OPTIMIZATION-BASED SUBDIVISION ALGORITHM FOR REACHABLE SETS

WOLFGANG RIEDL, ROBERT BAIER, AND MATTHIAS GERDTS

Abstract. Reachable sets for nonlinear control systems can be computed via the use of solvers for optimal control problems. The paper presents a new improved variant which applies adaptive concepts similar to the framework of known subdivision techniques by Dellnitz/Hohmann. Using set properties of the nearest point projection, the convergence and rigorousness of the algorithm can be proved without the assumption of diffeomorphism on a nonlinear mapping. The adaptive method is demonstrated by two nonlinear academic examples and for a more complex robot model with box constraints for four states, two controls and five boundary conditions. In these examples adaptive and non-adaptive techniques as well as various discretization methods and optimization solvers are compared. The method also offers interesting features, like zooming into details of the reachable set, self-determination of the needed bounding box, easy parallelization and the use of different grid geometries. With the calculation of a 3d funnel in one of the examples, it is shown that the algorithm can also be used to approximate higher dimensional reachable sets and the resulting box collection may serve as a starting point for more sophisticated visualizations or algorithms.

1. Introduction and preliminaries

Reachable sets (also named attainable sets or capture basins) being the set of end points of feasible trajectories of a nonlinear control problem or a differential inclusion appear in many theoretical works and practical applications. Studies on robustness, uncertain problems, switched systems or those with state discontinuities are based on reachable sets (see e.g., [22, 8, 2, 23]).

Applications of reachable sets range from the calculation of safety regions of cars or aircrafts ([32, 1, 20, 43, 4]), the design of discrete controllers for nonlinear plants [35], maneuvers in space by satellites and multi-boost launchers as in [19, 29, 15, 12] or models on climate change due to greenhouse gas emission [14, 3].

There is a rather big variety of methods, see e.g., the references in [7] giving an overview on methods applying ellipsoids, polytopes, zonotopes, boxes or applying the viability kernel algorithm, PDE solvers for the Hamilton-Jacobi-Bellman equation resp. level sets of reachable sets or support functions for the linear case.

Date: 20 December 2016.

2010 *Mathematics Subject Classification.* 93B03 49M37 (49M25 49J53 93C10).

Key words and phrases. reachable sets, subdivision, optimal control, direct discretization, nonlinear systems, nonlinear optimization.

We will focus on the method based on solvers for optimal control problems introduced in [6, 7] and further developed in [27, 29, 36, 34]. This method has advantages for those who are already regular users or experts in the application of optimizers which can be easily applied for these set-valued problems by calculating a series of suitable parametric optimization problems. In comparison to other approaches, that focus on the different use of optimizers, e.g. SQP methods [6, 7], interior point methods [27, 29] or approximation methods, based on support vector machines and reproducing kernel Hilbert spaces [34], we will present an adaptive technique which leads to a big performance win. This paper develops several ideas further on, which were already briefly presented in [36], especially the embedding in the class of subdivision methods is new.

The contents of the paper is as follows: In the remaining part of this section, we introduce the basic notions, define reachable sets and give a brief overview of the non-adaptive algorithm on which the idea was based.

In Section 2 we demonstrate, how the mentioned algorithm can be embedded into a subdivision framework. We will introduce a projection $\mathcal{P}$ that maps a set of points to their best approximations in the reachable set. The subdivision algorithm is also presented in this section and its convergence is proved.

Section 3 gives an overview of the implementation of the algorithm and the relation between the previously introduced map $\mathcal{P}$ and the end points of the optimal trajectories for the corresponding optimal control problems which are calculated by a chosen optimization solver after a suitable discretization. The flow of the subdivision algorithm is demonstrated on one of the examples and we will briefly discuss ways to make the algorithm more efficient and robust.

In Section 4, we discuss a selection of nonlinear control problems that were implemented to test the algorithm. We start with a simple bilinear model to demonstrate the efficiency of the subdivision algorithm. The model is very fast to solve and allows us to generate the reachable set even with a rather high number of grid points and compare both algorithms (with and without subdivision) in a justifiable amount of time.

The following Kenderov problem is a much more challenging problem to solve and the computational times are much longer than those in the bilinear problem, even with the adaptive version. The example demonstrates nicely, how the chosen Runge-Kutta discretization methods influence the quality of the generated approximation.

The third problem we introduce is the model of a planar industrial robot. Using this model, we demonstrate that the algorithm can handle problems with higher dimensional dynamics and is a viable choice for scenarios, where we want to calculate the reachable set only for some state variables in the dynamics of the system, or the reachable set of states, that are not directly part of the dynamics of a system, but are the result of a nonlinear transformation of some of the states. We show a direct approach to approximate the reachable set of the transformed problem and compare it to an indirect approach, which first generates the reachable set of the states, involved in the transformation. We will also use this example to briefly compare the results, generated by WORPH and Ipopt [13, 41].

Section 5 is a collection of features which the introduced method offers to generate reachable sets. We will show the use of different grid geometries, the zooming into interesting subsets of the reachable set, the automatic generation of a bounding box and the construction of a 3d solution funnel on some examples.

As we have seen in this short summary, we first have to introduce *reachable sets* and briefly have to talk about control problems first.

Our goal is to calculate the *reachable set* of a *parametric nonlinear control system (CP) with constraints* of the form

$$\begin{aligned} \dot{x}(t) &= f(t, x(t), u(t), p), \\ x(t_0) &\in X_0, \\ u(t) &\in U, \\ p_l &\leq p \leq p_u, \\ g_l &\leq g(t, x(t), u(t), p) \leq g_u, \\ \psi_l &\leq \psi(x(t_0), x(T), p) \leq \psi_u, \end{aligned}$$

for every $t \in [t_0, T]$, where $x(t) \in \mathbb{R}^n$ are the states and $u(t) \in U \subset \mathbb{R}^{n_u}$ are the controls of the system at time t. Furthermore we have state and control independent parameters $p \in \mathbb{R}^{n_p}$, the dynamics of the system given by the function f, nonlinear constraints on states and controls, modelled by the function g and boundary conditions given by the function ψ. All inequalities in the constraints of (CP) have to be understood componentwise. The parameter p and the corresponding constraints on p are used e.g., to simulate a free end time, but might also not be present in the considered control system.

Definition 1.1. The *reachable set* (sometimes also called *attainable set*) of a state-constrained nonlinear control system at a given end time T is defined as the set of the end points of all feasible trajectories of the control system and can be written as

$$R(T) := \{x(T) \in \mathbb{R}^n \,|\, (x, u, p) \text{ is a feasible solution of (CP) } \}.$$

The results gathered e.g., in [38, 16, 42], [14, Theorem 2.1.4] and [7, Proposition 3.2], state various mild conditions under which the reachable set is compact, nonempty and connected. Key assumptions are e.g. nonemptiness and compactness of the images of the right-hand side of the differential inclusion, the linear growth condition and local Lipschitzness of the right-hand side.

In this paper we use the approach shown in [7] by constructing the following *parametric optimal control problem* (OCP):

Minimize

$$\frac{1}{2}\|x(T) - g_\rho\|_2^2$$

subject to

$$\begin{aligned}\dot{x}(t) &= f(t, x(t), u(t), p),\\ x(t_0) &\in X_0,\\ u(t) &\in U,\\ p_l &\le p \le p_u,\\ g_l &\le g(t, x(t), u(t), p) \le g_u,\\ \psi_l &\le \psi(x(t_0), x(T), p) \le \psi_u\end{aligned}$$

for $t \in [t_0, T]$ which has to be solved for every point $g_\rho \in G$, where G is a grid in the state space of the system and ρ is a parameter indicating the mesh size. The objective function involves the Euclidean norm $\|\cdot\|_2$ for vectors in $\mathbb{R}^n$. The grid point g_ρ can be seen as an additional parameter besides p.

To solve this system we have to discretize the ODE with our system dynamics (e.g., using a Runge-Kutta method with grid points $t_i = t_0 + ih$ for $i = 1, 2, ..., N$, $N \in \mathbb{N}$ and stepsize $h = \frac{T-t_0}{N}$), leading to the following discrete optimal control problem $(\mathrm{OCP})_h$:

Minimize

$$\frac{1}{2}\|x_N - g_\rho\|_2^2$$

subject to

$$\begin{aligned}x_{i+1} &= x_i + h \cdot \Phi(t_i, x_i, x_{i+1}, u_i, u_{i+1}, p; h),\ \ i = 0, 1, \ldots, N-1,\\ x_0 &\in X_0,\\ u_i &\in U,\ \ i = 0, 1, \ldots, N,\\ p_l &\le p \le p_u,\\ g_l &\le g(t_i, x_i, u_i, p) \le g_u,\ \ i = 0, 1, \ldots, N,\\ \psi_l &\le \psi(x_0, x_N, p) \le \psi_u.\end{aligned}$$

Here, $\Phi(\cdot, \cdot, \cdot, \cdot, \cdot, \cdot; h)$ denotes the characterizing function of the discretization method.

Definition 1.2. The *discrete reachable set* of the aforementioned discrete optimal control problem can be written as

$$R_h(T) := \{x_N \in \mathbb{R}^n \,|\, ((x_i)_{i=0,\ldots,N}, (u_i)_{i=0,\ldots,N}, p) \text{ is a feasible solution of } (\mathrm{OCP})_h\}.$$

We can show for some cases, that the discrete reachable set converges to the reachable set (in the case of nonlinear control systems see e.g., [21, 40, 14, 9, 10] for set-valued Runge-Kutta methods and the discussion in [6] as well as [5] for state constrained differential inclusions) which makes algorithms calculating discrete reachable sets valuable tools for the approximation of the continuous reachable sets.

The aproaches in [14, 5, 9, 10, 6, 7, 29] used an *equidistant* grid G_ρ in the state space. The initial bounding box in the algorithm is usually an n-dimensional cuboid and can be discretized as

$$G_\rho := \left\{g_{\boldsymbol{\alpha}} = g^{(l)} + (\alpha_1 \cdot \rho_1, \ldots, \alpha_n \cdot \rho_n)^T \,|\, \alpha_i \in \{0, ..., N_i\} \subset \mathbb{N}_0,\ i = 1, ..., n\right\}$$

with the lower left corner $g^{(l)} \in \mathbb{R}^n$ and constant step sizes $\rho_1, ..., \rho_n \in \mathbb{R}$. The distance ρ between the grid points is coupled with the stepsize h and the order of the method Φ used to discretize the ODE with the system dynamics (see [14, 9, 7]). Using this algorithm to calculate an approximation for the discrete reachable set of the bilinear problem (Ex. 4.1) with a rather course grid, we get the result shown in Fig. 1.

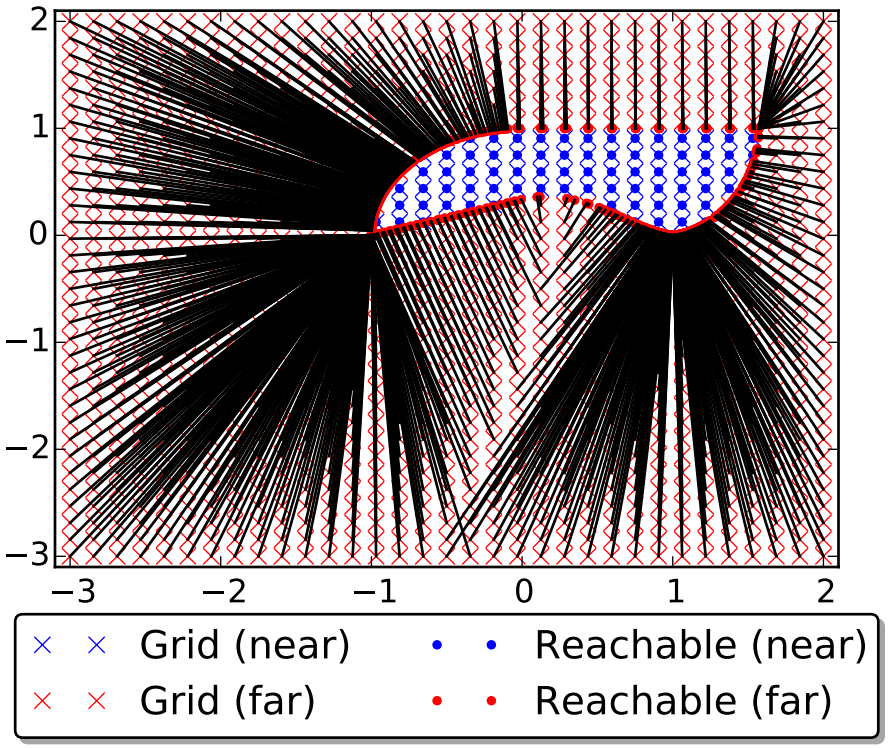

FIGURE 1. Reachable set of the bilinear problem for a full 33×33 grid.

