Essential Properties of Numerical Integration for Time-Optimal Path-Constrained Trajectory Planning

Peiyao Shen, Xuebo Zhang, and Yongchun Fang

***Abstract*—This letter summarizes *some known properties* and also presents *several new properties* of the numerical integration (*NI*) method for time-optimal trajectory planning along a specified path. The contribution is that *rigorous mathematical proofs* of these properties are presented, most of which have not been reported in existing literatures. We first give some properties regarding switch points and accelerating/decelerating curves of the *NI* method. Then, for the fact that when kinematic constraints are considered, the original version of *NI* that only considers torque constraints may results in failure of trajectory planning, we give the concrete failure conditions with rigorous mathematical proof. Accordingly, a failure detection algorithm is given in a "run-and-test" manner. Some simulation results on a unicycle vehicle are provided to verify those presented properties. Note that though those *known properties* are not discovered first, their mathematical proofs are given first in this letter. The detailed proofs make the theory of *NI* more complete and help interested readers to gain a thorough understanding of the method.**

***Index Terms*—Manipulation planning, motion and path planning, numerical integration, properties with rigorous proofs, time-optimal trajectory planning.**

I. Introduction

DUE to low computational complexity, decoupled planning [1]–[3] becomes a popular motion planning method, which consists of two stages. In the first stage, path planning methods are used to generate a geometric path with high level constraints including obstacle avoidance, curvature, and so on. In the second stage, trajectory planning, which aims to assign a motion time profile to the specified path, could be then simplified as a planning problem in two-dimensional path parametrization space $(s, \dot{s})$, with s and $\dot{s}$ being the path coordinate and path velocity respectively. To improve working efficiency, several time-optimal methods have been reported for the trajectory planning stage along a specified path, including *Dynamic Programming* [4]–[6], *Convex Optimization* [7]–[9] and *NI* [10]–[22]. In addition, the work in [23] proposes a real-time trajectory generation approach for omni-directional vehicles by constrained dynamic inversion.

In this letter, we will focus on the *NI* method, since its computational efficiency is shown in [13] to be better than other methods. The original version of *NI* is proposed at almost the same time in the works [10], [11], which aims to give a time-optimal trajectory along a specified path in the presence of only torque constraints for manipulators. Based on Pontryagin Maximum Principle, *NI* possesses a bang-bang structure of torque inputs, thus, the core of *NI* is the computation of switch points. The accelerating and decelerating curves, integrated from those switch points, constitute the time-optimal trajectory. The work in [12] presents three types of switch points: tangent, discontinuity and zero-inertia points. The detection methods for zero-inertia switch points are presented in [13]–[15]. Based on previous works, Pham [13] provides a fast, robust and open-source implementation for *NI* in C++/Python, which is subsequently extended to the case of redundantly-actuated systems in the work [16]. In addition to manipulators [17], [18], *NI* is also applied to spacecrafts [19] and humanoid robots [20]. In real applications, in addition to torque constraints, velocity constraints (bounded actuator velocity and path velocity) should also be considered. Under these constraints, the works in [21], [22] indicate that the original *NI* method with only torque constraints [10] can not be directly applied. Considering velocity and acceleration constraints, the work [18] focuses on detecting tangent, discontinuity and zero-inertia switch points on the speed limit curve decided by velocity constraints. However, rigorous proofs have not been reported to expose the conditions and underlying reasons of failure cases in aforementioned literatures.

In this brief, we summarize some known properties and amend corresponding mathematical proofs which have not been reported in existing literatures; in addition, some new properties are excavated and proven rigorously. Some properties indicate the evolution of accelerating/decelerating curves and their intersection points. And another important property indicates that, when kinematic constraints are considered, the original version of *NI* which only considers torque constraints, may result in failure of trajectory planning tasks. For this property, we first give the concrete failure conditions and detailed proofs. Accordingly, the failure detection algorithm is given in a 'run-and-test' manner. Simulation results on a unicycle vehicle are provided to verify these properties.

