Kinodynamic planning for robotic manipulators utilising set-based methods

Ryan McGovern and Nikolaos Athanasopoulos

School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast, Northern Ireland. E-mail: {rmcgovern03,n.athanasopoulos}@qub.ac.uk

Abstract— **We revisit the trajectory planning problem for general N-link robotic manipulators. Our approach focuses on characterising the whole set of states that can be transferred to a target set of velocities without violating constraints. To achieve this, we work in the commonly utilized two dimensional projected space of the Lagrangian dynamics on a specific predefined path. The produced dynamics can be viewed as a double integrator, with nonlinearities being pushed into a non-convex and state-dependent input constraint set. We approach the problem as a special *reach-avoid* set computation problem using ordering properties of the trajectories generated by a suitably parameterised state feedback control law. Our results are illustrated in simulation using a realistic model of the UR5 commercial robot.**

I. Introduction

In recent years, and especially in the context of Industry 4.0 [1], there has been growing demand for safer robots [2]. Challenges related to safety, inclusion of temporal specifications and reliability [3]–[5] still remain.

Path planning of robotic manipulators is a challenging task [6], where a collision free path has to be computed in a nonconvex environment in a sufficiently fast time. Given such a path, safe controllers can be computed taking into account constraints on speed, momentum and potential collision energies of the system [7]–[9]. In particular, trajectory planning is concerned with shaping the velocity profile as the robot moves from one configuration to another. Classically, the enforcement of hard constraints on the system dynamics are imposed at this stage to account for the fastest possible movements. The approach taken in our work aligns with the analysis of path dynamics, see, e.g., [10]–[12] for early works related to the generation of time-optimal and minimum energy trajectories of robotic manipulators. Roughly, the manipulator Lagrangian dynamics is projected in a specific path, resulting in a two dimensional abstract space. In our work we also abstract the system in an identical manner, with additionally imposed state dependent constraints. However, instead of solving a single optimal control problem as, e.g., in [10], [11], the main objective of our work is to find the whole admissible region of initial conditions for which a controller exists that can drive the system to a target set in finite time, without violating any input constraints. This set is known, in similar but different contexts, as the *reach-avoid set* [13]–[18].

Providing safety guarantees is not trivial with reach-avoid problems and greatly depends on the nature of the system with many techniques available for a variety of system types [19]–[21]. In our case, after a non-conservative state feedback parameterisation of the control strategy, we establish ordering relations between generated closed-loop trajectories in terms of (i) the initial conditions and (ii) the values of the control input. These properties constitute the building blocks of a procedure that recovers an approximation of the reach-avoid set with a finite set of operations. Compared to other important contributions [22], [23], our approach deals directly with nonlinear dynamics and constraints.

To illustrate our framework, in Figure 1a one can observe a two DOF planer manipulator moving in a circular arc within the workspace. In Figure 1b, the path in the joint space is shown that corresponds to the workspace motion. Figure 1c illustrates the two dimensional space where the dynamics of the joint space are projected, accounting for the pseudo-distance (s) and pseudo-velocity ($\dot{s}$) of the path. In Figure 1c the green region is the set of states for which the robot can remain on the path and reach a target set, shown as a red line segment, without any constraint violation. The blue boundary represents natural system constraints, while the orange curve accounts for additional safety constraints related to, e.g., speed or energy specifications. The purple line within the admissible region is a valid trajectory that allows the robot to successfully follow the path in Figure 1b.

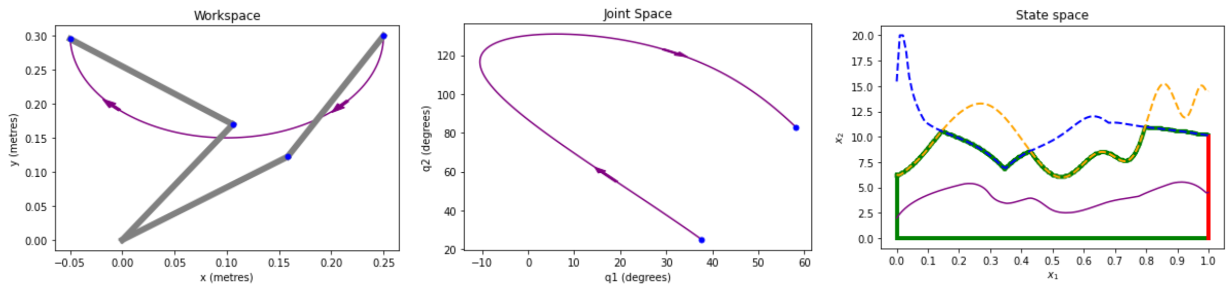
Apart from presenting, to the best of our knowledge, a new framework for trajectory tracking starting from a well-established methodology, we can summarise our contributions as follows: (i) We identify ordering properties between trajectories in the projected space by a new parameterisation of the control input. This constitutes the building block for proposing a convergent algorithm that returns a valid trajectory to a target set. (ii) We compute a reach-avoid set directly, i.e., without needing to discretise the path dynamics. (iii) As a byproduct, new smooth state feedback controllers can be induced that drive the system to a target set.

Notation: For a vector x, $\|x\|$ denotes its 2-norm and x_i its i-th element. For a vector $x_{\text{label}} \in \mathbb{R}^N$ with a label in their subscript, we will write, we write its i^{th} element as $(x_{\text{label}})_i$. For a set $\mathcal{S}$, $\text{conv}(\mathcal{S})$ denotes its convex hull, $\text{bd}(\mathcal{S})$ denotes its boundary and $\text{int}(\mathcal{S})$ denotes its interior. The ball centered at zero with radius ε is denoted by $\mathbb{B}(\varepsilon)$. We write sets with capital letters in italics. $\text{dist}(x, \mathcal{S}) = \min_{y \in \mathcal{S}} \|x - y\|$ denotes the distance of a point x from a compact set $\mathcal{S}$.

II. Preliminaries

We consider serial N-link manipulators, whose dynamics can be described by $M_L(q)\ddot{q} + C_L(q,\dot{q})\dot{q} + g_L(q) = \tau,$

Manuscript 438 submitted to 2022 European Control Conference (ECC).
Received November 8, 2021.



(a) Workspace (b) Joints space (c) s-$\dot{s}$ space

Fig. 1: (a) Workspace representation of a path for a two DOF (manipulator links are in grey color) (b) Joint space representation of the two DOF manipulator (c) State space representation of the constraint curves, natural constraints in blue, extra constraints in orange, target set in red, robot velocity profile in purple.

where $q \in \mathbb{R}^N$ corresponds to joint positions and $\tau \in \mathbb{R}^N$ collects in a vector the actuator torques at each joint. The matrices $M_L(q) \in \mathbb{R}^{N\times N}$, $C_L(q,\dot{q}) \in \mathbb{R}^{N\times N}$ are the inertial and centrifugal/Coriolis matrices that describe how the mass behaves when it is accelerated linearly or moved in circular arcs respectively. The gravitational force effects are collected into $g_L(q)$. The matrix and vector functions $M_L(q)$, $C_L(q)$ and $g_L(q)$ are globally Lipschitz, [24]–[26].

The Lagrangian dynamics can be projected to path dynamics: The first step is to consider an end effector path described by a function $q(s) : [0,1] \to \mathbb{R}^n$ known as a path parameterisation. Details can be found in the seminal papers [10], [11] and the book [27, Chapter 9].

Assumption 1 *The path parameterisation $q(s)$ is two times differentiable with respect to $s, \dot{s}$. Moreover, the parameterisation satisfies*

$$\left\| \frac{dq(s)}{ds} \right\| \geq \alpha_0,$$

for some constant positive number $\alpha_0 > 0$.

