ψ^{-1}. This problem can be efficiently solved using the multidimensional version of the Newton-algorithm. That is interpreting r in Algorithm 3 as the vector $\boldsymbol{r}$ and changing line 6 of Algorithm 3 to

$$\boldsymbol{r} \leftarrow \boldsymbol{r} - \left(\frac{\partial}{\partial \boldsymbol{r}} h_{\boldsymbol{y}}(\boldsymbol{r})\right)^{-1} .h_{\boldsymbol{y}}(\boldsymbol{r}). \tag{4.62}$$

The element-wise limiting of the current value of $\boldsymbol{r}$ is not necessary, but speeds up convergence in some cases. For this algorithm no strict error-bounds can be given, but empirical observations show that it usually converges ($normh(\boldsymbol{y})(\boldsymbol{r}) \leq 1e^{-6}$) within 5 to 10 iterations.

In contrast to the first method presented above, we do not have a final proof for the convergence of this algorithm, however it can be conjectured due to an empirical and an analytical finding. The empirical finding is that during the extensive use of this algorithm to compute the examples presented later on, it always terminated and has never produced inconsistent result. The analytical finding is that each element of $h(\boldsymbol{y})[i]$ is strictly monotone in $\boldsymbol{r}[i]$ independently of the current value of $\boldsymbol{r}[j]$, $i \neq j$. This suggests that the function has no local minima and converges to the global using the Newton-Euler algorithm.

4.5.4 Diffeomorphic Curve Matching

In section 4.4.2, we introduced a heuristic algorithm which constructs a diffeomorphism between two point sequences relying on point matching. That is, these points do not have an implicit or underlying structure, but the quality of the matching is defined by some weighted average of the distance between the image of a source point and the corresponding target point.

This is different in the framework of curve matching. In this case the point sequence represents a underlying curve in its discretized form. That is a demonstration, given as timed sequence $(t_i, \boldsymbol{y}_i)_{i\in[0,N-1]}$ of N points is interpreted as discretized version $\mathcal{C}^{dis}$ of some parametric curve $c\colon [0, T = t_{N-1}] \to \mathbb{R}^d$ for which we have $\boldsymbol{y}_i = c(t_i)$. As we have stated in the introduction and what is explained in greater detail in section 4.5.5, the diffeomorphism is only concerned with matching the geometry of the curves and they can therefore be arbitrarily re-parametrized using any valid parametrization function $h(.)$. Therefore, in this context, the matching quality between a (parametric) source curve $c_s(.)$ and target curve $c_t(.)$ is the minimal cost under a valid (re-) parametrization of the curve.

This is not a tractable problem, as there are infinitely many valid parametrization functions and moreover no easily identifiable but expressive structure can be imposed to reduce the complexity of the problem. We therefore propose a heuristic approach to construct a diffeomorphism based on iterative (re-) parametrization of the point sequence, which is inspired by the point matching Algorithm 4 and performs well in practice.

State-of-the-Art

Here again, most works adressing this issue stem from research in medical imaging. The state-of-the-art approaches (see Glaunès et al. [2008] or Sotiras et al. [2013]) to perform curve matching equally rely on constructing flow based diffeomorphisms, as presented in section 4.4.2. The main difference is the type of metric or cost function used to evaluate the closeness of the source and target.

While for point matching only the distance between source and target has to be accounted for, in curve matching the tangent direction has a meaning too and has therefore to be taken into account by the cost function. In Glaunès et al. [2008] this is done by looking at the natural action of the curve in its parametrised form $c\colon [0,1] : \mathbb{R}^d$ on a vector field $\omega\colon \mathbb{R}^d \to \mathbb{R}^d$:

$$\langle c|\omega\rangle = \int_0^1 c'(s).\omega(c(s))\, ds. \tag{4.63}$$

The value of this action is large if the direction of the vector field is similar to the direction of the curve. This can be used to compare curves by using the target curve to construct a vector field which naturally agrees with the curve and looking at the action between this vector field and the source curve. Then one can again maximize this action using gradient descent methods. Other measures to compare (plane) curves are derived in Bauer et al. [2014].

The drawbacks of this method, regarding our specific context, are basically the same as the ones in the point matching setting, they mainly stem from the flow-based construction of the diffeomorphism causing increased evaluation times, which also grow linearly with the number of points in the demonstrated trajectories/curves to be matched. Also the function to maximize (4.63) does not represent our needs, as it does not only take into account tangent direction, but also the norm of the tangent vector (the speed). We only seek to match the geometry and therefore this measure is not adapted.

Approximative Diffeomorphic Curve Matching

In this section the diffeomorphic curve matching algorithm is detailed. As outlined in the last section, actual curve matching suffers from multiple drawbacks that reduce its utility in this context. We therefore seek to emulate curve matching by iterative reparametrization of the discrete form of the curves to match. We first introduce the algorithm and then detail the different steps.

Consider the problem of curve matching the list of source curves $\left(X_j^{dis}\right)_j$ and target curves $\left(Y_j^{dis}\right)_j$ given in their discrete version. Without stint we can demand that the discretized forms have the same number of points and that the point density is high enough to capture the shape of the trajectory.

One of the key elements of Algorithm 5 is the procedure Naturalize. It takes a list of curves in their discretized form and ensures that $\forall t \in [0,T] : \; s = {}^t\!/_T = \int_0^t \|\boldsymbol{v}_c(z)\|_2\, dz / \int_0^T \|\boldsymbol{v}_c(z)\|_2\, dz$ holds. Or, in words, that the points are distributed equidistantly along the curve, causing the algorithm to match the shape of the curves, disregarding the given velocity profile.

The heuristic in Algorithm 4 chooses the centre and direction of the next LWT based on the largest distance between the current image of the source and target sequence. This is not desirable when confronted with trajectories showing significant variance between them as it can create local minima for the heuristic, resulting in drawback (b). To alleviate this problem we base the deduction of the next center and translation on statistical considerations, as proposed in Algorithm 6. Therefore the algorithm does not have the tendency to minimize the currently largest error between the sequences, but to improve regions with bad matching but similar error vectors (the error vectors between the image of the source curve and the target point in a similar direction having similar amplitudes).

We rely on Algorithm 6 to compute a center and a translation that is likely to improve the matching when used to construct the next locally weighted translation. To achieve this, a matrix is constructed by concatenating the points and errors of all current images of the source curves (line 4). We then cluster these points in the "position-error"-space resulting in the cluster centres $\boldsymbol{\mu}$. These cluster centres can then be interpreted as the concatenation of a point and a translations which is likely to coincide with the

Algorithm 5 Approximative diffeomorphic curve matching

Input $\left(X_j^{dis}\right)_j$, $\left(Y_j^{dis}\right)_j$
Output Φ
Parameter $N_K > 0$, $N_l > 0$ ▷ Number of LWMT N_K, and translations per LWMT N_l
Initialize $\Phi \leftarrow Id$ ▷ Φ is initialized to be the identity function
$\left(Y_j^{dis}\right)_j \leftarrow \textsc{Naturalize}(\left(Y_j^{dis}\right)_j)$ ▷ Compute the natural discrete versions of the targets
for k from 0 to $N_K - 1$ **do**
  $\left(R_j^{dis}\right)_j \leftarrow \left(\Phi\left(X_j^{dis}\right)\right)_j$ ▷ Compute the the current image of the source curves
  $\left(R_j^{dis}\right)_j \leftarrow \textsc{Naturalize}\left(\left(R_j^{dis}\right)_j\right)$
  $\Theta_k = ()$ ▷ Empty parameter vector for the current LWMT
  for l from 0 to $N_l - 1$ **do**
    $\left(Z_j^{dis}\right)_j \leftarrow \left(\Psi_{\Theta_k}\left(R_j^{dis}\right)\right)_j$ ▷ Update the image using the current LWMT
    $\left(Z_j^{dis}\right)_j \leftarrow \textsc{Naturalize}(\left(Z_j^{dis}\right)_j)$
    $(e_j)_j \leftarrow \left(Y_j^{dis} - Z_j^{dis}\right)_j$ ▷ Compute current error
    $\boldsymbol{c}_l, \boldsymbol{v}_l \leftarrow \textsc{ComputeCenterAndDirection}\left(\left(R_j^{dis}\right)_j, (e_j)_j\right)$
    $\beta_l^*, b_l^* \leftarrow \underset{\beta_j, b_j}{\operatorname{argmin}}\ \textsc{Cost}\left((e_j)_j - k_{\boldsymbol{\theta}_l, b_l}\left(\left(R_j^{dis}\right)_j - \boldsymbol{c}_j\right)((\beta_l \boldsymbol{v}_l)\right)$
    $\Theta_k \leftarrow (\Theta_k, (\boldsymbol{\theta}_l, b_l^*, \boldsymbol{c}_l, \beta_l^* \boldsymbol{v}_l))$ ▷ Add the found *optimal* translation
    $\Theta_k \leftarrow \textsc{EnsureDiffeo}(\Theta_k)$
  end for
  $\Phi \leftarrow \Psi_{\Theta_k} \circ \cdots \circ \Psi_{\Theta_0}$ ▷ Update diffeomorphism
  if $\textsc{converged}\left(\Phi, \left(X_j^{dis}\right)_j, \left(Y_j^{dis}\right)_j\right)$ **then**
    Return Φ
  end if
end for
Return Φ

Algorithm 6 ComputeCenterAndDirection

Input $\left(R_j^{dis}\right)_j$, $(e_j)_j$
Output $\boldsymbol{c}$, $\boldsymbol{v}$ ▷ Heuristic guess for next center and translation
Parameter N_c, $\epsilon > 0$ ▷ Number of clusters, Trade-off parameter
$\boldsymbol{D} = \begin{bmatrix} R_j^{dis} \\ e_j \end{bmatrix}_j$ ▷ Concatenate points and errors into one matrix
$\boldsymbol{T} = \left[\textsc{Tangent}\left(R_j^{dis}\right)\right]_j$ ▷ Compute the normalised tangent vectors
$(\boldsymbol{\mu}_i)_{i \in [0, N_c - 1]}, \boldsymbol{l} = \textsc{KMeans}(\boldsymbol{D})$ ▷ Compute clusters and labels
$\begin{pmatrix} \boldsymbol{c}_i \\ \boldsymbol{v}_i \end{pmatrix} = (\boldsymbol{\mu}_i)_i$ ▷ Separate center and translation
$i^* = \underset{i}{\operatorname{argmax}}\left(\|\boldsymbol{v}_i\|_2 + \epsilon\left(1. - \left|\frac{\boldsymbol{v}_i{}^{\mathrm{T}}.(\sum \boldsymbol{T}[:, \boldsymbol{l} == i]}{\|\boldsymbol{v}_i\|_2 N_{li}}\right|\right)\right)$
Return $\boldsymbol{c}_{i^*}$, $\boldsymbol{v}_{i^*}$

error vectors of all points in some neighbourhood (line 7). Finally the *best* such tuple has to be chosen. Here we trade-off between the norm of the translation (which is directly related to the norm of the error of the surrounding points) and the angle between the average tangent direction of the trajectories in the neighbourhood and the translation. Obviously translations perpendicular to the tangent vector of the directions are preferable as these correspond to actually changing the shape of the trajectory, whereas translations parallel to the tangent vector correspond rather to a (local) re-parametrization. In this work we use the k-Means clustering-algorithm based on Llyod's algorithm (as provided by Pedregosa et al. [2011]) as it is one of the fastest (average and worst-case complexity, see Lloyd [1982]). Also there exist very efficient "mini-batch"-versions of this algorithm, in case the number of data points is very large. Using GMMs as clustering methods delivers comparable results, but slows down the overall process considerably.

Once the *optimal* center and translation (direction) found, so after line 14 in Algorithm 5, a suitable base, or size of the kernel, has to be found. Also, in contrast to the heuristic in Algorithm 4, the relative (with respect to the current error vector) length of the translation is not fixed (parameter β). Instead the optimal base size (using gradient descent) is computed for some predefined values for β. We typically define around 10 values ranging from 0 to 1.1. The pair (β, b_β) achieving the lowest cost is then returned. Good choices for the cost are again the mean squared error or the largest eigenvalue of the errors, regularized by a trade-off giving priority to larger bases. If the variance within the given curves to be matched is very high an interesting alternative for the cost function is

$$\mathrm{dist}(X, Y) = \frac{1}{N} \sum_{i=0}^{N-1} \sqrt{\|\boldsymbol{x}_i - \boldsymbol{y}_i\|_2}. \tag{4.64}$$

The so defined cost function is more robust when faced with large regions of dynamical incompatibilities due to its degressive character. Such situations can occur, as for example in the "WShape"-dataset. Here the trajectory marked with the green arrow in Figure 4.16 in the middle image is dynamically incompatible with the other demonstrations due to the intersection and moreover the distance to the other demonstrations is "large". Using the mean squared error as cost function tends to focus on improving the matching for this curve, which is however not feasible and therefore leads to undesired results. The first iterations of this algorithm are showcased in Figure 4.21 and compared to the matching method described in Algorithm 4 in Figure 4.22.

Due to the increased complexity of the algorithm, training times are significantly higher. This increase in computation time is partly due to the necessary clustering, partly to the larger search space. In Algorithm 4, the parameter β (scaling factor for the largest error then used as translation) is predefined and in each iteration only ρ (size of the kernel) is optimized. In Algorithm 5, we search, basically using a grid search approach, for the best norm of the translation vector and the size of the kernel, increasing the complexity. Overall computation-time is however still reasonable with less than half a minute for 7000 points in the sequence X. The online evaluation time of Φ is comparable or even shorter.

4.5.5 Learning Globally Asymptotically Stable Nonlinear Dynamical Systems

In order to be able to take advantage of the variance within a set of trajectories demonstrating the same movement, we seek to construct globally asymptotically stable control-space dynamics which are more expressive than the prototypic dynamic function used within the One-Step learning. Then we show how this can be combined with a statistical model used to learn the magnitude of the velocity in the demonstration-space and a diffeomorphic transformation obtained with Algorithm 5.

Control-Space Dynamics

We have seen that we can match the geometry or shape of a list of source curves onto the corresponding target curves using a diffeomorphism performing curve matching. Moreover, recalling Theorem 4.4, it

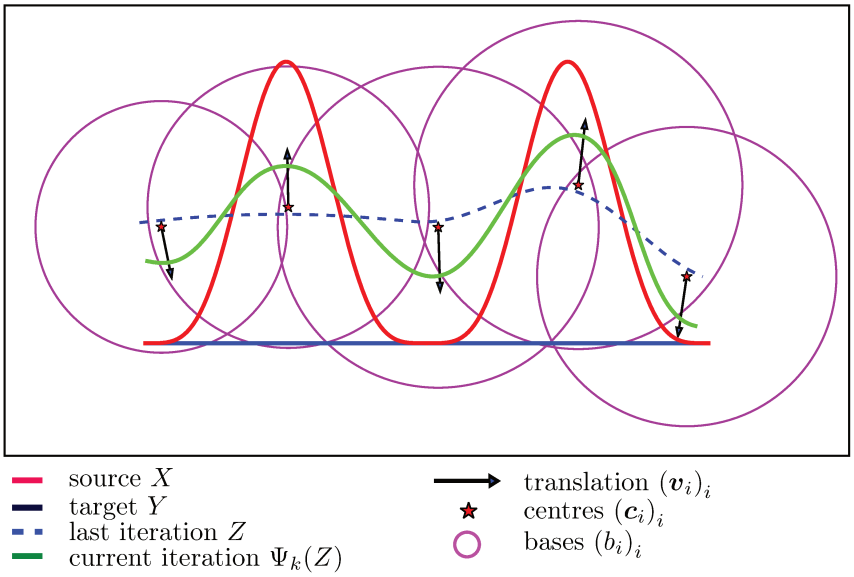


Figure 4.21 – This image depicts the first two iterations of the approximative curve matching using LWMT. As one can see the centres and translations deduced from difference of the image of the source curve from the last iteration Z and the target Y align well with the clusters of large errors, empirically justifying the clustering approach.

was shown that two dynamical systems which are geometrically equivalent via some $\mathcal{C}^1$-diffeomorphism are either both GAS or neither of them is GAS.

To obtain suitable control-space dynamics, wee seek to obtain the following properties.

- All trajectories converge with respect to the l_2-norm, or, equivalently, $\|\boldsymbol{x}\|_2$ is a Lyapunov function.
- To avoid the drift error, the dynamic has to be such that states converge towards the trajectories from within some neighbourhood around the trajectories.
- Trajectories which are close remain close or even converge, i.e. the vector field is contracting around the trajectories
- The control-space dynamics have to represent the variance of the given demonstrations.

The first point is important to ensure that both dynamical systems are GAS. As the approach has to be generic for all sorts of trajectories, the euclidean distance is a natural choice for the Lyapunov function, but other Lyapunov candidates could also be used. The second point has proven to be very desirable to obtain good reproductions but also to be able to reject possibly occurring perturbations if used to control a real system. The third point is important to ease the construction of the diffeomorphism. Finally the fourth point is important to avoid situations like the one shown in Figure 4.16, where a large variance within the demonstration set causes many reproductions to be of poor quality.

