Guaranteeing Constraints of Disturbed Nonlinear Systems Using Set-Based Optimal Control in Generator Space

Bastian Schürmann and Matthias Althoff

Department of Informatics, Technische Universität München, Boltzmannstr. 3, 85748 Garching, Germany (e-mail: bastian.schuermann@tum.de, althoff@tum.de).

Abstract: We address the problem of finding an optimal solution for a nonlinear system for a set of initial states rather than just for a single initial state. In addition, we consider state and input constraints as well as a set of possible disturbances. While previous optimal control techniques typically ignore the fact that the current state of a system is not exactly known, future safety-critical systems demand that all uncertainties including the initial state are considered; this is required for e.g. automated vehicles, surgical robots, or human-robot interaction. We present a new method that obtains optimal control inputs by finding optimal weights for generators that span the space reachable by the considered system. This solution routine can be used not only for a single initial state but also for a set of initial states – this is not possible using classical optimization techniques. We ensure that all constraints are met by using reachability analysis, which provides formal bounds for all possible system trajectories. We demonstrate the applicability of our approach with an example from automated driving; for this example, the result is obtained within a few seconds and outperforms a classical LQR approach.

Keywords: Control of Constrained Systems, Optimal Control, Nonlinear Control, Robust Control, Reach-Avoid Problems, Reachability Analysis, Zonotopes

1. INTRODUCTION

In many control problems, a system has to be controlled from a given initial state as close as possible to a desired final state in a fixed time while satisfying input constraints and avoiding forbidden regions in the state space. Such reach avoid-problems appear in many applications, e.g., in autonomous driving, where a car has to be steered from its initial position to a desired final position while obeying traffic rules and avoiding other traffic participants. For single initial states, these problems can be solved efficiently by using classical optimization techniques. In many scenarios, however, it is not enough to have a solution for a single initial state only: one needs solutions for a set of initial states, for example if the exact initial state is not known beforehand or if sensors are not accurate. If one deals with an initial set, efficient methods which work for single initial states no longer work, as they would have to be applied for all infinitely-many states of a continuous set.

Many optimal control techniques which are developed to deal with initial sets rely on dividing and discretizing the state space as discussed at the end of this section. Since this becomes computationally intractable for higher-dimensional systems, we present a novel optimal control approach, which combines the optimal control problem of a set of initial states into a single linear program that can be solved efficiently. By iteratively applying the open-loop optimal control in combination with reachability analysis, we find a novel way of obtaining feedback controllers which provide formal guarantees for the satisfaction of state and input constraints despite nonlinear dynamics and disturbances. This has the additional advantage that we do not need to prove stability of our controller, as safety is guaranteed by the included reachability analysis.

Our approach is not restricted to planning single maneuvers; it is also particularly useful in combination with safe maneuver automata as presented in Heß et al. (2014). Therein, reach-avoid problems are solved for different initial and final states and the resulting maneuvers are stored as states in a so-called maneuver automaton. There exists a transition between two states, i.e., two maneuvers can be combined, if the final reachable set of the first maneuver is completely contained in the initial set of the second maneuver. This allows one to compute and to verify maneuvers offline in advance. During the online control, path planning simplifies to combining the pre-computed maneuvers in a discrete search. To be able to follow each maneuver with as many other maneuvers as possible, it is important to obtain small reachable sets at the end of each maneuver primitive. Since this is hard for many classical control methods when dealing with constraints and disturbances, our novel control approach offers a solution to this problem.

There also exist other approaches for computing controllers for sets of initial states; however, as explained subsequently, most of them are either not able to provide guarantees for disturbed, nonlinear systems or their computational effort grows very fast with the number of dimensions. One way to systematically solve optimal con-

trol problems is by solving the Hamilton-Jacobi-Bellmann (HJB) equations or by using dynamic programming [Bertsekas (2005); Blanchini and Miani (2008); Lygeros et al. (1999); Kurzhanski et al. (2006)]. However, the analytic solution of the HJB equation is only possible for simple and small-dimensional systems. For higher-dimensional systems, especially with disturbances, the computation easily becomes too complex. Borrelli et al. (2015) use similar methods for explicit model predictive control (MPC), in particular for linear systems. In explicit MPC, the state space is divided into different parts, and an optimal control law is computed for each. Since the computational costs for this partitioning grow exponentially with the number of dimensions, this approach is also restricted to low-dimensional systems. This curse of dimensionality, and therefore the restriction to low-dimensional systems, is a common problem for all approaches which rely on discretizing the state space, including most abstraction-based control methods [Liu et al. (2012); Kloetzer and Belta (2008); Zamani et al. (2012)]. There, these discretizations are used to compute controllers which take complex specifications into account.

Other approaches compute funnels or tubes around trajectories and use different methods to provide guarantees. In Julius and Winn (2012), so-called trajectory robustness is used to steer an initial set around a trajectory; however, this only works for undisturbed and feedback-linearizable systems. A method which works for disturbed systems is tube-based MPC [Mayne et al. (2005); Rakovic et al. (2012)], where an additional feedback controller is used to keep all states in a tube around the optimized center trajectory despite disturbances. These methods work mainly for linear systems, though, and most approaches use a fixed feedback controller without further optimization of the tube size. Another way to compute tracking controllers around trajectories is presented in Tedrake et al. (2010) where sums-of-squares (SOS) programming techniques are used to compute LQR controllers to keep the states around a trajectory. This method is also successfully used to compute maneuver automata [Majumdar and Tedrake (2013)]; however, the computational effort of SOS methods grows very fast with the dimension of the system.

We have already considered the problem at hand in two previous works: In Schürmann and Althoff (2017a), we obtain control inputs for all states of an initial set by interpolating optimal open-loop inputs of the extreme points of a set, using convex combinations. Since the number of extreme points grows exponentially with the number of dimensions, we use generators in this work instead, which scale linearly with the dimension. In Schürmann and Althoff (2017b), we are able to obtain continuous feedback by directly optimizing the closed-loop dynamics, though only for linear dynamics.

The remainder of this paper is organized as follows: After a formal problem formulation in Sec. 2, we present our new algorithm in Sec. 3. Its applicability is shown in a numerical example in Sec. 4. We conclude with a discussion of the algorithm in Sec. 5 and a summary in Sec. 6.