The main contribution of this letter is the rigorous and detailed proofs for all presented properties:

Manuscript received September 6, 2016; accepted January 7, 2017. Date of publication January 19, 2017; date of current version February 7, 2017. This letter was recommended for publication by Associate Editor D. Balkcom and Editor H. Ding upon evaluation of the reviewers' comments. This work was supported by the National Natural Science Foundation of China under Grants 61573195, 613205017, and U1613210. *(Corresponding author: Xuebo Zhang.)*

The authors are with the Institute of Robotics and Automatic Information System, Nankai University, Tianjin 300071, China, and also with Tianjin Key Laboratory of Intelligent Robotics, Nankai University, Tianjin 300071, China (e-mail: 635467453@qq.com; zhangxuebo@nankai.edu.cn; fangyc@nankai.edu.cn).

Color versions of one or more of the figures in this letter are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/LRA.2017.2655580

2377-3766 © 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

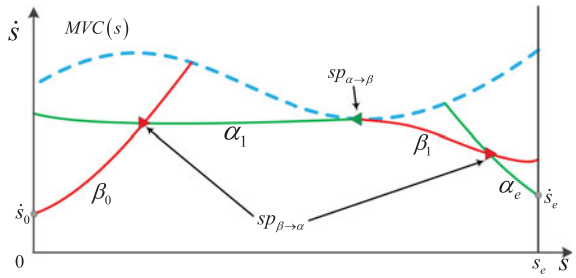

Fig. 1. Red solid curves β_0, β_1 represent accelerating $\beta\text{-}profiles$. Green solid curves α_1, α_e represent decelerating $\alpha\text{-}profiles$. Red $\triangleright$ and green $\triangleleft$ denote the points $sp_{\beta\to\alpha}$ and $sp_{\alpha\to\beta}$, respectively. The scalars $\dot{s}_0, \dot{s}_e$ are respectively the starting and terminal path velocity at the endpoints of the given path. The s_e is the total length of the given path.

1) For *Properties 2-3* reported in [24], we first give their mathematical proofs in Section III-A.
2) Some new properties are presented and proven, including *Property 4* in Section III-B and *Properties 5-6* in Section III-C.

These properties and proofs make the theory of *NI* more complete and help interested readers to gain an thorough understanding of the *NI* method.

The remainder of this letter is divided into four sections. Section II introduces notations and procedures of the *NI* method. Section III presents essential properties of the *NI* method, and provides rigorous mathematical proofs. In Section IV, simulation results are provided to verify the properties. Finally, Section V gives some conclusions.

II. Numerical Integration

In this section, we briefly introduce the notations and procedures of the *NI* method [10], [11]. For the time-optimal path-constrained trajectory planning problem, based on Pontryagin Maximum Principle, the original *NI* method [10], [11] uses a bang-bang structure of torque input to generate time-optimal trajectories. First, torque constraints are converted to path acceleration constraints along a given path. Then, switch points of path acceleration are found. Finally, a time-optimal trajectory is integrated numerically with maximum and minimum path acceleration. A stable and open-source implementation of *NI* method can be found in [13], and kinematic constraints are handled in the implementation with the method proposed in [21]. Yet, in the presence of kinematic constraints, the rigorous proof showing conditions of failure of the original *NI* with only torque constraints, has not been reported.

A. Time-Optimal Path-Constrained Trajectory Planning

The time-optimal path-constrained trajectory planning problem is to find a time-optimal velocity profile along the given path for a robot under various constraints such as torque constraints, velocity constraints, and so on.

B. Notations

In order to explain various notations clearer, we will refer to Fig. 1 in the following.

s: Path coordinate along a specified path.

$\dot{s}$: Path velocity along a specified path.

$\ddot{s}$: Path acceleration along a specified path.

$\boldsymbol{A}(s)\ddot{s} + \boldsymbol{B}(s)\dot{s}^2 + \boldsymbol{C}(s) \leq \boldsymbol{0}$ [10] : The inequality constraint along a specified path for $s, \dot{s}, \ddot{s}$, which is derived from torque constraints. As an example, for an n-dof manipulator, in order to guarantee torque constraints in a path-constrained trajectory planning task, the inequality constraint [10]

$$M(\boldsymbol{\xi})\ddot{\boldsymbol{\xi}} + \dot{\boldsymbol{\xi}}^{\mathrm{T}} P(\boldsymbol{\xi})\dot{\boldsymbol{\xi}} + \boldsymbol{Q}(\boldsymbol{\xi}) \leq \boldsymbol{0} \tag{1}$$