While the first and last point could also be accomplished by relying on SEDS to learn suitable control-space dynamics, the second and third point are beyond the scope of SEDS. Therefore we propose to use a statistical approach called locally converging directions (LCD). This approach learns a vector field which is such that states in some neighbourhood around the trajectory will converge towards it, however without stability guarantees. Constructing such vector fields around converging versions (with respect to the l_2-norm, construction shown in Figure 4.24) of the demonstrated trajectories, it is highly likely that the LCD admits the euclidean distance as Lyapunov function within some neighbourhood of the

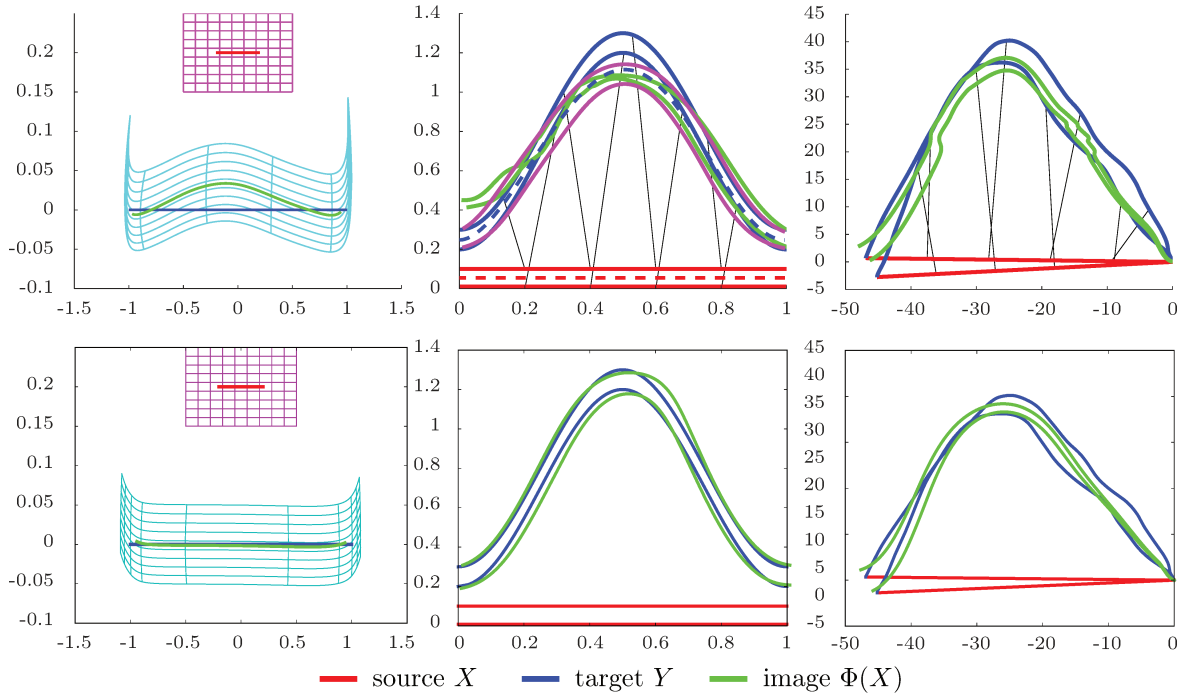


Figure 4.22 – Comparing the transformations resulting from different settings using Algorithm 4 (top row) and Algorithm 5 (bottom row). In the left column we seek the match two straight line segments with a limited number of transformations. In the upper row, 12 translations are used, in the bottom row three LWMT with 4,3 and 4 translations are used. As one can see the simultaneous application of multiple kernels and the heuristic based on clustering can lead to transformations with improved matching and smaller gradients. On the right matching results for dynamically compatible configurations are shown. The green lines correspond to the image of the source curves under the constructed dffeomorphism. Here (approximative) curve matching leads to smooth and accurate results, whereas the results in the top row get stuck in local minima leading to very jerky transformations. In the image top middle, the transformation resulting from averaging the given trajectories is shown (magenta lines are $\Phi(X)$) in addition to the result from seeking to match the simultaneously. Here the dynamical incompatibility is already large enough to significantly change the geometry of the trajectories, resulting in a non-representative meta-demonstration (dashed blue line). On the right the results for two trajectories from the "Angle"-demonstrations are shown. Here too the dynamical incompatibility prevents a successful simultaneous matching of the two point sequences.

demonstrations. As this is however hard to prove given the structure of the LCD, we apply a correction signal like the one in Khansari-Zadeh and Billard [2014] whenever the convergence criterion would be violated.

Locally converging directions are defined as weighted sum over a set components. The direction given by each component is computed using Algorithm 7 and is based on a weighting function, a center point and a base direction. In line 5, the distance vector between the point considered $\boldsymbol{x}$ and the center point $\boldsymbol{\mu}$ is projected into the nullspace of the base direction $\boldsymbol{d}^*$, denoted $\boldsymbol{dy} \in \mathbb{R}^{d-1}$. This vector is then element-wise split into its absolute value and sign. The absolute value is then lowered by the factor α and bounded from below by 0. The desired direction $\boldsymbol{d}$ is then given as the sum of the base direction and the sum of the basis vectors forming the nullspace (The i-th row of P is the i-th basis vector) multiplied with the elements of $\boldsymbol{dy}$. Finally we scale the resulting direction using the weight function w. The resulting

Algorithm 7 WeightedDirection

1: Input $\boldsymbol{x}$ ▷ Point for which to compute the direction
2: Parameter μ, $\boldsymbol{d}^*$, $\alpha \geq 0$, $\beta \leq 0$
3: Output $\boldsymbol{d}$
4: $P = \text{null}\left(\boldsymbol{d}^{*\mathrm{T}}\right)$
5: $\boldsymbol{dy} \leftarrow P.(\boldsymbol{x} - \boldsymbol{\mu})$
6: $\boldsymbol{dy}_s \leftarrow \text{sgn}(\boldsymbol{dy})$
7: $\boldsymbol{dy}_a \leftarrow \text{abs}(\boldsymbol{dy})$
8: $\boldsymbol{dy}_a \leftarrow \max(0, \boldsymbol{dy}_a - \alpha)$
9: $\boldsymbol{dy} \leftarrow \boldsymbol{dy}_a \cdot \boldsymbol{dy}_s$
10: $\boldsymbol{d} \leftarrow \boldsymbol{d}^* + \sum_i P^{\mathrm{T}}[:, i]\boldsymbol{dy}[i]$
11: $\boldsymbol{d} \leftarrow \boldsymbol{d}w(\boldsymbol{x} - \boldsymbol{\mu})$
12: Return $\boldsymbol{d}$

vector field is therefore parallel to the base direction in zone defined by $\boldsymbol{\mu}$, $\boldsymbol{d}^*$ and α. It converges towards the straight line going through $\boldsymbol{\mu}$ in the direction of $\boldsymbol{d}^*$. The convergence rate is defined by the parameter β and the resulting vector field is showcased in Figure 4.23.

Finally, in order to be able to obtain expressive vector fields, multiple such components are added up according to Algorithm 8. The resulting direction from adding up the components is neither guaranteed to be GAS nor to admit $\|\boldsymbol{x}\|_2^2$ as Lyapunov function. Therefore a correction signal is applied if the convergence criterion is modified. Here we have chosen a simple correction type (line 6), where $\gamma^*_{\boldsymbol{x},\boldsymbol{d}}$ is the smallest non-negative value for which $\boldsymbol{x}^{\mathrm{T}}.(\boldsymbol{d} - \gamma\boldsymbol{x}) < 0$ holds. As we use the geometric equivalence between vector fields, the norm of the resulting direction is arbitrary and therefore scaled to have unit norm.

Algorithm 8 CombinedWeightedDirec

1: Input $\boldsymbol{x}$
2: Output $\boldsymbol{d}$
3: Parameter $(\boldsymbol{\mu}_i, \boldsymbol{d}_i^*, \alpha_i, \beta_i)_i$, $\epsilon < 0$ ▷ List of parameters
4: $\boldsymbol{d} \leftarrow \epsilon \frac{\boldsymbol{x}}{\|\boldsymbol{x}\|_2}$ ▷ Base convergence
5: $\boldsymbol{d} \leftarrow \boldsymbol{d} + \sum_i \textsc{WeightedDirec}_{\boldsymbol{\mu}_i, \boldsymbol{d}_i^*, \alpha_i, \beta_i}(\boldsymbol{x})$ ▷ Sum over components
6: $\boldsymbol{d} \leftarrow \boldsymbol{d} - \gamma^*_{\boldsymbol{x},\boldsymbol{d}}\boldsymbol{x}$
7: $\boldsymbol{d} \leftarrow \frac{\boldsymbol{d}}{\|\boldsymbol{d}\|_2}$ ▷ Normalize
8: Return $\boldsymbol{d}$

Converging Trajectories Consider one or multiple trajectories for which parts are convergent with respect to the l_2-norm and other parts are not. In order to reduce the necessary state-space deformation induced by the diffeomorphism while admitting the l_2-norm as Lyapunov function, we seek to replace the non-convergent parts of the trajectories. As shown in Figure 4.24, this is achieved by the following method: consider the discretized curve $(\boldsymbol{y}_i, \boldsymbol{y}'_i)_i$ where $\boldsymbol{y}'_i$ denotes the normalized tangent vector at point $\boldsymbol{y}_i$. Assume that j is the first index for which ${\boldsymbol{y}_j}^{\mathrm{T}}.\boldsymbol{y}'_j < 0$ does not hold. Then we compute the first index j' for which $\left\|\boldsymbol{y}_{j'}\right\|_2 < \left\|\boldsymbol{y}_j\right\|_2 (1. - \epsilon)$ and ${\boldsymbol{y}_{j'}}^{\mathrm{T}}.\boldsymbol{y}'_{j'} < 0$ hold for the predefined convergence margin $\epsilon > 0$. Then the trajectory segment from index j to j' can be replaced by a converging arc segment. By repeating this procedures until the convergence criterion holds for all points, one obtains a converging version of the trajectory minimizing the difference between the resulting and the original trajectory while guaranteeing convergence. The so obtained trajectory can then be learned using LCD by distributing components along each trajectory. This approach therefore takes advantage of the variance between the

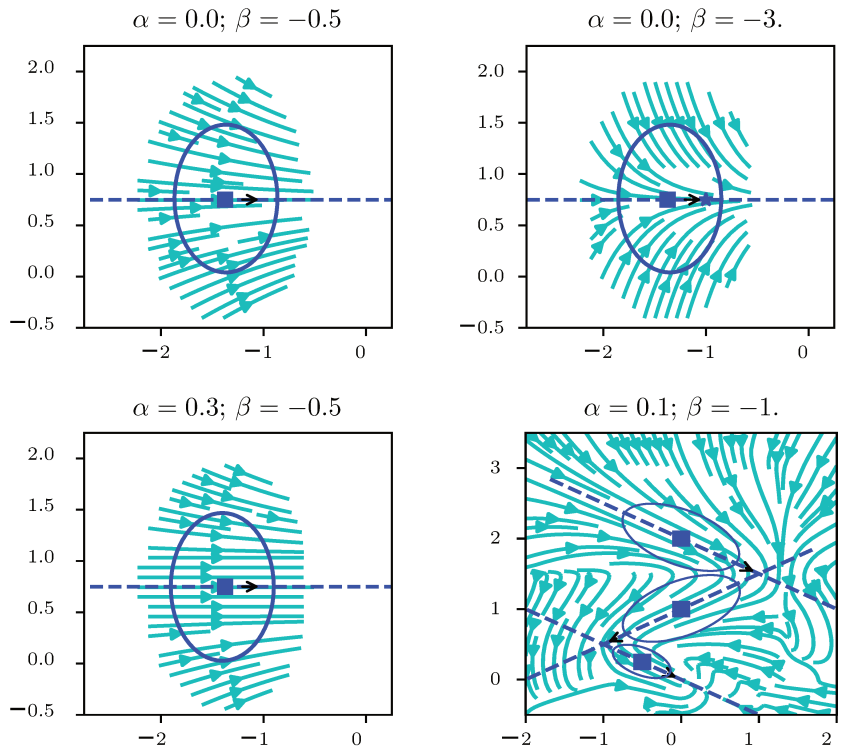


Figure 4.23 – These images depict the vector fields obtainable by using single components and adding up several of them. Each component is defined by the base direction (black arrow) and the center $\boldsymbol{\mu}$ (blue square). The first row depicts the influence of the parameter β. The zone resulting in parallel directions to the base direction defined by the parameter α is shown in the lower left image. The Gaussian kernel used as weighting function is depicted by the blue ellipsoid. Finally the vector field resulting from multiple (3) is depicted in the lower right image.

different demonstrations, as each of them is taken into account and also generalises into the surroundings of each demonstration. Moreover it averages the behaviour if multiple demonstrations are available for the same region. Therefore this approach solves the drawback (a), as a labelling of the demonstrations is no longer needed, and partly resolves drawback (b). It also resolves drawbacks (c), as the expressiveness of the control-space dynamics is no longer reduced when learning from multimodal demonstration sets.

Demonstration-Space Dynamics and Lyapunov Functions

Now all parts are in place and the demonstration-space dynamics can finally be constructed by performing the following steps:

1. Learn a model for the speed in the demonstration space denoted $m(\boldsymbol{y})$
2. Construct the converging trajectories $(Y_j^{conv})_j$ from the demonstrations $(Y_j)_j$
3. Deduce suitable control-space dynamics from the converging trajectories $\left(Y_j^{conv}\right)_j$ using LCD, denoted $f(\boldsymbol{x})$
4. Compute the forward trajectories of the initial point of each demonstration under $f(\boldsymbol{x})$ denoted $(X_j)_j$ with $X_j(0) = Y_j(0)$

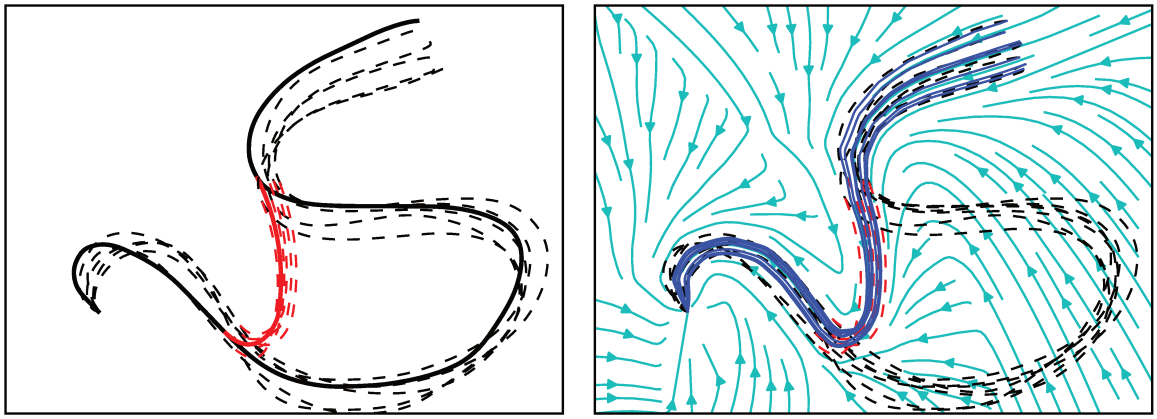


Figure 4.24 – On the left the construction of the converging trajectory is depicted. The original trajectories are shown in black (solid or dashed). The segments replacing the nonconvergent, almost looping part of the trajectory are shown in red (solid or dashed). Note that if the geometry of the trajectories is similar, the segments to be replaced and their replacement will be similar too. On the right the resulting vector field for placing 5 components along each trajectory is shown. α and β are chosen empirically, the influence region of the Gaussian kernel used as weighting function is adapted to the shape of the trajectory segment represented by it. As one can see, the reproductions follow closely the converging trajectories and trajectories starting in some neighbourhood around the trajectories tend to converge towards them.

5. Perform curve matching between the forward trajectories $(X_j)_j$ and the given demonstrations $(Y_j)_j$ to obtain the diffeomorphism Φ

6. Define the demonstration-space dynamics $\dot{\boldsymbol{y}} = g(\boldsymbol{y})$ according to Algorithm 9

In this work we use GMM's, trained via greedy insertion and EM as proposed in Verbeek et al. [2003], as speed models. In order to take into account that the speed, equivalent to the norm of the velocity, is always non-negative, we impose a small positive lower bound on the result. The resulting model is showcased in Figure 4.25.

Note that the learned control-space dynamics do not necessarily have to reproduce well the converging trajectories, as the forward trajectories resulting from it are used to construct the diffeomorphism (in contrast to the approach presented in the last section). However it is advantageous if the resulting trajectories show some similarity, reflecting the similarity of the original demonstrations as this facilitates the construction of the diffeomorphic transformation.

If the matching between the forward trajectories and the demonstrations as well as the model representing the speed were perfect, the forward trajectories under the demonstration-space dynamics starting at the initial point of the demonstration would perfectly reproduce it.