2. PROBLEM FORMULATION

We consider a disturbed, nonlinear, time-continuous system of the form

$$\dot{x}(t) = f(x(t), u(t), w(t)), \tag{1}$$

with states $x(t) \in \mathbb{R}^n$, inputs $u(t) \in \mathbb{R}^m$, and disturbances $w(t) \in \mathcal{W} \subset \mathbb{R}^d$ ($\mathcal{W}$ is compact, i.e., closed and bounded). We do not require any stochastic properties for $w(\cdot)$; we only assume that any possible disturbance trajectory is bounded at any point in time in the compact set $\mathcal{W}$. We denote this by $w(\cdot) \in \mathcal{W}$, which is a shorthand for $w(t) \in \mathcal{W}, \forall t \in [0, t_f]$, where $t_f \in \mathbb{R}_0^+$ is the final time. The same shorthand is also used for state and input trajectories throughout the paper. We denote the solution of (1) with initial state $x(0)$, input $u(\cdot)$, and disturbance $w(\cdot)$ at time t as $\xi(x(0), u(\cdot), w(\cdot), t)$. If we consider an undisturbed system, we use $\xi(x(0), u(\cdot), 0, t)$ to denote the solution without disturbances, i.e., $\mathcal{W} = 0$.

The task is to find a control law $u_{control}(x, t)$ for system (1) which guarantees that all states in an initial set $\mathcal{X}_0 \subset \mathbb{R}^n$ are steered into a final set $\mathcal{S}_f \subset \mathbb{R}^n$ around an end state $x^{(f)}$ after time t_f, despite the disturbance set $\mathcal{W}$. We minimize the size of the final set by solving

$$\min_{u_{control}} \max\{\|x - x^{(f)}\|_1 | x \in \mathcal{S}_f\}. \tag{2}$$

Furthermore, we consider convex constraints on the states and inputs, i.e.,

$$\xi(x(0), u(\cdot), w(\cdot), \cdot) \in \mathcal{X}, \tag{3}$$

$$u(\cdot) \in \mathcal{U}, \tag{4}$$

where $\mathcal{X}$ and $\mathcal{U}$ are both convex sets in $\mathbb{R}^n$ and $\mathbb{R}^m$, respectively.

Note that during offline computation, the locations of most non-convex constraints, such as other traffic participants in automated driving, are not known. Therefore, we use this approach to compute the maneuvers offline in advance, while taking convex input constraints, e.g., maximum acceleration or steering, and convex state constraints, e.g., maximum velocity, into account. The non-convex dynamical constraints are handled during the online planning using the maneuver automaton (see Fig. 1).

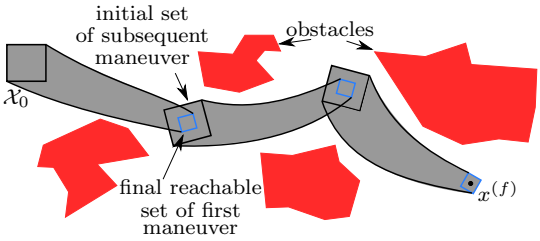


Fig. 1. Using maneuvers (gray) to steer all states from $\mathcal{X}_0$ to $x^{(f)}$ while avoiding nonconvex obstacles (red). The final set (blue box) of one maneuver is always contained inside the initial set (black box) of the following maneuver.

For most of the paper, we want to find a final set $\mathcal{S}_f$ which is as small as possible. If instead the task is to steer all states into a given final set, then we would have to adapt our algorithms by adding this as an additional constraint. In this case, however, it might be possible that no solution exists, depending on the choice of constraints, final time, and final set.

3. OPTIMAL CONTROL IN STATE SPACE

To prove safety and other important properties of continuous systems, formal verification in the form of reachability analysis [Althoff (2010), Asarin et al. (2006)] has proven to be useful. Therein, reachable sets are computed, and it is checked whether these sets satisfy certain properties, such as not intersecting with forbidden sets. The reachable set of a system is defined as:

Definition 1. (Reachable Set). *For a system* (1)*, the reachable set* $\mathcal{R}_{t,\mathcal{U},\mathcal{W}}(\mathcal{S}) \subset \mathbb{R}^n$ *for a time* t*, inputs* $u(\cdot) \in \mathcal{U} \subset \mathbb{R}^m$*, disturbances* $w(\cdot) \in \mathcal{W} \subset \mathbb{R}^d$*, and an initial set* $\mathcal{S} \subset \mathbb{R}^n$ *is the set of end states of trajectories starting in* $\mathcal{S}$ *after time* t*, i.e.,*

$$\mathcal{R}_{t,\mathcal{U},\mathcal{W}}(\mathcal{S}) = \{x(t) \in \mathbb{R}^n | \exists x(0) \in \mathcal{S}, u(\cdot) \in \mathcal{U}, w(\cdot) \in \mathcal{W} : \xi(x(0), u(\cdot), w(\cdot), t) = x(t)\}.$$

The reachable set over a time interval $[t_1, t_2]$ *is the union of all reachable sets for these time points, i.e.,*

$$\mathcal{R}_{[t_1,t_2],\mathcal{U},\mathcal{W}}(\mathcal{S}) = \bigcup_{t \in [t_1,t_2]} \mathcal{R}_{t,\mathcal{U},\mathcal{W}}(\mathcal{S}).$$

If we consider the reachable set for a system with feedback $u_{fb}(x)$, then we denote by $\mathcal{R}_{t,u_{fb},\mathcal{W}}(\mathcal{S})$ the reachable set obtained if we consider the closed-loop dynamics $\dot{x}(t) = f(x(t), u_{fb}(x(t)), w(t))$ and no open-loop inputs. If we consider systems without disturbances, then we use $\mathcal{R}_{t,\mathcal{U}}(\mathcal{S})$ as a shorthand for $\mathcal{R}_{t,\mathcal{U},0}(\mathcal{S})$, i.e., $\mathcal{W} = 0$. Since it is not possible to compute exact reachable sets for most systems [Platzer and Clarke (2007)], we compute over-approximations instead.