Assumption 1 is not very restrictive and is standard in the literature [10], [11], [27]. Intuitively, it requires that the configuration must be able to change such that the end effector can traverse a path, and that the rate at which the configuration is changing is also variable. The parameterised system is $M(s)\ddot{s} + C(s)\dot{s}^2 + g(s) = \tau$ where $M(s) \in \mathbb{R}^N$, $C(s) \in \mathbb{R}^N$, $g(s) \in \mathbb{R}^N$, are the transformed vectors, and $\tau \in \mathbb{R}^N$ describes the torques and forces produced by the actuators. We define $x = \begin{bmatrix} s & \dot{s} \end{bmatrix}^T = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^\top$, to discuss the system state. The torques are limited by the actuator dynamics as $\tau_i^{\min}(x) \leq \tau_i(x) \leq \tau_i^{\max}(x)$ where $\tau_i^{\min}(x)$ and $\tau_i^{\max}(x)$ describe the maximum torque that can be generated in either direction. The i^{th} element, $i = 1, ..., N$, of the above system generates constraints of the form

$$\tau_i^{\min}(x) \leq M_i(x_1)\dot{x}_2 + C_i(x_1)x_2^2 + g_i(x_1) \leq \tau_i^{\max}(x). \quad (1)$$

Assumption 2 *The state-dependent constraint bounds $\tau_i^{\min}(x)$ and $\tau_i^{\max}(x)$ are locally Lipschitz in x for all $i = 1, ..., N$.*

Assumption 2 is not generally restrictive for many electrical motors due to their continuous relationships between torque and speed, see, e.g., [28] [29, Chapter 10]. Rearranging (1), the limits on possible accelerations $\ddot{s}$ from any given state x can be formed as $A_i(x) = \frac{\tau_i^{\min}(x) - C_i(x_1)x_2^2 - g_i(x_1)}{M_i(x_1)}$,, $D_i(x) = \frac{\tau_i^{\max}(x) - C_i(x_1)x_2^2 - g_i(x_1)}{M_i(x_1)}$ for $M_i(x_1) < 0$, and $A_i(x) = \frac{\tau_i^{\max}(x) - C_i(x_1)x_2^2 - g_i(x_1)}{M_i(x_1)}$, $D_i(x) = \frac{\tau_i^{\min}(x) - C_i(x_1)x_2^2 - g_i(x_1)}{M_i(x_1)}$ for $M_i(x_1) > 0$. The case of zero inertia points, i.e., when $M_i(x_1) = 0$, for one or more indices $i = 1, .., N$ is taken into account by posing constraints acting directly on x [27]. This leads to inequality constraints $\tau_i^{\min}(x) \leq C_i(x_1)x_2^2 + g_i(x_1) \leq \tau_i^{\max}(x)$. By grouping the above, we can write

$$A(x) = \min_{\{i \in [1,N] : M_i(x) \neq 0\}} A_i(x), \quad (2)$$

$$D(x) = \max_{\{i \in [1,N] : M_i(x) \neq 0\}} D_i(x), \quad (3)$$

$$D(x) \leq \ddot{s} \leq A(x) \quad (4)$$

Under the aforementioned path parameterisation, we can formulate the path dynamics

$$\dot{x} = Ax + Bu(x), \quad (5)$$

with $A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$, $B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$. The state feedback $u(\cdot) : \mathbb{R}^2 \to \mathbb{R}$ accounts for the acceleration $\ddot{s}$, whose admissible range is state-dependent (4). We consider constraints

$$x \in \mathcal{X}', \quad u(x) \in \mathcal{U}_x, \quad (6)$$

where $\mathcal{X}' = \mathcal{C} \cap \mathcal{C}_0$, with $\mathcal{C} = \{x \in \mathbb{R}^2 : B(x) \leq 0\}$, $\mathcal{C}_0 = \{x \in \mathbb{R}^2 : (\exists j \in [1,...,N] : M_j(x_1) = 0, \quad \tau_j^{min} - C_j(x)x_2^2 - g_j(x) \leq 0, \quad -\tau_1^{max} + C_1(x)x_2^2 + g_1(x) \leq 0)\}$, and $\mathcal{U}_x = \{u(x) \in \mathbb{R} : D(x) \leq u(x) \leq A(x)\}$. The vector function $B(x)$ is $B(x) = \begin{bmatrix} -x_1 & x_1 - 1 & -x_2 & D(x) - A(x) \end{bmatrix}^\top$

We are now in a position to state the question we are addressing in this paper: Given an N-link robotic manipulator and a pre-defined path leading to the projected path dynamics (5), state and input constraints (6), which is the set of initial conditions for which an admissible state feedback controller exists such that the system (5) trajectory is transferred to a target set in finite time?

Manuscript 438 submitted to 2022 European Control Conference (ECC).
Received November 8, 2021.

III. Control parameterisation and properties

In this section, we establish Lipschitz continuity properties of the dynamics of the investigated dynamics, together with a parameterisation of the control input that will be instrumental in the developement of our results. The following simple facts are presented without proof.

Fact 1 *Suppose functions $f_1(\cdot) : \mathbb{R}^n \to \mathbb{R}$ and $f_2(.) : \mathbb{R}^n \to \mathbb{R}$ are locally Lipschitz continuous in a set $\mathcal{S} \subset \mathbb{R}^n$, with Lipschitz constants L_1 and L_2 respectively. The following hold:*

(i). *$f_3(x) = f_1(x) + f_2(x)$ is locally Lipschitz in $\mathcal{S}$.*
(ii). *$f_4(x) = f_1(f_2(x))$ is locally Lipschitz in $\mathcal{S}$.*
(iii). *Suppose that $f_1(x)$, $f_2(x)$ are bounded in $\mathcal{S}$. Then, $f_5(x) = f_1(x)f_2(x)$ is locally Lipschitz in $\mathcal{S}$.*
(iv). *Suppose that $|f_1(x)| > a$, $x \in \mathcal{S}$. Then the function $f_6(x) = \frac{1}{f_1(x)}$ is locally Lipschitz in $\mathcal{S}$.*
(v). *Consider $f_i(x)$, $i = 1, ..., N$ and the function $f(x) = \max_i\{f_i\}$. Let $f_i(x)$ be Locally Lipschitz in $\mathcal{S}_i = \{x \in \mathcal{S} : f_i(x) \geq f_j(x), \forall j = 1, .., N, j \neq i\}$. Then, f is locally Lipschitz in $\mathcal{S}$.*
(vi). *Consider $f_i(x)$, $i = 1, ..., N$ and the function $f(x) = \min_i\{f_i\}$. Let $f_i(x)$ be locally Lipschitz in $\mathcal{S}_i = \{x \in \mathcal{S} : f_i(x) \leq f_j(x), \forall j = 1, .., N, j \neq i\}$. Then, f is locally Lipschitz in $\mathcal{S}$.*

Lemma 1 *Under Assumption 1, for all $x \in \mathcal{X}'$ there exists at least one index $i^* \in \{1, 2, ..., N\}$ such that $||M_{i^*}(x_1)|| \geq \alpha_1$, for some $\alpha_1 > 0$.*

Proof By construction, it holds that $M(x_1) = M_L(x_1)\frac{dq(x_1)}{dx_1}$. Moreover, the symmetric mass matrix $M_L(x_1)$ is positive definite for all $x \in \mathcal{X}'$. Let us consider an arbitrary $y = \frac{dq(x_1)}{dx_1}$, such that $||y|| = \alpha_0$. We consider z_i, $i = 1, ..., N$, to be the normalised eigenvectors of $M_L(x_1)$ such that. $||z_i|| = \alpha_0, i = 1, .., N$. The eigenvectors z_i are mutually orthogonal as $M_L(x)$ is symmetric. Consequently, we can express y in the base of eigenvectors $y = \sum_{i=1}^N c_i \hat{z}_i$, with $c_i \geq 0$, $\sum_{i=1}^N c_i \geq 1$ and $\hat{z}_i = z_i$, or $\hat{z}_i = -z_i$. The last inequality holds since $\|y\| = \|z_i\|$. It follows that $M_L(x_1)y = \sum_{i=1}^N \lambda_i c_i z_i$, where $\lambda_i > 0$ are the eigenvalues of $M_L(x_1)$. Consider the set $\mathcal{S} = \operatorname{conv}(\pm z_1, .., \pm z_N)$. We can write $\mathcal{S} = \{x \in \mathbb{R}^N : \|Px\|_1 \leq 1\}$, where $P = \begin{bmatrix} z_1 & \cdots & z_N \end{bmatrix}^{-\top}$. Setting $d = \sum_{i=1}^N \lambda_i c_i$ we can write $M_L(x_1)y = d\sum_{i=1}^N \left(\frac{\lambda_i c_i}{d}\right) z_i$, or, $M_L(x_1)y \in d\,\mathrm{bd}(\mathcal{S})$. Since by norm equivalence we have $\|x\|_1 \leq N\|x\|_2$, it follows that $d\mathcal{S} \supseteq \mathbb{B}(\frac{1}{dN\sigma_{\max}(P)})$, thus, $|M_L(x_1)y|_2 \geq \frac{1}{dN\sigma_{\max}(P)}$, where $\sigma_{\max}(P)$ is the largest singular value of P. Consequently, there necessarily exists at least one element of $M_L(x_1)y$ such that $|M_L(x_1)y| \geq \alpha_1$ for some finite α_1 and for all $y \in \mathbb{B}(\alpha_0)$. The result for $\|y\| > \alpha_0$ follows directly by the homogeneity of the linear mapping. ∎