Algorithm 9 Demonstration-space dynamics $g(\boldsymbol{y})$

1: Input $\boldsymbol{y}$
2: Output $\dot{\boldsymbol{y}}$
3: Parameter $\Phi(.)$, $m(.)$, $f(.)$ ▷ Diffeomorphism, scaling function and control-space dynamics
4: $\dot{\boldsymbol{y}} \leftarrow J_\Phi.f\left(\Phi^{-1}(\boldsymbol{y})\right)$ ▷ Obtain the direction of $\dot{\boldsymbol{y}}$
5: $\dot{\boldsymbol{y}} \leftarrow \frac{m(\boldsymbol{y})}{\|\dot{\boldsymbol{y}}\|_2}\dot{\boldsymbol{y}}$ ▷ Normalize and scale
6: Return $\dot{\boldsymbol{y}}$

Due to the geometric equivalence between the control-space and demonstration-space dynamics via the $\mathcal{C}^1$-diffeomorphism Φ constructed by Algorithm 5, the global asymptotic stability of the control-space dynamics is directly transferred to the demonstration-space dynamics. As $V_{\mathcal{X}} = \|\boldsymbol{x}\|_2^2$ is imposed as control-space Lyapunov function, $V_{\mathcal{X}} \circ \Phi^{-1}$ is again a valid demonstration-space Lyapunov function.

The higher similarity between the forward trajectories of the learned control-space dynamics and the given demonstrations reduce the necessary deformation of the state-space induced by the diffeomorphism. This results in a higher safety margin ensuring that the diffeomorphism is invertible and therefore resolving drawback (e).

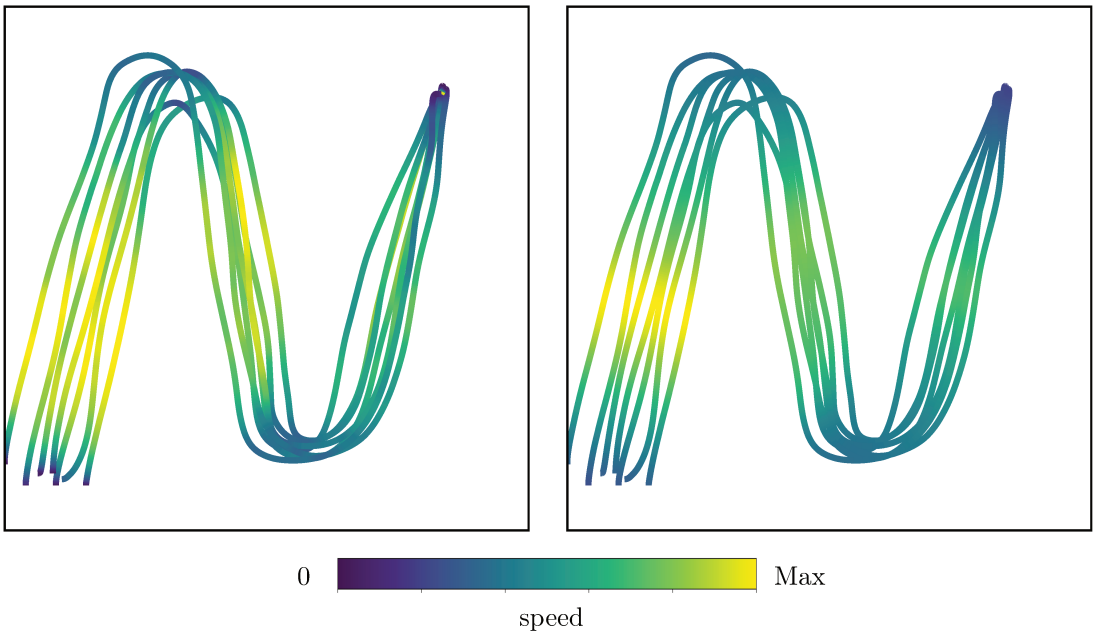


Figure 4.25 – On the left the velocity norm or speed of the given trajectories, encoded as color, are shown for the "NShape" dataset. On the right the learned speed using a GMM with 5 components trained via greedy insertion is shown. The colour corresponds to $m(\boldsymbol{y})$ evaluated at each of the given demonstrations. The learned velocity represents well the pattern of the given demonstrations, quantitatively and qualitatively. Note that the number of components in the GMM is not predefined, but components are iteratively inserted until the model suffices some convergence criterion.

4.5.6 Results

In this section we compare the results obtained using LWMT with approximative curve matching and learned control-space dynamics using LCD with the One-Step learning proposed in the last section. Special attention is paid to cases in which one significantly outperforms the other. Most of the shown test-cases are from the LASA-Dataset, others are inspired or derived from them.

At first we take a closer look at the resulting control-space dynamics and the deformations induced by the diffeomorphism and compare them to the ones obtained with Algorithm 4.

Results for the LASA-Dataset and Comparison with One-Step Learning

The first example shown in Figure 4.26 is the "GShape". Using the One-Step learning, this is a difficult dataset, as the trajectories resemble a spiral. Therefore many iterations are needed to match the

straight line segment onto the meta-demonstration. This results in a diffeomorphism with very high gradients and a poor matching quality at the beginning of the motion. Here the reproduced trajectories show some undesirable "pleats", before the vector field becomes smoother later on the trajectory. The demonstration-space Lyapunov function seems to have local minima, however this is not the case, here again the problem is the high deformation causing artificats in the plot. The Two-Step learning provides a demonstration-space vector field that is everywhere "smooth", even at the beginning of the trajectories. The converging trajectories have a smaller distance to the given demonstrations, facilitating the search for a diffeomorphism. Therefore the demonstration-space Lyapunov function is also closer to the control-space Lyapunov function $V_{\mathcal{X}}(\boldsymbol{x}) = \|\boldsymbol{x}\|_2^2$. This comes at the cost that the Lyapunov function is however less adapted to the dynamics, in the sense that the angle between the velocity and the level-set is stepper for the One-Step learning. The obtained demonstration-space dynamics generalise well the given demonstrations and provide good reproductions. The next demonstrations set, the "WShape" is chosen as it demonstrates well the ability of the Two-Step learning to better deal with or even profit from a higher variance within the data. One-Step learning is not able to generalise the meta-demonstration to a large enough region to cover all initial points of the demonstrations, causing large differences between the demonstrations and reproductions. Here Two-Step learning performs significantly better. Moreover we can see that the approach of clustering the error works well even when presented with significant differences in the geometry of the curves, which is the case here. For this demonstration-set it is also advantageous to use the cost function defined in (4.64) as one trajectory has a significantly different shape than the others (indicated by the green arrow). Both approaches result in comparable demonstration-space dynamics and reproductions for the last presented dataset, "Leaf_2". The main difference is the significantly reduced distortion of the state-space and the significantly reduced amount of kernels needed. For the One-Step learning around 100 LWT are needed to obtain good results (Here the result for 150 LWTs is shown), whereas in the Two-Step learning the diffeomorphism comprises 3 LWMT with a total of 24 translations.

Figure 4.27 shows the results obtained using Two-Step learning for a broader variety of demonstration-sets. Of special interest here are the multimodal demonstrations, which can be handled by the Two-Step algorithm without further information. The top row shows the results for "easy-to-learn" motions, like the "DoubleBendedLine" and the "Snake". The second row shows demonstration-sets which are harder to learn. Both are more difficult as the angle between the converging trajectories and the demonstrations are large at the beginning of the movement. For these cases the control-space dynamics cannot be very rich, as the demonstrations have large parts that diverge with respect to the l_2-norm. Moreover the demonstration-set "Leaf_1" is inherently difficult as some details of the motion (the top peak) are smaller than the variance within the demonstration. The last row shows demonstrations posing problems to the One-Step learning. The right configuration can only be learned using manual adjustments, whereas the "DoubleSharpC" is infeasible when relying on the proposed approach of computing meta-demonstrations and matching straight line segments.

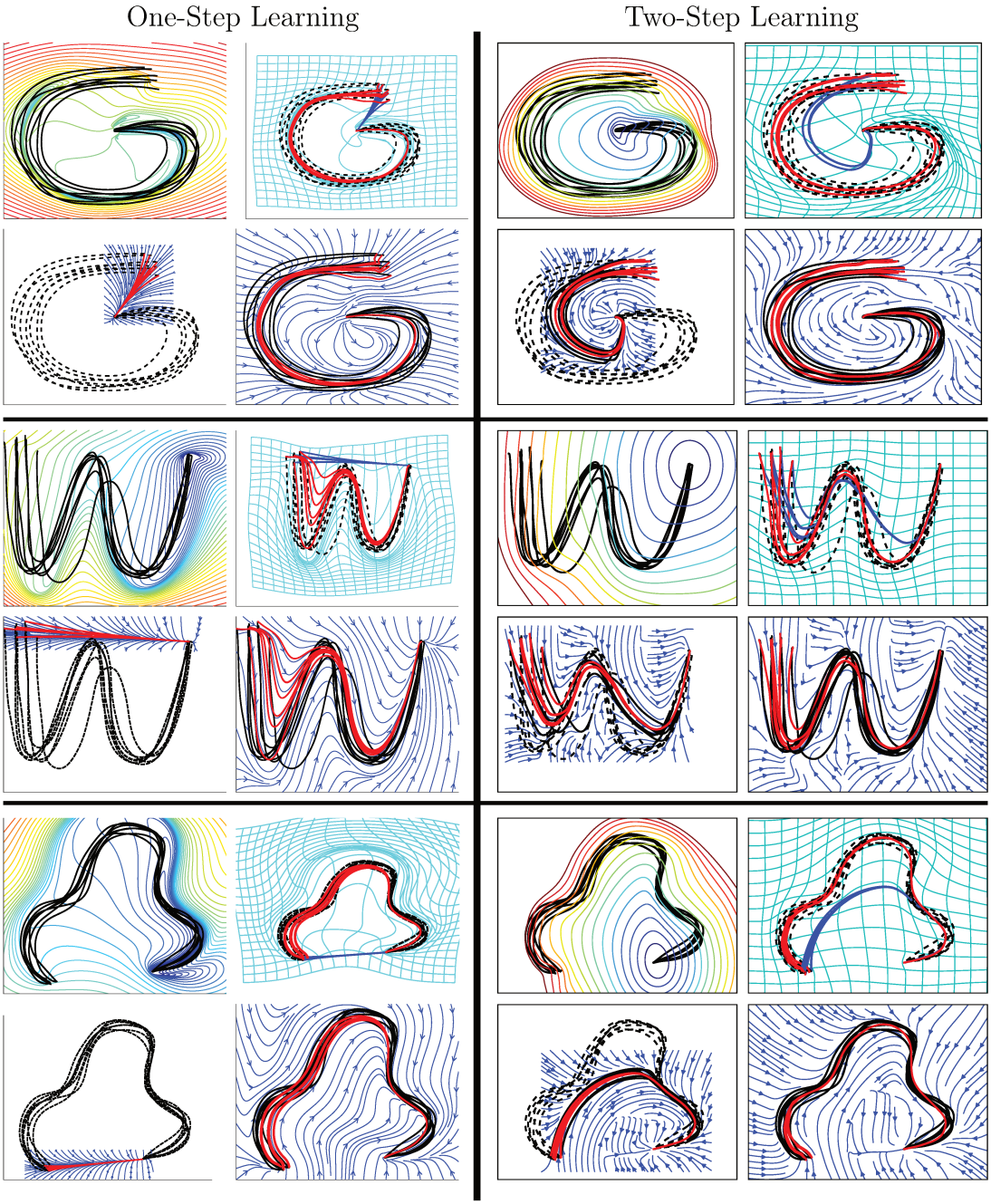


Figure 4.26 – The two left columns show the results obtained using One-Step learning, on the right the results for the presented Two-Step learning are shown. The considered datasets are "GShape", "WShape" and "Leaf_2". For each set we show: top left the demonstration-space Lyapunov function $V_{\mathcal{Y}}$; top right demonstrations (dashed black), the image of the control-space trajectories starting at the intial points of the trajectories (solid red) and the control-space trajectories (solid blue). Moreover the diffeomorphism depicted by showing the image of a rectangular grid under the transformation. Bottom left shows the control-space dynamics and trajectories (blue streamlines and solid red lines). For the Two-Step learning the converging trajectories are shown solid black. Finally bottom right depicts the obtained demonstration-space dynamics as streamplot (blue lines) and the obtained reproductions (solid red lines).

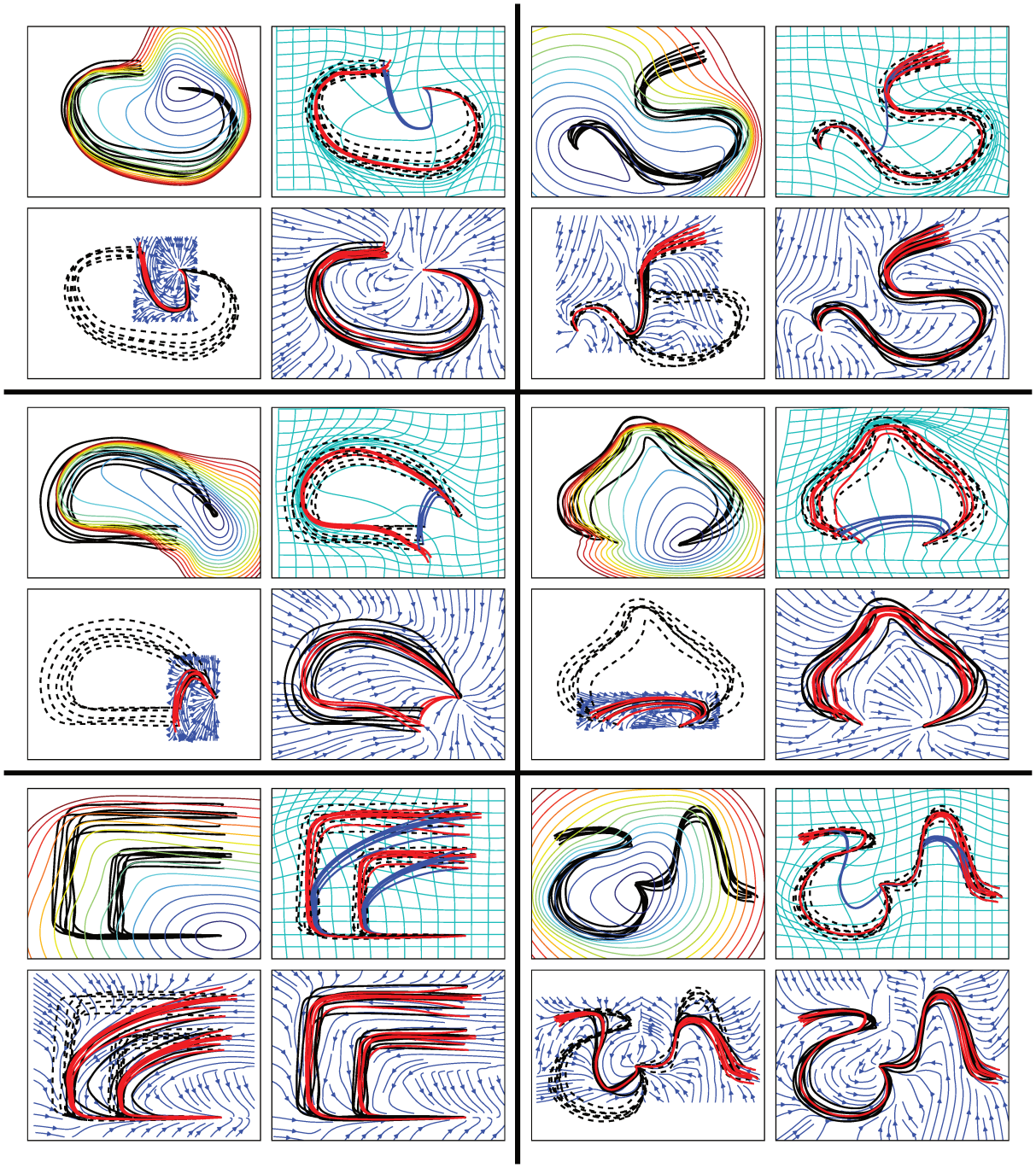


Figure 4.27 – Results obtained using Two-Step learning for demonstrations-set taken or inspired by the LASA-Dataset. The four images presented for each demonstration-set correspond to the ones in Figure 4.26

Evaluating the Dynamics

In Figure 4.26 and Figure 4.27 the ability of the Two-Step learning to reproduce and generalize the geometry of a given set of demonstrations is shown. In order to evaluate the quality of the learned dynamics, we again take a closer look at the generated velocity profiles in Figure 4.29 and compare some of them with the ones obtained via One-Step learning Figure 4.28.