Zonotopes are an efficient set representation for reachability analysis, as used for example by Althoff (2010):

Definition 2. (Zonotope). *A set is called a zonotope if it can be written as*

$$\mathcal{Z} = \left\{ x \in \mathbb{R}^n \middle| x = c + \sum_{i=1}^{p} \alpha_i g^{(i)}, \alpha_i \in [-1, 1] \right\}. \quad (5)$$

Therein $c \in \mathbb{R}^n$ *defines the center of the zonotope, and* $g^{(i)} \in \mathbb{R}^n, i \in \{1, \ldots, p\}$*, are* $p = on$ *generators, with* o *denoting the order of the zonotope. The generators can also be combined in the generator matrix* $G \in \mathbb{R}^{n \times p}$*, which contains the generators as its columns. We use* $< c, G >$ *as a more concise notation of* $\mathcal{Z}$*. A zonotope with* n *linearly independent generators is called a parallelotope.*

An overview of our main idea is illustrated in Fig. 2: In classical optimization of continuous dynamic systems, the cost function of a continuous trajectory is minimized in state space for a single initial state (Fig. 2 top-left). Currently, it is unknown how to solve a corresponding optimization problem when the possible set of initial states is provided (Fig. 2 top-right). Our idea is to reformulate the classical optimization problem as a problem of finding optimal weights of generators that span the reachable set of a system (Fig. 2 bottom-left). We refer to this problem space as the generator space; details of this transformation are presented in the following section. Based on the changed problem formulation, we can generalize the problem in a straightforward way to sets of initial states (Fig. 2 bottom-right). Thereby, we separately consider the partial reachable sets originating from the homogeneous solution and from the input solution. After obtaining the solution

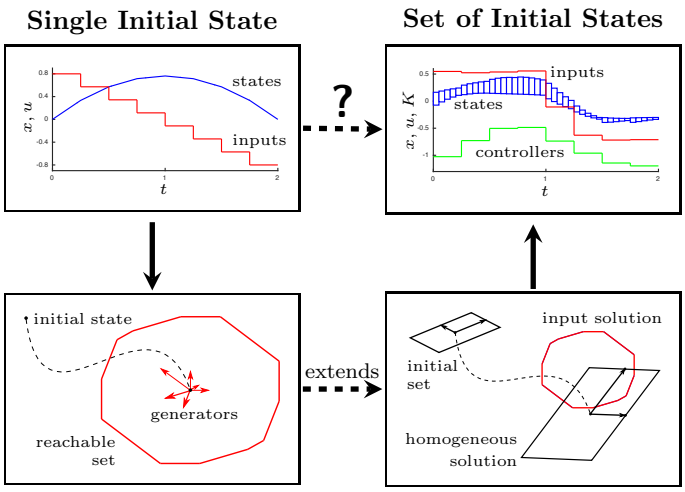


Fig. 2. Overview of solving optimal control problems.

in the generator space, we can map it back to the original state space to obtain the optimal solution. In the following, we discuss these steps in detail for different system classes.

3.1 Linear Systems with Single Initial State

Let us start with a single initial state and an undisturbed, continuous-time, linear time-invariant (LTI) dynamic of the form

$$\dot{x}(t) = A_c x(t) + B_c u(t),$$

with $x(t) \in \mathbb{R}^n, u(t) \in \mathbb{R}^m, A_c \in \mathbb{R}^{n \times n}$, and $B_c \in \mathbb{R}^{n \times m}$. To obtain a simple representation of the reachable set, let us first note that any continuous-time LTI system can be transformed into a discrete-time system of the form

$$x(t_{k+1}) = Ax(t_k) + Bu(t_k), \quad (6)$$

with $A = e^{A_c t_s}$ and $B = \int_0^{t_s} e^{A_c \tau} d\tau B_c$. Therein, $t_s = \frac{t_f}{N}$, $N \in \mathbb{N}$, denotes the sampling time, and we use t_k as a shorthand for kt_s, $k \in \{0, \ldots, N\}$. We assume that the control inputs are constant during each sampling period. For the discrete-time system (6), the solution can be expressed as

$$\xi(x(0), u(\cdot), 0, t_N) = A^N x(0) + A^{N-1} B u(t_0) + \cdots + B u(t_{N-1}).$$

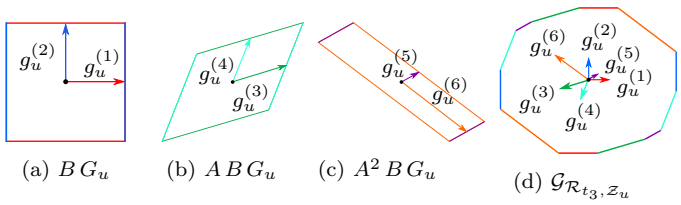
If we assume that the input set is given as a zonotope $\mathcal{U} = \mathcal{Z}_u =< c_u, G_u >$, then the solution can be written as

$$\begin{aligned} \xi(x(0), u(\cdot), 0, t_N) &= A^N x(0) + (A^{N-1} B + \cdots + B) c_u \\ &\quad + A^{N-1} B G_u \alpha(t_0) + \cdots + B G_u \alpha(t_{N-1}) \quad (7) \\ &= c_{\mathcal{R}_{t_N,\mathcal{Z}_u}} + G_{\mathcal{R}_{t_N,\mathcal{Z}_u}} \alpha. \end{aligned}$$

Therein $c_{\mathcal{R}_{t_N,\mathcal{Z}_u}} = A^N x(0) + (A^{N-1} B + \cdots + B) c_u$ and $G_{\mathcal{R}_{t_N,\mathcal{Z}_u}} = \left[A^{N-1} B G_u, \ldots, B G_u \right]$ are the center and generator matrix of the reachable set $\mathcal{R}_{t_N,\mathcal{Z}_u}(x(0))$, respectively. We illustrate how $G_{\mathcal{R}_{t_N,\mathcal{Z}_u}}$ consists of the input effects of different time steps in Fig. 3. The reachable set itself is shown in Fig. 4(b) as the combination of $G_{\mathcal{R}_{t_N,\mathcal{Z}_u}}$ and the homogeneous solution shown in Fig. 4(a). Since the set reachable by the system is spanned by $G_{\mathcal{R}_{t_N,\mathcal{Z}_u}}$, finding an input trajectory which steers $x(0)$ as close as possible to $x^{(f)}$ simplifies to finding weights α, which express the closest state to $x^{(f)}$ inside the generator space (see Fig. 4(c)). This is done in the following linear program

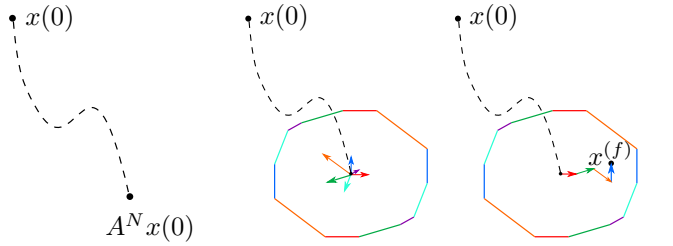
$$\min_{\alpha} \| \underbrace{c_{\mathcal{R}_{t_N,\mathcal{Z}_u}} + G_{\mathcal{R}_{t_N,\mathcal{Z}_u}} \alpha}_{\xi(x(0),u(\cdot),0,t_N),\ \text{see (7)}} - x^{(f)} \|_1 \tag{8}$$
$$\text{s.t.} |\alpha| \leq \mathbf{1},$$

where $\mathbf{1}$ denotes a vector containing only ones. Therein, absolute value and less or equal operators are both performed element-wise. The optimal input at any time t_k is then given as $u(t_k) = c_u + G_u \alpha(t_k)$.