Crosses indicate our used grid points, whereas blue and red dots will show the endpoint of the optimal trajectory from the OCP (also called *target point*). As a side effect of the algorithm, we can easily categorize, if a grid point is far away from the set or if it is inside or at least near to the set. The color at the grid point g_ρ and target point z (i.e. the end point $x(T)$ of the optimal solution generated from the OCP with parameter g_ρ) depends on whether $\text{dist}(g_\rho, z) > \varepsilon$ (red) or $\text{dist}(g_\rho, z) \leq \varepsilon$ (blue), for some chosen $\varepsilon > 0$. Therefore we can construct a rough approximation of the interior of the reachable set and the boundary of the reachable set for free by seperating these two cases. The black lines in the plots connect every grid point that is outside the reachable set to its corresponding target point from the OCP. All plots in this paper were generated using the Python library Matplotlib [28].

As we can see in Table 1 this algorithm leads to a lot of grid points the finer the grid gets. The number of grid points also determines the number of optimization problems we have to solve and has a huge impact on the performance of the algorithm with the actual optimization process being the most expensive part. The

grid	number of grid points
3×3	9
5×5	25
9×9	81
17×17	289
33×33	1089
65×65	4225
129×129	16641
257×257	66049

TABLE 1. Number of grid points for a few different grids.

following section 2 will now introduce the idea for an extension of this algorithm to reduce the number of needed grid points by using a *subdivision algorithm* similar to [17], [26, Secs. 6.3 and 7.5]. Using this algorithm we can generate the reachable set with much less grid points far away from the reachable set. The left image in Fig. 2 shows the non-adaptive version with an equidistant 129×129 grid and the right image the result created by seven steps of the subdivision algorithm. Both algorithms approximate the interior of the reachable set with the same density of grid points, but we can see that the subdivision algorithm chooses the density according to the distance of the grid point to the reachable set.

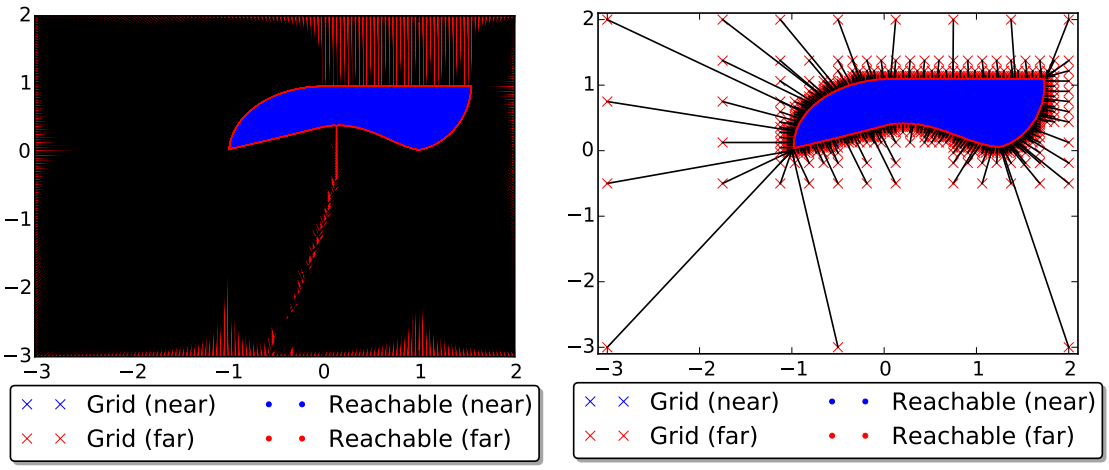

FIGURE 2. Reachable set of the bilinear problem, generated with (right) and without (left) the subdivision algorithm.

Furthermore we will prove convergence of this variant of the subdivision algorithm.

2. GRID CONSTRUCTION VIA SUBDIVISION

The main goal of this algorithm is to reduce the number of grid points needed to calculate the reachable set. As seen in Fig. 1 and the left image in Fig. 2 the

grid points far away from the reachable set are mapped to the boundary of the set. On the one hand this creates a nice visualization which grid point is mapped to which target point, but we waste a lot of time creating this dense outline when we are only interested to get points of the set with a certain distance of each other like in the interior of the shown sets.

First of all we will introduce a map $\mathcal{P}$ by the following definition.

Definition 2.1. Let $R \subset \mathbb{R}^n$ be the closed and nonempty reachable set of (CP) and $G \subset \mathbb{R}^n$ be a compact set. We define the map $\mathcal{P}$ as a *projection* of G onto R, i.e. every point $g \in G$ is mapped to one of its nearest points of the reachable set R:

$$\mathcal{P}(G) := \bigcup_{g \in G} \{\pi_R(g)\}, \quad \pi_R(g) \in \Pi_R(g) := \operatorname*{argmin}_{z \in R} \operatorname{dist}(g, z),$$

where $\operatorname{dist}(g, z) = \|g - z\|_2$ is the distance function. The set-valued map Π_R coincides with the optimal-set mapping defined in [37, Example 5.22]. We know especially

$$\mathcal{P}(\emptyset) = \emptyset.$$

Using the previously defined map $\mathcal{P}$ we can show some important properties that are needed to formulate the subdivision algorithm and to prove the convergence.

Lemma 2.2. *Let R, G and $\mathcal{P}$ be as in Definition 2.1 and assume that R is closed. Then we can show that*

(1) $\mathcal{P}(G) \subset R \ \forall G \subset \mathbb{R}^n$,
(2) $\mathcal{P}(G) = \emptyset \Leftrightarrow G = \emptyset$,
(3) $\mathcal{P}(G \cap R) = G \cap R$,
(4) $\mathcal{P}(G) \cap G = G \cap R$ *and especially* $G \cap R \neq \emptyset \Leftrightarrow \mathcal{P}(G) \cap G \neq \emptyset$.
(5) $G \subset R \Leftrightarrow \mathcal{P}(G) = G$ *and especially* $\mathcal{P}(R) = R$, *if R is compact.*

Proof.

(1) The first part follows directly from the definition of the map $\mathcal{P}$:
$\operatorname*{argmin}_{r \in R} \operatorname{dist}(g, r) \subset R \ \forall g \in G \neq \emptyset$ and therefore $\pi_R(g) \in R$ for all $\pi_R(g)$ from Definition 2.1. This leads to

$$\mathcal{P}(G) = \bigcup_{g \in G} \{\pi_R(g)\} \subset R. \tag{1}$$

$G = \emptyset$ results in $\mathcal{P}(G) = \emptyset$ and (1) is obviously true in that case.

(2) For the second part, we use $\mathcal{P}(\emptyset) = \emptyset$ from the definition of the map, and only have to prove, that $\mathcal{P}(G) = \emptyset$ leads to $G = \emptyset$. The assumption $G \neq \emptyset$ results in $\Pi_R(g) = \operatorname*{argmin}_{z \in R} \operatorname{dist}(g, z) \neq \emptyset$ since $R \neq \emptyset$ by definition. Therefore exists a $\pi_R(g) \in \Pi_R(g)$ and $\mathcal{P}(G) = \bigcup_{g \in G} \{\pi_R(g)\} \neq \emptyset$.

(3) To prove the third equality $\mathcal{P}(G \cap R) = G \cap R$ we have to look at the following two cases:
- $G \cap R = \emptyset$: From the definition of $\mathcal{P}$ it follows trivially $\mathcal{P}(G \cap R) = \emptyset$.

- $G \cap R \neq \emptyset$: We can show that for every $g \in G \cap R$

$$\operatorname*{argmin}_{r \in R} \operatorname{dist}(g, r) = \{g\},$$

since $\operatorname{dist}(g, r) > 0$ for every $r \neq g$, but $\operatorname{dist}(g, g) = 0$. Therefore

$$\mathcal{P}(G \cap R) = \bigcup_{g \in G \cap R} \{g\} = G \cap R.$$

(4) For the proof of the equality $\mathcal{P}(G) \cap G = G \cap R$, we also have to split it up:

- Assume $R \cap G \neq \emptyset$: From the first part of the lemma we already know that $\mathcal{P}(G) \subset R$ and therefore

$$\mathcal{P}(G) \cap G \subset R \cap G.$$

To show that $R \cap G \subset \mathcal{P}(G) \cap G$ we assume, that we have a $z \in R \cap G$ with $z \notin \mathcal{P}(G) \cap G$.

$$z \in R \cap G \Rightarrow \operatorname*{argmin}_{\tilde{z} \in R} \operatorname{dist}(z, \tilde{z}) = \{z\}$$

which leads to $\pi_R(z) = z$ and results in

$$z \in \bigcup_{g \in G} \{\pi_R(g)\} = \mathcal{P}(G).$$

By assumption $z \in G$ holds and therefore

$$z \in \mathcal{P}(G) \cap G,$$

which is a contradiction. Now we can conclude

$$R \cap G \subset \mathcal{P}(G) \cap G,$$

$$\Rightarrow \mathcal{P}(G) \cap G = R \cap G$$

- $R \cap G = \emptyset$: We know from the first part of the lemma that $\mathcal{P}(G) \subset R$. Assume there exists $r \in \mathcal{P}(G) \cap G$

$$\Rightarrow r \in R \text{ and } r \in G.$$

But then we have found an element $r \in R \cap G$ which is a contradiction to our assumtion that $R \cap G = \emptyset$.

$$\Rightarrow \mathcal{P}(G) \cap G = \emptyset.$$

We have seen, that

$$\mathcal{P}(G) \cap G = R \cap G,$$

which naturally includes

$$R \cap G \neq \emptyset \iff \mathcal{P}(G) \cap G \neq \emptyset.$$

(5) The last part of the lemma can be easily shown by using some of the previous results.

We have already seen in 3. that $\mathcal{P}(G \cap R) = G \cap R$. If $G \subset R$ then

$$G \cap R = G$$

and

$$\mathcal{P}(G) = \mathcal{P}(G \cap R) = G \cap R = G.$$

We have also shown in 4. that $\mathcal{P}(G) \cap G = G \cap R$. If $\mathcal{P}(G) = G$ then

$$\mathcal{P}(G) \cap G = G$$

and therefore

$$G \cap R = \mathcal{P}(G) \cap G = G \Rightarrow G \subset R.$$

The equality $\mathcal{P}(R) = R$ is trivial, since we may choose $G = R$.

□

Now we can use the projection from Definition 2.1 to introduce our subdivision algorithm using a similar approach as in, e.g., [17, 18] and [26, Secs. 6.3 and 7.5]. The goal of the algorithm is to generate a sequence $\mathcal{B}_0, \mathcal{B}_1, \ldots$ of finite, nonempty collections of compact sets with shrinking diameter, which converges to the reachable set R, i.e.,

$$\operatorname{diam}(\mathcal{B}_k) = \max_{B \in \mathcal{B}_k} \operatorname{diam}(B) \to 0 \quad \text{for } k \to \infty,$$

$$\bigcup_{B \in \mathcal{B}_k} B \to R \quad \text{for } k \to \infty.$$

Algorithm 2.3. Subdivision algorithm for reachable sets.

Initialize the algorithm with a *bounding set* $B_0 \supset R$, the collection of sets $\mathcal{B}_0 = \{B_0\}$ and set $k = 1$.

Subdivision: Construct a new collection of sets $\mathcal{S}_k$ such that

$$\bigcup_{S \in \mathcal{S}_k} S = \bigcup_{B \in \mathcal{B}_{k-1}} B$$

and

$$\operatorname{diam}(\mathcal{S}_k) = \theta_k \cdot \operatorname{diam}(\mathcal{B}_{k-1}),$$

where $0 < \theta_{\min} \leq \theta_k \leq \theta_{\max} < 1$.

Selection: Define a new collection of sets $\mathcal{B}_k$ by

$$\mathcal{B}_k = \{B \in \mathcal{S}_k : \mathcal{P}(B) \cap B \neq \emptyset\} \tag{2}$$

Increase k by 1 and go to *Subdivision*.

The only difference between this subdivision algorithm and the algorithm in [17] is the selection step. In practical applications, we usually use box collections since it is very easy to decide if a point is inside a box or not, but other variants of set collections are also possible.

Remark 2.4. It can be shown, that the selection step

$$\widetilde{\mathcal{B}}_k = \{B \in \mathcal{S}_k : \exists \hat{B} \in \mathcal{B}_{k-1}, \exists g \in \hat{B} \text{ such that } \pi_R(g) \in B\},$$

that was used in [36], is equivalent to the selection criterium (2).

Remark 2.5. The algorithm assumes, that the reachable set is compact and therefore can be bounded by a compact initial bounding set $B_0 \supset R$. In case of a closed but not bounded reachable set R, the algorithm still can be used, but only to approximate the compact set $R \cap B_0$. The convergence of the algorithm, shown in this section, still holds, which can be proved by substituting every R with $R \cap B_0$. The case $R \cap B_0 = \emptyset$ is trivial, since it yields an empty box collection after the first selection step and we can stop the algorithm at that point and return $\emptyset$ as the result. To avoid confusion with the notation when dealing with those special cases, we only look at the case of a compact reachable set R and a bounding box B_0 that completely covers R here.

Proving the convergence of the subdivision algorithm for reachable sets is in some cases similar to the proof seen in [17], since our reachable set R plays a similar role as the relative global attractor A_Q in [17], but also differs in some parts. One of the most important differences is that the map in aforementioned paper had to be a diffeomorphism and the projection $\mathcal{P}$ we are using here has in general no inverse. We start with a proposition similar to [17, Lemma 3.2]:

Proposition 2.6. *Let R be the reachable set and let $\mathcal{B}_0$ be a finite, nonempty collection of closed subsets with*

$$R \subset Q_0 := \bigcup_{B \in \mathcal{B}_0} B.$$

Then the sets

$$Q_k := \bigcup_{B \in \mathcal{B}_k} B$$

generated by the subdivision algorithm contain the reachable set R.