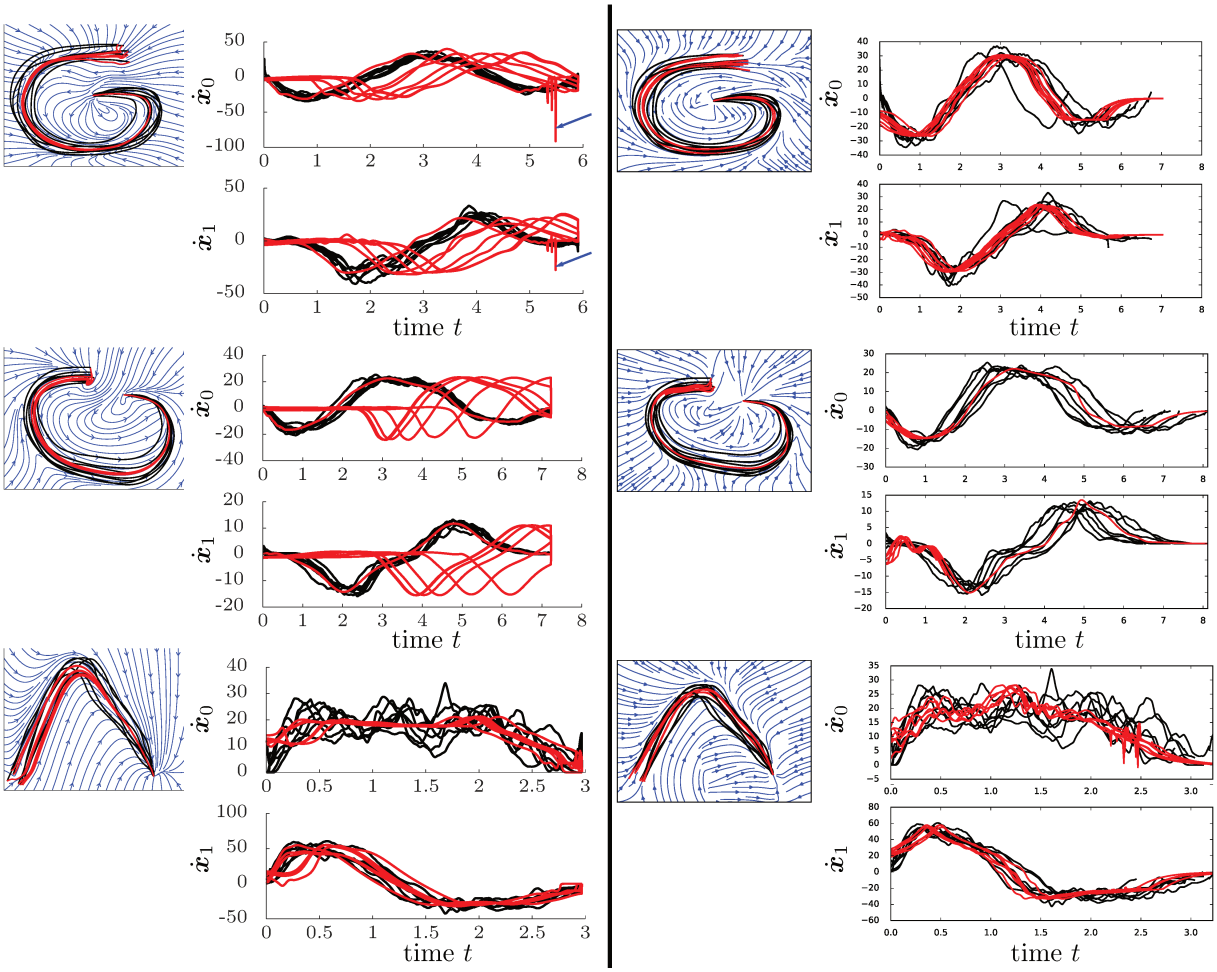


Figure 4.28 – The resulting velocity profiles for the One-Step learning are shown on the left, those corresponding to Two-Step learning appear on the right.

In Figure 4.28 one can see that the Two-Step learning approach significantly reduces the error between the velocities given in the demonstrations and the reproductions at the very beginning of the movement. This helps to prevent "time-delays" between the reproduction and the demonstration. The higher regularity of the transformation prevents (seldomly) occurring velocity peaks as the ones indicated by the blue arrow in the top left image. In general, the Two-Step learning allows to learn velocity profiles which are very similar to those given by the demonstrations, but have the tendency to have high frequency variations of low amplitude. This is partly caused by the use of Polynomial Radial Basis function which have a finite base and are $\mathcal{C}^2$ compared to the $\mathcal{C}^\infty$ kernels used in One-Step learning, partly by the construction of control-space dynamics as sum over different components defining the direction and whose

blending causes the oscillations.

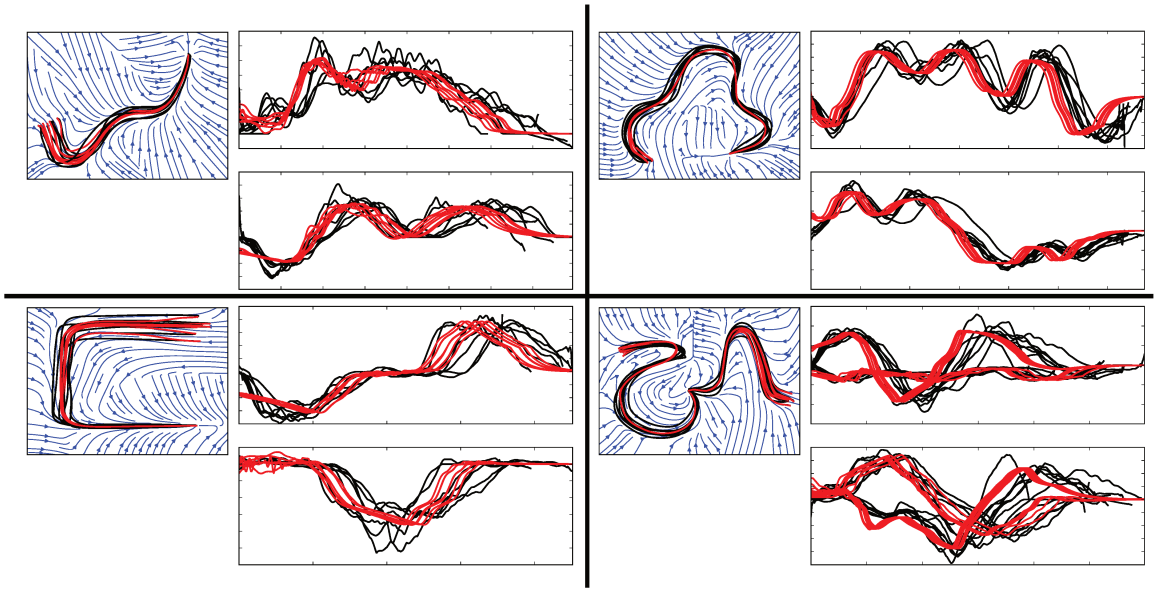


Figure 4.29 – Resulting velocity profiles for the Two-Step learning for four different demonstration-sets.

The demonstration-sets shown in Figure 4.29 all have certain particularities, but they can nonetheless be effectively learned using the proposed Two-Step approach, even when looking at the velocity profile. The trajectories of "Khamesh" in the top left image are only divergent (with respect tot the l_2-norm) at the beginning of the trajectories. The particularity is that this region has a similar size as the variance between the demonstrations is therefore more difficult to learn. The "SharpC" (left bottom) is difficult due to the high curvature in the corners, which is not present in the converging trajectories. To accurately represent "Leaf_2" (top right), the velocities in $\dot{\boldsymbol{x}}_0$ and $\dot{\boldsymbol{x}}_1$ must be well synchronised, posing typically problems for methods where the dynamics of each dimension are decoupled like for dynamic motion primitives Ijspeert et al. [2003] or Schaal [2006]. Finally the results for a multimodal demonstrations set is shown (bottom right). All learning processes (control-space dynamics $f(\boldsymbol{x})$, speed model $m(\boldsymbol{y})$ and the diffeomorphism Φ) are learned or constructed based on unlabelled data.

Higher Dimensional Data-Sets

An advantage of the here proposed One-Step and Two-Step learning approaches is that they scale well with higher dimensional problems. These approaches rely on heuristic algorithms to construct the diffeomorphic transformation and, in the case of Two-Step learning, also the control-space dynamics. This stands in contrast to other learning from demonstration methods in the literature, in which the problem of finding the parameters of dynamical system is reformulated as an optimization problem. In this case the number of variables in the optimization grows at least quadratically (Khansari-Zadeh and Billard [2011, 2014]) or even binomial (Blocher et al. [2017][5]). The optimizations performed within the proposed heuristics Algorithm 4 and Algorithm 6 are used to optimize a scalar variable using line or grid searches and are therefore not effected by increasing the dimension[6] of the control and demonstration space.

[5]The number of coefficients in the polynomials used to generate the contraction metric

[6]By "not affected" we mean, that the computation needs the same amount of atomic function calls. That for instance $\|.\|_2$ is called equally often no matter the dimension of the state, but of course each atomic function call takes longer to compute in a higher dimensional spaces.

In order to evaluate the ability to learn nonlinear vector fields from demonstration in high-dimensional spaces, we concatenate 5 different trajectories from the LASA-Dataset as to obtain demonstration trajectories of dimension 10. Specifically we concatenate the demonstrations "Leaf_2", "GShape", "BendedLine", "DoubleBendedLine", "Leaf_1", "Sharpc", "Snake" and "NShape", so that the first and second dimension correspond to "Leaf_2", the third and forth to "GShape" and so forth.

In this case learning, that is constructing the control-space dynamics and the diffeomorphism takes about 30% longer than for two-dimensional datasets, forward integrating the a trajectory takes about three times longer[7]. Matching results and generalisation, as shown in Figure 4.30, are of comparable quality than for 2d datasets, using the same parameters. As we propose to learn the vector field from demonstrations given as trajectories, that is curves embedded in the demonstration space, one could expect to obtain worse generalisation of the movements when the dimension of the demonstration space increases. This would be natural, as the curve, so a one-dimensional manifold, contains (relatively) less and less information when the dimension grows, however it seems that the control-space dynamics based on locally contracting directions seems to counter this problem.

[7]The construction of the diffeomorphism is implemented in python and has therefore a significant computational overhead. Therefore the time needed to construct the diffeomorphism grows only slightly even though the dimension was significantly increased. The forward integration is implemented in C++ relying on Eigen and has therefore less overhead and computation times show a higher correlation with the dimension.

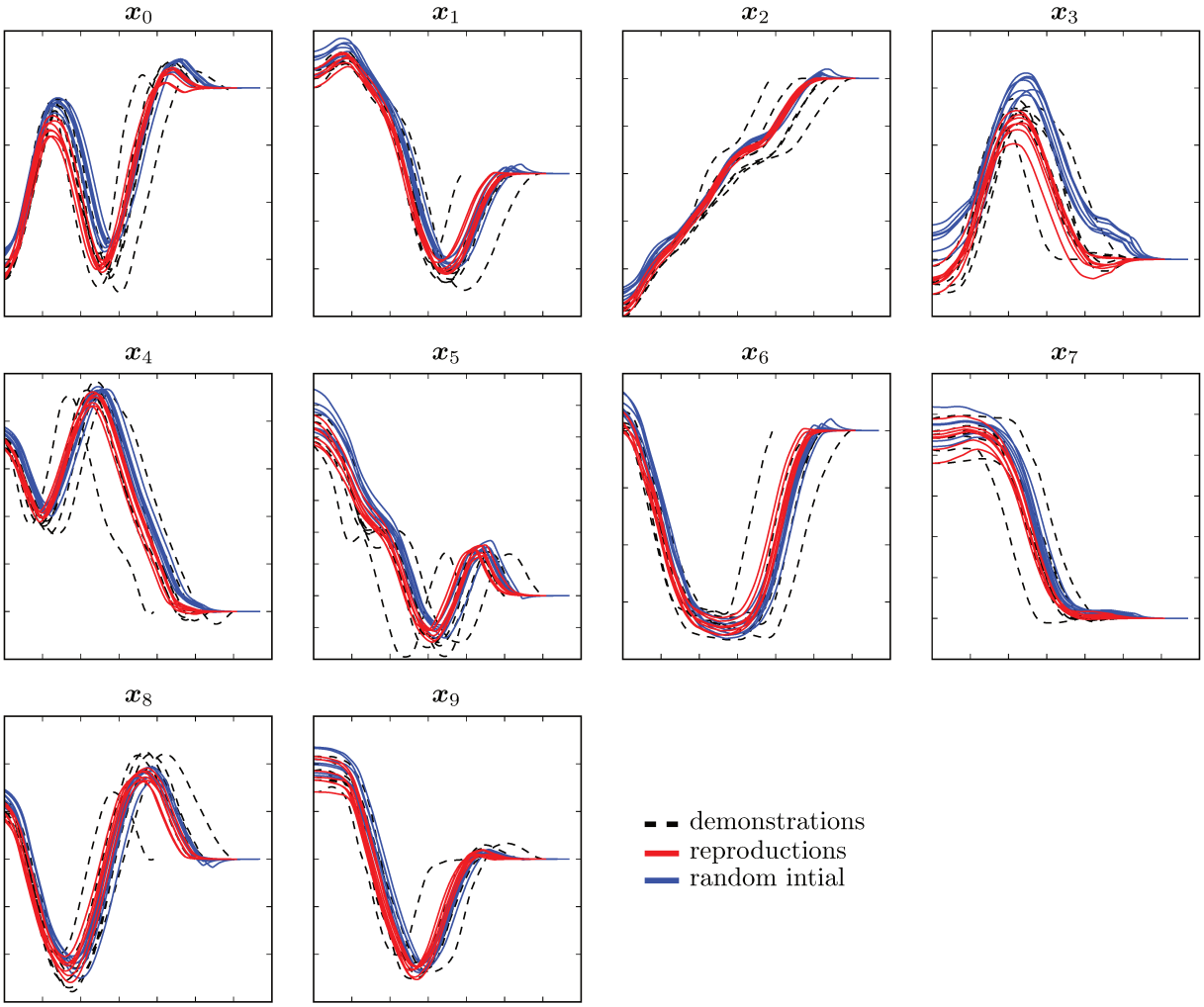


Figure 4.30 – Resulting reproductions for learning 10 dimensional demonstrations. As the vector field and the trajectories can only be plotted in a meaningfull way for two dimensional problems, each coordinate $\boldsymbol{x}_i$ is plotted as a function of time. As we can see the demonstrations are successfully learned, by increasing only very slightly the overall computational complexity. The diffeomorphism found for these demonstrations is composed of 5 LWMT with a total of 30 translations.

4.5.7 Robot Experiments

To validate the approach we also performed experiments on a robotic platform, the humanoid robot Sigmaban. This robot is developed at the informatics laboratory LaBRI at University of Bordeaux 1 and is a multiple times Robocup champion in the kid size league, see Fabre et al. [2015] and Rouxel et al. [2016].

These experiments have been performed before the final version of the Two-Step learning has been established and are therefore based on a slightly different algorithms and methods. It can be seen as an intermediate step between the presented One-Step and Two-Step learning approach. The main differences are

- The control-space dynamics are learned using a GMM trained via greedy insertion (instead of LCD).
- The diffeomorphic matching is based on LWMTs and clustering to behave better when faced with variances within the demonstration-set, but point matching is performed. That is, the matching is based on Algorithm 5, but the procedure NATURALIZE does not change the point sequence.
- The piecewise Polynomial Radial Basis function was defined as piecewise continuous and continuously differentiable function inside of the base, so the resulting kernel is $\mathcal{C}^1$ (Instead of $\mathcal{C}^3$ for the examples presented so far).

Robotic Platform and Experimental Set-Up

The humanoid robot Sigmaban has a total of 20 degrees of freedom (DoF), six per leg, three per arm and two for the head. All DoF are actuated using Dynamixel™ servo-motors, based on small DC-motors and integrated gearboxes with a gear ratio of about 200. The servos are controlled using specifically designed integrated circuits in order to allow for reasonably high control frequencies around 100Hz.

The servomotors used in the legs are Dynamixel MX106 which are position controlled. That is only a goal position for the servomotor can be externally set and an internal control loop based on a PID (proportional-integral-derivative)-controller tracks the goal position by modifying the duty-cycle of the pulse-width-modulation (PWM) powering the actual DC-motor.

In this test we are only interested in performing a kick-like motion with one foot involving hip, knee and ankle, basically forming a 3R-robot. As there is no stability control mechanism in the control architecture, but the walking and performed shots have to be open-loop stable, the robot is fixed during the experiments.

Dynamic Model and Feedforward-Control The proposed approach constructs a vector fields that allows to accurately reproduce the given demonstrations. That is for a given point in the state-space the learned model returns a desired velocity. This is however not directly compatible with the position control of the servomotors. The most direct approach of simply setting the target position q^*_{k+1} to the current position q_k plus the currently desired velocity multiplied with the cycle time of the control loop $\delta q_k = \dot{q}^*_k \delta t$ does not work either. As the dynamical model of the robot is not accounted for in the internal control law of the servo-motor, the joint torques necessary to compensate for gravity and accelerations forces are not taken into account directly but have to generate a position error before being compensated for by the internal PID-controller. During dynamic movements or in positions necessitating "large" torques to compensate for gravity, this error is order of magnitudes larger than the position offset δq_k which is therefore basically disregarded. For these reasons a different approach based on feedforward control and model identification is used.