(a) BG_u (b) ABG_u (c) A^2BG_u (d) $\mathcal{G}_{\mathcal{R}_{t_3,\mathcal{Z}_u}}$

Fig. 3. We combine the effects from the inputs of the current step (a) and the previous time-steps (b,c) into a single zonotope (d).



(a) $c_{\mathcal{R}_{t_3,\mathcal{Z}_u}}$ (b) $\mathcal{R}_{t_3,\mathcal{Z}_u}(x(0))$ (c) $\xi(x(0), u(\cdot), 0, t_3)$

Fig. 4. Illustration of our approach for a single initial state, $N = 3$ and $c_u = 0$.

The standard form of a linear program is $\max_y c^T y$ subject to $Dx = e,\ y \geq 0$, with $c, y \in \mathbb{R}^n, D \in \mathbb{R}^{n \times m}$, and $e \in \mathbb{R}^m$ [Dantzig and Thapa (2006)]. As shown in (Dantzig and Thapa, 2006, Ch. 6) and Grant et al. (2006), any optimization problem of the form (8) can be transformed into a linear program, which is not shown due to space restrictions.

For discrete-time LTI systems, the batch approach (Borrelli et al., 2015, Sec. 9.2) is well known and often used for MPC. It is similar to our approach and also uses an explicit expression for future states, but encloses inputs in boxes or polytopes. However, it does not extend to the case of initial sets.

3.2 Linear Systems with Initial Sets

Let us now apply the idea from the previous section to initial sets $\mathcal{X}_0 =< c_x, G_x >$ instead of to a single initial state $x(0)$. Introducing the Minkowski sum ($\mathcal{A} \oplus \mathcal{B} = \{a + b | a \in \mathcal{A}, b \in \mathcal{B}\}$) allows us to express the reachable set similarly to (7):

$$\mathcal{R}_{t_N,\mathcal{U}}(\mathcal{X}_0) = A^N \mathcal{X}_0 \oplus A^{N-1} B \mathcal{Z}_u \oplus \cdots \oplus B \mathcal{Z}_u.$$

As for the single initial state case, we have a part which we cannot influence with our controller and another part which results from our control inputs. Let us denote by $\tilde{\mathcal{R}} =< \tilde{c}, \tilde{G} >$ the resulting reachable set, which we want to minimize in (2). Since $\tilde{\mathcal{R}}$ is a zonotope, minimizing (2) is equivalent to minimizing

$$\max \|\tilde{c} - x^{(f)} + \bigoplus_{i=1}^{p} \{-\tilde{g}^{(i)}, \tilde{g}^{(i)}\}\|_1, \tag{9}$$

with $\bigoplus$ denoting the sum symbol for Minkowski sum, which we use to refer to all 2^p possible extreme points. This means we want to minimize the maximum distance of any extreme point and $x^{(f)}$. While this optimization problem becomes too complex, we are able to obtain

$$\|\tilde{c} - x^{(f)}\|_1 + \sum_{i=1}^{p} \|\tilde{g}^{(i)}\|_1 \tag{10}$$

as an upper bound for (9), where (9) $\leq$ (10) follows from the triangular inequality. This is also one reason why we use the 1-norm, as this inequality does not hold for the 2-norm. This shows that by steering the center as close as possible to $x^{(f)}$ and by minimizing the length of the generators, we minimize an upper bound for (2). After introducing $c_x = x(0)$, steering c_x as close as possible to $x^{(f)}$ results in the same optimization problem (8) as for the single initial state. Equivalently, we minimize the length of the generators in

$$\min_{\alpha} \|A^N g_x^{(i)} + G_{\mathcal{R}_{t_N,\mathcal{Z}_u}} \alpha\|_1, \forall i \in \{1, \ldots, p\}, \tag{11}$$

which we obtain by replacing $c_{\mathcal{R}_{t_N,\mathcal{Z}_u}}$ in (8) with the homogeneous solution of the generator $A^N g_x^{(i)}$ and the desired final state $x^{(f)}$ with the origin $\mathbf{0}$, which therefore can be neglected. This is illustrated in Fig. 5, where we use the input solutions (shown as concatenated arrows in red, green, and blue) to steer the center and the generators of the homogeneous solution as close as possible to $x^{(f)}$ and the origin, respectively.

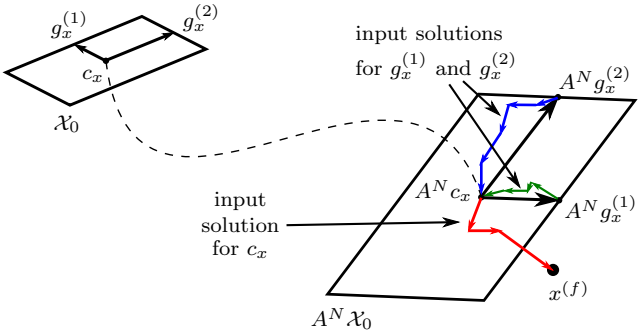


Fig. 5. The approach from Fig. 4 extended to initial sets.

Since any state $x \in \mathcal{X}_0$ is a superposition of c_x and $g_x^{(i)}$, we cannot compute the α weights in (8) and (11) independently of each other. Otherwise, the superposed inputs might violate the input constraints. Instead, we use the fact that $\left\| \begin{bmatrix} a \\ b \end{bmatrix} \right\|_1 = \|a\|_1 + \|b\|_1$ to combine (8) and (11) in a single linear program which minimizes (10) while taking the coupled inputs into account:

$$\min_{\substack{\alpha(c_x),\alpha(g_x^{(1)}),\\ \ldots,\alpha(g_x^{(p)})}} \left\| \begin{bmatrix} c_{\mathcal{R}_{t_N,\mathcal{Z}_u}} + G_{\mathcal{R}_{t_N,\mathcal{Z}_u}} \alpha(c_x) - x^{(f)} \\ A^N g_x^{(1)} + G_{\mathcal{R}_{t_N,\mathcal{Z}_u}} \alpha(g_x^{(1)}) \\ \vdots \\ A^N g_x^{(p)} + G_{\mathcal{R}_{t_N,\mathcal{Z}_u}} \alpha(g_x^{(p)}) \end{bmatrix} \right\|_1 \tag{12}$$