Proof. We know by definition, that $R \subset Q_0$. Assume there exists $r \in R \subset Q_{k-1}$ with $r \notin Q_k$ for $k > 0$. Then we can find a set $B \in \mathcal{S}_k$ that contains r and this set B is removed in the selection step of the subdivision algorithm, i.e.

$$\mathcal{P}(B) \cap B = \emptyset.$$

But this is a contradiction to Lemma 2.2 (4) due to $r \in R \cap B$ and therefore

$$R \subset Q_k.$$

□

The implication "$B \subset \mathcal{P}(B) \Rightarrow B \subset R$" from [17, Lemma 3.3] is trivial for Algorithm 2.3.

Since the subdivision algorithm constructs a nested sequence of compact sets $Q_k \subset Q_{k-1}$ we know that for every $l > 0$

$$Q_l = \bigcap_{k=0}^{l} Q_k$$

and we define the limit for the set by

$$Q_\infty := \bigcap_{k=0}^{\infty} Q_k.$$

The limit set with respect to the Hausdorff distance exists by [37, Exercise 4.3 and Example 4.13].

Our goal is to show that $R = Q_\infty$. The strategy shown in [17] would require the existence of $\mathcal{P}^{-1}$ (and especially $\pi_R(g)^{-1}$) and cannot be used for our case. The following theorems will show an alternative route to prove the convergence.

Theorem 2.7. *Let R be the reachable set, $k \in \mathbb{N}_0$ and Q_k the compact set generated by the subdivision algorithm after k steps. Then*

$$Q_k \subset R + \operatorname{diam}(\mathcal{B}_k) \cdot B_1(0),$$

where $\operatorname{diam}(\mathcal{B}_k) \cdot B_1(0)$ is a ball with radius $\operatorname{diam}(\mathcal{B}_k)$ around the origin and $R + \operatorname{diam}(\mathcal{B}_k) \cdot B_1(0)$ is the Minkowski sum of the reachable set and this ball.

Proof. Take $x \in Q_k$. We know from the selection step of the subdivision algorithm, that we can find a $B \in \mathcal{B}_k$ and $x \in B$ with

$$\mathcal{P}(B) \cap B \neq \emptyset \overset{\text{Lemma } 2.2}{\Rightarrow} R \cap B \neq \emptyset.$$

Applying Lemma 2.2 (3),

$$\operatorname{dist}(x, R) \leq \operatorname{dist}(x, R \cap B) = \operatorname{dist}(x, \mathcal{P}(B) \cap B) \leq \max_{y \in B} \operatorname{dist}(x, y) \leq \operatorname{diam}(B)$$

holds and with $\operatorname{diam}(B) \leq \operatorname{diam}(\mathcal{B}_k)$, the assertion easily follows.

□

This result can now be used to prove the convergence of Algorithm 2.3.

Theorem 2.8. *Let R be the reachable set and $(Q_k)_{k \in \mathbb{N}_0}$ the sequence of compact sets generated by the subdivision algorithm. Then*

$$Q_\infty = \bigcap_{k=0}^{\infty} Q_k = R$$

Proof. In the subdivision step of the algorithm we constructed our new box collections with

$$\operatorname{diam}(\mathcal{S}_k) = \theta_k \cdot \operatorname{diam}(\mathcal{B}_{k-1})$$

and

$$0 < \theta_{\min} \leq \theta_k \leq \theta_{\max} < 1$$

for all $k \in \mathbb{N}$. With $\theta = \max_{k\in\mathbb{N}} \theta_k$, we can construct an estimate for the diameter, since $0 < \theta < 1$ and

$$\operatorname{diam}(\mathcal{B}_k) \leq \theta^k \cdot \operatorname{diam}(\mathcal{B}_0) \xrightarrow{k\to\infty} 0. \tag{3}$$

Using the result of Theorem 2.7 we get

$$Q_\infty = \bigcap_{k=0}^{\infty} Q_k \subset \bigcap_{k=0}^{\infty} (R + \operatorname{diam}(\mathcal{B}_k) \cdot B_1(0))$$

and we can show, that $R = \bigcap_{k=0}^{\infty} (R + \operatorname{diam}(\mathcal{B}_k) \cdot B_1(0))$. Although this last equation seems obvious, we couldn't find a reference and therefore provide a short proof:

"$\subset$" Let $r \in R$ be a point of the reachable set, then $r \in R + \operatorname{diam}(\mathcal{B}_k) \cdot B_1(0)$ for every $k \in \mathbb{N}_0$ and therefore

$$r \in \bigcap_{k=0}^{\infty} (R + \operatorname{diam}(\mathcal{B}_k) \cdot B_1(0)) .$$

"$\supset$" Since R is closed, we have $\operatorname{dist}(r, R) = \delta > 0$ for every $r \notin R$, i.e.,

$$r \in R + \delta \cdot B_1(0).$$

Using the convergence of the diameter in (3) we can find a $k_0 \in \mathbb{N}_0$ with $\operatorname{diam}(\mathcal{B}_k) \leq \frac{\delta}{2}$ for every $k \geq k_0$. Due to the minimality of δ we know, that $r \notin R + \operatorname{diam}(\mathcal{B}_k) \cdot B_1(0)$ for all $k \geq k_0$, hence, r is not an element of $\bigcap_{k=0}^{\infty} (R + \operatorname{diam}(\mathcal{B}_k) \cdot B_1(0))$.

Therefore $R = Q_\infty$. □

In conclusion, we have seen in this section that the set collections, generated by the subdivision algorithm, converge to the reachable set. We also have an a priori estimate, how many steps of the algorithm are needed to achieve a given accuracy of the representation of the set by the box collection.

3. Implementation

To implement the subdivision-algorithm we need to create an approximation of the map $\mathcal{P}$ defined in the previous section of the paper. Although the subdivision algorithm allows the usage of a collection of sets with many different shapes, we use boxes in our implementation, since the computational effort to decide, if a given point is in the interior of a box or not, is very low. We use the approach in [6, 7] mentioned in the first section on a course grid on the elements of the box collections, generated by the subdivision algorithm.

To demonstrate the algorithm, we can take a look at the first few steps of generating the reachable set of one specific nonlinear example, the bilinear problem 4.1 shown in more detail in Sec. 4, where we collect all of the examples. To initialize

the subdivision algorithm we start with the box $[-3,2]^2 \subset \mathbb{R}^2$ and subdivide it into four smaller boxes as shown in Fig. 3 (i.e., the factor θ_k from the subdivision algorithm above is set here to 0.5 for every step). Now, we approximate the map $\mathcal{P}$ using only the corners of the smaller boxes, i.e., we have to solve nine OCPs using a framework, that can handle problems of the structure shown in Sec. 1 (in our case, we used Ipopt [41]).

Now we can take the end points of the optimal trajectories, provided by the optimizer to realize our selection step. We subdivide every box that contains at least one target point (in our case the orange boxes $[-3,0.5] \times [-0.5,2]$ and $[-0.5,2]^2$ were chosen for further subdivision in the selection step) and skip all the others (i.e., in our example we drop the two lower boxes $[-3,-0.5]^2$ and $[-0.5,2] \times [-3,-0.5]$).

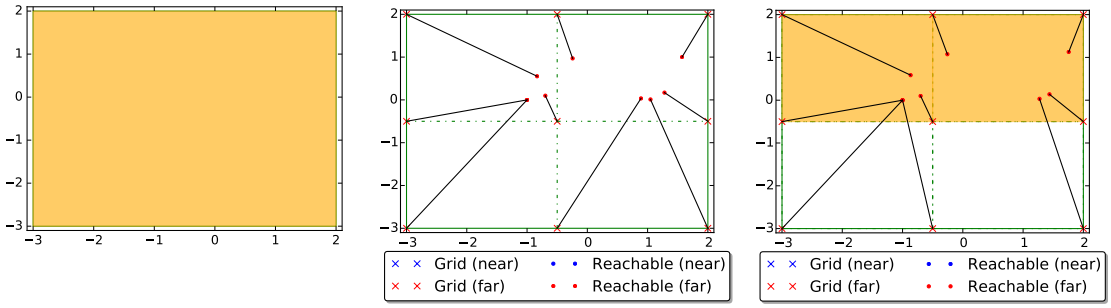

FIGURE 3. First step of the subdivision algorithm. Choose an initial bounding box (left), solve the optimization problems to approximate the map $\mathcal{P}$ (middle) and select the boxes for the next step (right).

Fig. 4 illustrates the further steps of the algorithm until we reach the desired density of grid points around the reachable set (here the distance between neighboring grid points is ≈ 0.0391 which is equivalent to what can be achieved by using a full 129×129 grid) seen in Fig. 5.

Fig. 5 visualizes both approximations of the reachable set that were generated by the subdivision algorithm, i.e., an inner approximation using the target points (left) and the outer approximation using the box collection Q_7 (right).

For solving the optimal control problems associated to the presented algorithm, we tested OCPID-DAE1 [24] (used in the computations in [36]) and, after direct discretization of the optimal control problem, the solution of the discrete nonlinear programming problem by local optimizers as Ipopt [41] and WORHP [13] as well as by the global methods MCS and PSO from The NAG FORTRAN library ([39]). Ipopt, based on interior-point algorithms, and WORHP, based on sparse sequential quadratic programming techniques, are designed for large-scale optimization problems. MCS uses a multilevel coordinate search based on a local optimizer, whereas PSO is a particle swarm optimizer.

Numerical experiments [11] showed that the use of local optimizers was sufficient to provide good approximations for our examples.

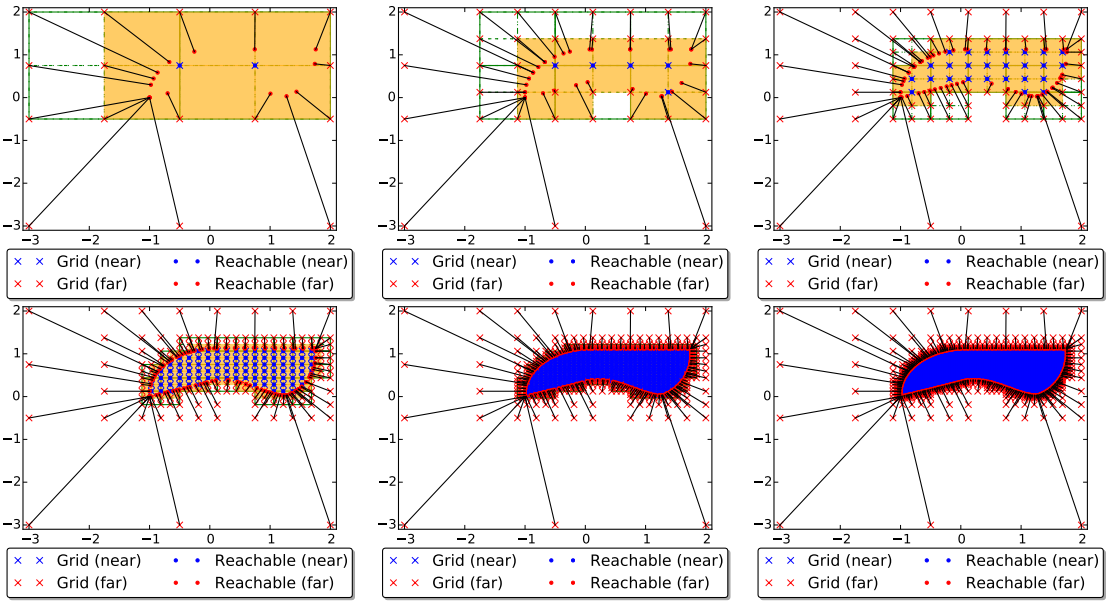

FIGURE 4. Selection of boxes for the next subdivision steps.