Given the standard dynamics equation of a robotic system based on the rigid-body dynamics given as

$$M_{\boldsymbol{q}}.\ddot{\boldsymbol{q}} + C_{\boldsymbol{q},\dot{\boldsymbol{q}}}.\dot{\boldsymbol{q}} + g_{\boldsymbol{q}} = \boldsymbol{\tau}$$

with $M_{\boldsymbol{q}}$ being the mass matrix, $C_{\boldsymbol{q},\dot{\boldsymbol{q}}}$ the matrix of nonlinear effects, $g_{\boldsymbol{q}}$ denoting the gravity vector and $\boldsymbol{\tau}$ the vector of joint torques (generalised efforts, however all joints are revolute joints).

Therefore if one disposes of the current position and velocity as well as the desired acceleration, one can compute the necessary joint torques. In order to generate these torques with the given servomotors, we need to finely control the duty cycle of PWM driving the motor and relate it to the generated torque via some mathematical model.

To achieve this, first consider the simple electrical model of a DC-motor, in which the dynamics of the electrical parts are neglected due to their significantly smaller time constants:

$$\tau = \frac{k_\tau}{R}U - \frac{k_\tau^2}{R}\omega = \frac{k_\tau}{R}U_S\alpha_{\mathrm{PWM}} - \frac{k_\tau^2}{R}\omega \tag{4.65}$$

with k_τ being the torque constant of the DC-motor, R the ohmic resistance of the motor and ω the rotation speed. The tension at the motor U is simply the supply tension times the duty cycle of the PWM. This means that we can control the servomotor on a torque level if we can control the duty cycle of the PWM, as we can hope to achieve the desired torque by setting α_{PWM}[8].

To obtain the desired duty cycle we use only the proportional control of the internal PID by setting the coefficients of the integral and derivative to zero. Then the duty cycle of the PWM α_{PWM} only depends on the current error, the proportional gain k_P and some internal factor γ, so

$$\alpha_{\mathrm{PWM}} = -\gamma k_P(q^* - q). \tag{4.66}$$

By setting $q^* = q - \frac{\alpha_{\mathrm{PWM}}}{\gamma k_P}$ we get exactly the desired duty cycle for this instant. As we have a control loop frequency of about 100Hz, we set the actual goal position to $q^* = q + \frac{\dot{q}}{100} - \frac{\alpha_{\mathrm{PWM}}}{\gamma k_P}$. In order to ensure that the desired torque stays approximately constant during the control loop, k_P has to be chosen small. This way $\frac{\alpha_{\mathrm{PWM}}}{\gamma k_P}$ is significantly larger than the change of the position during one cycle, causing the duty cycle of the PWM to stay (more or less) constant.

To summarize, in a first step the desired torque

$$\boldsymbol{\tau} = M_{\boldsymbol{q}}.\ddot{\boldsymbol{q}}^{des} + C_{\boldsymbol{q},\dot{\boldsymbol{q}}}.\dot{\boldsymbol{q}} + g_{\boldsymbol{q}} \tag{4.67}$$

is computed based on the mathematical model of the robot, as a function of the desired acceleration $\ddot{\boldsymbol{q}}^{des}$ as well as the current position and velocity. On the actual servomotor we then try to emulate control on the torque level by using a model for the servomotor and "abusing" the internal proportional controller of the servomotor.

Model Identification The above described approach only works if the mathematical model of the robot is approximately correct. The rigid body dynamics equations used above need to be improved by taking into account a (simple) model for dry and viscous friction. This is very important as the high ratio gearboxes of the servomotors induce considerable friction torques, that can, in certain cases, even dominate the dynamics. Therefore the dynamics equation becomes

$$\boldsymbol{\tau} = M_{\boldsymbol{q}}.\ddot{\boldsymbol{q}}^{des} + C_{\boldsymbol{q},\dot{\boldsymbol{q}}}.\dot{\boldsymbol{q}} + g_{\boldsymbol{q}} + \mu_{\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}}^{des}} \tag{4.68}$$

where $\mu_{\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}}^{des}}$ denotes the vector of friction torques. There exists no analytical formula to compute the friction torques, but one has to rely on simplified models, as for instance Stribeck [1902].

Secondly the parameters of the dynamical model (mass, center of gravity, inertia matrix) of each body within the chain are not well known. To reduce this uncertainty and calibrate the friction model we propose to use model identification, as described in, for instance, Khalil and Dombre [2004] or Wu et al. [2010].

Certain particularities, mostly caused by the servomotors, which make the identification process more difficult have to be taken into account.

[8] In order to have a precise enough model, some other details need to be taken into account, like the voltage drop at the H-bridge. As these details are not the core interest here, they are omitted.

- The high backlash in the gears causes vibrations in the whole chain every time the sign of the acceleration of one of the actuations changes
- The estimation of the velocity and in particular the acceleration is difficult due to the discrete measurement of the position in addition to the backlash
- The gearboxes induce high friction torques, which are additionally nonlinear and can only be modelled approximatively

Many different approaches for model identification and the generation of movements facilitating identification, such as Kostic et al. [2004], exist. Due to the above cited particularities an approach comparable to the one described in Schwarz and Behnke [2013] is chosen. The main idea is to perform identification on periodic movements and modulate the input using an iterative learning approach until the difference between the reference movement and the observed movement is negligible. Each improvement step of the iterative learning is basically a proportional and derivative control step of the input based on the current error and its derivative as, see Oh et al. [1988] and Liu [1994]. Once the learning sequence converged, see Figure 4.31, it offers the advantage that the velocity and acceleration do no longer have to be estimated from the measured data but can be directly taken as the derivatives of the reference trajectory. Moreover we chose the reference trajectory as a superposition of different sine functions with different frequencies similar to the approach presented in Swevers et al. [1996]. This overall movement is conceived such that all DoF change sign of the acceleration at the same moment, minimizing the impact of the backlash onto the measurements.

The identification itself is then a minimization problem and we seek simultaneously for the parameters of the robot and the servomotors. In order to prevent the minimization problem to be unbounded, we fix the torque constant of the DC-motors to the one obtained from the datasheet, as it is most likely the parameter showing the least uncertainty. The one deduced from the datasheet is moreover coherent with the ones reported in Rao [2016]. The results of the identification process is shown in Figure 4.32. The obtained ameliorations of the tracking may not seem to be very significant, but as the approach relies on an accurate tracking of the desired velocity, the increased responsiveness and reduced error increases the overall performance.

An additional remark concerns the power supply of the DC-Motors. All servomotors of one leg share the same powerline. Therefore the supply voltage of the servomotor is not equal to the battery voltage, as the resistance of all connectors and the cable is not negligible. The voltage drop at each resistance (connector/cable) add up and therefore the effective supply voltage of each servomotor depends on the current load of all motors in the chain and its position within the chain. Moreover, the capacity integrated in the electrical circuit of the servomotor is not large enough stabilize to voltage even for duty cycles of only 20%, causing high-frequency oscillations of the effective supply voltage. The evolution of the effective supply voltage depending on the desired torques can be taken into account within the feedforward computation, as this tension is measured by the servomotor every 0.1s. The high frequency oscillations of the supply tension on the other hand can be reduced by locally adding a capacity to the servomotor as they cannot be directly measured or be accounted for otherwise.

Obtaining Accelerations The robotic system is a second order system controlled on the acceleration or torque level. The feedforward control necessitates the computation of the currently desired acceleration. This acceleration is composed of two parts: the first one corresponds to the acceleration resulting from the demonstration space dynamics, the second part corresponds to an additional correction term.

To compute the first part, a naive approach is to define the acceleration as the finite difference

$$\ddot{\boldsymbol{q}}^{traj} = \frac{g(\boldsymbol{q} + \delta t \dot{\boldsymbol{q}}) - g(\boldsymbol{q})}{\delta t}. \tag{4.69}$$

This definition however is not adapted to the diffeomorphic transformation used, as the $\mathcal{C}^1$-kernels induce a high variance in the computed acceleration. We therefore computed the desired acceleration for a

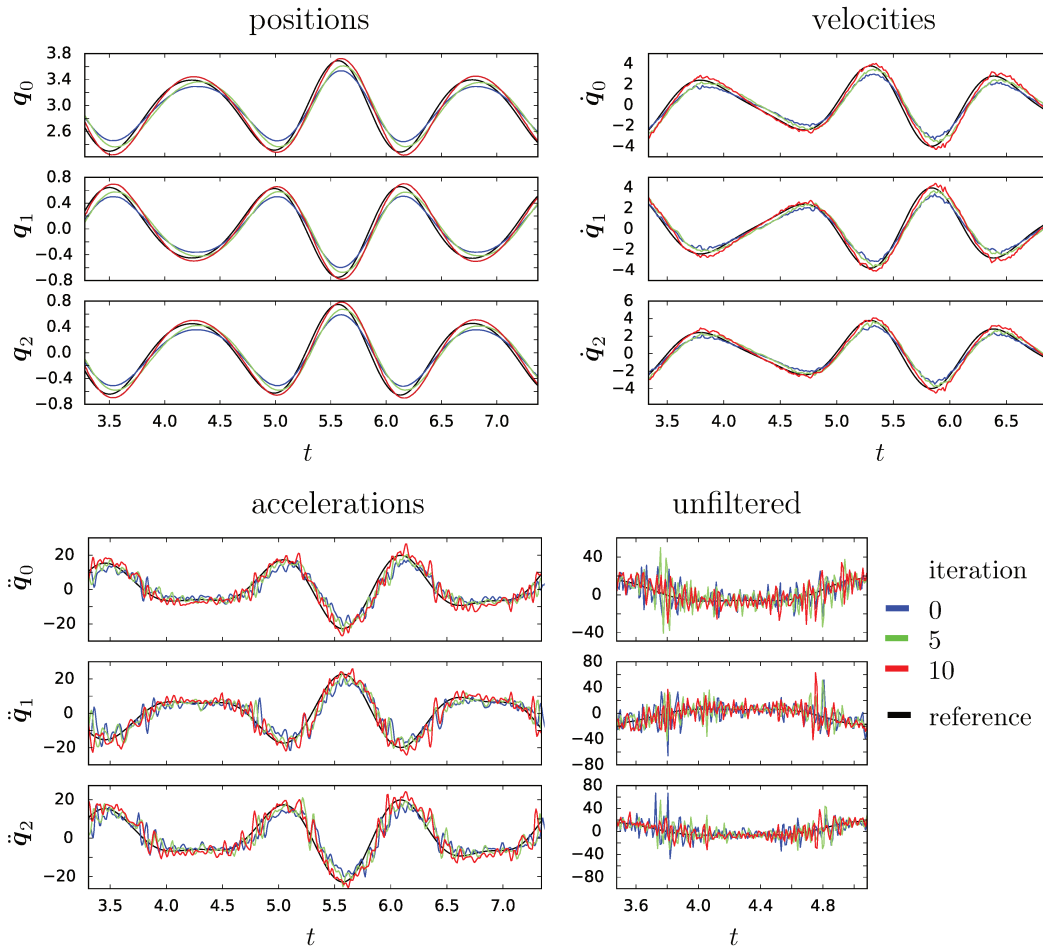


Figure 4.31 – In this picture the iterative learning approach used to obtain values for the joint velocity and acceleration without having to rely on the noisy measurements. The periodic movement is executed multiple times and at each iteration the input signal is updated to drive the trajectory closer to the reference.

list of time points $[t - N\delta t, t - (N-1)\delta t, \cdots t + (N-1)\delta t, t + N\delta t]$ and filtering it using a shifted finite impulse response (fir) filter. We say the filter is shifted, as it takes into account as many accelerations lying in the future as in the past, and it is therefore noncausal. As we obtain the accelerations relying on a forward and backward integration of demonstration space dynamics within the control loop, this is not a problem in this case. This shifting causes the filter to have zero-phase, which is very important in this case as it prevents a time-delay in the acceleration signal. In order to be able to obtain a suitable cut-off frequency as well as minimal amplitude distortion in the passband, at least 81 accelerations covering a time-span of 0.4 seconds had to be computed. This was achieved without slowing down the control loop frequency demonstrating the low computational cost of the proposed control law and the diffeomorphic transformation.

The second part accounts for the difference of the control level: usually we interpret the learned vector field as a first order system, that is we control directly the velocity which is not the case here. Therefore an additional acceleration part is defined as a proportional control between the current and the desired

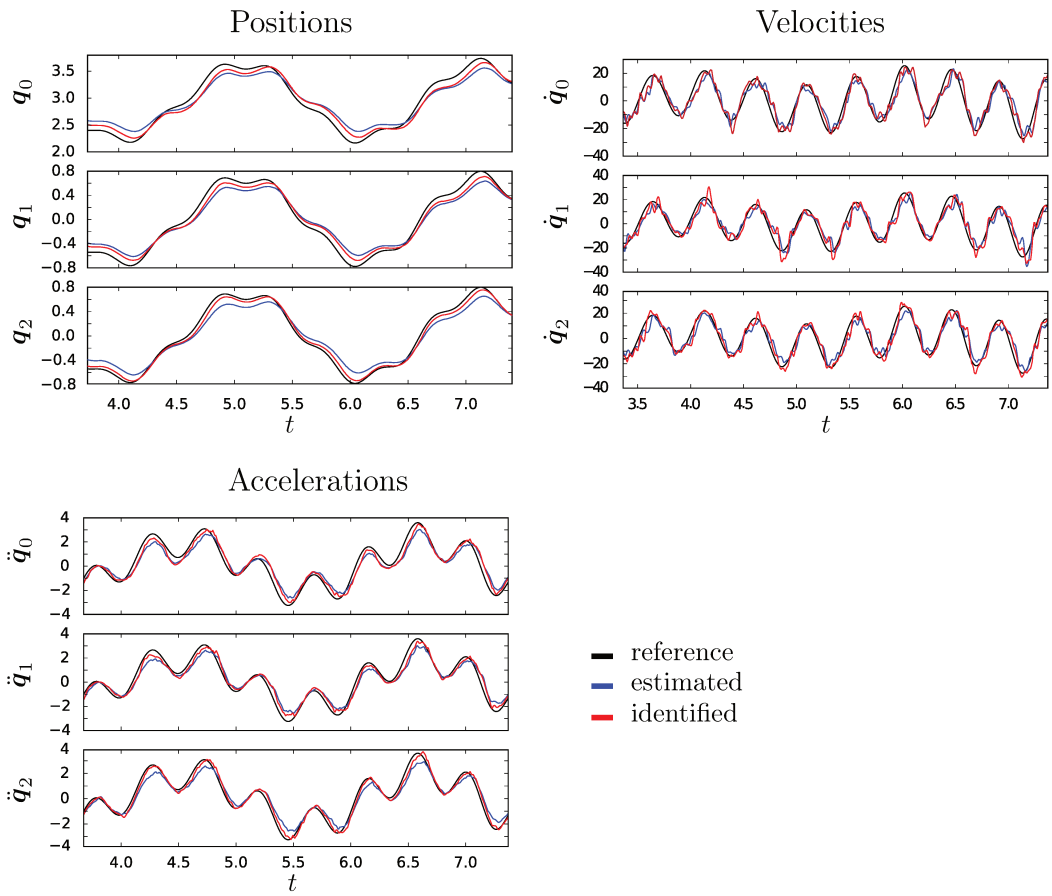


Figure 4.32 – The black line shows the desired reference trajectory. The blue line results from using feedforward control with the parameters of the robot and the DC-motors taken from the datasheets or computed from the technical specifications. The trajectory resulting from feed-forward control using the estimated parameters is shown in red. The difference in the position error is relatively small, but as one can see the velocity profile is better tracked, which is very desirable for our use case. Due to the measurement noise induced by the discretization the difference in the accelerations is difficult to evaluate. The raw data for the acceleration (not shown) is by far too noisy, the smoothened data (third column) might suffer from a phase delay and overly smooth out the actual peaks.

velocity

$$\ddot{\boldsymbol{q}}^{ctrl} = -k_P(\dot{\boldsymbol{q}}^{traj} - \dot{\boldsymbol{q}}) \tag{4.70}$$

with k_P chosen empirically. In total, the current desired acceleration is defined as $\ddot{\boldsymbol{q}}^{des} = \ddot{\boldsymbol{q}}^{traj} + \ddot{\boldsymbol{q}}^{ctrl}$.

Results In Figure 4.33 the obtained results for following a 3D trajectory for the hip, knee and ankle joint is shown. The demonstration trajectories (dashed black lines) are generated using splines whose nodes have a constant part shared by all demonstration and an additional randomized component. Each plot presents the position or velocity of the trajectory measured on the robot (red lines) and the trajectory resulting from the numerical integration of the demonstration space dynamics starting at the same initial point (blue lines). Results are shown for three different initial positions, with the hip-joint having an offset of $0rad$, $-0.1rad$ and $0.1rad$, using the same learned dynamics. As one can see, these offsets lie

within the region for which the demonstrations are well generalized, as the reproductions correspond to the demonstrated movement.