$$\text{s.t. } |\alpha(c_x)| + \sum_{i=1}^{p} |\alpha(g_x^{(i)})| \leq \mathbf{1}, \tag{13}$$

where $\alpha(c_x)$ denotes the vector containing the corresponding $\alpha(c_x, t_0), \dots, \alpha(c_x, t_{N-1})$ and analogously for $\alpha(g_x^{(i)})$. Therefore, the corresponding input for any state $x = c_x + \sum_{i=1}^{p} g_x^{(i)} \beta(x) \in \mathcal{X}_0$ is simply given as $u(x, t_k) = c_u + G_u \alpha(c_x, t_k) + \sum_{i=1}^{p} G_u \alpha(g_x^{(i)}, t_k) \beta_i(x)$.

3.3 Nonlinear Dynamics and Disturbances

In order to extend the approach developed for linear systems to nonlinear systems with disturbances, we first compute a reference trajectory and linearize the system along this trajectory. To fully consider the disturbances and the linearization errors, we apply the control law iteratively for short time horizons and compute the reachable set for the times in between. This way, we obtain feedback and guarantee that it satisfies the constraints at all points in time. The approach for nonlinear systems consists of the following three steps:

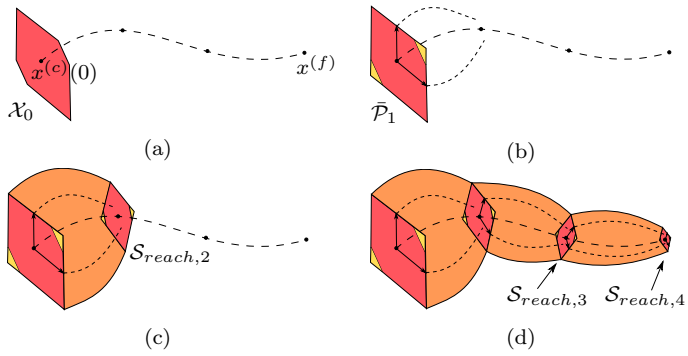


Fig. 6. Illustration of our control approach extended to disturbed, nonlinear systems.

Step 1: We first compute a reference trajectory steering the center c_x of the initial set $\mathcal{X}_0$ as close as possible to the desired final state $x^{(f)}$ (see Fig. 6(a)) by solving the following nonlinear optimization problem:

$$\min_{u^{(c)}(\cdot)} \|\xi(c_x, u^{(c)}(\cdot), 0, t_f) - x^{(f)}\|_1 + \gamma \|u^{(c)}(\cdot)\|_1, \quad (14)$$
$$\text{s.t. } u^{(c)}(\cdot) \in \mathcal{U}$$

where $\gamma \in \mathbb{R}^+$ denotes a weighting factor for the input costs.

We use the center trajectory $x^{(c)}(\cdot) = \xi(c_x, u^{(c)}(\cdot), 0, \cdot)$ as a reference for our system and divide it into N parts of length t_s. Since our linear programming approach requires discrete-time, linear dynamics for each time step, we first linearize the system dynamics along the reference trajectory, i.e., $\forall k \in \{1, \dots, N\}$

$$A_{c,k} = \frac{\partial f(x,u,0)}{\partial x}\Big|_{\substack{x=x^{(c)}(t'_k),\\ u=u^{(c)}(t'_k)}}, \; B_{c,k} = \frac{\partial f(x,u,0)}{\partial u}\Big|_{\substack{x=x^{(c)}(t'_k),\\ u=u^{(c)}(t'_k)}},$$

with $t'_k = t_{k-1} + \frac{1}{2} t_s$, and then time-discretize $(A_{c,k}, B_{c,k})$ to obtain (A_k, B_k), as done for linear systems.

Step 2: We consider a time horizon of h time steps, with $N = Mh$ and $M, h \in \mathbb{N}$. At each iteration step $l \in \{1, \dots, M\}$, we apply the set-based optimal control approach using the linearized dynamics. As the initial set, we use the reachable set $\mathcal{S}_{reach,l}$ from the last time step and over-approximate it by a parallelotope $\bar{\mathcal{P}}_l$, i.e., $\mathcal{S}_{reach,l} \subseteq \bar{\mathcal{P}}_l$. We initialize our approach with $\mathcal{S}_{reach,1} = \mathcal{X}_0$. The initial set is over-approximated by a parallelotope to reduce the number of generators which are considered in the optimization problem and to have unique control inputs when applying the controller online, as discussed later. There exist efficient ways to formally compute parallelotope over-approximations of zonotopes, see e.g. Althoff (2010). Since we compute over-approximations, our approach is still sound, and we ensure the satisfaction of the input constraints for all states.

At each step, we use the previously described set-based optimal control approach for linear dynamics to control $\bar{\mathcal{P}}_l$ as close as possible to $x^{(c)}(t_{lh})$ (see Fig. 6(b)) by solving (12) with (13), wherein we replace $x^{(f)}$ with $x^{(c)}(t_{lh})$ and A^N with $A_{lh} \dots A_{(l-1)h+1}$ since the linearized dynamics are time-varying.

Step 3: We then compute an over-approximation of the reachable set $\bar{\mathcal{R}}$ for the original nonlinear system with disturbances if controlled with the previously computed control inputs $u_{control}(\cdot)$ (see Fig. 6(c)). We use the over-approximated reachable set as the initial set of the next iteration step $S_{reach,l+1} = \bar{\mathcal{R}}_{t_h, u_{control}, \mathcal{W}}(\mathcal{S}_{reach,l})$ and continue with Step 2.

By iterating Steps 2 and 3 for all M time steps, we obtain with $\mathcal{S}_f = \mathcal{S}_{reach,M+1}$ the over-approximation of the final reachable set of all states starting in the initial set $\mathcal{X}_0$ despite disturbances (see Fig. 6(d)). We store the generators of the parallelotope over-approximations $\bar{\mathcal{P}}_l$ and the input zonotopes at sampling times in a look-up table.