should hold, where the state $\boldsymbol{\xi}$ is an n-dimensional vector, M is an $m \times n$ matrix, P is an $n \times m \times n$ tensor and $\boldsymbol{Q}$ is an m-dimensional vector. Along the specified path, the robot state and its differentials are

$$\boldsymbol{\xi} = \boldsymbol{\xi}(s),\ \dot{\boldsymbol{\xi}} = \boldsymbol{\xi}_s \dot{s},\ \ddot{\boldsymbol{\xi}} = \boldsymbol{\xi}_{ss}\dot{s}^2 + \boldsymbol{\xi}_s \ddot{s} \tag{2}$$

where $\boldsymbol{\xi}_s = d\boldsymbol{\xi}/ds$, $\boldsymbol{\xi}_{ss} = d\boldsymbol{\xi}_s/ds$. After substituting (2) into (1), we obtain that

$$\boldsymbol{A}(s)\ddot{s} + \boldsymbol{B}(s)\dot{s}^2 + \boldsymbol{C}(s) \leq \boldsymbol{0}, \tag{3}$$

with

$$\begin{aligned}
\boldsymbol{A}(s) &= M(\boldsymbol{\xi}(s))\boldsymbol{\xi}_s(s), \\
\boldsymbol{B}(s) &= M(\boldsymbol{\xi}(s))\boldsymbol{\xi}_{ss}(s) + \boldsymbol{\xi}_s(s)^{\mathrm{T}} P(\boldsymbol{\xi}(s))\boldsymbol{\xi}_s(s), \\
\boldsymbol{C}(s) &= \boldsymbol{Q}(\boldsymbol{\xi}(s)).
\end{aligned}$$

$\alpha(s,\dot{s}), \beta(s,\dot{s})$: In order to guarantee torque constraints, the scalars $s, \dot{s}, \ddot{s}$ should satisfy the inequality (3). Therefore, given the path coordinate s and path velocity $\dot{s}$, the path acceleration $\ddot{s}$ satisfies the following inequality

$$\alpha(s,\dot{s}) \leq \ddot{s} \leq \beta(s,\dot{s}), \tag{4}$$

where the minimum path acceleration $\alpha(s,\dot{s})$ and maximum path acceleration $\beta(s,\dot{s})$ are computed as

$$\alpha(s,\dot{s}) = \max\left\{\alpha_i | \alpha_i = \frac{-B_i(s)\dot{s}^2 - C_i(s)}{A_i(s)}, A_i(s) < 0\right\}, \tag{5}$$

$$\beta(s,\dot{s}) = \min\left\{\beta_i | \beta_i = \frac{-B_i(s)\dot{s}^2 - C_i(s)}{A_i(s)}, A_i(s) > 0\right\}, \tag{6}$$

wherein the integer $i \in [1, m]$, with m being the dimension of the vector $\boldsymbol{A}$. Please see the detailed description in [11].

MVC: The maximum velocity curve in the plane $(s,\dot{s})$ is represented as

$$MVC(s) = \min\{\dot{s} \geq 0 | \alpha(s,\dot{s}) = \beta(s,\dot{s})\},\ s \in [0, s_e]. \tag{7}$$

For instance, the cyan dash curve in Fig. 1 is MVC. If the robot state is on the MVC, there exists at least one saturated actuator torque.

AR: The admissible region, in the plane $(s,\dot{s})$, is enclosed by the curve MVC and the lines $\dot{s} = 0, s = 0, s = s_e$. Within the AR except for the boundary MVC, all actuator torques are between lower and upper bounds ($\alpha(s,\dot{s}) < \beta(s,\dot{s})$).

$\alpha\text{-}profile$: The decelerating curve, in the plane $(s,\dot{s})$, is integrated backward with minimum acceleration $\alpha(s,\dot{s})$ in (5). The slope k_α is computed as $k_\alpha = d\dot{s}/ds = \alpha(s,\dot{s})/\dot{s}$.

β-$profile$: The accelerating curve, in the plane $(s, \dot{s})$, is integrated forward with maximum acceleration $\beta(s, \dot{s})$ in (6). The slope k_β is computed as $k_\beta = d\dot{s}/ds = \beta(s, \dot{s})/\dot{s}$.