Lemma 2 *The functions $A(x)$, $D(x)$ are locally Lipschitz continuous in $\mathcal{X}'$.*

Proof sketch The proof of Lemma 2 is based on Lemma 1 and the properties outlined in Fact 1 combined with the fact that $M_L(\cdot)$, $C_L(\cdot)$ and $g_L(\cdot)$ are locally Lipschitz functions [24]–[26]. Roughly, we can show that for each $x \in \mathcal{X}$, both functions $A(x)$, $D(x)$ are finite, and in particular correspond to some index $i \in [1, N]$ for which $|M_i|$ is bounded from below. Moreover, both $A(x)$, $D(x)$ can be considered as sums and products of locally Lipschitz functions, thus reaching the result. We note a formal proof will be included in an extended version of our work for completeness, and is omitted here due to space limitations and its straightforward nature.

Within our framework we can incorporate additional constraints, leading to a set $\mathcal{X} \subseteq \mathcal{X}'$, defined below,

$$\mathcal{X} = \left\{x \in \mathbb{R}^2 : C^u(x_1) \leq x_2 \leq C^l(x_1)\right\}. \tag{7}$$

We organise the constraints into two functions, $C^u(x_1) : \mathbb{R} \to \mathbb{R}$ and $C^l(x_1) : \mathbb{R} \to \mathbb{R}$ which represent upper and lower bound constraints respectively. The sets representing the upper and lower boundary of the $\mathcal{X}$ are defined below

$$\mathcal{V}^u = \{x \in \mathcal{X} : C^u(x_1) = x_2\}, \tag{8}$$
$$\mathcal{V}^l = \left\{x \in \mathcal{X} : C^l(x_1) = x_2\right\}. \tag{9}$$

Assumption 3 *The functions $x_2 = C^u(x_1)$ and $x_2 = C^l(x_1) = x_2$ are continuous, and bijections.*

We note this assumption might be restrictive as, e.g., islanding may appear in the constraint set. Nevertheless, it covers many realistic cases including polynomial and trigonometric constraints.

In agreement with approaches in the literature [10], [27], we make the choice to set $u(x) = \dot{x_2}$ as an input variable as this acceleration can be set via the actuators. We further parameterise the input using an *actuation level function* $\lambda(x) : \mathcal{X} \to [0, 1]$ as

$$u(x, \lambda(x)) = D(x) + \lambda(x)(A(x) - D(x)). \tag{10}$$

We state the following Corollary, which follows straightforwardly by combination of Lemma 2 and Fact 1 (i), (iii).

Corollary 1 *Let $\lambda(x) : \mathcal{X} \to [0, 1]$ be a locally Lipschitz continuous function, then* (10) *is locally Lipschitz continuous in $\mathcal{X}$.*

The state space equations for the system can now be written for the forward and backwards dynamics as

$$\dot{x} = Ax + Bu(x, \lambda(x)), \tag{11}$$
$$\dot{x} = -Ax - Bu(x, \lambda(x)), \tag{12}$$

respectively, where A, B are defined in (5).

IV. Trajectories ordering properties

Given a particular choice of the actuation level $\lambda(x)$ (10) and an initial condition $x_0 \in \mathcal{X}$, the solution of (11) is $\phi_{\mathrm{f}}(t;x_0,\lambda(x)) = e^{At}x_0 + \int_0^t e^{A(t-\tau)}Bu(x,\lambda(x))d\tau$. Correspondingly, solution of the backwards dynamics (12) is $\phi_{\mathrm{b}}(t;x_0,\lambda(x)))\} = e^{-At}x_0 - \int_0^t e^{-A(t-\tau)}Bu(x,\lambda(x))d\tau$.

We define the trajectory of the closed loop system in the forward direction as

$$\mathcal{T}_{\mathrm{f}}(x_0,\lambda(x)) = \{x \in \mathcal{X} : (\exists t \geq 0 : x = \phi_{\mathrm{f}}(t;x_0,\lambda(x)))\}, \tag{13}$$

while the trajectory formed by the backward dynamics is represented by

$$\mathcal{T}_{\mathrm{b}}(x_0,\lambda(x)) = \{x \in \mathcal{X} : (\exists t \geq 0 : x = \phi_{\mathrm{b}}(t;x_0,\lambda(x)))\}. \tag{14}$$

In the remainder, $\mathcal{T}(x_0,\lambda(x))$ is used to describe a trajectory given by either (13) or (14). This is useful when discussing trajectories without the need to be specific about the direction in which they are formed. Moreover, we define a slice of a set to be used within the following two lemmas

$$\mathcal{X}_v(n_1^*) = \mathcal{X} \cap \{x : x_1 = n_1^*\}. \tag{15}$$

Lemma 3 *Consider an $n_1^* \in [0,1]$ and the slice $\mathcal{X}_v(n_1^*)$. Consider two states $x_u, x_l \in \mathcal{X}_v(x_1^*)$ such that $(x_u)_2 > (x_l)_2$, and a constant actuation level $\lambda(x) = \lambda \in [0,1]$. Then,*

$$\mathcal{T}(x_u,\lambda) \cap \mathcal{T}(x_l,\lambda) = \emptyset.$$

Proof By Corollary 1, $u(x)$ is locally Lipschitz continuous in $\mathcal{X}$. Consequently, by Fact 1 (i), (iii) the backward dynamics $f(x) = -Ax - Bu(x)$ is also locally Lipschitz continuous. Therefore by the Picard–Lindelöf theorem the solution of $\dot{x} = f(x)$ is unique for a given x [30] [31]. Suppose an intersection between two trajectories occurs at x, the backward dynamics must produce two solutions which is a contradiction, thus the result is shown. ■

Lemma 4 *Consider a number $n_1^* \in [0,1]$ and two states $x_u, x_l \in \mathcal{X}_v(n_1^*)$ such that $x_u \neq x_l$. Consider two fixed actuation levels $\lambda_u, \lambda_l \in [0,1]$, with $\lambda_u > \lambda_l$. Then, the trajectories can intersect at most once in* $\mathrm{int}\,(\mathcal{X})$*, i.e., either $\mathcal{T}(x_u,\lambda_u) \cap \mathcal{T}(x_l,\lambda_l) = \{x^\star\}$, for some $x^\star \in \mathcal{X}$, or $\mathcal{T}(x_u,\lambda_u) \cap \mathcal{T}(x_l,\lambda_l) = \emptyset$.*

Proof Suppose there are two trajectories that intersect defined as $\mathcal{T}_u = \mathcal{T}_u(x_u,\lambda_u)$ and $\mathcal{T}_l = \mathcal{T}_l(x_l,\lambda_l)$ with $\mathcal{T}_l \cap \mathcal{T}_u = \{x^*\}$. We describe these two trajectories with four pieces that emanate from x^*, namely, as $\mathcal{T}_u = \mathcal{T}_{b,u} \cup \mathcal{T}_{f,u}$ and $\mathcal{T}_l = \mathcal{T}_{b,l} \cup \mathcal{T}_{f,l}$. Specifically, $\mathcal{T}_{b,u} = \mathcal{T}_b(x^*,\lambda_u)$, $\mathcal{T}_{f,u} = \mathcal{T}_f(x^*,\lambda_u)$ and $\mathcal{T}_{b,l} = \mathcal{T}_b(x^*,\lambda_l)$, $\mathcal{T}_{f,l} = \mathcal{T}_f(x^*,\lambda_l)$. We observe that for all $x \in int(\mathcal{X})$, $\frac{\partial u(x,\lambda)}{\partial \lambda} = A(x) - D(x) > 0$. Thus, $\lambda_u > \lambda_l$ implies $u(x^*,\lambda_u) > u(x^*,\lambda_l)$.