Online Use of the Controller In order to apply the controller online, we have to compute the corresponding control input for a given state. If the initial set is a parallelotope, then the control input is unique. Given a state $x(t_{(l-1)h}) \in \bar{P}_{reach,l} =< c_{x,l}, G_{x,l} >$, any state can be uniquely expressed by the parameter vector β as

$$x(t_{(l-1)h}) = c_{x,l} + G_{x,l} \beta(x(t_{(l-1)h}))$$
$$\Rightarrow \beta(x(t_{(l-1)h})) = G_{x,l}^{-1}(x(t_{(l-1)h}) - c_{x,l}).$$

Therefore, the corresponding input for the time interval $[t_{(l-1)h}, t_{lh})$ can be computed as

$$u_{control}(x, t_k) = c_u + G_u \alpha(cu, t_k) \quad (15)$$
$$+ G_u [\alpha(g_x^{(1)}, t_k), \dots, \alpha(g_x^{(n)}, t_k)] G_{x,l}^{-1}(x(t_{(l-1)h}) - c_{x,l}),$$

for all $k \in \{(l-1)h, \dots, lh-1\}$. Since our controller is open-loop during each iteration step $[t_{(l-1)h}, t_{lh})$, we compute h piecewise-constant control inputs with one measurement $x(t_{(l-1)h})$, as can be seen in (15). Note that all the matrices and their inverses can be computed offline, as they do not depend on the current state $x(t_{(l-1)h})$. The inverse $G_{x,l}^{-1}$ always exists since $G_{x,l}$ has full rank, as $\bar{P}_{reach,l}$ is a parallelotope.

3.4 Further Extensions

During the presentation of the approach, we have focused on the main idea. Depending on the practical application

some additional goals or constraints might be relevant as well. In the following, we discuss four of them.

State Constraints: Since state constraints can be simply included as additional constraints to the linear program, they do not change the basic approach. In order to have finitely many constraints, we want to ensure the satisfaction of the state constraints for all time points despite checking them only at the sampling times. For linear systems, this is possible by using some tighter state constraints $\hat{\mathcal{X}} \subseteq \mathcal{X}$, which are computed to take the maximal curvature of trajectories between the sampling times into account, see e.g. Darup (2015). To check the satisfaction of the state constraints, we simply check if $\forall k \in \{1, \dots, N\}$: $\xi(x(0), u(\cdot), 0, t_k) \in \hat{\mathcal{X}}$ or $\mathcal{R}_{t_k, \mathcal{Z}_u}(\mathcal{X}_0) \subseteq \hat{\mathcal{X}}$, respectively. To check if the reachable set in the form of a zonotope $\mathcal{Z} =< c, G >$ is inside a polytope $\mathcal{P} = \{x | Cx \leq d\}$, we check if $Cc + \sum_{i=1}^{p} |Cg^{(i)}| \leq d$.

For nonlinear dynamics, we check the state constraints similarly. We compute the reachable set using the time-varying, linearized dynamics and check the satisfaction of the state constraints at sampling times again. For undisturbed, nonlinear systems, this can be done using the results from (Magni and Scattolini, 2004, Thm. 3). To ensure that the disturbed, nonlinear dynamics will satisfy the real state constraints, we have to tighten the state constraints even more. Since no general closed-formula exists for disturbed, nonlinear systems, we can use the tightened constraints based on the disturbance effects on the linearized dynamics as an initial guess. As we compute everything offline, we can check if the actual constraints are satisfied, and if not, adapt the tightened state constraints. By iterating this offline until all constraints are satisfied, we obtain a formally correct controller for the online use.

Input Costs: So far, we did not consider any input costs, since we are looking for a minimal reachable set which satisfies the input constraints. In the maneuver automata scenario, this maximizes the number of maneuvers which can follow a given maneuver. If the input costs do matter, we can take them into account by adding a term $\|\hat{c} + \alpha\|_1$ to the cost function (possibly with a weighting factor). Therein the vector $\hat{c}$ is used if $c_u \neq \mathbf{0}$ to translate the input costs, such that $\alpha_i = -\hat{c}_i$ corresponds to an input $u_i = 0$ and therefore has no costs.

Weighting Matrices: Depending on the physical meaning of the states, the size of the resulting reachable set in certain dimensions might be more critical than others. We can consider this by using a weighting matrix Q by which we multiply the cost function, analogously to the weighting matrices for LQR controllers.

Center Inputs: Depending on the dynamics and disturbances, it might be better in the nonlinear case to use the inputs computed for the reference trajectory directly as the inputs for the center. In this case, only the inputs for the generators are computed in each iteration step to counteract the disturbances. Therefore, we remove the first line of (12) but still consider the fixed $\alpha(c_x)$ in (13) to take the center inputs into account when optimizing the generator inputs.

4. NUMERICAL EXAMPLE

We demonstrate our new controller for an autonomous vehicle example. As a model, we choose a kinematic car model, which covers the most important dynamics of a car:

$$\dot{v} = a + w_1,\ \dot{\Psi} = b + w_2,\ \dot{x} = v\,cos(\Psi),\ \dot{y} = v\,sin(\Psi),$$

where the states v, Ψ, x, and y are the velocity, the orientation, and the positions in x and in y directions, respectively. The acceleration a and the normalized steering angle b are the inputs, and w_1 and w_2 are additive disturbances. They are constrained to lie in the intervals $a \in [-9.81, 9.81]\frac{m}{s^2}$, $b \in [-0.4, 0.4]\frac{rad}{s}$, $w_1 \in [-0.5, 0.5]\frac{m}{s^2}$, and $w_2 \in [-0.02, 0.02]\frac{rad}{s}$.

We construct a maneuver automaton for this car, where we restrict ourselves for space limitation to three maneuvers: "turn left", "drive straight", and "turn right". For all maneuvers, we consider the initial set $\mathcal{X}_0 = [19.8, 20.2]\frac{m}{s} \times [-0.02, 0.02]rad \times [-0.2, 0.2]m \times [-0.2, 0.2]m$. We want to steer all states from $\mathcal{X}_0$ after $1s$ as close as possible to the final states $x^{(f)} = \left[20\,\frac{m}{s}, \pm 0.2\,rad, 19.87\,m, \pm 1.99\,m\right]^T$, for the "turn left" and "turn right" maneuvers and $\left[20\,\frac{m}{s}, 0\,rad, 20\,m, 0\,m\right]^T$ for the "drive straight" maneuver. To be able to use this approach to obtain a maneuver automaton, the final set has to be completely contained in the initial set, see Fig. 1 and [Heß et al. (2014)]. Thereby, we can shift the final state in the Ψ, x, and y coordinates, since the car dynamics are independent of the absolute orientation and position. We divide the one second time horizon into $N = 40$ time steps and consider $M = 10$ time intervals of horizon $h = 4$ time steps. As described in Sec. 3.4, we apply the reference input $u^{(c)}(\cdot)$ directly as input for the center and also use weights for the states. While we do not consider input weights, the input constraints are still satisfied by the way we construct the controller.