$sp_{\alpha\to\beta}$ [12]: Switch points from decelerating to accelerating curves, such as the green $\triangleleft$ in Fig. 1. They are on the MVC curve, and there are three different types: tangent, discontinuity or zero-inertia points. At tangent points, the slope $k_{\text{mvc}} = d\dot{s}/ds$ of MVC is equal to $k_\alpha (= k_\beta)$. At discontinuity points, MVC is discontinuous. At zero-inertia points, at least one $A_i(s) = 0$ holds for the corresponding path coordinate s.

$sp_{\beta\to\alpha}$: Switch points from accelerating to decelerating curves, such as the red $\triangleright$ in Fig. 1.

C. Numerical Integration Algorithm

Procedures of the original version of *NI* [10] which only considers torque constraints are given as follows:

NI-1. In the plane $(s, \dot{s})$, starting from $(s = 0, \dot{s} = \dot{s}_0)$, the accelerating curve β-$profile$ is integrated forward with maximum path acceleration $\beta(s, \dot{s})$ until one of the following cases occurs:
- the curve MVC is hit, and go to **NI-2**;
- the line $\dot{s} = 0$ is hit, and output that this path is not traversable;
- the line $s = s_e$ is hit, and go to **NI-3**.

NI-2. From the hitting point, searching forward along MVC, the first tangent, discontinuity or zero-inertia point found is $sp_{\alpha\to\beta}$.
- If $sp_{\alpha\to\beta}$ is detected, from the switch point, an α-$profile$ is integrated backward with $\alpha(s, \dot{s})$ until it intersects the generated β-$profile$ in **NI-1, NI-2** at a point $sp_{\beta\to\alpha}$, and one new β-$profile$ is integrated forward as **NI-1**.
- If $sp_{\alpha\to\beta}$ is not detected, go to **NI-3**.

NI-3. Starting from $(s = s_e, \dot{s} = \dot{s}_e)$, the decelerating curve α-$profile$ is integrated backward with minimum path acceleration $\alpha(s, \dot{s})$ until it intersects the generated β-$profile$ in **NI-1, NI-2** at a point $sp_{\beta\to\alpha}$. Finally, output a time-optimal trajectory consisting of those generated accelerating and decelerating curves in **NI-1, NI-2** and **NI-3**.

Note that chattering is usually caused by infinite control switches [25]. Fortunately, the work in [10] has indicated that switch points are finite for the *NI* method due to the predefined smooth path constraint. Therefore, the *NI* method [10] does not encounter this chattering problem. Please see detailed explanation in the reference [10].

III. Properties

In this section, essential properties of *NI* are provided with rigorous mathematical proofs. Note that *Property 1* has been presented and proven in [24]; *Properties 2-3* have been reported in [24], and we will amend their mathematical proofs; *Properties 4-5* are first exposed in this letter with rigorous proofs; in the presence of kinematic constraints, the failure of the original *NI* with only torque constraints is reported in [21], [22], and we will give the mathematical conditions and underlying reasons of failure cases in *Property 6*. See the proofs of *Properties 2-5* in Appendix A-D, respectively.

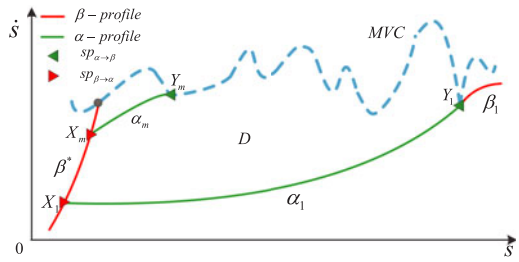

Fig. 2. $m > 1$: Intersection points X_i, $i \in [1, m]$ are on β^*, wherein X_j, $1 < j < m$ is on $\overset{\frown}{X_1 X_m}$. Each X_i has one corresponding decelerating curve α_i and one switch point Y_i. The region D is enclosed by α_1, β^* and MVC.