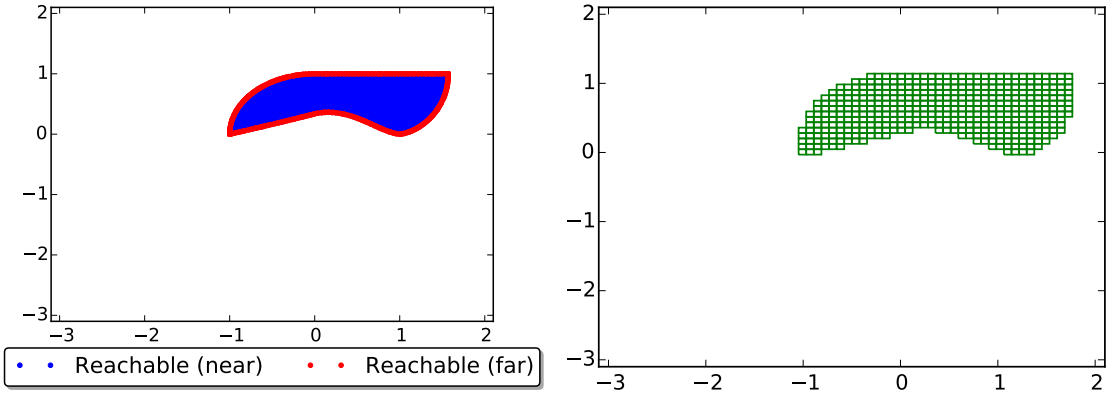

FIGURE 5. Inner and outer approximation of the reachable set

Remark 3.1. It has to be noted that a naive implementation of the subdivision step requires to evaluate some grid points more than once (e.g., the center of the grid $(-0.5, 0.5)$ shown in Fig. 3 will also be evaluated in the following two subdivision steps). The optimization process, however, is usually the most expensive step of the whole algorithm, but will yield the same result every time this point is evaluated. Therefore the computational effort of the algorithm can be reduced significantly by storing the result of the optimization process after the first evaluation (in practice we used an AVL tree to store the grid point, its corresponding target point and the distance between these two, similar to [17, 18]). Fig. 6 demonstrates this graphically by showing the second subdivision step for the bilinear model in detail. On the left we see the result after the first selection step of the subdivision algorithm, the middle picture shows the nine optimization problems we have to newly solve in the second step and the right image shows all

the points stored in the tree that can be taken into consideration for the selection step now. In the shown case of the second step of the algorithm, we only have to solve nine optimization problems instead of the 15 that a naive approach would require. In the next step of the algorithm we can now reuse the results of the twelve corner points of the shown orange boxes on the right, and so on.

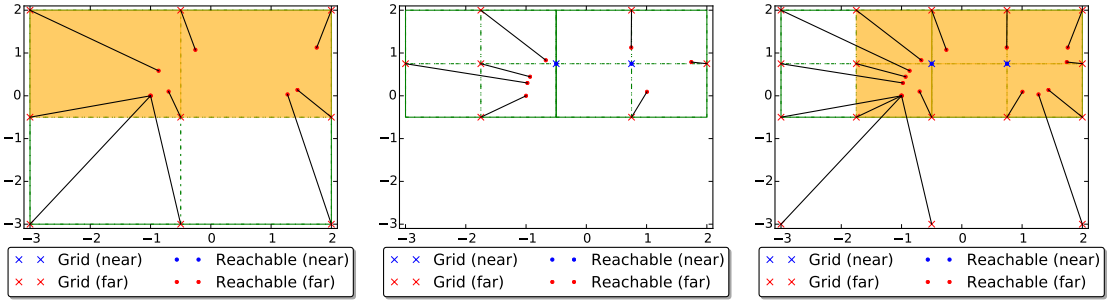

FIGURE 6. Demonstration of reusability of the results within the subdivision algorithm.

Another problem we have in the implementation of Algorithm 2.3 is that we cannot calculate the mapping $\pi_R(g)$ of every point g in every box, but the box has to be discretized. Similar to the approach shown in the PhD-thesis of O. Junge [30], we can change the algorithm in a way to guarantee that we will not cut away parts of the reachable set, when only mapping and considering the corners of every box in every step. This also means that we don't have to look at every target point generated from the other boxes, but only those we were calculating in this step which decreases the amount of lookups significantly.

Lemma 3.2. *Let $B \subset \mathbb{R}^n$ be a polygon with corners $g_i \in \mathbb{R}^n$, $i \in \{1, \dots, m\}$, let m be the number of the corners of B and $d = \operatorname{diam}(B) > 0$ the diameter of the box B.*

Then $B \cap R = \emptyset$, if $\operatorname{dist}(g_i, R) > \dfrac{d}{2}$ for every $i \in \{1, \dots, m\}$.

Proof. Since $\operatorname{dist}(g_i, R) > \dfrac{d}{2}$ for every corner g_i, we know, using the minimality of the distance-function, that there is no $r \in R$ with $r \in \dfrac{d}{2} \cdot B_1(g_i)$, for every i. We can also show easily, that $B \subset \bigcup_i \dfrac{d}{2} \cdot B_1(g_i)$, which leads to

$$\nexists\, r \in R \text{ with } r \in B.$$

□

This can be implemented into the selection step of the algorithm by not only looking for target points inside the box, but also accounting points which are close to the box. For the rigorous case, the selection step is changed to

$$\mathcal{B}_k = \left\{ B \in \mathcal{S}_k : \mathcal{P}(B) \cap \left(B + \frac{\operatorname{diam}(B)}{2} \cdot B_1(0) \right) \neq \emptyset \right\}.$$

Remark 3.3. In practice, we treat the rigorous enlargement of the box as a parameter, provided by the user. Numerical examples showed, that for most of the problems, much smaller values than $\frac{1}{2}\operatorname{diam}(B)$ are sufficient to provide good results without increasing the numerical overhead too much, by selecting in general more boxes in each step.

4. Numerical examples

As a first example in this paper, we take a look at the following bilinear system already mentioned in Sections 1 and 3.

Example 4.1. Bilinear Problem

The bilinear problem can be described with the two ODEs

$$\begin{aligned} \dot{x}_1 &= \pi \cdot x_2, \\ \dot{x}_2 &= -\pi \cdot x_1 \cdot u, \end{aligned}$$

where $x := (x_1, x_2)$ are the states and $u \in [0, 1]$ is the control of the system.

The Bilinear Problem is a nonlinear system with no state constraints and boundary conditions, which can be used to demonstrate small-time convexity of the reachable set (e.g., [7]), but we will only use it as an easy nonlinear introductory example in this paper. The reachable set can be calculated with the following parametric OCP:

Minimize

$$\frac{1}{2}\left\|x(T) - g_\rho\right\|_2^2$$

subject to:

$$\left.\begin{aligned} \dot{x}_1(t) &= \pi \cdot x_2(t), \\ \dot{x}_2(t) &= -\pi \cdot x_1(t) \cdot u(t), \\ x_1(0) &= -1, \\ x_2(0) &= 0, \\ u(t) &\in [0, 1], \end{aligned}\right\} t \in [0, T].$$

As the result of the algorithm for final time $T = 1$, we find the sets shown in Fig. 2 for the subdivision algorithm and using an equidistant grid in the state space. We can use this example to compare the computational effort of both strategies as seen in Table 2. To discretize the ODE constraints of the system, the usage of explicit Euler with only 21 timesteps was sufficient to provide good results. We used Ipopt to solve the optimization problem for every grid point, since it provided the best results for this specific example.

The next problem we want to discuss was created by Petar Kenderov during his visit at the University of Bayreuth 2001/2002 (e.g., [14, Example 5.2.1]). Since the reachable set of this problem is a part of the one-dimensional boundary of a circle in $\mathbb{R}^2$ (i.e., the reachable set of the continuous problem has no interior). It is very challenging for the optimizer to solve this system, but it has been identified as a excellent benchmark to demonstrate the influence of the used discretization scheme on the numerical approximation of the reachable set.

Number of optimization problems

Grid	N	S	S/N
3×3	9	9	1
5×5	25	18	0.72
9×9	81	41	0.506
17×17	289	92	0.318
33×33	1 089	231	0.212
65×65	4 225	635	0.150
129×129	16 641	1948	0.119
257×257	66 049	6822	0.103
513×513	263 169	25477	0.097
1025×1025	1 050 625	98040	0.093

CPU time

Grid	N	S	S/N
3×3	0.13s	0.13s	1
5×5	0.35s	0.25s	0.714
9×9	1.2s	0.57s	0.475
17×17	4.3s	1.24s	0.288
33×33	16.12s	3.12s	0.194
65×65	62.74s	8.55s	0.136
129×129	253.53s	26.29s	0.104
257×257	1128.09s	92.63s	0.082
513×513	8234.45s	355.29s	0.043
1025×1025	94221.4s	1741.58s	0.018

TABLE 2. Comparison of the computational effort of the algorithm with (S) and without (N) using the subdivision algorith for a few different grids.

Example 4.2. Kenderov Problem

The Kenderov Problem is a two-dimensional CP which can be described by the ODEs

$$\begin{aligned} \dot{x}_1 &= 8 \cdot (a_{11}x_1 + a_{12}x_2 + 2a_{12}x_2u)\,, \\ \dot{x}_2 &= 8 \cdot (-a_{12}x_1 + a_{11}x_2 - 2a_{12}x_1u) \end{aligned}$$

with constants $a_{11} = \sigma^2 - 1$, $a_{12} = \sigma \cdot \sqrt{1-\sigma^2}$ and $\sigma = 0.9$.

With $x(0) = (2, 2)$ as initial point for this system, we can construct the OCP for the reachable set in a similar way as shown in the bilinear model.

The plots in Fig. 7 were generated using WORHP, since it gave better results than Ipopt for this example, especially when using implicit Euler (middle picture, cpu time: 7 894.26s, see mapping for the grid point $(1, -1)$) with only a few timesteps for discretizing the ODE constraints. The use of explicit Euler (left picture, cpu time: 2 178.26s) yields a set spiraling to the outside, whereas the implicit Euler generates a set spiraling to the inside. Using the implicit trapezoidal rule (right picture, cpu time: 730.18s), even with only 16 timesteps, generates almost the circle we expect as reachable set of the continuous control problem.

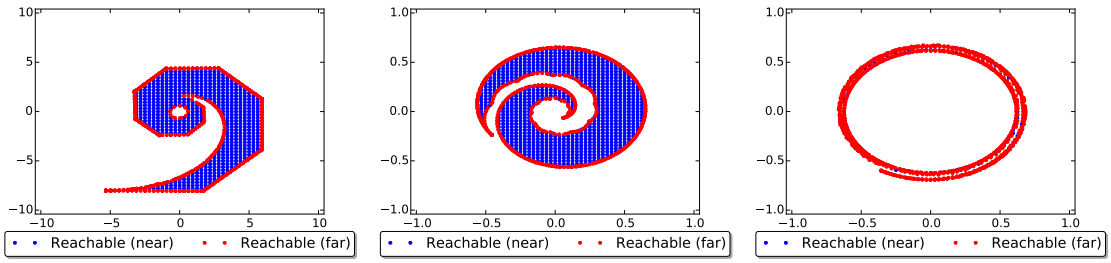

FIGURE 7. Kenderov problem using different discretization methods for the ODE-constraints with 16 timesteps each: Explicit Euler (left), implicit Euler (middle), implicit trapezoidal rule (right).

Increasing the number of timesteps to 261 will improve the results when using explizit (left plot, cpu time: 32 655.8s) and implicit Euler (middle plot: cpu-time: 16 190.4s) by reducing the spiraling behavior as seen in Fig. 8, but also increases the computational effort. Using the implicit trapezoidal rule here (right plot, cpu-time: 10 282.6s) is the recommended method to find the reachable set of this system and we can get an almost perfect result in this case.

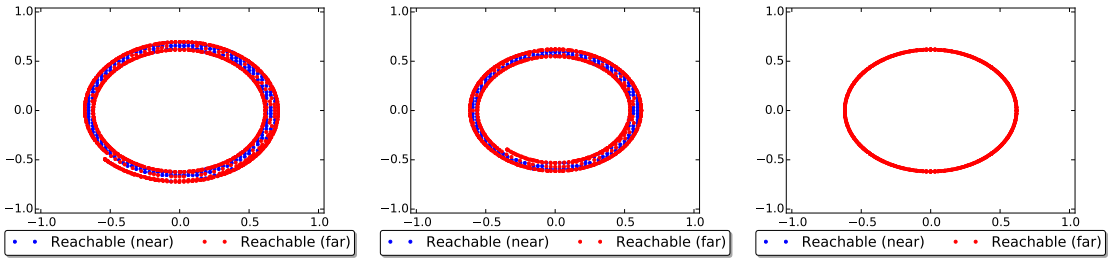

FIGURE 8. Kenderov problem using explicit Euler (left), implicit Euler (middle) and the implicit trapezoidal rule (right) with 261 timesteps each.

As a more practical example, we can take a look at the robot described in [25, Sec. 5.2].