4.6 Conclusion

In this chapter two different approaches for learning globally asymptotically stable vector fields from demonstration are presented. They are based on a novel method to construct diffeomorphic via the composition of simple sub-transformation called locally weighted (multi-)translation. In contrast to state-of-the-art techniques which construct diffeomorphisms as on flows based on the transport equation, the presented method drastically reduces computation time for computing the forward and inverse transformation as no integration is needed for the evaluation. This speed-up allows the proposed approach to be integrated into the control-loop of a robotic system while achieving a high control frequency, necessary in many applications. We moreover propose two heuristic algorithms that can be used to construct such diffeomorphic transformation based on point- (One-Step learning) or approximative curve matching (Two-Step learning).

The advantage of the proposed approach however lies not only in its reduced calculation time, but also in its ability to generalize shown movements into some neighbourhood of the demonstration and to be able to learn a movement successfully from as view a one demonstration. This is typically a problem for learning strategies based on statistical inference, as the data is often too sparse in this case. As the proposed approach is exploiting the geometry of the given trajectories, having a "sparse" dataset is not a hindrance to successful learning. In fact in the case of One-Step, learning a sparse dataset with only one demonstration is even the "optimal" case, as this approach has a limited capability to cope with the variance and probable incompatibility (in a diffeomorphic sense) often arising when considering several trajectories demonstrating the same movement. This is not true for the second method proposed, called Two-Step learning, which alleviates this and some other drawbacks by interweaving diffeomorphic (curve) matching with statistical learning. This approach allows to extract more information from the diversity within the demonstration set while still being able to successfully learn from a sparse dataset.

It was shown that the approaches can be used to learn complex (loop-free) 2D and 3D reaching motions, partly taken from the publicly available LASA-Dataset. By performing manual adjustments, we are also able to learn limit cycles and can therefore extend the approach to periodic motions. The proposed algorithms scale well with dimension as its parameters are dimension independent. The only except to this is the clustering necessary in Algorithm 5. In this work the k-Means clustering is used, whose average runtime is dimension independent, but its worst-case runtime scales exponentially with dimension. This should give the approach an advantage when dealing with higher dimensional data, as the number of variables for statistical approaches usually scales at least quadratically with the dimension.

For these reasons we believe it can be applied with ease to efficiently learn a large variety of globally asymptotically stable autonomous systems, with applications in dynamic movement primitives construction or more generally in control design.

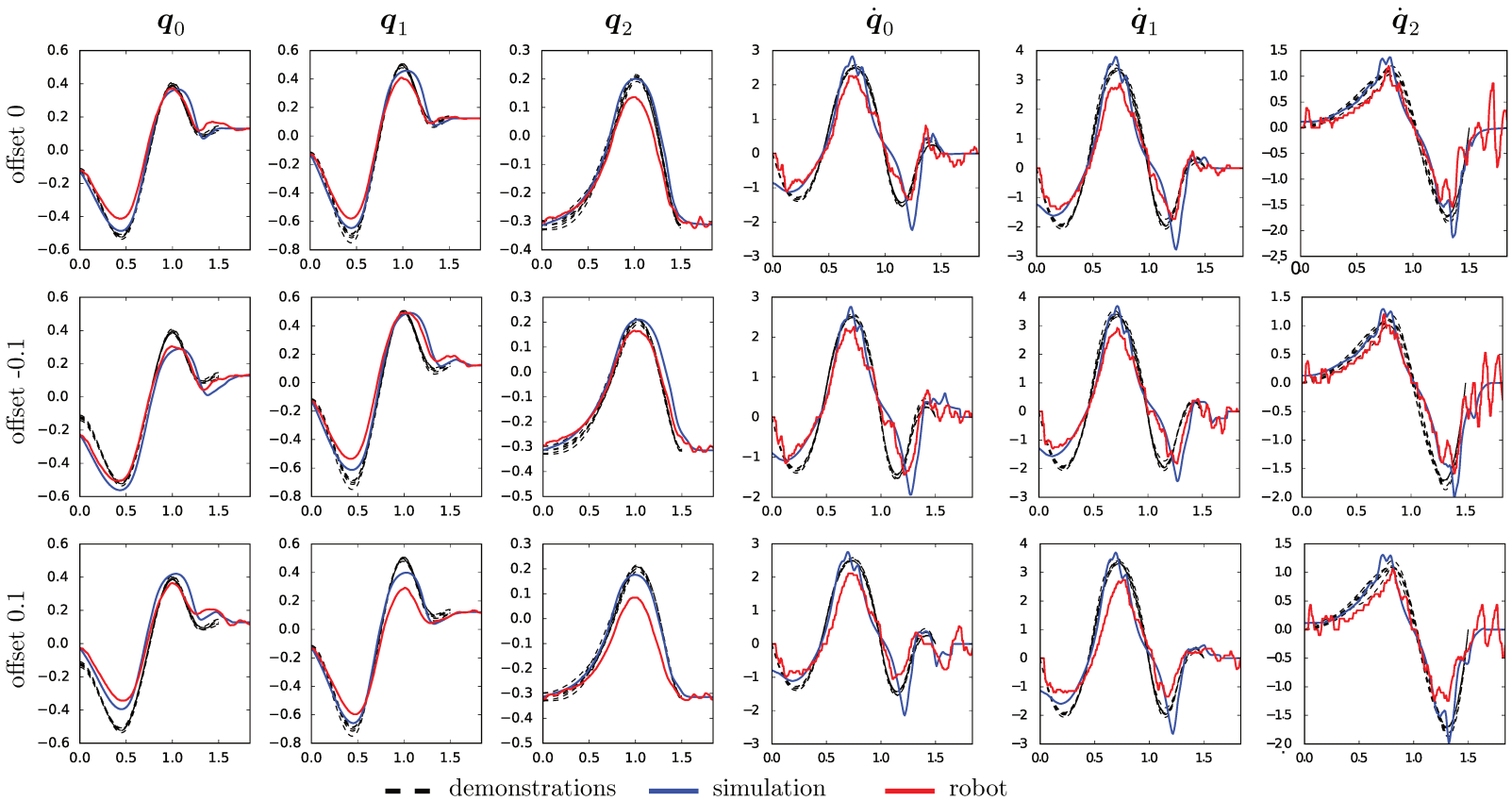


Figure 4.33 – The black lines depict the given demonstrations. The blue line corresponds to the ideal demonstration space trajectory (so the trajectory resulting from integrating $\dot{\boldsymbol{y}} = g(\boldsymbol{y})$) with the boundary condition that $\boldsymbol{y}(0)$ corresponds to the actual initial robot position. The red line corresponds to the actual trajectory performed by the robot. As one cann see the position and velocity profiles are well matched, except for the velocity a the end of the trajectory. Here, due to the low velocity, the high backlash induces a frequent change of the acceleration sign and the entire kinematic chain starts to vibrate.

Chapter 5

Conclusion and Future Work

Control strategies for modern torque controlled robotic systems have usually a layered or hierarchical structure, decomposing the problem of achieving some predefined task into sub-problems. These sub-problems notably have very different time-scales. In this thesis different contributions to the overarching objective of automated control design providing formal guarantees with a focus on stability and safety are proposed. To conclude this work, the different approaches are discussed with respect to this hierarchical control structure and the advantages or disadvantages over existing methods are and how these approaches contribute to the safety of the overall system is recapitulated. At the same time, the current limitations of the proposed approaches are pin-pointed and ways how to overcome these limitations are discussed.

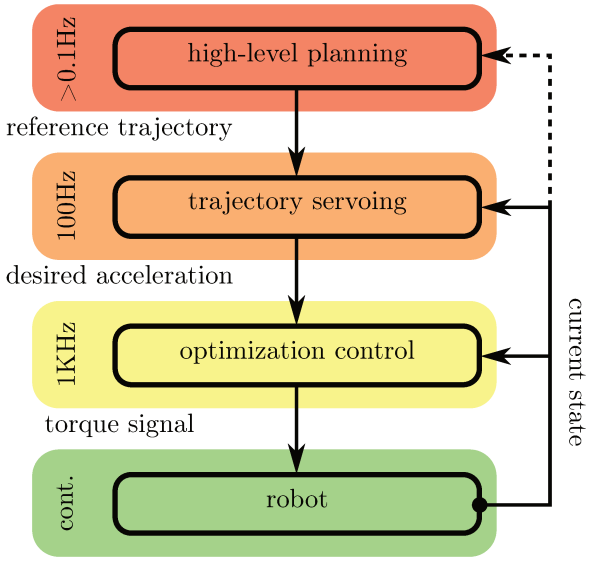


Figure 5.1 – Layered control structure typically found in torque controlled robotic systems, adapted from Lober [2017].

The layers typically found in the control architecture of a robotic system can be summed up into the three layers high-level planning, trajectory servoing and optimization based control as shown in Figure 5.1. The first layer, the high-level planning is typically the computationally most complex and is therefore executed at a slow rate (replanning) or corresponds to pure "offline" computation, that is the plan is first conceived, then executed without further changes. This layer typically outputs a reference trajectory,

which is then handed down to the trajectory servoing layer. This layer can already be executed at a higher frequency and usually takes into account the current state of the robotic system and is therefore always part of the closed loop controller. This layer is necessary, as the given reference trajectories cannot simply be executed in an open-loop fashion, as this is almost guaranteed to fail due to modelling imprecisions or external perturbations. Therefore the trajectory servoing layer computes desired accelerations or velocities, or sequences thereof, that (are supposed to) bring the state of the system back to the reference trajectory. The last layer, which is executed at the highest rate, is typically a quadratic program. It computes the desired control torque that allows to achieve the desired accelerations or velocities (if possible) while respecting the input constraints. Furthermore this layer is typically regularized by taking into account the overall control effort.

5.1 High-level Planning

The high-level planning addresses the problem of converting a specification into a reference trajectory for the robotic system verifying this specification. There exists a large variety of different approaches, which are able to take into account more or less expressive specifications and system dynamics.

The "simplest" specifications are pure motion planning problems, so generating a (theoretically) collision-free trajectory from A to B and approaches to solve this problems range from early works like A* searching over predefined nodes (Hart et al. [1968]) in which no underlying model of the robot is used, over approaches considering the kinematic model of the robot like Dubins [1957] to ones in which the full dynamical robot model is accounted for (as in KPIECE Şucan and Kavraki [2012] or Rapidly exploring Random Trees (RRT) see LaValle [1998]). In Tedrake et al. [2010b] the motion planning problem is solved by constructing trees of funnels. In difference to the works cited before, this method does not only provide a reference trajectory, but also the guarantee that all states within the funnel can be driven to the target configuration, providing higher robustness. The approach presented in Le Ny and Pappas [2012] uses funnels within a RRT-framework, achieving robust motion planning.

More complex specifications, or more precisely (fragments of) Linear Temporal Logic can be handled by approaches like the ones proposed in Kress-Gazit et al. [2009] for kinematic models of the form $\dot{\boldsymbol{x}} = \boldsymbol{u}$ or Kloetzer and Belta [2008] for linear systems. The difference between the two approaches is the reactivity. In the latter a plan to verify a given formula is computed, whereas the approach in Kress-Gazit et al. [2009] has an additional specification defining the environment, and by computing a winning strategy for the two player game (controller versus environment) a reactivate control strategy adapting to the environment is generated.

Timed-automata Abstraction The approach proposed in chapter 2 can also be used as such a high-level planner. By abstracting the dynamical system to a timed automaton where the discrete states of the automaton correspond to positively invariant subsets of the state-space of the dynamical system, e.g. the joint-space of a robotic manipulator, we obtain several advantages over existing methods. Notably due to the properties of the abstraction, we obtain similar robustness and safety properties as the approaches in Tedrake et al. [2010b]; Le Ny and Pappas [2012]. Specifically, the proposed abstraction of the dynamical system to a funnel timed transition system, which can then be reduced to a timed automaton, is proven to be sound with respect to reachability. That is, if the set of undesired states (such as collisions between the robot and its environment) is not reachable in to automaton, it is guaranteed that the associated events will not occur on the real system (the robot will not collide with the environment, no matter the circumstances). Moreover, our approach keeps a continuous notion of time and is therefore able to synthesize control strategies for tasks necessitating quantitative and exact timing constraints, like the Pick-and-Place scenario presented in section 2.6.2. It therefore allows more expressive specifications than, for instance, Kress-Gazit et al. [2009] our other approaches relying on (fragments of) LTL to specify the task and the environment. It is important to note that all approaches proposed in this work rely on a basic assume-guarantee principle: its is guaranteed that the given specification is verified by the synthesized

control strategy if a) the underlying (mathematical) model is correct and b) the actual environment evolves in accordance with the specification describing it. This in turns indicates that every execution on the real system that fails to verify the specification supplies information about the limits or errors in the modelling of the system. This way the proposed approach can contribute to a better understanding of the overall system and its environment by iteratively refining the modelling, synthesizing new control strategies and executing them on the real system.

The drawbacks of the proposed approach are essentially that the abstraction is not complete and its overall computational complexity. The size of the constructed automaton grows rapidly with the dimension of the system or when seeking to represent all possible trajectories of the controlled dynamical system (seeking to approach completeness). This is making it impossible to treat higher dimensional systems using the proposed approach in its current form. Even though it is in theory possible to compute winning strategies considering an adversarial environment as showcased in section 2.6.1, more complex environments like the ones considered in Kress-Gazit et al. [2007, 2009] are currently intractable with the proposed approach.

A secondary, not negligible, source of complexity stems directly from abstracting the controlled dynamical system to (time-dependent) invariants, that is to construct the funnel system. This abstraction is generic in the sense that it is not restricted to a specific class of dynamical systems, but relies on properties directly provided by the funnel system. This stands in contrast to most of the approaches in the literature, as these often rely on specific properties guaranteed by a suitable choice of the class of admissible dynamical systems. The property of (conjectured) positive invariance of the funnels necessary for our approach is closely linked to Lyapunovs stability criterion. This property is established before the actual synthesis (checking reachability on the associated automaton) in a "preprocessing" step when constructing the funnels and generating the funnel timed transition system. This can be efficiently done for linear systems (see section 2.5.1), in which case the time necessary to create the funnel system is negligible compared to the verification of the TA. But this step is already significantly more complex when considering polynomial systems (see chapter 3) and no generic approach exists for general nonlinear systems. To deal with such systems, we proposed the use of bounding funnels with conjectured properties. It provides a frame work to deal with general nonlinear dynamics, however the problem of obtaining not only a (hopefully) correct but also useful conjecture is a complicated problem in it self.

Nonetheless the provided examples (see section 2.6) showcase the utility of precise timing constraints within a Pick-and-Place scenario, often encountered in industrial applications. Here the approach has proven its capability of synthesizing non-trivial strategies to control second-order systems under logical and timing constraints. In the example synchronizing sine-waves we have shown the possibility of computing winning strategies relying on the proposed abstraction method and an auxiliary automaton describing the possible evolution of the (adversarial) environment. Here the specifications can be more expressive than the ones considered in Kress-Gazit et al. [2009], however due to the complexity only "toy-examples", like the one proposed, can be solved in practice.

In order to go one step further and adapt the proposed method to larger problems with nonlinear system dynamics, several promising directions exist.

The first one is closely intertwined with the method proposed in chapter 3 and is concerned with finding computationally less complex certificates of positive invariance for polynomial systems. Any advance on this subject could significantly reduce the time needed to construct the funnel system and enlarge the class of dynamical systems actually treatable.

The generation of certificates of positive invariance for polynomial systems is also interesting in combination with conjectured funnels. As it is shown in chapter 3, the in general nonlinear dynamics of robotic systems and in particular robotic manipulators are well approximated by its Taylor expansion around a reference trajectory. This indicates that the certificate for the Taylor expansion provides a good conjecture for the nonlinear system and allows to have a high confidence in the conjecture with only very few additional numerical simulations. This is an important step, as the method of covering the funnel with regularly spaced initial points for numerical evaluation as done in the examples (Dubins' car) scales badly to higher dimensions.

Another key to achieve better scalability is to identify ways to generate funnels adapted to the given task specification. That is instead of constructing funnels in a brute force manner and purely rely on the verification tool to find a suitable sequence of funnels, it is interesting to construct a smaller automaton representing a "suitable" funnel system with respect to the task. This way, the abstraction could approach completeness without blowing up the size of the automaton. One could even seek to directly interweave the construction of the funnel system and the verification process, by generating suitable funnels on the fly. The size of the automaton would not be defined in advance, but would grow during the verification.

Globally Asymptotically Stable Vector Fields Restricting the expressiveness of the task specification, the methods proposed in chapter 4 for learning nonlinear globally asymptotically stable vector fields can also be seen as a way to achieve high-level planning. As these vector fields guarantee that all states converge exponentially fast to the unique global attractor, and therefore attain an arbitrarily small neighbourhood of the target point in finite time, they bear resemblance to the "eventually" operator in LTL. By constructing several such vector fields with different attractors, it can be used for path-planning under simple LTL constraints.

An advantage of this method is that, as the learned vector field reproduces the given demonstrations and generalizes the movement to some neighbourhood, the vector field is likely to be suitable for the dynamical system if the given demonstrations are. Therefore this approach not only allows to learn complex movements from demonstration, but can also be seen as a method to generate suitable and converging reference trajectories for the dynamical system.