Let $\dot{x} = f(x)$ as given by (11). Consider the gradients defined as $f^u = f(x^*,\lambda_u) = \begin{bmatrix} x_2^* & u(x^*,\lambda_u)\end{bmatrix}^\top$ and $f^l = f(x^*,\lambda_l) = \begin{bmatrix} x_2^* & u(x^*,\lambda_l)\end{bmatrix}^\top$. We consider the *angle of emanation* from x^* for some vector $f \in \mathbb{R}^2$ as $\theta(f) = \tan^{-1}(\frac{f_2}{f_1})$. At intersection it holds that $f_1^u = f_1^l$, and $f_2^u > f_2^l$, thus $\frac{f_2^u}{f_1^u} > \frac{f_2^l}{f_1^l}$, or, $\theta(f^u) > \theta(f^l)$. Consequently, for any $\delta > 0$ small enough and the slice $\mathcal{X}_f = \mathcal{X}_v(x_1^* + \delta)$ with $\bar{x}_u = \mathcal{T}_{f,u} \cap \mathcal{X}_f$ and $\bar{x}_l = \mathcal{T}_{f,l} \cap \mathcal{X}_f$, it holds that $\bar{x}_{u,2} > \bar{x}_{l,2}$.

Similarly, for the slice $\mathcal{X}_b = \mathcal{X}_v(x_1^* - \delta)$ with $\tilde{x}_u = \mathcal{T}_{b,u} \cap \mathcal{X}_b$ and $\tilde{x}_l = \mathcal{T}_{b,l} \cap \mathcal{X}_b$ it holds that $\tilde{x}_{u,2} < \tilde{x}_{l,2}$. The proof is complete if one considers that the trajectories are continuous and thus a second intersection cannot happen. ■

Lemma 4 shows that if two trajectories evolve from a single state with different constant actuation levels the one with the higher actuation level will remain above the other for the duration the system runs. Specifically, for any $x_1^\star \in [0,1]$, any $\lambda_b \leq \lambda_a \leq 1$ it holds that $x_b \leq x_a$, where $x_a = \mathcal{X}(x_1^*) \cap \mathcal{T}_f(x_0,\lambda_a)$, $x_b = \mathcal{X}(x_1^*) \cap \mathcal{T}_f(x_0,\lambda_b)$.

Moreover, suppose that we consider two trajectories $\mathcal{T}_1 = \mathcal{T}(x_u,\lambda_u), \mathcal{T}_2 = \mathcal{T}(x_l,\lambda_l)$, such that x_u, x_l lie on a slice $\mathcal{X}_v(x_1)$ defined by equation (15) such that $x_u \neq x_l$ and $\lambda_u > \lambda_l$. Lemma 4 shows that in the case where $x_{u,2} > x_{l,2}$, we can guarantee that $\mathcal{T}_1 \cap \mathcal{T}_2 = \emptyset$.

V. Approximation of the Reach-avoid set

Lemmas 3 and 4 provide core ordering properties between trajectories that are used in the development of the algorithm to produce an approximation of the reach-avoid set. We first define the target set $\mathcal{X}_{\mathrm{T}}$ as an interval within $\mathcal{X}$, i.e.,

$$\mathcal{X}_{\mathrm{T}} = \{x \in \mathcal{X} : x_1 = c, x_u^* \leq x_2 \leq x_l^*\}. \tag{16}$$

where $c \leq 1$. We define the reach-avoid set to be the set of states that can be transferred to a target set in finite time via state feedback, without violating the state and input constraints. Formally, the reach-avoid set is defined below.

Definition 1 *Consider the system* (11)*, the constraint set $\mathcal{X}$* (7)*, a target set $\mathcal{X}_T$* (16) *and the control parameterisation* (10)*. The reach-avoid set is*

$$\begin{aligned}\mathcal{R}(\mathcal{X}_T) = &\{x_0 \in \mathcal{X} : (\exists\lambda(\cdot), \exists t^* > 0 : \phi_f(t^*,x_0,\lambda(x)) \in \mathcal{X}_T,\\ &\phi_f(t,x_0,\lambda(x)) \in \mathcal{X}, \forall t \in [0,t^*])\}.\end{aligned} \tag{17}$$

Next, we define the sets of states that lie above or below a set $\mathcal{Z}$ in $\mathcal{X}$.

Definition 2 *Consider a set $\mathcal{Z} \subseteq \mathcal{X}$. We define the following sets*

$$\mathcal{R}_u(\mathcal{Z}) = \{x \in \mathcal{X} : (\exists y \in \mathcal{Z} : x_1 = y_1, x_2 \leq y_2\},$$
$$\mathcal{R}_l(\mathcal{Z}) = \{x \in \mathcal{X} : (\exists y \in \mathcal{Z} : x_1 = y_1, x_2 \geq y_2\}.$$

Next, we define the set-valued mapping $\mathcal{Q}(\cdot)$ that returns the 'leftmost' state of a trajectory $\mathcal{T} = \mathcal{T}(x_0,\lambda(x))$

$$\mathcal{Q}(\mathcal{T}) = \{x^* \in \mathcal{X} \cap \mathcal{T} : x_1^* \leq x_1, \forall x \in \mathcal{X} \cap \mathcal{T}\}. \tag{18}$$

We are now in a position to present Algorithm 1 for deriving an approximation of $\mathcal{R}(\mathcal{X}_{\mathrm{T}})$. Initially, the two trajectories T_l^* and T_u^* are produced from the extreme points of $\mathcal{X}_{\mathrm{T}}$ denoted are computed in Lines 3 and 5 respectively. Lines 4 and 6 provide the points where the boundary is intersected at x_a, x_d. The vectors x_u^* and x_l^* represent the upper and

Algorithm 1 Approximation of the reach-avoid set

```
1: Input : constraint sets $\mathcal{X}$, $\mathcal{V}^l$, $\mathcal{V}^u$ target set $\mathcal{X}_\mathrm{T}$, $\epsilon > 0$
2: Output : approximation $\mathcal{R}_\epsilon(\mathcal{X}_\mathrm{T})$ of the reach-avoid set
3: $\mathcal{T}_u^* \leftarrow \mathcal{T}_\mathrm{b}(x_u^*, 0)$,
4: $\{x_d\} \leftarrow \mathcal{Q}(\mathcal{T}_u^*)$
5: $\mathcal{T}_l^* \leftarrow \mathcal{T}_\mathrm{b}(x_l^*, 1)$
6: $\{x_a\} \leftarrow \mathcal{Q}(\mathcal{T}_l^*)$
7: if $x_a \in \mathcal{V}^l$ and $x_d \in \mathcal{V}^l$ then
8:     $\mathcal{Z}_l \leftarrow \mathcal{T}_l^* \cup \text{extend}(\mathcal{V}^l, [(x_d)_1, (x_a)_1], 1, \epsilon)$
9:     $\mathcal{Z}_u \leftarrow \mathcal{T}_u^*$
10: else if $x_a \in \mathcal{V}^u$ and $x_d \in \mathcal{V}^u$ then
11:     $\mathcal{Z}_u \leftarrow \mathcal{T}_u^* \cup \text{extend}(\mathcal{V}^u, [(x_d)_1, (x_a)_1], 0, \epsilon)$
12:     $\mathcal{Z}_l \leftarrow \mathcal{T}_l^*$
13: else
14:     if $x_a \in \mathcal{V}^l$ then
15:         $\mathcal{Z}_l \leftarrow \mathcal{T}_l^* \cup \text{extend}(\mathcal{V}^l, [0, (x_a)_1], 1, \epsilon)$
16:     else
17:         $\mathcal{Z}_l \leftarrow \mathcal{T}_l^*$
18:     if $x_d \in \mathcal{V}^u$ then
19:         $\mathcal{Z}_u \leftarrow \mathcal{T}_u^* \cup \text{extend}(\mathcal{V}^u, [0, (x_d)_1], 0, \epsilon)$
20:     else
21:         $\mathcal{Z}_u \leftarrow \mathcal{T}_u^*$
22: return $\mathcal{R}_\epsilon(\mathcal{X}_\mathrm{T}) \leftarrow \mathcal{R}_l(\mathcal{Z}_u) \cap \mathcal{R}_u(\mathcal{Z}_l)$
```