Example 4.3. Robot Problem

A planar two-linked robot (see Fig. 9), controlled by two torques $(u_1, u_2) \in [-25, 25]^2$ operating in the joints M_1 and M_2, can be described by the following ODEs:

$$\dot{q}_1 = w_1, \tag{4}$$

$$\dot{q}_2 = w_1 + w_2, \tag{5}$$

$$\dot{w}_1 = \frac{J_{22}\left(u_1 - u_2 + J_{12}\sin(q_2)w_2^2\right) - J_{12}\cos(q_2)\left(u_2 - J_{12}\sin(q_2)w_1^2\right)}{J_{11}J_{22} - J_{12}^2\cos^2(q_2)}, \tag{6}$$

$$\dot{w}_2 = \frac{J_{11}\left(u_2 - J_{12}\sin(q_2)w_1^2\right) - J_{12}\cos(q_2)\left(u_1 - u_2 + J_{12}\sin(q_2)w_2^2\right)}{J_{11}J_{22} - J_{12}^2\cos^2(q_2)}, \tag{7}$$

with two angles q_1 and q_2 between -3 and 3, two angular velocities w_1 and w_2 between -5 and 5, constants

$$\begin{array}{llllllllll} m_1 & = 24 & [kg], & m_2 & = 15 & [kg], & m & = 6 & [kg], \\ J_1 & = 1.6 & [kgm^2], & J_2 & = 0.43 & [kgm^2], & J_3 & = 0.01 & [kgm^2], \\ l_1 & = 0.4 & [m], & l_2 & = 0.25 & [m], & & & \\ a_1 & = 0.2 & [m], & a_2 & = 0.125 & [m] & & & \end{array}$$

and

$$\begin{aligned} J_{11} &= J_1 + (m_2 + m)l_1^2, \\ J_{12} &= m_2 a_2 l_1 + m l_1 l_2, \\ J_{22} &= J_2 + J_3 + m l_2^2. \end{aligned}$$

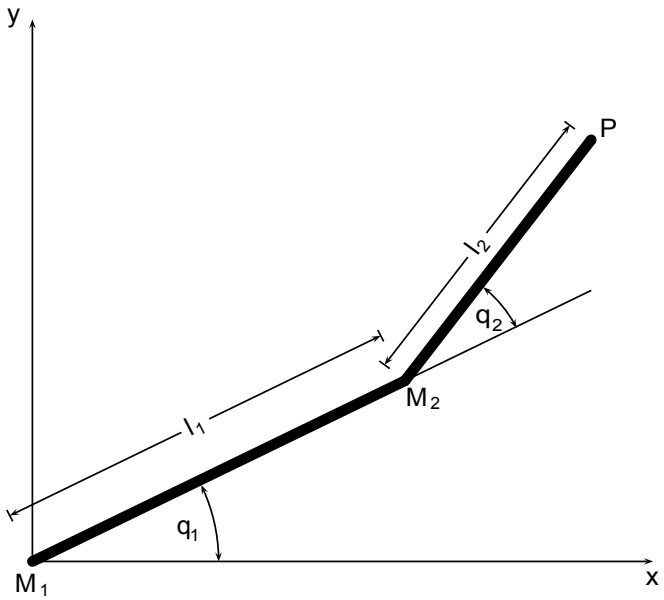

FIGURE 9. Illustration of the robot from Ex. 4.3.

As we can see, we have to deal with a four dimensional system, but in practice, we are only interested in the possible location of a payload at the end point $P \in \mathbb{R}^2$ of the robot. The coordinates (x, y) of this point can be described with the following transformation

$$P = \mathcal{T}(q_1, q_2) = \begin{pmatrix} l_1 \cos(q_1) + l_2 \cos(q_1 + q_2) \\ l_1 \sin(q_1) + l_2 \sin(q_1 + q_2) \end{pmatrix}.$$

Furthermore, we want the robot to stop at the desired final time T, i.e., $w_1(T) = w_2(T) = 0$ which can be formulated as a boundary condition

$$w_1(T)^2 + w_2(T)^2 = 0.$$

As initial values for q_1, q_2, w_1, and w_2, we model the scenario to perform an emergency shutdown of the fully accelerated robot arm passing through one point of the system. The calculated reachable set is the dangerous region, where the robot arm might come to a halt, whereas the complement is safe. This results in the following parametric OCP to calculate the reachable set at final time $T = 2$s:

Minimize

$$\frac{1}{2}\|\mathcal{T}(q_1, q_2) - g_\rho\|_2^2 \tag{8}$$

subject to:

$$
\begin{aligned}
\dot{q}_1 &= w_1, \\
\dot{q}_2 &= w_1 + w_2, \\
\dot{w}_1 &= \frac{J_{22}\left(u_1 - u_2 + J_{12}\sin(q_2)w_2^2\right) - J_{12}\cos(q_2)\left(u_2 - J_{12}\sin(q_2)w_1^2\right)}{J_{11}J_{22} - J_{12}^2\cos^2(q_2)}, \\
\dot{w}_2 &= \frac{J_{11}\left(u_2 - J_{12}\sin(q_2)w_1^2\right) - J_{12}\cos(q_2)\left(u_1 - u_2 + J_{12}\sin(q_2)w_2^2\right)}{J_{11}J_{22} - J_{12}^2\cos^2(q_2)}, \\
q_i(0) &= 0, \ i = 1,2 \\
w_i(0) &= 5, \ i = 1,2 \\
u_i(t) &\in [-25, 25], \ i = 1,2 \\
-3 &\leq q_i(t) \leq 3, \ i = 1,2 \\
-5 &\leq w_i(t) \leq 5, \ i = 1,2 \\
0 &\leq w_1^2(T) + w_2^2(T) \leq 0,
\end{aligned}
$$

for every time $t \in [0, 2]$ and grid point g_ρ, using $[-1, 1]^2$ as initial bounding box.

Using the implicit trapezoidal rule (Lobatta IIIA of order 2) with 200 timesteps, we get the approximation of the reachable set shown in Fig. 10 (distance between grid points $\frac{1}{16}$ as in a full 33×33 grid) and Fig. 11 (distance between grid points $\frac{1}{62}$ as in a full 129×129 grid). The use of Euler or implicit Euler would require a rather fine time discretization/small step size in time which would cause longer computation times.

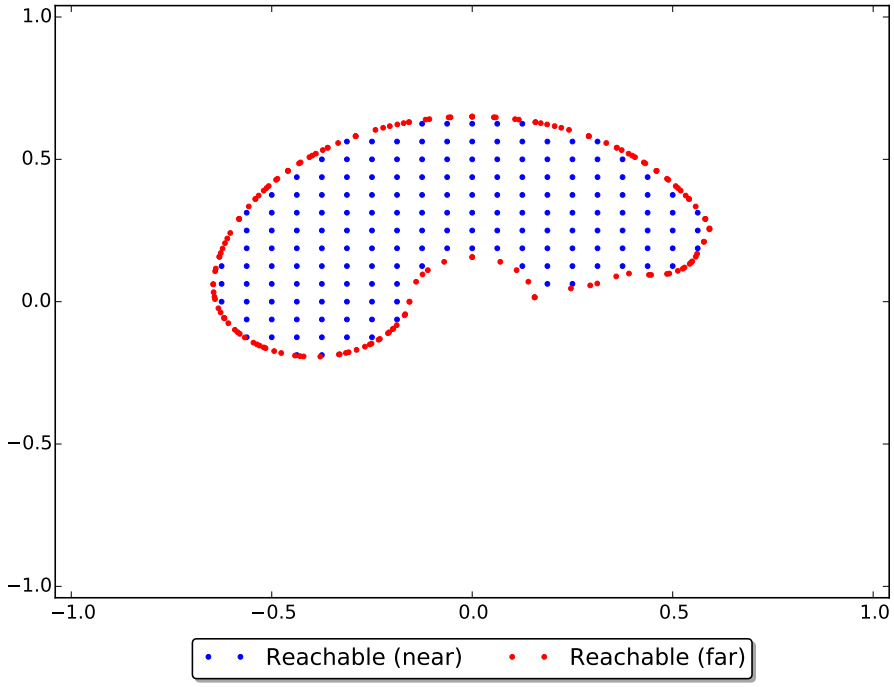

FIGURE 10. Rough approximation of the reachable set of the robot problem.

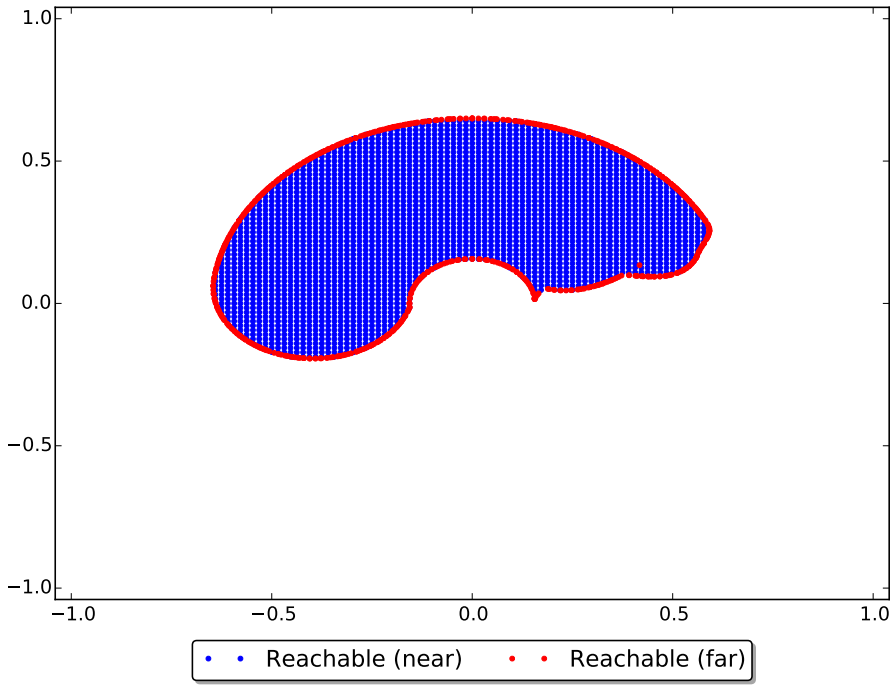

FIGURE 11. Finer approximation of the reachable set of the robot problem.

The OCP generated by this problem is rather challenging for the used optimizers and the quality of the results depends heavily on the used initial guess. To demonstrate the difficulty we first look at the reachable set for the angles q_1 and q_2 (left picture in Fig. 12) and perform the transformation $\mathcal{T}$ of the angles q_1 and q_2 into the coordinates x and y afterwards, when plotting the results (right picture in Fig. 12).

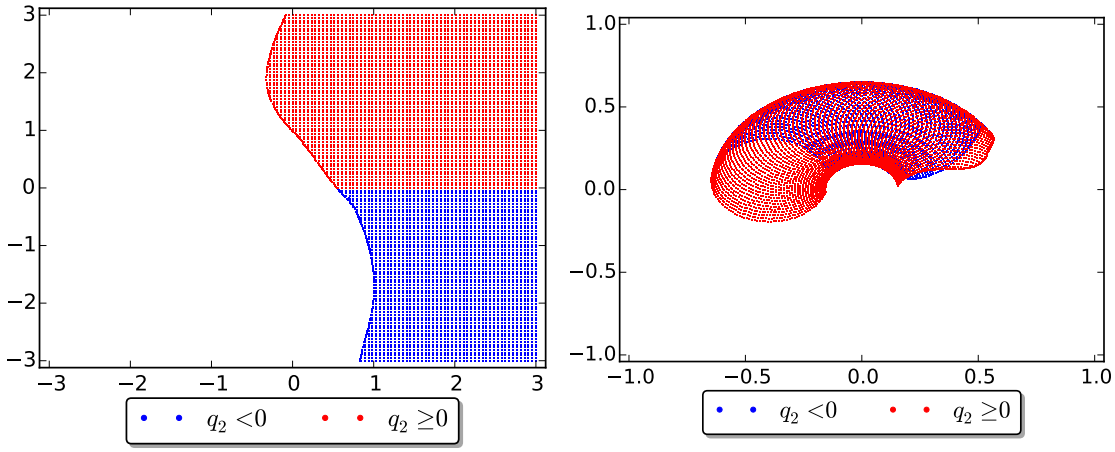

FIGURE 12. Reachable set of the robot without the transformation in the objective function (left) and the transformed set using the same results and colorcode (right).

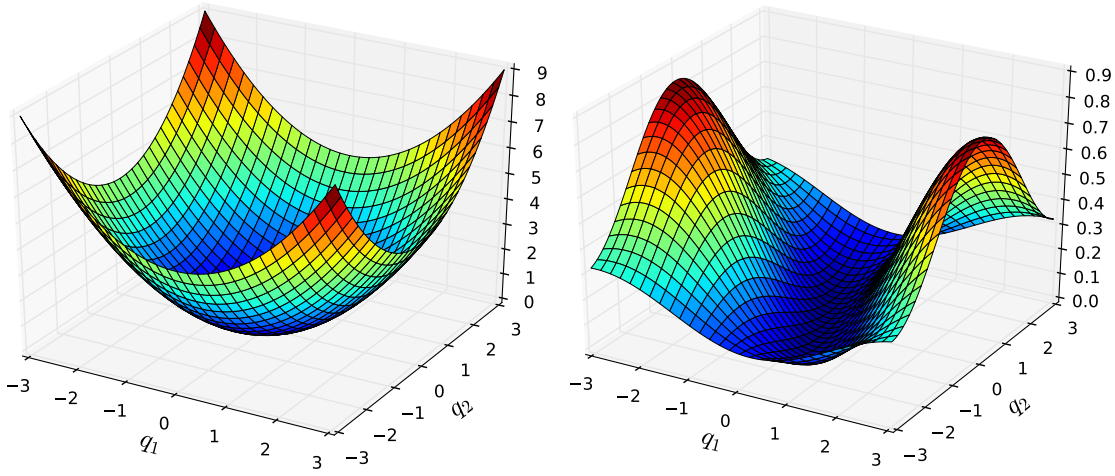

FIGURE 13. Objective function of the non-transformed robot problem (left) and the transformed version (right).

We can see (indicated by the two different colors) that there are two major strategies (i.e., q_2 is positive or negative at final time T) which generate a partially overlapping part of the transformed reachable set, but on the right boundary, there are also some parts that can only be reached by following a trajectory leading to either a positive or negative angle q_2.