In contrast to the timed-automata abstraction, the computational complexity of both approaches (One-Step and Two-Step learning) scales very well to higher dimensions, as the number of parameters optimized within the iterative algorithms is independent of the dimension. Moreover, the approaches can successfully learn nonlinear vector fields from as few as a single demonstration, which is not necessarily the case for learning methods relying on statistical models.

To go further, the approaches could be extended to create vector fields used for navigation as the ones created in Conner et al. [2003], with the difference that they (indirectly) take into account the system dynamics, instead of supposing unconstrained kinematic systems. This could be achieved by adding repulsive fields around the obstacles, similar to the ideas proposed in Khansari-Zadeh and Billard [2012]. To obtain the (almost everywhere) asymptotic stability, it would be interesting to construct the diffeomorphism not only based on the demonstrations, but at the same time seek to transform the, possibly non-convex, obstacles into convex shapes (similar to the idea in Rimon and Koditschek [1991]), and add repulsive fields around the convex images, as this avoids creating spurious attractors.

Many of the draw-backs of the One-Step learning are already tackled by the Two-Step learning, however there is still room for improvement. The current weak point of the Two-Step learning is the construction of the control-space dynamics based on the given demonstrations. The fact of relying on the locally contracting directions introduced in section 4.5.5 has the advantage that it provides a contracting vector field that converges with respect to the euclidean norm, but it introduces a number of user-defined parameters, which have to be hand-tuned to some extent. Also, the number of components defining the control-space dynamics grows linearly with the number of demonstrations (the number of trajectories, not data points), which is, due to their small computational cost, acceptable, but not desirable. It is an interesting research direction to replace this with another machine learning technique providing similar benefits (locally contracting while globally stable, taking into account that we seek to learn a direction, so a unit vector), while being independent of the number of demonstrations.

5.2 Trajectory Servoing and Optimization based Control

As already stated above, the trajectory servoing layer computes desired accelerations or velocities for the robotic system as a function of its current state and reference trajectory. This is typically done relying on one of the following methods. The most traditional approach, which is still very present especially

in manufacturing robots, is the proportional-integral-derivative (PID) controller. This method typically provides acceptable error rejection at the cost of the often necessary hand-tuning of the parameters. Moreover the output is not guaranteed to respect input or state constraints of the robot. Another commonly used approach is to use (nonlinear) model predictive control (Morari and Lee [1999]), which computes a sequence of cost optimal accelerations or velocities for a finite horizon. Here the computational cost depends on the model used (linear/nonlinear) and the size of the time window considered.

The last layer computes desired torques using the inverse dynamics model of the robot taking into account the current position and velocity as well as the desired accelerations. This optimization has to be computationally cheap as it is typically executed in the real-time loop of the robot, which is why many approaches rely on quadratic programming (see Righetti and Schaal [2012]; Liu et al. [2016]).

Using Funnels for Trajectory Servoing The funnels, defined as timed-dependent, stabilizable regions, proposed in chapter 3 can be seen as a way to perform trajectory servoing directly within the optimization layer. In chapter 3 it was shown how to construct an as large as possible region around a given reference trajectory for which it is proven that there exists an control input driving all states to the (time-dependent) reference point. These regions are defined by quadratic Lyapunov functions and it was shown that the decisive constraint guaranteeing convergence can be formulated as a simple linear constraint on the torques (control inputs), see (3.57c). This constraint therefore not only ensures (exponential) convergence, but also provides a direct and easily interpretable link between the acceleration/torque, the current difference of the current state and the reference state and the instantaneous convergence.

In the proposed examples, treating both polynomial approximations of nonlinear systems and truly polynomial systems, it was shown that reasoning directly about stabilizability for a (suitable) given Lyapunov function candidate instead of searching for a couple Lyapunov function and (polynomial) control law in a iterative fashion can be beneficial. Even thought we rely on simple methods to compute the Lyapunov function candidate, which only takes into account the linearisation of the system, our approach is able to find significantly larger (in terms of the enclosed volume) regions of stabilizability then the regions of attractions obtained by Majumdar et al. [2013b], a state-of-art algorithm.

To obtain smooth control trajectories, we embed the linear constraint ensuring convergence into a quadratic programming based control law, trading off instantaneous convergence (with respect to the funnel) and control effort. In a more general setting, when working with funnels the trajectory servoing layer can be dropped, if the optimization based layer allows to add the convergence constraint. As this constraint is linear, and therefore the most basic constraint form, this should always be possible. This allows the control law to minimize its objective as long as no safety issue can arise, that is as long as convergence is guaranteed.

Reducing the Computational Cost of Funnels To reduce the computational cost of this method even further, several directions can be of interest.

To reduce the number of linear constraints necessary to obtain tight bounds, we seek to combine LMI constraints derived from the Theory of Moments (see section 3.8.4) with the linear constraints derived from the modified Reformulation and Linearization technique. This could be interesting, as the gap between the true optimum of the polynomial expression and its relaxation by the Theory of Moments decreases when increasing the maximal degree occurring in the relaxation. By fixing a low maximal degree of the relaxation the gap resulting from the application of the Theory of Moments can possibly be reduced by adding the linear constraints, while reducing the overall complexity of the optimization problem.

When computing a funnel around a reference trajectory, the subset of the state space having the worst convergence usually stays the same when considering two consecutive time step, as the reference trajectory and the system and control dynamics are continuous. This can be exploited by using heuristics within the line-search performed to find an as large as possible inner approximation of the true region

of stabilizability. This way the number of "atomic" convergence proofs is reduced, reducing overall computation time.

Finally it would be interesting to search for ways to modify the Lyapunov function candidate found based on the linearisation of the system around the reference point, especially when constructing a funnel around a dynamic reference trajectory. This could be done by considering the point having currently the worst convergence (found as a by-product of the optimization, as it is the minimizer of the expression) and its velocity. By adapting the funnel shape in such a way that the gradient of the Lyapunov function at this point aligns better with the velocity at this point, the sub-levelset could be further enlarged.

Bibliography

A. A. Ahmadi. Non-monotonic lyapunov functions for stability of nonlinear and switched systems: theory and computation. Master's thesis, Massachusetts Institute of Technology, 2008.

A. A. Ahmadi. *Algebraic relaxations and hardness results in polynomial optimization and Lyapunov analysis*. PhD thesis, Massachusetts Institute of Technology, 2012.

A. A. Ahmadi and A. Majumdar. Dsos and sdsos optimization: Lp and socp-based alternatives to sum of squares optimization. In *Information Sciences and Systems (CISS), 2014 48th Annual Conference on*, pages 1–5. IEEE, 2014.

M. Althoff, O. Stursberg, and M. Buss. Reachability analysis of nonlinear systems with uncertain parameters using conservative linearization. In *Proc. of the 47th IEEE Conference on Decision and Control*, 2008.

M. Althoff, O. Stursberg, and M. Buss. Computing reachable sets of hybrid systems using a combination of zonotopes and polytopes. *Nonlinear analysis: hybrid systems*, 4(2):233–249, 2010.

R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994a.

R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994b.

M. Andersen, J. Dahl, and L. Vandenberghe. Cvxopt: A python package for convex optimization. *abel. ee. ucla. edu/cvxopt*, 2013.

M. S. Andersen, J. Dahl, and L. Vandenberghe. Cvxopt: A python package for convex optimization, version 1.1. 6 (2013).

MOSEK ApS. *The MOSEK optimization toolbox for MATLAB manual. Version 8.1.*, 2017. URL `http://docs.mosek.com/8.1/toolbox/index.html`.

B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

E. Asarin, O. Maler, and A. Pnueli. Reachability analysis of dynamical systems having piecewise-constant derivatives. *Theoretical Computer Science*, 138(1):35–65, 1995.

E. Asarin, O. Maler, A. Pnueli, and J. Sifakis. Controller synthesis for timed automata. In *Proc. IFAC Symposium on System Structure and Control*, pages 469–474. Elsevier, 1998.

E. Asarin, T. Dang, and A. Girard. Reachability analysis of nonlinear systems using conservative approximation. In *International Workshop on Hybrid Systems: Computation and Control*, pages 20–35. Springer, 2003.

C. G. Atkeson and S. Schaal. Robot learning from demonstration. In *ICML*, volume 97, pages 12–20, 1997.

J. P. Aubin. Viability tubes. In *Modelling and Adaptive Control*, pages 27–47. Springer, 1988.

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.

R. Barbuti and L. Tesei. Timed automata with urgent transitions. *Acta Informatica*, 40(5):317–347, 2004.

M. Bauer, M. Bruveris, S. Marsland, and P. W. Michor. Constructing reparameterization invariant metrics on spaces of plane curves. *Differential Geometry and its Applications*, 34:139–165, 2014.

G. Behrmann, A. David, K. G. Larsen, J. Håkansson, P. Pettersson, W. Yi, and M. Hendriks. Uppaal 4.0. In *Proc. 3rd International Conference on Quantitative Evaluation of Systems (QEST'06)*, pages 125–126. IEEE, 2006.

G. Behrmann, A. Cougnard, A. David, E. Fleury, K. G. Larsen, and D. Lime. UPPAAL-Tiga: Time for playing games! In *Proc. 19th International Conference on Computer Aided Verification (CAV'07)*, volume 4590 of *LNCS*, pages 121–125. Springer, 2007.

C. Berge. Espaces topologiques, fonctions multivoques (dunod, paris, 1959). *Google Scholar*, 1970.

A. Billard, S. Calinon, R. Dillmann, and S. Schaal. Robot programming by demonstration. In *Springer handbook of robotics*, pages 1371–1394. Springer, 2008.

C. Blocher, M. Saveriano, and D. Lee. Learning stable dynamical systems using contraction theory. In *Proceedings of the International Conference on Ubiquitous Robots and Ambient Intelligence (URAI 2017)*, 2017.

V. D. Blondel, E. D. Sontag, M. Vidyasagar, and J. C. Willems. *Open problems in mathematical systems and control theory*. Springer Science & Business Media, 2012.

P. Bouyer, C. Dufourd, E. Fleury, and A. Petit. Updatable timed automata. *Theoretical Computer Science*, 321(2-3):291–345, 2004.

P. Bouyer, F. Laroussinie, and P.-A. Reynier. Diagonal constraints in timed automata: Forward analysis of timed systems. In *International Conference on Formal Modeling and Analysis of Timed Systems*, pages 112–126. Springer, 2005.

P. Bouyer, N. Markey, N. Perrin, and P. Schlehuber-Caissier. Timed-automata abstraction of switched dynamical systems using control funnels. In *International Conference on Formal Modeling and Analysis of Timed Systems*, pages 60–75. Springer, 2015.

P. Bouyer, N. Markey, N. Perrin, and P. Schlehuber-Caissier. Timed-automata abstraction of switched dynamical systems using control invariants. *Real-Time Systems*, 53(3):327–353, 2017.

S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. Linear matrix inequalities in system and control theory, 1994.

M. Bozga, C. Daws, O. Maler, A. Olivero, S. Tripakis, and S. Yovine. Kronos: A model-checking tool for real-time systems. In *International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems*, pages 298–302. Springer, 1998.

J. R. Büchi and L. H. Landweber. Solving sequential conditions by finite-state strategies. *Transactions of the American Mathematical Society*, 138:295–311, 1969.

A. Butz. Higher order derivatives of liapunov functions. *IEEE Transactions on automatic control*, 14(1): 111–112, 1969.

S. Calinon, F. D'halluin, E. Sauser, D. Caldwell, and A. Billard. A probabilistic approach based on dynamical systems to learn and reproduce gestures by imitation. *IEEE Robotics and Automation Magazine*, 17(2):44–54, 2010.

X. Chen. *Reachability analysis of non-linear hybrid systems using taylor models*. PhD thesis, PhD thesis, RWTH Aachen University, 2015.

X. Chen, E. Ábrahám, and S. Sankaranarayanan. Flow*: An analyzer for non-linear hybrid systems. In *International Conference on Computer Aided Verification*, pages 258–263. Springer, 2013.

G. Chesi. Estimating the domain of attraction for non-polynomial systems via lmi optimizations. *Automatica*, 45(6):1536–1541, 2009.

G. Chesi. Lmi techniques for optimization over polynomials in control: a survey. *IEEE Transactions on Automatic Control*, 55(11):2500–2510, 2010.

N. G. Chetaev. *The stability of motion*. Pergamon Press, 1961.

A. Church. Logic, arithmetic and automata. In *Proceedings of the international congress of mathematicians*, volume 1962, pages 23–35, 1962.

T. Cimen. State-dependent riccati equation (sdre) control: A survey. *IFAC Proceedings Volumes*, 41(2): 3761–3775, 2008.

D. C. Conner, A. A. Rizzi, and H. Choset. Composition of local potential functions for global robot control and navigation. In *Intelligent Robots and Systems (IROS), 2003 IEEE/RSJ International Conference on*, volume 4, pages 3546–3551. IEEE, 2003.

H. De Jong, M. Page, C. Hernandez, and J. Geiselmann. Qualitative simulation of genetic regulatory networks: Method and application. In *IJCAI*, pages 67–73, 2001.

J. DeCastro and H. Kress-Gazit. Synthesis of nonlinear continuous controllers for verifiably-correct high-level, reactive behaviors. *IJRR*, 34(3):378–394, 2014.

G. DeJong and R. Mooney. Explanation-based learning: An alternative view. *Machine learning*, 1(2): 145–176, 1986.

L. E. Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of mathematics*, 79(3):497–516, 1957.

P. : Duggirala, S. Mitra, and M. Viswanathan. Verification of annotated models from executions. In *Embedded Software (EMSOFT), 2013 International Conference on*, pages 1–10. IEEE, 2013.

P. Dupuis, U. Grenander, and M. I. Miller. Variational problems on flows of diffeomorphisms for image matching. *Quarterly of applied mathematics*, pages 587–600, 1998.

C. Edwards and S. Spurgeon. *Sliding mode control: theory and applications*. Crc Press, 1998.

SV. Emel'yanov and VI. Utkin. Stability of motion of a class of variable structure control systems. *Izv. AN SSSR, Tech. Cyber*, (2):140–142, 1964.

E. B. Erdem. *Analysis and real-time implementation of state-dependent Riccati equation controlled systems*. PhD thesis, Citeseer, 2001.

R. Fabre, H. Gimbert, L. Gondry, L. Hofer, O. Ly, S. N'Guyen, G. Passault, and Q. Rouxel. Rhoban football club–team description paper. *Humanoid KidSize League, Robocup 2015 Hefei*, 2015.

S.-C. Fang and S.-Y. Wu. Solving min-max problems and linear semi-infinite programs. *Computers & Mathematics with Applications*, 32(6):87–93, 1996.

E. Frazzoli, M. A. Dahleh, and E. Feron. Maneuver-based motion planning for nonlinear systems with symmetries. *IEEE Trans. Robotics*, 21(6):1077–1091, 2005.

G. Frehse, C. Le Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler. Spaceex: Scalable verification of hybrid systems. In *International Conference on Computer Aided Verification*, pages 379–395. Springer, 2011.

N. Fulton, S. Mitsch, J.-D. Quesel, M. Völp, and A. Platzer. Keymaera x: An axiomatic tactical theorem prover for hybrid systems. In *International Conference on Automated Deduction*, pages 527–538. Springer, 2015.

K. Gatermann and P. A. Parrilo. Symmetry groups, semidefinite programs, and sums of squares. *Journal of Pure and Applied Algebra*, 192(1-3):95–128, 2004.

J. Glaunes. The matchine software (c). 2005.

J. Glaunes, A. Trouvé, and L. Younes. Diffeomorphic matching of distributions: A new approach for unlabelled point-sets and sub-manifolds matching. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–II. Ieee, 2004.

J. Glaunès, A. Qiu, M. I. Miller, and L. Younes. Large deformation diffeomorphic metric curve mapping. *International journal of computer vision*, 80(3):317, 2008.

R. Grimshaw. *Nonlinear ordinary differential equations*. Routledge, 2017.

D. D. Grossman. *Programmimg a Computer Controlled Manipulator by Guiding Through the Motions*. IBM, 1977.

Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

D. Henrion and A. Garulli. *Positive polynomials in control*, volume 312. Springer Science & Business Media, 2005.

D. Henrion and M. Korda. Convex computation of the region of attraction of polynomial control systems. *IEEE Transactions on Automatic Control*, 59(2):297–312, 2014.

T. A. Henzinger, P.-H. Ho, and H. Wong-Toi. Algorithmic analysis of nonlinear hybrid systems. *IEEE transactions on automatic control*, 43(4):540–554, 1998.

A. J. Ijspeert, J. Nakanishi, and S. Schaal. Learning attractor landscapes for learning motor primitives. In *Advances in neural information processing systems*, pages 1547–1554, 2003.

A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal. Dynamical movement primitives: learning attractor models for motor behaviors. *Neural computation*, 25(2):328–373, 2013.

Z. Jarvis-Wloszek, R. Feeley, W. Tan, K. Sun, and A. Packard. Some controls applications of sum of squares programming. In *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*, volume 5, pages 4676–4681. IEEE, 2003.

T. A. Johansen, I. Petersen, and O. Slupphaug. On explicit suboptimal lqr with state and input constraints. In *Decision and Control, 2000. Proceedings of the 39th IEEE Conference on*, volume 1, pages 662–667. IEEE, 2000.