lower extremes of $\mathcal{X}_\mathrm{T}$ respectively, namely, $(x_u^*)_1 = (x_l^*)_1$ and $(x_u^*)_2 \geq (x_l^*)_2$.

Lines 7, 10 and 13 represent the cases that can occur in relation to the location x_d and x_a. Figure 2 illustrates all possible situations. In particular, condition in Line 7 covers the situation illustrated by Figure 2a, where x_d, x_a are in $\mathcal{V}^l$. In this case the upper boundary requires no additional extension, thus, $\mathcal{Z}_u = \mathcal{T}_u^*$. On the other hand, the lower bound needs extended to obtain $\mathcal{Z}_l$.

Line 10 covers the situation illustrated by Figure 2b where x_d, x_a are in $\mathcal{V}^u$. In this situation the lower bound is complete, thus, $\mathcal{Z}_l = T_l^*$. On the other hand, the upper bound needs to be extended to obtain $\mathcal{Z}_u$.

The Lines 13-21 cover all the cases where $\mathcal{T}_u^*$ and $\mathcal{T}_l^*$ do not intersect the same boundary $\mathcal{V}^u$ or $\mathcal{V}^l$ as illustrated by Figures figures 2c - 2f and applies the extend function to the relevant sections, which is described in detail below.

After all the relevant extensions have been made Line 22 returns $\mathcal{R}_\epsilon(\mathcal{X}_\mathrm{T})$ which is constructed as the intersection of the sets $\mathcal{R}_l(\mathcal{Z}_u)$, and $\mathcal{R}_u(\mathcal{Z}_l)$.

We note the overall objective of this algorithm is to produce a set $\mathcal{R}_\epsilon(\mathcal{X}_\mathrm{T})$ in which the target interval $\mathcal{X}_\mathrm{T}$ represents the only *admissible exit facet*. This *admissible exit facet* represents the area of the boundary that through which some trajectory $\mathcal{T}_f(x_0, \lambda(x))$ may escape in finite time, under appropriate control $\lambda(x)$ emanating from $x_0 \in \mathcal{R}_\epsilon(\mathcal{X}_\mathrm{T})$. Several existing works on *control-to-facet* strategies focus on non-linear hybrid systems with sets defined by simplices or other simple polynomial descriptions of the set boundaries [32]–[34].

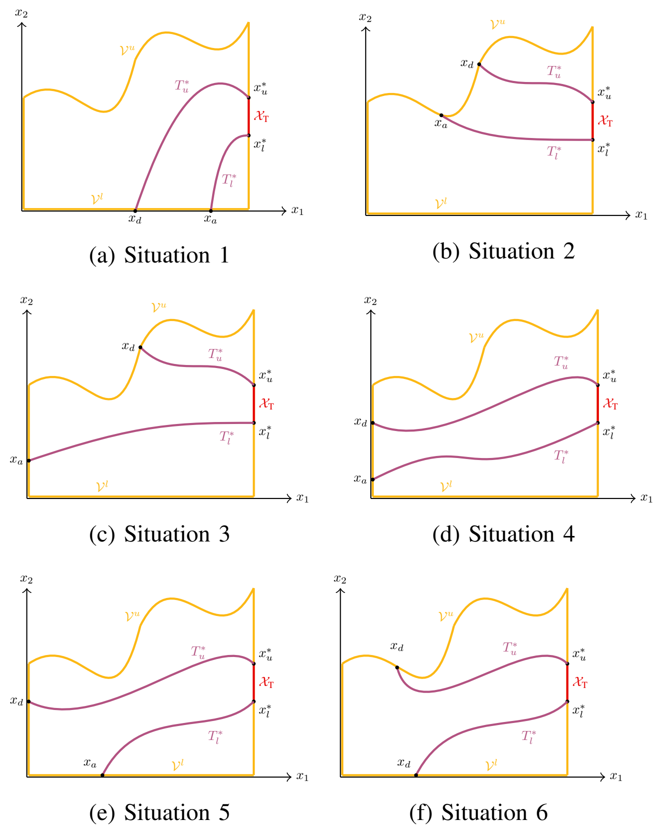


Fig. 2: Illustration of the possible configurations that can result from Lines 3-5 from Algorithm 1.

A. *Extension of bounds of $\mathcal{R}_\epsilon(\mathcal{X}_T)$*

The procedure extend($\mathcal{V}$, $[(x_\text{end})_1, (x_\text{start})_1]$, λ, ϵ) is used in Algorithm 1 on Lines 8, 11, 15, and 19. It relates to extending the upper and lower bounds of $\mathcal{R}_\epsilon(\mathcal{X}_\mathrm{T})$ in the cases where x_a or x_d lie on $\mathcal{V}^u$ or $\mathcal{V}^l$. Within the procedure, a set of states $\mathcal{Z}$ are constructed in the x_1 interval $[(x_\text{end})_1, (x_\text{start})_1]$. Roughly, the set $\mathcal{Z}$ is a curve that can either be followed exactly or not crossed by any trajectory $\mathcal{T}(x_0, \lambda(x))$, with $x_0 \in \mathcal{R}_\epsilon(\mathcal{X}_\mathrm{T})$, when $\lambda(x)$ is suitably chosen.

In particular, $\mathcal{Z}$ is constructed from a union of extreme trajectories and segments of $\mathcal{V}$ via back propagation of the system dynamics. In order to decide how the algorithm will run we first present a method to analyse how trajectories emanating from states on $\mathcal{V}$ will behave. To this purpose, we utilise Bouligand's tangent cone, see. e.g., [35, page 122].

Definition 3 *Consider the set $\mathcal{X}$* (7). *The tangent cone to $\mathcal{X}$ at a vector $x \in \mathcal{X}$ is*

$$\mathcal{B}_v(x) = \left\{ z : \liminf_{\tau \to 0} \frac{dist(x + \tau z, \mathcal{X})}{\tau} = 0 \right\}. \qquad (19)$$

Next, we proceed by expressing the system dynamics (11) as an inclusion. In detail, we consider the dynamics given by (11) and write them as $\dot{x} = f(x, u(x)) = \begin{bmatrix} x_2 & u(x) \end{bmatrix}^\top$, with $u(x) \in \mathcal{U}_x$.

Definition 4 *Consider a state $x \in \mathcal{V}$ and the system $\dot{x} = f(x, u(x))$. We define the set $\mathcal{F}(x, u(x))$*

$$\mathcal{F}(x,u(x)) = \{y \in \mathbb{R}^2 : y = f(x,u(x)), u(x) \in \mathcal{U}_x, \\ \|f(x,u(x))\| > 0\}, \tag{20}$$

where $\mathcal{U}_x$ is defined in (6).

We can utilise (19), (20) to create the condition verify whether

$$\mathcal{B}_v(x) \cap \mathcal{F}(x,u(x)) \neq \emptyset. \tag{21}$$

In essence, condition (21) verifies if for a particular state x on the boundary of $\mathcal{X}$ there exists an input that will allow the system to remain in $\mathcal{X}$.