The non-transformed problem is a little bit easier to solve for the optimizer (some fine tuning is still required), since the objective function doesn't have as many local extrema as the transformed problem (as we can see in Fig. 13), but we don't get a result with almost equidistant target points inside of the reachable set this way. If we want to directly solve the transformed version of this problem to generate a nicer approximation, we have to be very careful to not only generate a part of the reachable set (numerical experiments like Fig. 14 showed that due to the choice of the initial value, all our used optimizers tend to only generate the red part in the right picture of Fig. 12 most of the time).

In our case, we circumvented the problem by solving each OCP more than once with multiple initial guesses which drive the system into both strategies and only took the result for which the target point had the minimal distance to the grid point. Although this strategy will raise the computational effort of the program, it is a better strategy than working with warm starts from former processed grid points and risking to follow the wrong strategy that way. As initial guesses for the transformed problem, we chose control sequences, that were generated by using the non-transformed problem. We reused the results from Fig. 12 for this purpose, but for practical applications, it is not necessary to generate the non-transformed reachable set with such a dense grid. We only have to solve the (OCP) for grid points in the vicinity of problematic regions, to construct appropriate estimates.

In Fig. 15 we show the results we get, if we generate the reachable set with Ipopt and WORHP. As we can see, both optimizers have some difficulties to find

the global minimum for some grid points, but we still get a good approximation of the expected reachable set with both when using various initial guesses.

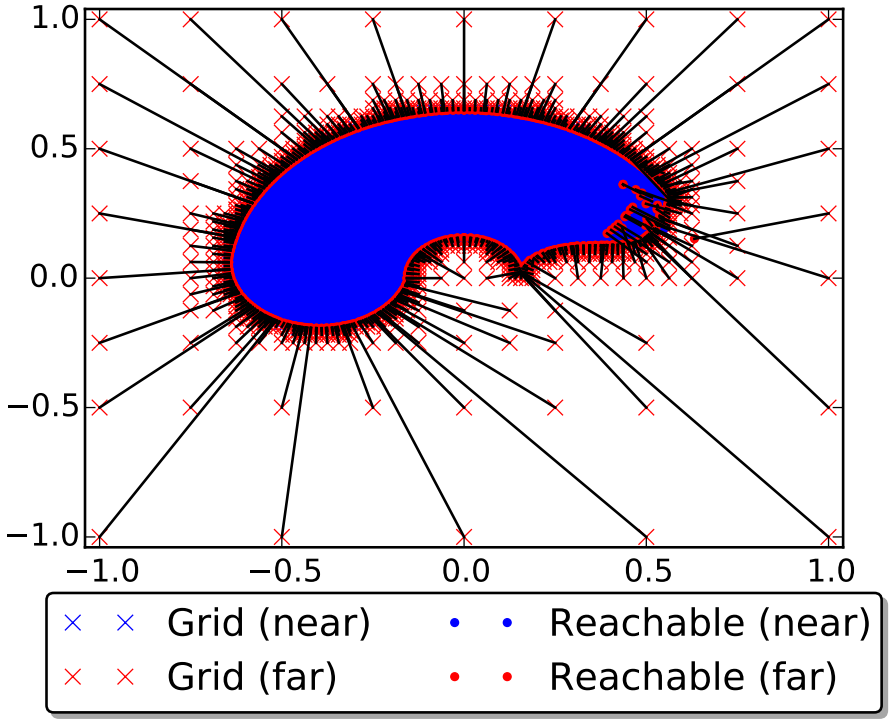

FIGURE 14. Approximated transformed reachable set of the robot with a less elaborate strategy for initial guesses.

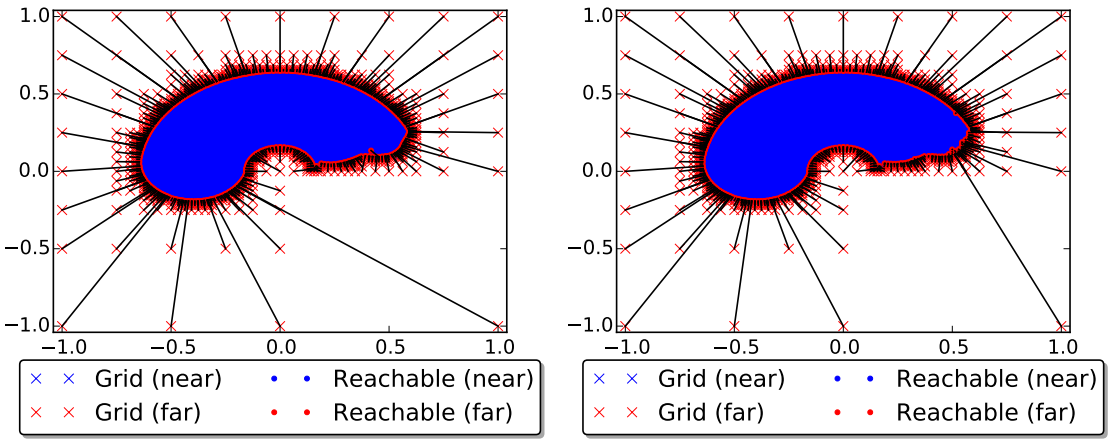

FIGURE 15. Reachable set of the robot problem, generated with Ipopt (left) and WORHP (right).

5. Advantages of the algorithm

In this section, we will collect some nice features of the subdivision algorithm, some of which (e.g. zooming and parallelization) also work without subdivision. Example 4.3 also shows that the algorithm can handle higher dimensional dynamics, especially if we are interested in a lower-dimensional projection of the reachable set. The state discretization is only done for the lower-dimensional bounding box (in this example 2d), not for the reachable set of the complete system (4d). In the robot example these are the x- and y-coordinates of the payload at the end point of the robot arm.

5.1. **Transformed grids.** Although most of the examples in this paper only use boxes in the set collection, the algorithm (with or without subdivision) can also be used with triangulations of different shapes. A prime example is the robot problem introduced in Ex. 4.3. Due to physical constraints the feasible region is a circle around the origin with outer radius $l_1+l_2 = 0.65$ and the different lengths of the two parts of the arm create a hole in the middle, since the robot cannot reach the inner circle with radius $l_2 - l_1 = 0.15$. A natural approach is to use a circular grid with the bigger radius $l_1 + l_2$, since the reachable set has to be a subset of it (see Fig. 16). The left picture shows the used grid points in this case and the right picture shows the approximation of the reachable set using the generated target points.

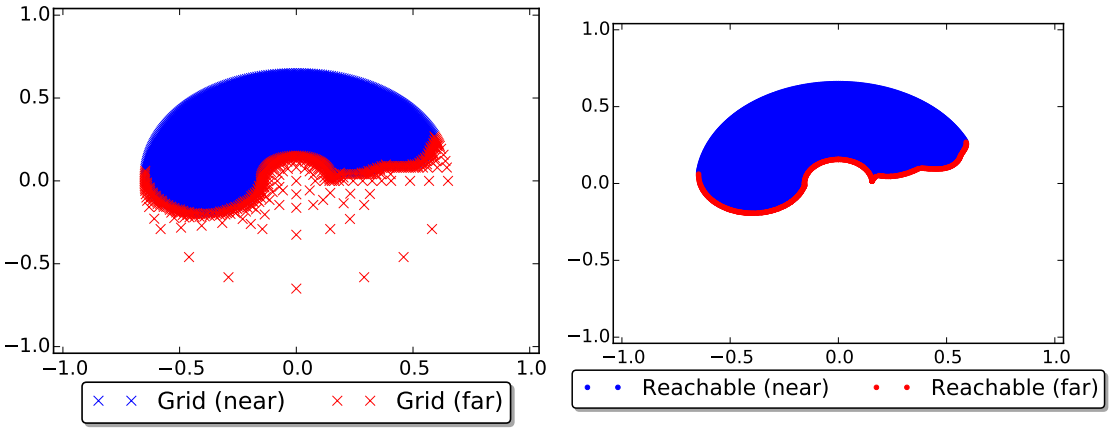

Figure 16. Robot model using a circular grid.

The circular grid is created by choosing a box shaped grid (in the robot example: $[-1,1]^2$) and transforming it to a circular grid with radius s, using the mapping

$$x \longmapsto \tilde{x} = \begin{cases} 0 & \text{if } x = 0, \\ s \cdot \frac{\|x\|_\infty}{\|x\|_2} \cdot x & \text{if } x \neq 0. \end{cases}$$

It can be easily shown that the inverse of this transformation can be written as

$$\tilde{x} \longmapsto x = \begin{cases} 0 & \text{if } \tilde{x} = 0, \\ \frac{1}{s} \cdot \frac{\|\tilde{x}\|_2}{\|\tilde{x}\|_\infty} \cdot \tilde{x} & \text{if } \tilde{x} \neq 0 \end{cases}$$

which allows us to implement the subdivision algorithm for the circular grid very easily. Subdivision and selection will be performed by using the initial bounding box $[-1,1]^2$. A chosen grid point g in this box will be transformed to its respective point $\tilde{g}$ in the circular grid, we solve the minimization problem "minimize $\frac{1}{2}\|\mathcal{T}(q_1,q_2)-\tilde{g}\|_2^2$" and use the inverse to transform the generated end point of the trajectories back to our used boxes.

Using this method, we can also use grids created by, e.g., a rotation around some point in the state space, sheared grids, etc., which can be very beneficial when dealing with oddly shaped reachable sets.

5.2. **Zooming.** Sometimes, we are not interested in the complete reachable set, but only want to take a closer look at one specific part of the set. As an example, we take a closer look at the Kenderov Problem, generated by the algorithm using the explicit Euler with 16 time steps shown in Fig. 7 (left). We are interested to see, how the set looks like inside of the box $[-1.0,1.0]\times[0.5,2.0]$, but a graphical zoom into that picture does not reveal any detail and we only get a meaningless collection of points with a huge gap in between as shown in Fig. 17 on the left.

To get a better picture we could generate the whole reachable set with a very dense grid (as it would be in many other methods approximating reachable sets) which takes a lot of time, or – more efficiently – we can set the initial bounding box of Algorithm 2.3 to the interesting region $B_0=[-1.0,1.0]\times[0.5,2.0]$, start the subdivision algorithm and only generate the part of the reachable set, that lies near that box.

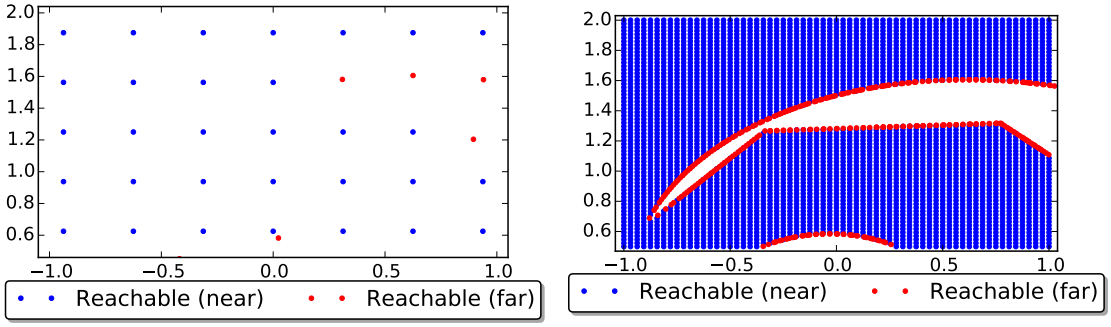

FIGURE 17. Graphical (left) and computational (right) zoom on the example of the Kenderov Problem.

5.3. **Determination of a bounding box.** One restriction of the algorithm is, that we have to provide an initial bounding box, where we expect the reachable set. Finding this box is not always an easy task when analytic estimates are not at hand. The subdivision algorithm provides a good way to automatically find a suitable bounding box via trial and error which is illustrated in Figs. 18 and 19.

The strategy behind this approach is to start with any box in the state space and perform the subdivision algorithm. If the reachable set, or parts of it, are contained within this box, we will get an approximation as shown in the previous examples. If we have the situation, like shown in Fig. 18 (cpu time: 0.15s), where

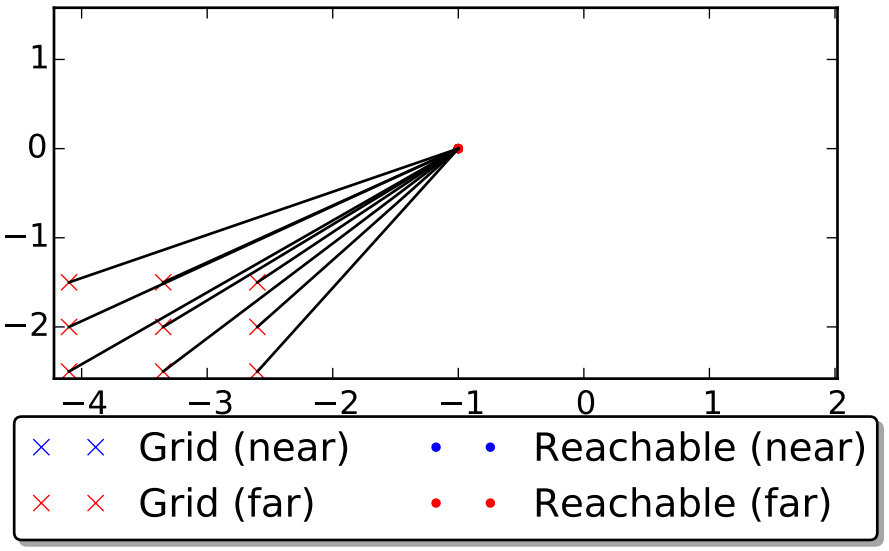

FIGURE 18. Self-finding of the bounding box.