S. C. Joshi and M. I. Miller. Landmark matching via large deformation diffeomorphisms. *IEEE transactions on image processing*, 9(8):1357–1370, 2000.

A. A. Julius and G. J. Pappas. Trajectory based verification using local finite-time invariance. In *Hybrid Systems: Computation and Control*, volume 5469 of *LNCS*, pages 223–236. Springer, 2009.

I. Kezurer, S. Z. Kovalsky, R. Basri, and Y. Lipman. Tight relaxation of quadratic matching. In *Computer Graphics Forum*, volume 34, pages 115–128, 2015.

H. K. Khalil. Noninear systems. *Prentice-Hall, New Jersey*, 2(5):5–1, 1996.

W. Khalil and E. Dombre. *Modeling, identification and control of robots*. Butterworth-Heinemann, 2004.

S. M. Khansari-Zadeh and A. Billard. Bm: An iterative algorithm to learn stable non-linear dynamical systems with gaussian mixture models. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 2381–2388. IEEE, 2010.

S. M. Khansari-Zadeh and A. Billard. Learning stable nonlinear dynamical systems with gaussian mixture models. *IEEE Transactions on Robotics*, 27(5):943–957, 2011.

S. M. Khansari-Zadeh and A. Billard. A dynamical system approach to realtime obstacle avoidance. *Autonomous Robots*, 32(4):433–454, 2012.

S. M. Khansari-Zadeh and A. Billard. Learning control lyapunov function to ensure stability of dynamical system-based robot reaching motions. *Robotics and Autonomous Systems*, 62(6):752–765, 2014.

M. Kloetzer and C. Belta. A fully automated framework for control of linear systems from temporal logic specifications. *IEEE Transactions on Automatic Control*, 53(1):287–297, 2008.

P. Koiran, M. Cosnard, and M. Garzon. Computability with low-dimensional dynamical systems. *Theoretical Computer Science*, 132(1):113–128, 1994.

G. Konidaris, S. Kuindersma, R. Grupen, and A. Barto. Robot learning from demonstration by constructing skill trees. *The International Journal of Robotics Research*, 31(3):360–375, 2012.

D. Kostic, B. De Jager, M. Steinbuch, and R. Hensen. Modeling and identification for high-performance robot control: An rrr-robotic arm case study. *IEEE Transactions on Control Systems Technology*, 12 (6):904–919, 2004.

H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas. Where's waldo? sensor-based temporal logic motion planning. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 3116–3121. IEEE, 2007.

H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas. Temporal-logic-based reactive mission and motion planning. *IEEE transactions on robotics*, 25(6):1370–1381, 2009.

J. La Salle and S. Lefschetz. *Stability by Liapunov's Direct Method with Applications by Joseph L Salle and Solomon Lefschetz*, volume 4. Elsevier, 2012.

G. Lafferriere, G. J. Pappas, and S. Yovine. *Decidable hybrid systems*. Citeseer, 1998.

J. P. LaSalle. *The stability of dynamical systems*, volume 25. SIAM, 1976.

J. B. Lasserre. Global optimization with polynomials and the problem of moments. *SIAM Journal on Optimization*, 11(3):796–817, 2001.

J. B. Lasserre. Moments and sums of squares for polynomial optimization and related problems. *Journal of Global Optimization*, 45(1):39–61, 2009.

Steven M LaValle. Rapidly-exploring random trees: A new tool for path planning. 1998.

J. L. Le Ny and G. J. Pappas. Sequential composition of robust controller specifications. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 5190–5195. IEEE, 2012.

J.-S. Liu. Joint stick-slip friction compensation for robotic manipulators by iterative learning. In *Intelligent Robots and Systems (IROS), 1994 IEEE/RSJ International Conference on*, volume 1, pages 502–509. IEEE, 1994.

M. Liu, Y. Tan, and V. Padois. Generalized hierarchical control. *Autonomous Robots*, 40(1):17–31, 2016.

S. Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

R. Lober. *Task Compatibility and Feasibility Maximization for Whole-Body Control*. PhD thesis, UPMC, 2017.

W. Lohmiller and J.-J. E. Slotine. On contraction analysis for non-linear systems. *Automatica*, 34(6): 683–696, 1998.

R. Longchamp. Stable feedback control of bilinear systems. *IEEE Transactions on Automatic Control*, 25(2):302–306, 1980.

L. Lovász and A. Schrijver. Cones of matrices and set-functions and 0–1 optimization. *SIAM Journal on Optimization*, 1(2):166–190, 1991.

T. Lozano-Perez. Robot programming. *Proceedings of the IEEE*, 71(7):821–841, 1983.

J. Lunze. *Regelungstechnik 2: Mehrgrößensysteme Digitale Regelung*. Springer-Verlag, 2013.

A. M. Lyapunov. The general problem of motion stability. *Annals of Mathematics Studies*, 17, 1892.

D. G. Macharet, A. A. Neto, V. F. da Camara Neto, and M. FM. Campos. Nonholonomic path planning optimization for dubins' vehicles. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 4208–4213. IEEE, 2011.

A. Majumdar and R. Tedrake. Robust online motion planning with regions of finite time invariance. In *Algorithmic Foundations of Robotics X*, volume 86 of *STAR*, pages 543–558. Springer, 2013.

A. Majumdar and R. Tedrake. Funnel libraries for real-time robust feedback motion planning. *The International Journal of Robotics Research*, 36(8):947–982, 2017.

A. Majumdar, A. A. Ahmadi, and R. Tedrake. Control design along trajectories with sums of squares programming. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 4054–4061. IEEE, 2013a.

A. Majumdar, A. A. Ahmadi, and R. Tedrake. Control design along trajectories with sums of squares programming. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 4054–4061, 2013b.

O. Maler and G. Batt. Approximating continuous systems by timed automata. In *Formal methods in systems biology*, volume 5054 of *LNBI*, pages 77–89. Springer, 2008.

O. Maler, Z. Manna, and A. Pnueli. From timed to hybrid systems. In *Real-time: theory in practice*, volume 600 of *LNCS*, pages 447–484. Springer, 1992.

I. R. Manchester and J.-J. E. Slotine. Control contraction metrics: Convex and intrinsic criteria for nonlinear feedback design. *IEEE Transactions on Automatic Control*, 62(6):3046–3053, 2017.

S. Manschitz, M. Gienger, J. Kober, and J. Peters. Mixture of attractors: A novel movement primitive representation for learning motor skills from demonstrations. *IEEE Robotics and Automation Letters*, 2018.

M. Marshall. *Positive polynomials and sums of squares*. Number 146. American Mathematical Soc., 2008.

M. T. Mason. The mechanics of manipulation. In *Robotics and Automation (ICRA), 1985 IEEE International Conference on*, volume 2, pages 544–548. IEEE, 1985.

A. Meguenani, V. Padois, J. Da Silva, A. Hoarau, and P. Bidaud. Energy based control for safe human-robot physical interaction. In *International Symposium on Experimental Robotics*, pages 809–818. Springer, 2016.

M. Morari and J. H. Lee. Model predictive control: past, present and future. *Computers & Chemical Engineering*, 23(4-5):667–682, 1999.

T. S. Motzkin. The arithmetic-geometric inequality. *Inequalities (Proc. Sympos. Wright-Patterson Air Force Base, Ohio, 1965)*, pages 205–224, 1967.

Y. Nesterov and A. Nemirovskii. *Interior-point polynomial algorithms in convex programming*, volume 13. SIAM, 1994.

K. Neumann and J. J. Steil. Learning robot motions with stable dynamical systems under diffeomorphic transformations. *Robotics and Autonomous Systems*, 70:1–15, 2015.

S.-R. Oh, Z. Bien, and I. H. Suh. An iterative learning control method with application to robot manipulators. *IEEE Journal on Robotics and Automation*, 4(5):508–514, 1988.

B. O'Donoghue, E. Chu, N. Parikh, and S. Boyd. Conic optimization via operator splitting and homogeneous self-dual embedding. *Journal of Optimization Theory and Applications*, 169(3):1042–1068, 2016.

P. M. Pardalos and S. A. Vavasis. Quadratic programming with one negative eigenvalue is np-hard. *Journal of Global Optimization*, 1(1):15–22, 1991.

P. C. Parks. Am lyapunov's stability theory—100 years on. *IMA journal of Mathematical Control and Information*, 9(4):275–303, 1992.

P. A. Parrilo. *Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization*. PhD thesis, California Institute of Technology, 2000.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python . *Journal of Machine Learning Research*, 12: 2825–2830, 2011.

N. Perrin and P. Schlehuber-Caissier. Fast diffeomorphic matching to learn globally asymptotically stable nonlinear dynamical systems. *Systems & Control Letters*, 96:51–59, 2016.

A. Pisano and E. Usai. Sliding mode control: A survey with applications in math. *Mathematics and Computers in Simulation*, 81(5):954–979, 2011.

N. Piterman, A. Pnueli, and Y. Sa'ar. Synthesis of reactive (1) designs. In *International Workshop on Verification, Model Checking, and Abstract Interpretation*, pages 364–380. Springer, 2006.

E. Plaku, L. E. Kavraki, and M. Y. Vardi. Motion planning with dynamics by a synergistic combination of layers of planning. *IEEE Transactions on Robotics*, 26(3):469–482, 2010.

A. Platzer and J.-D. Quesel. Keymaera: A hybrid theorem prover for hybrid systems (system description). In *International Joint Conference on Automated Reasoning*, pages 171–178. Springer, 2008.

A. Pnueli. The temporal logic of programs. In *Foundations of Computer Science, 1977., 18th Annual Symposium on*, pages 46–57. IEEE, 1977.

S. Prajna, A. Papachristodoulou, and F. Wu. Nonlinear control synthesis by sum of squares optimization: A Lyapunov-based approach. In *Control Conference, 2004. 5th Asian*, volume 1, pages 157–165. IEEE, 2004.

M. O. Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions of the american Mathematical Society*, 141:1–35, 1969.

A. Z. Rao. *Realization of Dynamixel Servo Plant Parameters to Improve Admittance Control for a Compliant Human-robot Interaction*. PhD thesis, New Jersey Institute of Technology, Department of Biomedical Engineering, 2016.

H. Ravanbakhsh and S. Sankaranarayanan. Robust controller synthesis of switched systems using counterexample guided framework. In *ACM/IEEE Conference on Embedded Software (EMSOFT)*, pages 8:1–8:10, 2016.

H. Ravichandar, I. Salehi, and A. Dani. Learning partially contracting dynamical systems from demonstrations. In *Conference on Robot Learning*, pages 369–378, 2017.

L. Righetti and S. Schaal. Quadratic programming for inverse dynamics with optimal distribution of contact forces. In *Humanoid Robots (Humanoids), 2012 12th IEEE-RAS International Conference on*, pages 538–543. IEEE, 2012.

E. Rimon and D. E. Koditschek. The construction of analytic diffeomorphisms for exact robot navigation on star worlds. *Transactions of the American Mathematical Society*, 327(1):71–116, 1991.

Q. Rouxel, G. Passault, L. Hofer, S. N'Guyen, and O. Ly. Learning the odometry on a small humanoid robot. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 1810–1816. IEEE, 2016.

G. Rozenberg and A. Salomaa. *Handbook of Formal Languages: Volume 3 Beyond Words*. Springer Science & Business Media, 2012.

S. Schaal. Dynamic movement primitives-a framework for motor control in humans and humanoid robotics. In *Adaptive motion of animals and machines*, pages 261–280. Springer, 2006.

S. Schaal, A. J. Ijspeert, and A. Billard. Computational approaches to motor learning by imitation. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 358(1431):537–547, 2003.

A. Scheuer and T. Fraichard. Continuous-curvature path planning for car-like vehicles. In *Intelligent Robots and Systems (IROS), 1997 IEEE/RSJ International Conference on*, volume 2, pages 997–1003. IEEE, 1997.

P. Schlehuber-Caissier and N. Perrin. Computing regions of stabilizability for nonlinear control systems with input constraints. In *2018 Annual American Control Conference (ACC)*, pages 2869–2876. IEEE, 2018.

M. Schwarz and S. Behnke. Compliant robot behavior using servo actuator models identified by iterative learning control. In *Robot Soccer World Cup*, pages 207–218. Springer, 2013.

P. OM. Scokaert and J. B. Rawlings. Infinite horizon linear quadratic control with constraints. *IFAC Proceedings Volumes*, 29(1):5905–5910, 1996.

A. Segre and G. DeJong. Explanation-based manipulator learning: Acquisition of planning ability through observation. In *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, volume 2, pages 555–560. IEEE, 1985.

H. D. Sherali and W. P. Adams. A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. *SIAM Journal on Discrete Mathematics*, 3(3): 411–430, 1990.

H. D. Sherali and B. MP. Fraticelli. Enhancing rlt relaxations via a new class of semidefinite cuts. *Journal of Global Optimization*, 22(1-4):233–261, 2002.

H. D. Sherali and C. H. Tuncbilek. A global optimization algorithm for polynomial programming problems using a reformulation-linearization technique. *Journal of Global Optimization*, 2(1):101–112, 1992.

H. D. Sherali and C. H. Tuncbilek. A reformulation-convexification approach for solving nonconvex quadratic programming problems. *Journal of Global Optimization*, 7(1):1–31, 1995.

H. D. Sherali, E. Dalkiran, and J. Desai. Enhancing rlt-based relaxations for polynomial programming problems via a new class of v-semidefinite cuts. *Computational Optimization and Applications*, 52(2): 483–506, 2012.

S. Singh, A. Majumdar, J.-J. Slotine, and O. Pavone. Robust online motion planning via contraction theory and convex optimization. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 5883–5890. IEEE, 2017.

C. Sloth and R. Wisniewski. Abstraction of continuous dynamical systems utilizing lyapunov functions. In *Decision and Control (CDC), 2010 49th IEEE Conference on*, pages 3760–3765. IEEE, 2010a.

C. Sloth and R. Wisniewski. Timed game abstraction of control systems. Technical Report 1012.5113, ArXiv, 2010b.

C. Sloth and R. Wisniewski. Complete abstractions of dynamical systems by timed automata. *Nonlinear Analysis: Hybrid Systems*, 7(1):80–100, 2013.

J. Smith, S. L .and Tümová, C. Belta, and D. Rus. Optimal path planning for surveillance with temporal-logic constraints. *The International Journal of Robotics Research*, 30(14):1695–1708, 2011.

E. D. Sontag. *Mathematical control theory: deterministic finite dimensional systems*. Springer, 1998.

A. Sotiras, C. Davatzikos, and N. Paragios. Deformable medical image registration: A survey. *IEEE transactions on medical imaging*, 32(7):1153–1190, 2013.

P. Soueres, A. Balluchi, and A. Bicchi. Optimal feedback control for route tracking with a bounded-curvature vehicle. *International Journal of Control*, 74(10):1009–1019, 2001.

M. W. Spong. The swing up control problem for the acrobot. *IEEE control systems*, 15(1):49–55, 1995.

R. Stribeck. Die wesentlichen eigenschaften der gleit-und rollenlager. *Zeitschrift des Vereines Deutscher Ingenieure*, 46:1341–1348, 1902.

J. F. Sturm. Using sedumi 1.02, a matlab toolbox for optimization over symmetric cones. *Optimization methods and software*, 11(1-4):625–653, 1999.

I. A. Şucan and L. E. Kavraki. A sampling-based tree planner for systems with complex dynamics. *IEEE Transactions on Robotics*, 28(1):116–131, 2012.

I. A. Şucan, M. Moll, and L. E. Kavraki. The open motion planning library. *IEEE Robotics & Automation Magazine*, 19(4):72–82, 2012. http://ompl.kavrakilab.org.

J. Swevers, C. Ganseman, J. De Schutter, and H. Van Brussel. Experimental robot identification using optimised periodic trajectories. *Mechanical Systems and Signal Processing*, 10(5):561–577, 1996.

R. Tedrake and the Drake Development Team. Drake: A planning, control, and analysis toolbox for nonlinear dynamical systems, 2016. URL `http://drake.mit.edu`.

R. Tedrake, I. R. Manchester, M. Tobenkin, and J. W. Roberts. LQR-trees: Feedback motion planning via sums-of-squares verification. *IJRR*, 29(8):1038–1052, 2010a.

R. Tedrake, I. R. Manchester, M. Tobenkin, and J. W. Roberts. Lqr-trees: Feedback motion planning via sums-of-squares verification. *The International Journal of Robotics Research*, 29(8):1038–1052, 2010b.

J. J. Verbeek, N. Vlassis, and B. Kröse. Efficient greedy learning of gaussian mixture models. *Neural computation*, 15(2):469–485, 2003.

W. Wang, J. Yi, D. Zhao, and D. Liu. Design of a stable sliding-mode controller for a class of second-order underactuated systems. *IEE Proceedings-Control Theory and Applications*, 151(6):683–690, 2004.

J. Wu, J. Wang, and Z. You. An overview of dynamic parameter identification of robots. *Robotics and computer-integrated manufacturing*, 26(5):414–419, 2010.