It is worth noting that in our setting we can straightforwardly verify algebraically (21) using the extreme inputs $A(x)$ when x is on $\mathcal{V}^l$, or $D(x)$ when x is on $\mathcal{V}^u$. In particular, it is possible to define a function $S(x)$ as the inner product of the system dynamics (11) with the relevant extreme input, and a normal vector to the boundary at this point to form $S(x)$. In this case, $S(x) \leq 0$ implies nonempty intersection of (21), while $S(x) > 0$ implies the opposite.

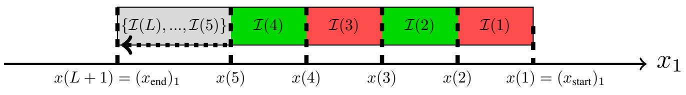
To consider the entire boundary $\mathcal{V}$ of the constraint set $\mathcal{X}$, we identify the intervals where (21) holds.

Definition 5 *Consider an ordered set of intervals $\mathcal{I} = \{\mathcal{I}(1), ..., \mathcal{I}(L)\}$, $\mathcal{I}(i) \subseteq [(x_{end})_1, (x_{start})_1]$, $i = 1, .., L$. Let $\mathcal{I}$ be constructed from two sets of intervals, namely, $\mathcal{I}_{in} = \{\mathcal{I}_{in}(1), ..., \mathcal{I}_{in}(M)\}$ and $\mathcal{I}_{out} = \{(\mathcal{I}_{out})(1), ..., \mathcal{I}_{out}(P)\}$, such that $|\mathcal{I}_{in}| + |\mathcal{I}_{out}| = M + P = L$, with $\mathcal{I}_{in}(i) \neq \mathcal{I}_{out}(j)$, for all $i = 1, ..., M, j = 1, ..., P$. Moreover,*

$$x \in (\mathcal{I}_{in})_i \Rightarrow \mathcal{B}_v(x) \cap F(x,u(x)) \neq \emptyset$$
$$x \in (\mathcal{I}_{out})_j \Rightarrow \mathcal{B}_v(x) \cap F(x,u(x)) = \emptyset,$$

for all $i = 1, ..., M$, $j = 1, ..., P$.

Remark 1 *For convenience, $\mathcal{I}$ is an ordered set of intervals such that each interval is addressable with $i \in [1, L]$. We define the order such that that we have the interval $[(x_{end})_1, (x_{start})_1]$ described in terms of alternating I_{in} and I_{out} intervals as illustrated in Figure 3.*



$\{\mathcal{I}(L), ..., \mathcal{I}(5)\}$
$\mathcal{I}(4)$
$\mathcal{I}(3)$
$\mathcal{I}(2)$
$\mathcal{I}(1)$
x_1
$x(L+1) = (x_{\text{end}})_1$
$x(5)$
$x(4)$
$x(3)$
$x(2)$
$x(1) = (x_{\text{start}})_1$

Fig. 3: Illustration of an example of how the intervals are to be ordered green implies the interval is in $\mathcal{I}_{\text{in}}$, red implies it is in $\mathcal{I}_{\text{out}}$, gray implies the remaining collection of intervals from $\mathcal{I}$.

The extension procedure is presented in Algorithm 2. The algorithm contains a loop that checks all generated interval. Lines 8 -18 deal intervals in $\mathcal{I}_{in}$, and Line 20-23 deal with intervals in $\mathcal{I}_{out}$. The loop ends when we have run out of intervals.

We are in a position to state the main result of our work.

Algorithm 2 Extend bound, extend($\mathcal{V}$, $[(x_{\text{end}})_1, (x_{\text{start}})_1]$, λ, ϵ)

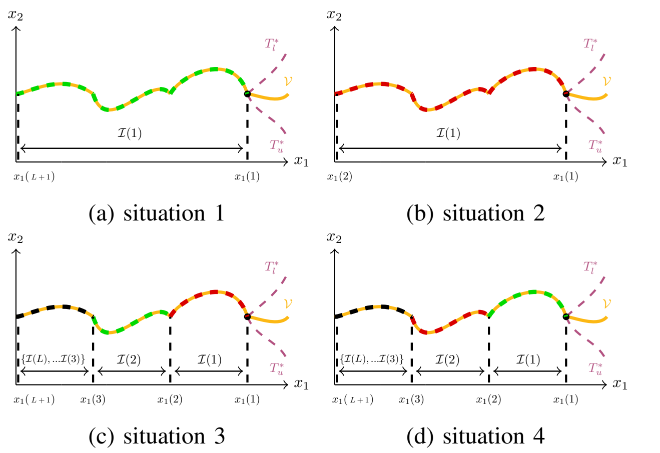
```
1:  Input : Set $\mathcal{V}$ (either $\mathcal{V}^u$ (8) or $\mathcal{V}^l$ (9)), start state $x_{\text{start}}$, end state $x_{\text{end}}$, step size $\varepsilon > 0$, $\delta = (-1)^{\lambda+1}\varepsilon$.
2:  Output : set $\mathcal{Z}$
3:  $\mathcal{Z} \leftarrow \emptyset$
4:  $i \leftarrow 1$
5:  $y \leftarrow \{x \in \mathcal{V} : x_1 = (x_{\text{start}})_1\}$
6:  while $i < L + 1$ do
7:      if $\mathcal{I}(i) \in \mathcal{I}_{\text{in}}$ then
8:          if $y \in \mathcal{V}$ then
9:              $\mathcal{Z} \leftarrow \mathcal{Z} \cup \{x \in \mathcal{V} : (\exists y \in \mathcal{I}(i) : x_1 = y_1)\}$
10:         else
11:             $\mathcal{T}_\mathcal{I} = \mathcal{T}_b(y, \lambda) \cap \{x \in \mathbb{R}^2 : (\exists y \in \mathcal{I}(i) : x_1 = y_1)\}$
12:             if $\mathcal{T}_\mathcal{I} \cap \mathcal{V} \neq \emptyset$ then
13:                 $x_{\text{int}} \leftarrow \mathcal{Q}(\mathcal{T}_\mathcal{I} \cap \mathcal{V})$
14:                 $\mathcal{T}_{\text{int}} \leftarrow \mathcal{T}_\mathcal{I} \cap \{x \in \mathcal{T}_\mathcal{I} : x_1 \geq x_{\text{int}}\}$
15:                 $\mathcal{V}_{\text{int}} \leftarrow \{x \in \mathcal{V} : x_1 \leq x_{\text{int}}, x \in \mathcal{I}(i)\}$
16:                 $\mathcal{Z} \leftarrow \mathcal{Z} \cup \mathcal{T}_{\text{int}} \cup \mathcal{V}_{\text{int}}$
17:             else
18:                 $\mathcal{Z} \leftarrow \mathcal{Z} \cup \mathcal{T}_\mathcal{I}$
19:     else if $\mathcal{I}(i) \in \mathcal{I}_{\text{out}}$ then
20:         if $y \in \mathcal{V}$ then
21:             $y \leftarrow \begin{bmatrix} y_1 & y_2 + \delta \end{bmatrix}^\top$
22:         $\mathcal{T}_\mathcal{I} = \mathcal{T}_b(y, \lambda) \cap \{x \in \mathbb{R}^2 : (\exists y \in \mathcal{I}(i) : x_1 = y_1)\}$
23:         $\mathcal{Z} \leftarrow \mathcal{Z} \cup \mathcal{T}_\mathcal{I}$
24:     $i \leftarrow i + 1$
25:     $y \leftarrow \mathcal{Q}(\mathcal{Z})$
26: return $\mathcal{Z}$
```

Theorem 1 *Given the system* (11)*, the state constraints $\mathcal{X}$* (7)*, input constraints $\mathcal{U}_x$ (6), the target set $\mathcal{X}_T$ (16). The following hold.*