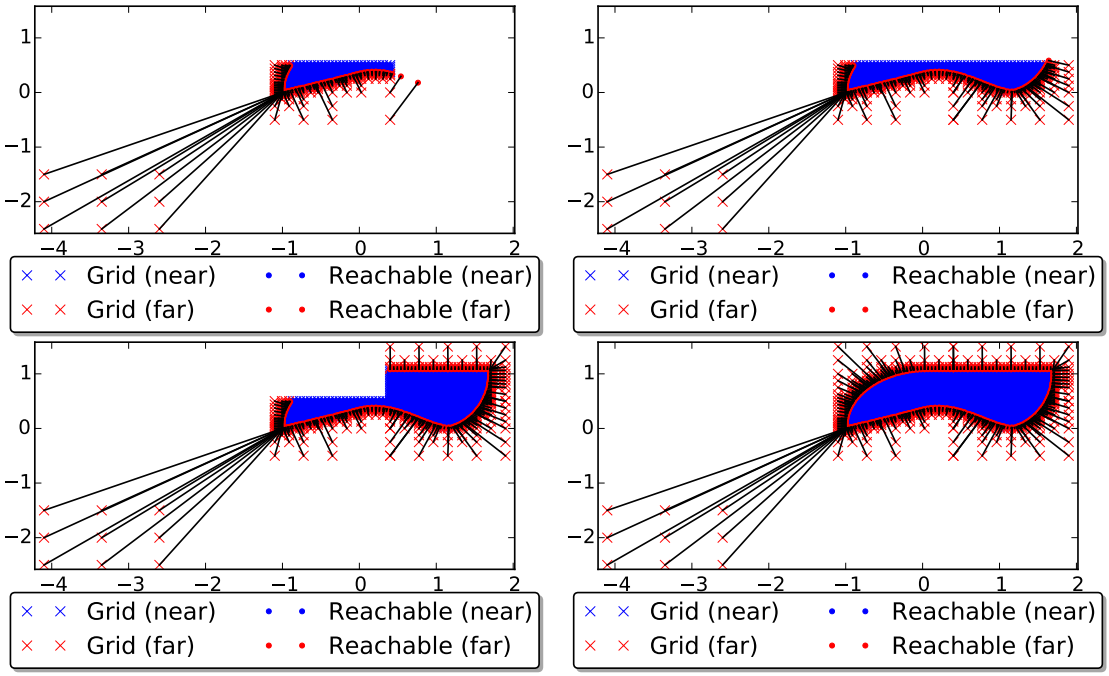

FIGURE 19. Piecewise construction of the whole reachable set, by choosing new bounding boxes at locations containing target points from previous runs or cut off looking edges.

the grid points were far away from the reachable set (i.e. $B_0 \cap R = \emptyset$ with $B_0 = [-4.1, -2.6] \times [-2.5, -1.5]$), the algorithm will stop after the first selection step, but we do not have to start with another blind guess again. We can use our results to create a bounding box around the target points (in our example we only found the point approximately at $(-1, 0)$) we got from the failed algorithm, since

they are part of the reachable set that we are looking for. We may not find the whole set with this new bounding box either, but every time we see parts of the reachable set being cut off by the chosen bounding box or we find target points outside of the box, we can restart the algorithm again at those locations, until we find a set that is completely surrounded by target points created from grid points far away from the set. Fig. 19 shows this strategy for the bilinear problem 4.1, where new boxes where chosen at $[-1.1, 0.4] \times [-0.5, 0.5]$, $[0.4, 1.9] \times [-0.5, 0.5]$, $[0.4, 1.9] \times [0.5, 1.5]$ and $[-1.1, 0.4] \times [0.5, 1.5]$. We only translated our initial bounding box to new locations to achieve a consistent density of grid points, without having to change the number of subdivision steps. Merging the results after every step creates the same reachable set as shown in Fig. 2.

5.4. **Parallelization.** Due to the structure of the algorithm, it is quite easy to parallelize the algorithm. Most of the effort to generate the reachable set lies in solving the OCPs at every grid point, but if we are not relying on warm start strategies, these are all independent of each other. The algorithm without subdivision can easily be split up into many tasks (the number of tasks that way is only bounded by the number of grid points in the state space) and those can be distributed to different threads, processes or even machines which allows us to use the whole spectrum of parallelization (e.g., OpenMP[1], Pthreads[2], Open MPI[3]). Other parallelization approaches in using optimizers for the calculation of reachable sets can be found in [14, Sec. 4.4] (master/slave system with PVM[4] and [29, Chap. 4–6] (parallelization on GPUs with CUDA[5]).

The subdivision algorithm is a little bit more difficult to parallelize, due to the fact, that we cannot specify every grid point a priori. A possible – and very crude but feasible – way to implement parallelization is a similar idea to the one seen in Sec. 5.3. Every time we reach the selection step of the algorithm, we could distribute the selected boxes to parallel tasks performing the subdivision and only collect all the results in the end. As a small scale example, we can recycle the results from Fig. 19. The four top right boxes can be calculated in parallel and we were measuring the times spent to create the individual parts (Tab. 3) using a workstation with a Intel®Core™2 Duo CPU E8600 (3.33GHz) and 8GB RAM. The whole reachable set, with the same density ($\frac{3}{64}$ in x-direction and $\frac{1}{32}$ in y-direction) of grid points, can be generated in 33.12 s. Although the summed up times for the smaller boxes is slightly higher, solving the OCPs in the four smaller boxes in parallel leads to a faster computation finishing after approx. 11 s. Table 3 shows that the computational time for each subregion differs due to skipped load balancing and the varying number of grid points lying in each subregion.

This idea can also be used for the non-adaptive version, where the complete set with the same density of grid points can be computed in 70.44 s on the same

[1]Open Multi-Processing, http://www.openmp.org/

[2]Native POSIX Thread Library, http://www.opengroup.org/content/posix%C2%AE/

[3]Open Message Passing Interface, https://www.open-mpi.org/

[4]Parallel Virtual Machine, http://www.csm.ornl.gov/pvm/

[5]Compute Unified Device Architecture, http://www.nvidia.com/object/cuda_home_new.html

Box	Times (with subdivision)	Times (without subdivision)
$[-1.1, 0.4] \times [-0.5, 0.5]$	$7.49\,s$	$18.21\,s$
$[0.4, 1.9] \times [-0.5, 0.5]$	$7.00\,s$	$17.58\,s$
$[-1.1, 0.4] \times [0.5, 1.5]$	$9.35\,s$	$18.12\,s$
$[0.4, 1.9] \times [0.5, 1.5]$	$10.55\,s$	$18.69\,s$

TABLE 3. Computational times for generating parts of the reachable set with the shown boxes.

machine. Looking at the results in the last column of Table 3, we can see that we get the set in less than $19\,s$ by splitting it up into four threads. Here, the computational time of each subregion is very similar, since the non-adaptive version uses a uniform grid and has to solve the same number of optimization problems to solve on every box.

5.5. **Solution funnel in 3d.** Another advantage of the algorithm (with and without subdivision) is that we are not restricted to two dimensional grids. As an example we can expand the bilinear model 4.1 to calculate not only the reachable set at a fixed time, but the solution funnel of the system (i.e., the reachable set up to a set time). We can show, that this can be achieved by introducing another state x_t and the (virtual) control u_t in the following way:

$$\begin{aligned} \dot{x}_t &= u_t, \\ \dot{x}_1 &= u_t \cdot \pi \cdot x_2, \\ \dot{x}_2 &= u_t \cdot (-\pi \cdot x_1 \cdot u), \end{aligned}$$

with $u_t \in [0, 1]$ on the time interval $[0, 1]$. Using the algorithm on this system will give us target points (now three dimensional) which can be used to describe the reachable set (as seen in Fig. 20). By considering, e.g., the virtual control

$$u_t(\tau) := \begin{cases} 1 & \text{for } \tau \in [0, 0.5], \\ 0 & \text{else}, \end{cases}$$

the corresponding solution $x(\cdot)$ fulfills that $x_t(\tau) = 0.5$ and $(x_1(\tau), x_2(\tau))$ lies in all reachable sets $R(\tau)$ for $\tau \geq 0.5$. This well-known technique to approximate the *union of reachable sets*, i.e., *reachable points up to the time* $\tau \in [0, 1]$, is also applied, e.g., in [31].

The subdivision algorithm has the big advantage, that it also creates the box collections approximating the reachable set which are much easier to handle in practical applications. Working with the generated point cloud can be rather difficult, since we cannot assume the set to be convex like in linear control problems. Even visualization of the results turns out to be a rather problematic task. Fig. 20 only shows the slices at $(x_t)_i = \frac{1}{8}i$ $(i = 0, 1, ..., 8)$ through the set, i.e., the reachable sets of the bilinear problem with final times $(x_t)_i$. This was mostly

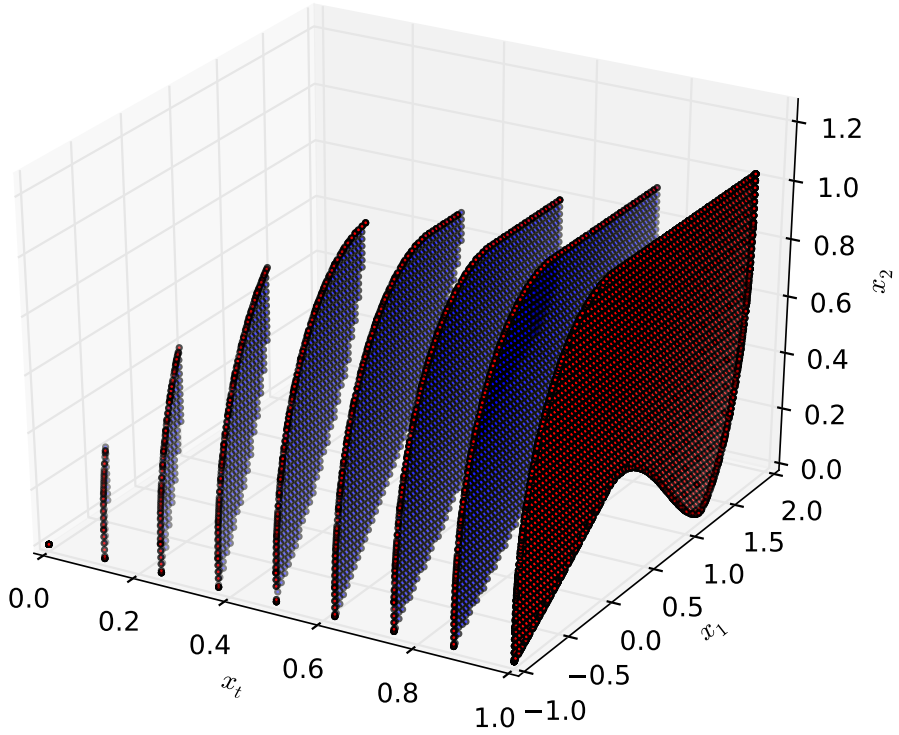

FIGURE 20. Solution funnel of the bilinear problem for $T \in [0, 1]$. The plot only shows the slices at $t_i = \frac{1}{8} \cdot i,\ i \in \{0, 1, \ldots, 8\}$ to make it easier to read (cpu-time 103 534s).

done to keep the visualization of the results in this example a little bit more clear. In the calculation of the funnel the distance between grid points on the x_t-axis is much smaller. When using the box collections, we can see that it is much easier to handle the generated approximation. If we have to check, if a point is (almost) reachable, we only have to check if it is contained in one of the boxes (convex and therefore easy to handle) which were generated by the algorithm. It is also much easier to export the box collections into a format (e.g., OBJ) that can be handled by e.g., 3d modelling software or interactive 3d applications, than having to deal with a point cloud. As an example, we took the final box collection from the calculation of Fig. 20 and visualized it using the raytracer POV-Ray [33]. The image in Fig. 21 was created in a very crude way, by putting every generated box into a POV-Ray union (i.e., a structure that unifies multiple objects into a single object for the rendering process). One could also postprocess the box collection first and merge neighboring boxes into bigger objects or remove completely hidden boxes to save on rendering time by reducing the number of objects in the scene.