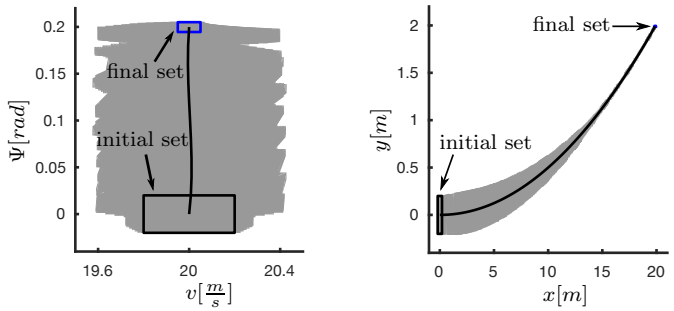
4.1 Results

We implement the nonlinear control approach in MATLAB, where we use CVX to specify and solve the linear program [Grant and Boyd (2014)]. The reachability analysis is computed using the CORA toolbox [Althoff (2015)]. The computations are performed on a computer with a 3.1 GHz dual-core i7 processor and 16 GB memory. The offline computation of the controller for each maneuver takes around ten seconds without using parallel computing. Since the online computation consists only of multiplying precomputed matrices with vectors, it can be performed very fast in around 0.01 milliseconds.

We show the resulting reachable sets in Fig. 7. In addition, we show the initial set (black) and the final reachable set (blue) in Fig. 8, where we shift the final sets by the desired final states $x^{(f)}$ to have a better comparison. We see that the (shifted) reachable sets are completely contained inside the initial set for all maneuvers. Therefore, we are able to connect all maneuvers with each other and obtain a fully connected maneuver automaton.

4.2 Comparison with LQR Tracking Controller

For comparison, we also implement an LQR tracking controller. To do so, we use the same reference trajectory and



(a) Turn Left, (v, Ψ) plane (b) Turn Left, (x, y) plane

Fig. 7. Resulting reachable set for the turnleft maneuver with our controller.

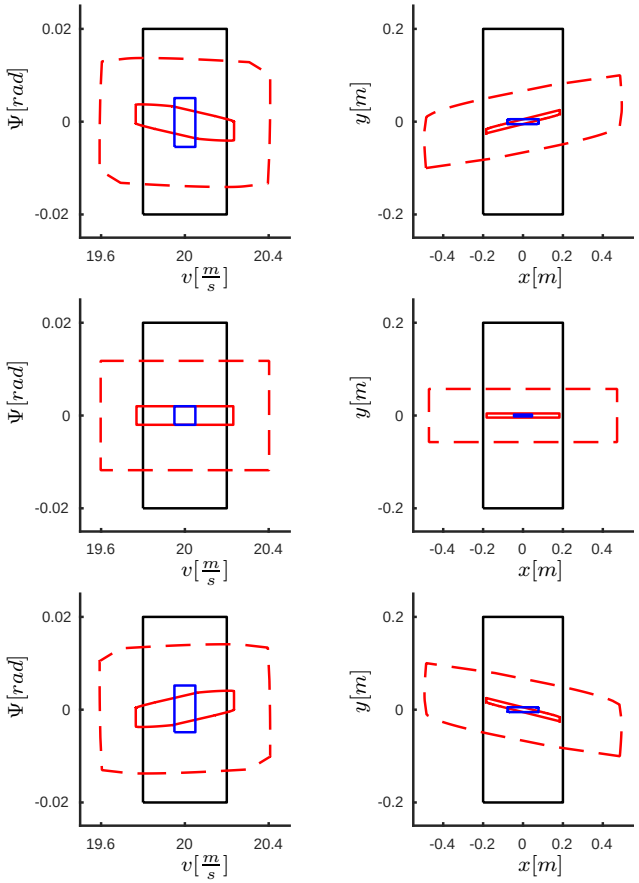


Fig. 8. Initial (black) and shifted final sets (blue) for our controller, projected to the (v, Ψ) and the (x, y) planes, for the "turn left" (top), "drive straight" (center), and "turn right" (bottom) maneuvers. For comparison the final sets of two LQR controllers (red).

linearizations at the sampling times as for our controller. For each linearization, we compute an LQR controller to track the system along the reference trajectory. First, we simply use the same state weights as for our controller and weight the inputs with the identity matrix. The resulting reachable set is plotted in Fig. 8 (red, solid), and it is not completely inside the initial set. Moreover, it uses inputs of $b = 1.53\frac{rad}{s}$, which is more than 3.5 times the allowed inputs. If we increase the input weights until the input constraints are satisfied, we obtain the reachable set which is plotted with a red, dashed line in Fig. 8, and which is much larger than the initial set. We see that we cannot achieve both a small final set and input satisfaction with LQR controllers. This shows the advantage of our new control approach, which minimizes the reachable set while ensuring the satisfaction of the constraints.

5. DISCUSSION OF THE ALGORITHM

Let us now discuss the complexity and optimality of our algorithm.

5.1 Complexity

Reachability Analysis: The reachability analysis from Althoff (2010) which we use in our algorithm has a complexity of $\mathcal{O}(n^3)$, where n denotes the number of states.

Linear Programs: The complexity of solving linear programs depends on the exact implementation and the number and type of constraints. There exist algorithms with polynomial time complexity in the number of optimization variables and constraints. If we consider zonotopes of a fixed order, then the number of optimization variables and number of constraints grows polynomial as well (with $\mathcal{O}(n^2)$, if $n > m$). Therefore, the whole optimization algorithm has a polynomial complexity.

Center Trajectory: The center trajectory is computed using nonlinear programming. Since this is a nonlinear and nonconvex optimization problem, one cannot give bounds on the computational complexity. However, there exist very efficient implementations using direct optimization which solve the problem fast and converge for many initial states [Betts (2010)]. Since we have to compute this center trajectory only a single time for a single state, and we are not dependent on an optimal solution, we can neglect the complexity of the center trajectory for the overall approach.

Overall Approach: If we neglect the computation of the center trajectory, then the overall complexity is polynomial as well. This is better than many other optimal control approaches which provide guarantees for sets of initial states and whose complexities are exponential with the number of states as discussed in the literature review.