(i) *Algorithm 1 terminates in finite time.*
(ii) *For each initial condition x_0 in $\mathcal{R}_\epsilon(\mathcal{X}_T)$, there exists a control law $\lambda(x)$ such that the solution to the system $\phi(t; x_0, \lambda(x))$ reaches the target set $\mathcal{X}_T$ in finite time t^*, i.e., $\phi(t^*; x_0, \lambda(x)) \in \mathcal{X}_T$ without violating the constraints, i.e., $\phi(t; x_0, \lambda(x)) \in \mathcal{X}$, for all $t \in [0, t^*]$.*

Proof (i) In Lines 3 and 5 of Algorithm 1, the trajectories from the extreme points of the target set interval are calculated. The trajectories are calculated in finite time as $x_2 > 0$, the state equations $\dot{x}_1 = -x_2$ and $\dot{x}_2$ will be equal $A(x)$ or $D(x)$, with the solution escaping $\mathcal{X}$ in finite time. If Lines 8, 12, 17 or 22 are reached in Algorithm 1 the previously calculated trajectory has to be extended to produce a boundary of the $\mathcal{R}_\epsilon(\mathcal{X}_\text{T})$. In this case Algorithm 2 is called, taking as an input the set of L intervals described in definition 5. Each of these intervals must either satisfy condition (21) or not. Consequently, to show finite termination of Algorithm 1, it is sufficient to show that L is finite.

In order to show that L is finite we must analyse all the states in $\mathcal{V}$. If we find any equilibrium points i.e. $\dot{x} = 0$.



(a) situation 1 (b) situation 2

(c) situation 3 (d) situation 4

Fig. 4: Illustration of all the typical formats of $\mathcal{I}(i)$. Green regions imply $\mathcal{I}(\cdot) \in \mathcal{I}_{in}$, whilst red regions imply $\mathcal{I}(\cdot) \in \mathcal{I}_{out}$. The black region represents the continuation remaining intervals until $\mathcal{I}(L)$ is reached.

These equilibrium points should form intervals contained within $\mathcal{I}_{out}$ from definition 5.

Let us define the inner product $S(x)$ of the extreme dynamics at non equilibrium points and a normal vector to the the upper and lower bound as

$$S(x) = \begin{cases} D(x) - m_u(x)x_2 \text{ when } x \in \mathcal{V}^u, \|f(x,0)\| > 0, \\ m_l(x)x_2 - A(x) \text{ when } x \in \mathcal{V}^l \; |f(x,1)\| > 0, \end{cases}$$

where $m_u(x)$ and $m_l(x)$ represent the right-derivative of $C^u(x)$ and $C^l(x)$ from (7) respectively. If $S(x)$ has a finite number of roots the number of intervals is also finite. Since $m_l(x)x_2$, $-m_u(x)x_2$, $A(x)$ and $D(x)$, are locally Lipschitz continuous functions, by Fact 1 (i) have $S(x)$ is also locally Lipschitz continuous. Thus, its derivative is bounded and therefore in a finite interval the number of roots must also be bounded. This bounded number of roots translates to a bounded number of intervals L, and consequently, the Algorithm 1 terminates in finite time.

(ii) Consider an initial condition state x in $\mathcal{X}$ It holds $x_1 \geq 0$, $x_2 \geq 0$ for all x in $\mathcal{X}$ and, the condition $x_1^* > x_1$, for all x^* in $\mathcal{X}_\mathrm{T}$. To show $\mathcal{X}_\mathrm{T}$ will be reached in finite time we construct function $\lambda(x)$: We consider the state x in $\mathcal{R}_\epsilon(\mathcal{X}_\mathrm{T})$, and forming the slice $\mathcal{X}_v(x_1)$ given by (15), we define the vectors $x^u(x) \in \mathcal{X}_v(x)$, $x^l(x) \in \mathcal{X}_v(x)$ on the upper and lower boundary of $\mathcal{R}_\epsilon(\mathcal{X}_\mathrm{T})$. This allows the control law to be defined as $\lambda(x) = \frac{x_2 - (x^u(x))_2}{(x^l(x))_2 - (x^u(x))_2}$. The control law is constructed such than when $x_2 = x_2^l$, $\lambda(x) = 1$ and thus $\dot{x}_2 = u(x, \lambda(x)) = A(x)$. Likewise, if $x_2 = x_2^u$, $\lambda(x) = 0$ and thus $\dot{x}_2 = u(x, \lambda(x)) = D(x)$. Due to the construction of $\mathcal{R}_\epsilon(\mathcal{X}_\mathrm{T})$, if the state lies on the upper or lower boundary the choice of these inputs will have the closed-loop system dynamics pointing inside $\mathcal{X}$. Moreover, since $\dot{x}_1 = x_2 \geq 0$ the value of x_1 increases with time when $x_2 > 0$. For the case when $x_2 = 0$, x is necessarily on the lower boundary, thus, $\dot{x}_2 = A(x) > 0$, thus, the value of x_1 will increase in finite time. Last, taking into account that the right boundary of $\mathcal{R}_\epsilon(\mathcal{X}_\mathrm{T})$ is $\mathcal{X}_\mathrm{T}$, there is necessarily a finite time t^* such that the solution to the system $\phi_\mathrm{f}(t^*; x_0, \lambda(x))$ is in $\mathcal{X}_\mathrm{T}$. ■

Naturally, $\mathcal{R}_\epsilon(\mathcal{X}_\mathrm{T})$ is a subset of $\mathcal{R}(\mathcal{X}_\mathrm{T})$, making it an inner approximation of the reach-avoid set.

VI. Numerical Implementation

We illustrate the results with the challenging commercial UR5 robot model contained within the Matlab Robotics Systems toolbox [1], using the joint torque limitations found on the manufacturers website [2]. The chosen robotic manipulator has six degrees of freedom. The implementation was made in Python [3] using the Matlab engine API for Python [4]. The path to be followed is a straight line in the joint space such from $q(0) = \begin{bmatrix}\pi/2 & \pi/2 & \pi/3 & 2\pi/3 & -\pi/2 & -\pi/3\end{bmatrix}$, to $q(1) = \begin{bmatrix}0 & 0 & 0 & 0 & 0 & 0\end{bmatrix}$.

To illustrate, Figure 5 shows the set $\mathcal{R}_\epsilon(\mathcal{X}_\mathrm{T})$ (yellow), together with the constraint set $\mathcal{X}$ (green). We choose the target set $\mathcal{X}_\mathrm{T} = \{x \in \mathbb{R}^2 : x_1 = 0.9, 2 \leq x_2 \leq 4\}$ and set $\epsilon = 0.01$. The trajectories formed from the extreme points resulted in both the upper an lower bounds requiring extension. In our example, Situation 6 is triggered in Algorithm 1, as shown in Figure 2f. In this case, during the extension procedure of Algorithm 2, the entire lower boundary was valid, which is the situation illustrated by Fig. 4a. On the other hand, the upper boundary generated 49 intervals over which which extension was required. Figure 5 also illustrates a trajectory of the closed-loop system for the initial condition $x_0 = \begin{bmatrix}0.1 & 2\end{bmatrix}^\top$, generated via the simple state feedback controller utilised in the proof of Theorem 1.

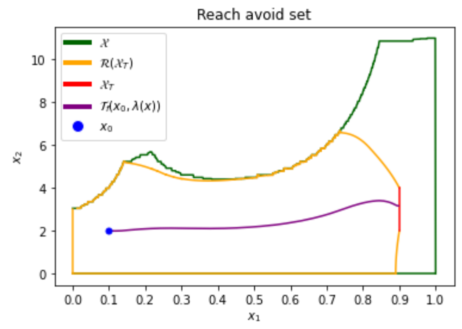


Fig. 5: Typical control trajectory (purple) within the reach-avoid set $\mathrm{bd}(\mathcal{R}_\epsilon(\mathcal{X}_\mathrm{T}))$ (yellow), with $\mathrm{bd}((\mathcal{X}))$ represented with the green boundary

VII. Conclusions

In this paper we explored the trajectory planning problem for robotic manipulators using a well established methodology from a new perspective. In specific, we formulated the problem of finding safe state-feedback controllers as a reach-avoid problem in the projected path dynamics, and

[1] https://uk.mathworks.com/products/robotics.html
[2] https://www.universal-robots.com/articles/ur/robot-care-maintenance/max-joint-torques/
[3] https://github.com/rmcgovern50/Robotic_Manipulator_Analysis.git
[4] https://uk.mathworks.com/help/matlab/matlab-engine-for-python.html

consequently solved it using ordering properties of closed-loop trajectories under a suitable parameterisation of the control law. The established algorithm is practicable and terminates in finite time, while numerical experiments using realistic manipulators show our method works well. Our future work will focus on the problem of composition of trajectories between different paths, using the inherent flexibility offered by the framework, ultimately addressing the more complicated path planning problem in an efficient way. Last, we aim to relax some of the assumptions on the shapes of the constraint and target sets posed herein, effectively increasing the generalith of our approach.