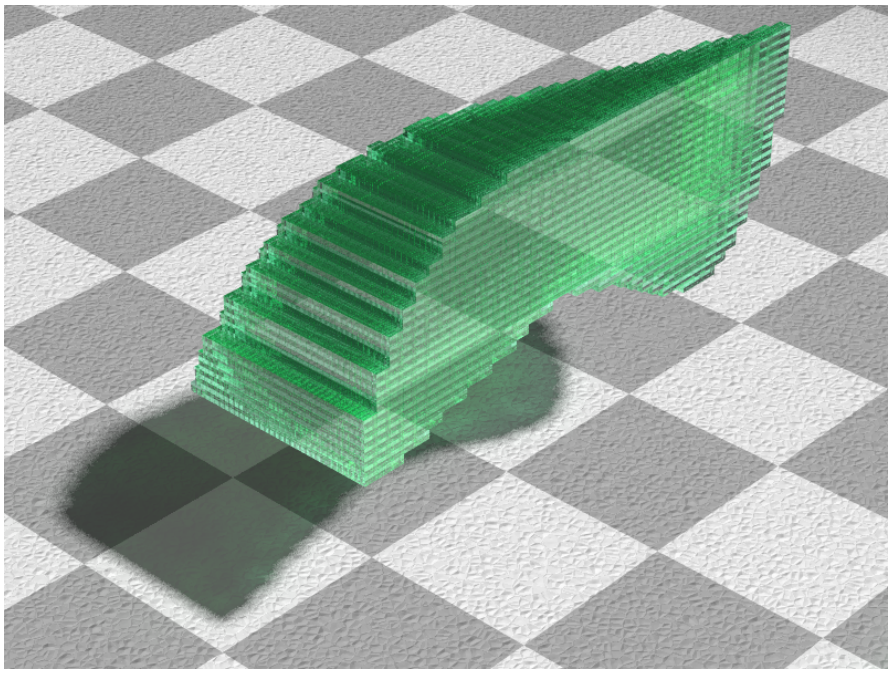

FIGURE 21. Final box collections of the solution funnel of the bilinear problem for $T \in [0, 1]$, rendered with POV-Ray.

6. CONCLUSION

As we can see in this paper, the subdivision algorithm is an extension to the reachable set algorithm shown in [7] that doesn't only reduce the computational effort for most cases, but also generates an additional approximation of the set. We can show convergence of the set collection, generated by the subdivision algorithm, to the reachable set and have an a priori estimate about the accuracy. This allows us to use in practical examples as in 21, e.g., a box collection (instead of a point cloud which is much more difficult to handle). Due to the subdivision approach, the algorithm inherits all positive aspects of the subdivision technique, especially the computational applicability to higher-dimensional control systems if only a low-dimensional projected reachable set is of interest (see Example 4.3). Even with the non-adaptive optimization approach, those low-dimensional projected reachable sets can be calculated, e.g., the single-track model with 7 states and 2 controls in [43, Sec. 2,1] as well as the docking maneuver of a satellite with 13 states and 6 controls in [29, Sec. 7.1].

We have also seen that the algorithm can handle more difficult examples, e.g., reachable sets without an interior (Kenderov Problem 4.2) or transformed problems (Robot 4.3) without changing the subdivision strategy. The main problem lies in solving the optimization problems which can be, depending on the problem, more or less tricky. It is imperative to ensure good results of the underlying optimization process which can be achieved by using one of the many well tested optimization frameworks available. Some parameter adjustments or trial and error with initial guesses may be necessary, depending on the difficulty of the problem.

The examples also were used to demonstrate the differences between the adaptive and non-adaptive version as well as the influence of the choice of the discretizer or OCP-solver (i.e., WORHP or Ipopt) on the quality of the results.

Acknowledgments

The research leading to these results had received funding from the European Union's Seventh Framework Programme under grant agreement no. 338508.

References

[1] M. Althoff. *Reachability Analysis and its Application to the Safety Assessment of Autonomous Cars*. PhD thesis, Fakultät für Elektrotechnik und Informationstechnik, Technische Universität München, Munich, Germany, August 2010. http://nbn-resolving.de/urn/resolver.pl?urn:nbn:de:bvb:91-diss-20100715-963752-1-4.

[2] J.-P. Aubin, A. M. Bayen, and P. Saint-Pierre. *Viability theory. New directions*. Springer, Heidelberg, second edition, 2011. First edition: J.-P. Aubin in *Systems & Control: Foundations & Applications*, Birkhäuser Boston Inc., Boston, MA, 2009.

[3] J.-P. Aubin, T. Bernado, and P. Saint-Pierre. A viability approach to global climate change issues. In A. Haurie and L. Viguier, editors, *The Coupling of Climate and Economic Dynamics. Essays on Integrated Assessment*, volume 22 of *Advances in Global Change Research*, pages 113–143. Springer, Dordrecht–Berlin–Heidelberg–New York, 2005.

[4] J.-P. Aubin and A. Désilles. *Traffic networks as information systems. A viability approach*. Mathematical Engineering. Springer-Verlag, Berlin, 2017.

[5] R. Baier, I. A. Chahma, and F. Lempio. Stability and convergence of Euler's method for state-constrained differential inclusions. *SIAM J. Optim.*, 18(3):1004–1026, 2007. D. Dentcheva, J. Revalski (eds.), special issue on "Variational Analysis and Optimization".

[6] R. Baier and M. Gerdts. A computational method for non-convex reachable sets using optimal control. In *Proceedings of the European Control Conference (ECC) 2009, Budapest (Hungary), August 23–26, 2009*, pages 97–102, Budapest, 2009. European Union Control Association (EUCA). http://ieeexplore.ieee.org/document/7074386/.

[7] R. Baier, M. Gerdts, and I. Xausa. Approximation of reachable sets using optimal control algorithms. *Numer. Algebra Control Optim.*, 3(3):519–548, 2013.

[8] M. Bardi and I. Capuzzo-Dolcetta. *Optimal control and viscosity solutions of Hamilton-Jacobi-Bellman equations*. Systems & Control: Foundations & Applications. Birkhäuser Boston Inc., Boston, MA, 1997. With appendices by Maurizio Falcone and Pierpaolo Soravia.

[9] W.-J. Beyn and J. Rieger. Numerical fixed grid methods for differential inclusions. *Computing*, 81(1):91–106, 2007.

[10] W.-J. Beyn and J. Rieger. The implicit Euler scheme for one-sided Lipschitz differential inclusions. *Discrete Contin. Dyn. Syst. Ser. B*, 14(2):409–428, 2010.

[11] M. Bodenschatz. Berechnung erreichbarer mengen mit globalen optimierungsverfahren [calculation of reachable sets using global optimization methods]. Diploma thesis, Department of Mathematics, University of Bayreuth, Bayreuth, Germany, March 2014. http://num.math.uni-bayreuth.de/en/thesis/2014/Bodenschatz_Michael/.

[12] O. Bokanowski, E. Bourgeois, A. Désilles, and H. Zidani. HJB approach for a multi-boost launcher trajectory optimization problem. In *IFAC Proc.*, volume 49-17, pages 456–461. 20th IFAC Symposium on Automatic Control in Aerospace – ACA 2016, Aug 2016, Sherbrooke, Quebec, Canada., 2016.

[13] C. Büskens and D. Wassel. The ESA NLP solver WORHP. In *Modeling and optimization in space engineering*, volume 73 of *Springer Optim. Appl.*, pages 85–110. Springer, New York, 2013.

[14] I. A. Chahma. Set-valued discrete approximation of state-constrained differential inclusions. *Bayreuth. Math. Schr.*, 67:3–162, 2003.
[15] Christian M. Chilan and Bruce A. Conway. A reachable set analysis method for generating near-optimal trajectories of constrained multiphase systems. *J. Optim. Theory Appl.*, 167(1):161–194, 2015.
[16] J. L. Davy. Properties of the solution set of a generalized differential equation. *Bull. Austral. Math. Soc.*, 6:379–398, 1972.
[17] M. Dellnitz and A. Hohmann. A subdivision algorithm for the computation of unstable manifolds and global attractors. *Numer. Math.*, 75(3):293–317, 1997.
[18] M. Dellnitz, A. Hohmann, O. Junge, and M. Rumpf. Exploring invariant sets and invariant measures. *Chaos*, 7(2):221–228, 1997.
[19] M. Dellnitz, O. Junge, M. Post, and B. Thiere. On target for Venus — set oriented computation of energy efficient low thrust trajectories. *Celestial Mech. Dynam. Astronom.*, 95(1-4):357–370, 2006.
[20] A. Désilles, H. Zidani, and E. Crück. Collision analysis for an UAV. In *AIAA Guidance, Navigation, and Control Conference 2012 (GNC-12), 13–16 August 2012 in Minneapolis, Minnesota*, page 23 pages. American Institute of Aeronautics and Astronautics, August 2012. http://arc.aiaa.org/doi/book/10.2514/MGNC12.
[21] A. L. Dontchev and E. M. Farkhi. Error Estimates for Discretized Differential Inclusions. *Computing*, 41(4):349–358, 1989.
[22] A. L. Dontchev and F. Lempio. Difference methods for differential inclusions: A survey. *SIAM Rev.*, 34(2):263–294, 1992.
[23] M. Falcone and R. Ferretti. *Semi-Lagrangian Approximation Schemes for Linear and Hamilton-Jacobi Equations*, volume 133 of *Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), 2014.
[24] Matthias Gerdts. OCPID-DAE1 – Optimal control and parameter identification with differential-algebraic equations of index 1, 2013. http://www.optimal-control.de/.
[25] Matthias Gerdts and Martin Kunkel. A nonsmooth Newton's method for discretized optimal control problems with state and control constraints. *J. Ind. Manag. Optim.*, 4(2):247–270, 2008.
[26] L. Grüne. *Asymptotic behavior of dynamical and control systems under perturbation and discretization*, volume 1783 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin, 2002.
[27] L. Grüne and T. Jahn. Computing reachable sets via barrier methods on SIMD architectures. In J. Eberhardsteiner, H. J. Böhm, and F. G. Rammerstorfer, editors, *Proceedings of the 6th European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS 2012) Held at the University of Vienna, Vienna, Austria, September 10–14, 2012*, pages 2076–2095, Vienna, Austria, 2012. Vienna University of Technology. http://www.eccomas.org/spacehome/1/7/, Paper No. 1518, e-book.
[28] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing In Science & Engineering*, 9(3):90–95, 2007.
[29] T. U. Jahn. *A Feasibility Problem Approach for Reachable Set Approximation*. PhD thesis, Fakultät für Mathematik, Physik und Informatik, Bayreuth, Germany, December 2014. http://nbn-resolving.de/urn/resolver.pl?urn=urn:nbn:de:bvb:703-epub-2087-4.
[30] O. Junge. *Mengenorientierte Methoden zur numerischen Analyse dynamischer Systeme*. PhD thesis, University of Paderborn, März 2000. http://www.shaker.de/de/content/catalogue/index.asp?lang=de&ID=8&ISBN=978-3-8265-7081-0.
[31] I. M. Mitchell, A. M. Bayen, and C. J. Tomlin. A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games. *IEEE Trans. Automat. Control*, 50(7):947–957, 2005.
[32] I. M. Mitchell and C. J. Tomlin. Overapproximating reachable sets by Hamilton-Jacobi projections. *J. Sci. Comput.*, 19(1–3):323–346, 2003. Special issue in honor of the sixtieth birthday of Stanley Osher.

[33] Persistence of Vision Pty. Ltd. Persistence of vision raytracer (version 3.7). Computer software. Retrieved from http://www.povray.org/download/, 2004.
[34] M. Rasmussen, J. Rieger, and K. N. Webster. Approximation of reachable sets using optimal control and support vector machines. *J. Comput. Appl. Math.*, 311:68–83, 2017.
[35] Gunther Reißig. Computing abstractions of nonlinear systems. *IEEE Trans. Automat. Control*, 56(11):2583–2598, 2011.
[36] W. Riedl. Optimization-based subdivision algorithm for reachable sets. *Proc. Appl. Math. Mech.*, 14(1):937–938, Dezember 2014.
[37] R. T. Rockafellar and R. J.-B. Wets. *Variational analysis*, volume 317 of *Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]*. Springer-Verlag, Berlin, 1998.
[38] E. Roxin. Stability in general control systems. *J. Differ. Equ.*, 1(3):115–150, 1965.
[39] The Numerical Algorithms Group (NAG). The NAG Fortran Library. http://www.nag.com/.
[40] V. M. Veliov. Second order discrete approximations to strongly convex differential inclusions. *Systems Control Lett.*, 13(3):263–269, 1989.
[41] A. Wächter and L. T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Math. Program. Ser. A*, 106(1):25–57, 2006.
[42] P. R. Wolenski. The exponential formula for the reachable set of a Lipschitz differential inclusion. *SIAM J. Control Optim.*, 28(5):1148–1161, 1990.
[43] I. Xausa. *Verification of Collision Avoidance Systems using Optimal Control and Sensitivity Analysis*. PhD thesis, Fakultät für Luft- und Raumfahrttechnik, Universität der Bundeswehr München, Munich, Germany, December 2015. http://nbn-resolving.de/urn/resolver.pl?urn:nbn:de:bvb:706-4394.

(Wolfgang Riedl) Lehrstuhl für Ingenieurmathematik, Mathematisches Institut, Universität Bayreuth, 95440 Bayreuth, Germany
E-mail address: wolfgang.riedl@uni-bayreuth.de

(Robert Baier) Lehrstuhl für Angewandte Mathematik, Mathematisches Institut, Universität Bayreuth, 95440 Bayreuth, Germany
E-mail address: robert.baier@uni-bayreuth.de

(Matthias Gerdts) Ingenieurmathematik, Institut für Mathematik und Rechneranwendung (LRT-1), Universität der Bundeswehr München, 85577 Neubiberg/München, Germany
E-mail address: matthias.gerdts@unibw.de