A. Property of α-$Profiles$ and β-$Profiles$

In the admissible region except for the boundary MVC $(AREM)$, the inequality $\alpha(s, \dot{s}) < \beta(s, \dot{s})$ holds, which indicates $k_\alpha < k_\beta$ for each point. Three known properties are summarized in the following, and we will give rigorous mathematical proofs for *Properties 2-3*, which have not been reported in existing literatures.

Property 1: In the $AREM$ region, any two α-$profiles$ never intersect with each other. Neither do β-$profiles$. (Please see the proof in [24].)

Property 2: In the $AREM$ region, if an α-$profile$ intersects another β-$profile$ at a point $(s = s_c, \dot{s} = \dot{s}_c)$, in terms of path velocity, the α-$profile$ is greater than the β-$profile$ in the left neighborhood of s_c, but less than the β-$profile$ in the right neighborhood of s_c.

Property 3: In the $AREM$ region, an α-$profile$ is not tangent to another β-$profile$ at any point.

B. Property of $sp_{\beta\to\alpha}$

The point $sp_{\beta\to\alpha}$ is the intersection point between α-$profile$ and β-$profile$, denoted as the red $\triangleright$ in Fig. 2. In the iterative process of *NI* (see Section II-C), a β-$profile$ may intersect a finite number of integrated backward α-$profiles$ [10]. Which one of these intersection points is finally chosen as $sp_{\beta\to\alpha}$ on the β-$profile$? The answer is given in *Property 4*, which is first presented in this letter with rigorous proof.

Property 4: If one β-$profile$ intersects $m \geq 1$ α-$profiles$, respectively at points $X_i, i \in [1, m]$, and in terms of path coordinate, X_i is less than X_j, $1 \leq i < j \leq m$, then, X_1 is finally chosen as $sp_{\beta\to\alpha}$ on the β-$profile$ after finite iterations.

Remark 1: In Fig. 2, Y_1 may be the terminal point $(s_e, \dot{s}_e)$. In this special situation, *Property 4* still holds. The case $m = 1$ obviously holds. Meanwhile, the case $m > 1$ also holds since the trajectory starting from $X_i, i > 1$ cannot arrive at the terminal point according to *Properties 1-3*. In addition, other points Y_i, $i > 1$ cannot be the terminal point $(s_e, \dot{s}_e)$ since *NI* has completed all procedures at the terminal point (see the procedure **NI-3** in Section II-C). ■

The *Properties 1–4* show the evolution of accelerating/decelerating curves and their intersection points.

C. Property of NI With Kinematic Constraints

The original *NI* method [10] generates a time-optimal trajectory with bounded torque. However, when velocity constraints

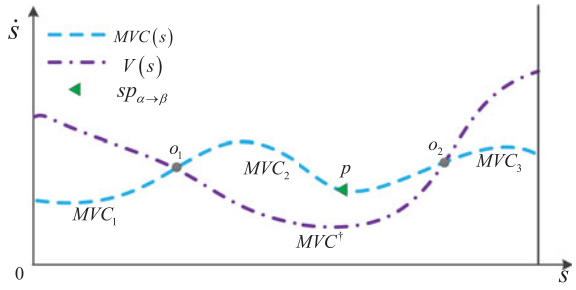

Fig. 3. The maximum velocity curve is altered from $MVC = MVC_1 + MVC_2 + MVC_3$ to $MVC^* = MVC_1 + MVC^\dagger + MVC_3$.

(bounded actuator velocity and path velocity) are taken into account, the original *NI* (as shown in Section II-C) may result in a failure. Several examples for this property are shown in the work in [21], [22], *but the concrete failure conditions and the detailed proof have not been reported.* In the subsequent *Property 6*, we will elaborate the failure conditions and provide the mathematical proof. Note that *Property 5* is first presented and proven to indicate the existence of tangent switch points on the limit curve considering velocity constraints, which is used in the proof of *Property 6*.