References

[1] V. Villani, F. Pini, F. Leali, and C. Secchi, "Survey on human–robot collaboration in industrial settings: Safety, intuitive interfaces and applications," *Mechatronics*, vol. 55, pp. 248–266, 2018.

[2] J. E. Michaelis, A. Siebert-Evenstone, D. W. Shaffer, and B. Mutlu, in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, 2020, pp. 1–12.

[3] E. M. E. Alegue, "Human-robot collaboration with high-payload robots in industrial settings," in *IECON 2018-44th Annual Conference of the IEEE Industrial Electronics Society*. IEEE, 2018, pp. 6035–6039.

[4] L. Kunze, N. Hawes, T. Duckett, M. Hanheide, and T. Krajník, "Artificial intelligence for long-term robot autonomy: A survey," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4023–4030, 2018.

[5] Z. Bi, M. Luo, Z. Miao, B. Zhang, W. Zhang, and L. Wang, "Safety assurance mechanisms of collaborative robotic systems in manufacturing," *Robotics and Computer-Integrated Manufacturing*, vol. 67, p. 102022, 2021.

[6] S. M. LaValle, *Planning Algorithms*, 05 2006.

[7] L. Joseph, V. Padois, and G. Morel, "Towards x-ray medical imaging with robots in the open: safety without compromising performances," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 6604–6610.

[8] ——, "Experimental validation of an energy constraint for a safer collaboration with robots," in *International Symposium on Experimental Robotics*. Springer, 2018, pp. 575–583.

[9] R. Rossi, M. P. Polverini, A. M. Zanchettin, and P. Rocco, "A pre-collision control strategy for human-robot interaction based on dissipated energy in potential inelastic impacts," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 26–31.

[10] Kang Shin and N. McKay, "Minimum-time control of robotic manipulators with geometric path constraints," *IEEE Transactions on Automatic Control*, vol. 30, no. 6, pp. 531–541, 1985.

[11] J. E. Bobrow, S. Dubowsky, and J. S. Gibson, "Time-optimal control of robotic manipulators along specified paths," *The international journal of robotics research*, vol. 4, no. 3, pp. 3–17, 1985.

[12] F. Pfeiffer and R. Johanni, "A concept for manipulator trajectory planning," *IEEE Journal on Robotics and Automation*, vol. 3, no. 2, pp. 115–123, 1987.

[13] J. D. Gleason, A. P. Vinod, and M. M. K. Oishi, "Underapproximation of reach-avoid sets for discrete-time stochastic systems via lagrangian methods," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, 2017, pp. 4283–4290.

[14] B. Landry, M. Chen, S. Hemley, and M. Pavone, "Reach-avoid problems via sum-or-squares optimization and dynamic programming," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 4325–4332.

[15] J. F. Fisac, M. Chen, C. J. Tomlin, and S. S. Sastry, "Reach-avoid problems with time-varying dynamics, targets and constraints," in *Proceedings of the 18th international conference on hybrid systems: computation and control*, 2015, pp. 11–20.

[16] L. Shamgah, T. G. Tadewos, A. Karimoddini, and A. Homaifar, "Path planning and control of autonomous vehicles in dynamic reach-avoid scenarios," in *2018 IEEE Conference on Control Technology and Applications (CCTA)*. IEEE, 2018, pp. 88–93.

[17] B. HomChaudhuri, M. Oishi, M. Shubert, M. Baldwin, and R. S. Erwin, "Computing reach-avoid sets for space vehicle docking under continuous thrust," in *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE, 2016, pp. 3312–3318.

[18] N. Kariotoglou, D. M. Raimondo, S. Summers, and J. Lygeros, "A stochastic reachability framework for autonomous surveillance with pan-tilt-zoom cameras," in *2011 50th IEEE Conference on Decision and Control and European Control Conference*. IEEE, 2011, pp. 1411–1416.

[19] S. Summers and J. Lygeros, "Verification of discrete time stochastic hybrid systems: A stochastic reach-avoid decision problem," *Automatica*, vol. 46, no. 12, pp. 1951–1961, 2010.

[20] S. N. Krishna, A. Kumar, F. Somenzi, B. Touri, and A. Trivedi, "The reach-avoid problem for constant-rate multi-mode systems," in *International Symposium on Automated Technology for Verification and Analysis*. Springer, 2017, pp. 463–479.

[21] L. Shamgah, T. G. Tadewos, A. Karimoddini, and A. Homaifar, "A symbolic approach for multi-target dynamic reach-avoid problem," in *2018 IEEE 14th International Conference on Control and Automation (ICCA)*. IEEE, 2018, pp. 1022–1027.

[22] Q. Pham, "A general, fast, and robust implementation of the time-optimal path parameterization algorithm," *IEEE Transactions on Robotics*, vol. 30, no. 6, pp. 1533–1540, 2014.

[23] H. Pham and Q. Pham, "A new approach to time-optimal path parameterization based on reachability analysis," *IEEE Transactions on Robotics*, vol. 34, no. 3, pp. 645–659, 2018.

[24] J.-Y. Choi, J. Uh, and J. S. Lee, "Iterative learning control of robot manipulator with i-type parameter estimator," in *Proceedings of the 2001 American Control Conference.(Cat. No. 01CH37148)*, vol. 1. IEEE, 2001, pp. 646–651.

[25] F. Bouakrif, D. Boukhetala, and F. Boudjema, "Velocity observer-based iterative learning control for robot manipulators," *International Journal of Systems Science*, vol. 44, no. 2, pp. 214–222, 2013.

[26] M. de Mathelin, R. Lozano, and D. Taoutaou, "Commande adaptative et applications," *Hermès Science*, 2001.

[27] K. M. Lynch and F. C. Park, *Modern Robotics*, 2017.

[28] E. Hughes, J. Hiley, K. Brown, and I. M. Smith, *Hughes Electrical and Electronic Technology*, 07 2021.

[29] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Modeling and Control*, 11 2005.

[30] G. Teschl, *Ordinary Differential Equations and Dynamical Systems*, 08 2012.

[31] E. Lindelöf, "Sur l'application de la méthode des approximations successives aux équations différentielles ordinaires du premier ordre," *Comptes rendus hebdomadaires des séances de l'Académie des sciences*, vol. 116, no. 3, pp. 454–457, 1894.

[32] L. Habets, P. J. Collins, and J. H. van Schuppen, "Reachability and control synthesis for piecewise-affine hybrid systems on simplices," *IEEE Transactions on Automatic Control*, vol. 51, no. 6, pp. 938–948, 2006.

[33] L. Habets, M. Kloetzer, and C. Belta, "Control of rectangular multi-affine hybrid systems," in *Proceedings of the 45th IEEE Conference on Decision and Control*. IEEE, 2006, pp. 2619–2624.

[34] C. Sloth and R. Wisniewski, "Control to facet for polynomial systems," in *Proceedings of the 17th international conference on Hybrid systems: computation and control*, 2014, pp. 123–132.

[35] F. Blanchini and S. Miani, *Set-Theoretic Methods in Control*, 07 2015.

Manuscript 438 submitted to 2022 European Control Conference (ECC).
Received November 8, 2021.