5.2 Optimality

Linear programs are convex optimization problems, for which the solutions converge to the global optimal solution. Therefore, for linear systems, we obtain the global optimal solution under the restriction of piecewise-constant inputs. Since we solve the linear programs based on the linearized dynamics, we cannot guarantee global optimality for the nonlinear case. However, for disturbed, nonlinear systems, no efficient method exists which can guarantee global optimality [Schütze (2004); Bertsekas (2005)]. We are, however, able to provide a very efficient way to control all states based on the linearized dynamics in an optimized manner. Since we only have to consider the control inputs for n generators, we are able to obtain fast solutions. Moreover, we are able to compute the reachable set offline, and if the solution does not satisfy our expectations, then we can introduce weights for the different dimensions and reiterate the algorithm to find a better controller. We only apply the controller online after we are satisfied with the offline computations.

6. CONCLUSION

We have presented a novel approach to control all states of an initial set as close as possible to a desired final state. We are able to provide formal guarantees for constraint satisfaction even for disturbed nonlinear systems. The first novelty is that we are able to transform the optimal control problem into a search problem in generator space. Thus, we directly know the set of reachable states and optimize the solutions in this fixed set. By using zonotopes as an efficient set representation, we are able to leverage similar, efficient approaches for single initial states to a set of initial states. The second novelty is that we are able to obtain a robust feedback controller for disturbed nonlinear systems by iteratively applying the open-loop control approach in combination with reachability analysis, which leads to formal guarantees. As we see in the numerical example, we are able to compute the controllers very fast offline, and the approach scales polynomial with the number of dimensions. This is much better than most other approaches which control sets of solutions. Moreover, the online complexity is the same as for time-variant feedback matrices and therefore allows for fast sampling times.

ACKNOWLEDGEMENTS

The author gratefully acknowledges financial support from the European Commission project UnCoVerCPS under grant number 643921.

REFERENCES

Althoff, M. (2015). An introduction to CORA 2015. In *Proc. of the Workshop on Applied Verification for Continuous and Hybrid Systems*, 120–151.

Althoff, M. (2010). *Reachability Analysis and its Application to the Safety Assessment of Autonomous Cars*. Ph.D. thesis, Technische Universität München.

Asarin, E., Dang, T., Frehse, G., Girard, A., Le Guernic, C., and Maler, O. (2006). Recent progress in continuous and hybrid reachability analysis. In *Proc. of the IEEE Conference on Computer Aided Control Systems Design*, 1582–1587.

Bertsekas, D.P. (2005). *Dynamic Programming and Optimal Control*. Athena Scientific Belmont, MA, 3rd edition.

Betts, J.T. (2010). *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*. Siam.

Blanchini, F. and Miani, S. (2008). *Set-Theoretic Methods in Control*. Springer.

Borrelli, F., Bemporad, A., and Morari, M. (2015). *Predictive Control for linear and hybrid systems*. Cambridge University Press. URL `http://www.mpc.berkeley.edu/mpc-course-material`.

Dantzig, G.B. and Thapa, M.N. (2006). *Linear Programming 1: Introduction*. Springer.

Darup, M.S. (2015). Efficient constraint adaptation for sampled linear systems. In *IEEE Conference on Decision and Control*, 1402–1408.

Grant, M. and Boyd, S. (2014). CVX: Matlab software for disciplined convex programming, version 2.1. `http://cvxr.com/cvx`.

Grant, M., Boyd, S., and Ye, Y. (2006). Disciplined convex programming. In *Global Optimization: from Theory to Implementation*, 155–210. Springer.

Heß, D., Althoff, M., and Sattel, T. (2014). Formal verification of maneuver automata for parameterized motion primitives. In *Proc. of the International Conference on Intelligent Robots and Systems*, 1474–1481.

Julius, A.A. and Winn, A.K. (2012). Safety controller synthesis using human generated trajectories: Nonlinear dynamics with feedback linearization and differential flatness. In *Proc. of the American Control Conference*, 709–714.

Kloetzer, M. and Belta, C. (2008). A fully automated framework for control of linear systems from temporal logic specifications. *IEEE Transactions on Automatic Control*, 53(1), 287–297.

Kurzhanski, A.B., Mitchell, I.M., and Varaiya, P. (2006). Optimization techniques for state-constrained control and obstacle problems. *Journal of Optimization Theory and Applications*, 128(3), 499–521.

Liu, J., Topcu, U., Ozay, N., and Murray, R.M. (2012). Reactive controllers for differentially flat systems with temporal logic constraints. In *Proc. of the Conference on Decision and Control*, 7664–7670.

Lygeros, J., Tomlin, C., and Sastry, S. (1999). Controllers for reachability specifications for hybrid systems. *Automatica*, 35(3), 349–370.

Magni, L. and Scattolini, R. (2004). Model predictive control of continuous-time nonlinear systems with piecewise constant control. *IEEE Transactions on Automatic Control*, 49(6), 900–906.

Majumdar, A. and Tedrake, R. (2013). Robust online motion planning with regions of finite time invariance. In *Algorithmic Foundations of Robotics X*, 543–558. Springer.

Mayne, D.Q., Seron, M.M., and Raković, S.V. (2005). Robust model predictive control of constrained linear systems with bounded disturbances. *Automatica*, 41(2), 219 – 224.

Platzer, A. and Clarke, E.M. (2007). The image computation problem in hybrid systems model checking. In *International Workshop on Hybrid Systems: Computation and Control*, 473–486.

Rakovic, S.V., Kouvaritakis, B., Cannon, M., Panos, C., and Findeisen, R. (2012). Parameterized tube model predictive control. *IEEE Transactions on Automatic Control*, 57(11), 2746–2761.

Schürmann, B. and Althoff, M. (2017a). Convex interpolation control with formal guarantees for disturbed and constrained nonlinear systems. In *Proc. of Hybrid Systems: Computation and Control*.

Schürmann, B. and Althoff, M. (2017b). Optimal control of sets of solutions to formally guarantee constraints of disturbed linear systems. In *Proc. of the American Control Conference*.

Schütze, O. (2004). *Set Oriented Methods for Global Optimization*. Ph.D. thesis, Univ. Paderborn.

Tedrake, R., Manchester, I.R., Tobenkin, M., and Roberts, J.W. (2010). LQR-trees: Feedback motion planning via sums-of-squares verification. *The International Journal of Robotics Research*, 29(8), 1038–1052.

Zamani, M., Pola, G., Mazo Jr., M., and Tabuada, P. (2012). Symbolic models for nonlinear control systems without stability assumptions. *IEEE Transactions on Automatic Control*, 57(7), 1804–1809.