Actuator velocity constraints are transformed into path velocity constraints by kinematic models of robots, therefore, velocity constraints are represented as

$$V(s) : s \to \dot{s}, \quad s \in [0, s_e], \tag{8}$$

where the scalar $V(s)$ is the maximum path velocity. Due to velocity constraints, the maximum velocity curve is altered as

$$MVC^*(s) = \min(MVC(s), V(s)), \quad s \in [0, s_e]. \tag{9}$$

In the region enclosed by $MVC^*, s = 0, s = s_e, \dot{s} = 0$, accelerating and decelerating curves satisfy all velocity and torque constraints. In addition, the part of MVC^*, which is different from MVC, is represented as

$$MVC^\dagger(s) = \{MVC^*(s) | MVC^*(s) < MVC(s), s \in [0, s_e]\}. \tag{10}$$

For instance, the dash-dot curve $o_1\overset{\frown}{o_2}$ is $MVC^\dagger$ in Fig. 3.

The maximum velocity curve altering from MVC to MVC^* results in a decreasing number of tangent switch points. For instance, the tangent switch point p disappears due to $MVC^\dagger$ instead of MVC_2 in Fig. 3.

Property 5: Tangent switch points are nonexistent on $MVC^\dagger$.

The *NI* method [10] with torque constraints, generates a time-optimal trajectory T. After considering velocity constraints $V(s)$, the maximum velocity curve is altered from MVC to MVC^*, which causes that tangent switch points on the part $MVC^\dagger$ of MVC^* are nonexistent according to *Property 5*. It has negative effects on searching switch points in the *NI* method, and may result in failure of trajectory planning tasks.

Property 6: If the following conditions hold:

$C_1 : \exists s^* \in [0, s_e], \quad MVC^*(s^*) < T(s^*)$,
C_2 : discontinuity and zero-inertia switch points on $MVC^\dagger$ are nonexistent,

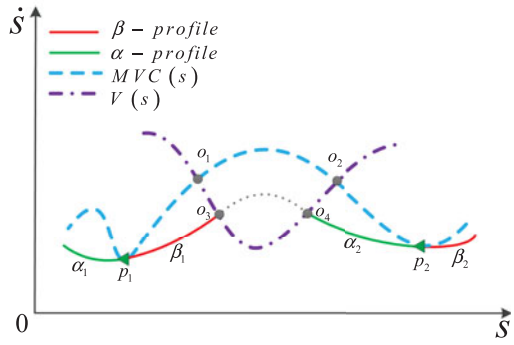

Fig. 4. Left: β-$profile$, Right: α-$profile$.

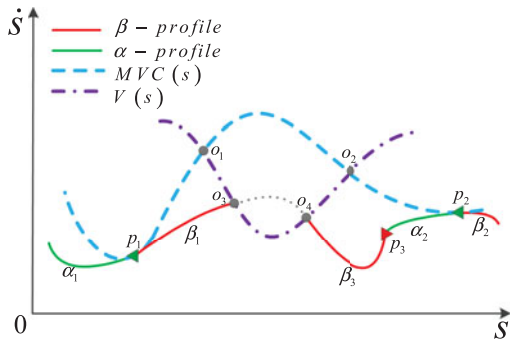

Fig. 5. Left: β-$profile$, Right: β-$profile$.

then, the *NI* method using MVC^* fails to find out a feasible trajectory.

Proof: The proof is proceeded in 4 steps.

Step 1: To show four essential cases of C_1.

The time-optimal trajectory T, which is obtained from the *NI* method using MVC, consists of several α-$profiles$ and β-$profiles$, satisfying $T(s) \le MVC(s), s \in [0, s_e]$. According to (10) and C_1, there must exist one part of $MVC^\dagger$ below T, and this part is called as $MVC^\ddagger$. On both sides of $MVC^\ddagger$, the trajectory T could be accelerating or decelerating, thus there are four essential cases for the condition C_1 as shown in Figs. 4–7. The purple dash-dot line $o_3\overset{\frown}{o_4}$ represents $MVC^\ddagger$. The gray dotted line $\overline{o_3o_4}$ and red/green solid lines constitute the trajectory T. The green $\lhd$ and red $\rhd$ are $sp_{\alpha\to\beta}$ and $sp_{\beta\to\alpha}$ points respectively. These essential cases are regarded as basic components of other complex cases, thus, for clarify, the proof of complex cases will be presented in *Remark 2*.

Step 2: To prove that for each essential case, the NI method using MVC^* must run to the procedure **NI-2** in Section II-C, which is to search forward switch point along $MVC^\dagger$ (the purple dash-dot curve $o_1\overset{\frown}{o_2}$) from the hitting point.

Let p_1 be the neighborhood $sp_{\alpha\to\beta}$ of T on the left side of $MVC^\dagger$, which also can be $p_1 = (0, \dot{s}_0)$. Starting from p_1, *NI* integrates β_1 forward. For essential cases as Figs. 4–5, the accelerating curve β_1 hits $MVC^\dagger$, then *NI* will search forward $sp_{\alpha\to\beta}$ along $MVC^\dagger$ from the hitting point of $o_1\overset{\frown}{o_3}$. For essential cases as Figs. 6–7, all accelerating curves including β_1 and β-$profiles$ from $p_1\overset{\frown}{o_1}$ cannot intersect α_3 for *Property 2*, thus, it must intersect $p_1\overset{\frown}{o_1}$ or $o_1\overset{\frown}{o_3}$. If the hit curve is $o_1\overset{\frown}{o_3}$, then *NI* will search forward $sp_{\alpha\to\beta}$ from the hitting point along $MVC^\dagger$.

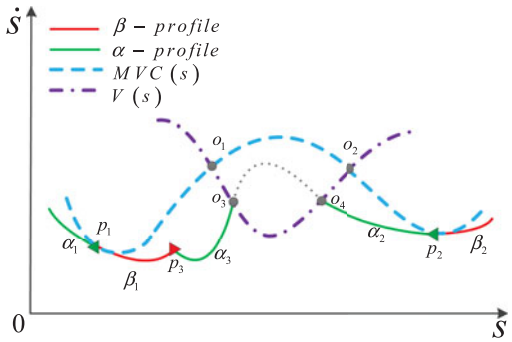

Fig. 6. Left: α-$profile$, Right: α-$profile$

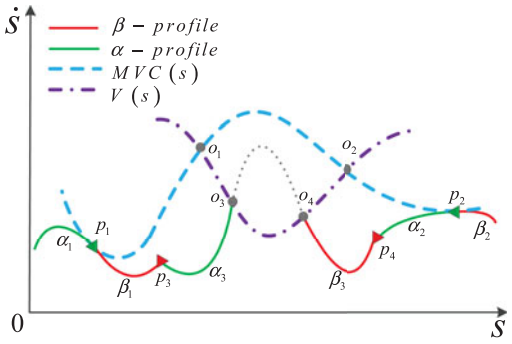

Fig. 7. Left: α-$profile$, Right: β-$profile$

If the hit curve is $p_1\overset{\frown}{o_1}$, then *NI* will search forward $sp_{\alpha\to\beta}$ from the hitting point along $p_1\overset{\frown}{o_1}$. The number of switch points on $p_1\overset{\frown}{o_1}$ is finite [10], therefore switch points on $p_1\overset{\frown}{o_1}$ will run out, and *NI* will go on searching forward $sp_{\alpha\to\beta}$ along $MVC^{\dagger}$.

Step 3: To prove that those α-$profiles$, starting from the right side of o_4, cannot intersect β-$profiles$, which are on the left side of o_3 and generated in iterative process of NI.

Based on *Property 5* and condition C_2 of *Property 6*, switch points on $MVC^{\dagger}$ are nonexistent. Thus, *NI* will go on searching forward switch points on the right side of o_2. In Figs. 4–7, the integrated backward α-$profile$, starting from the $sp_{\alpha\to\beta}$ of $o_2\overset{\frown}{p_2}$, cannot intersect α_2, β_3 according to *Properties 1–2*, thus, it must hit the purple dash-dot $o_4\overset{\frown}{o_2}$ or cyan dash $o_2\overset{\frown}{p_2}$.

Moreover, in the admissible region, from the right side of p_2, integrated backward α-$profiles$ cannot intersect those β-$profiles$ generated by *NI* on the left side of o_3, which is proven by contradiction. Assume that in the admissible region, from the right side of p_2, an integrated backward α-$profile$ intersects one of those β-$profiles$. Then, the α-$profile$ should be part of the trajectory T based on *Property 4*. However, in terms of path velocity, the α-$profile$ is less than MVC^{*}, which contradicts with C_1. Thus, this assumption is invalid, and α-$profiles$ from the right side of p_2 cannot intersect those β-$profiles$ generated on the left side of o_3.

In addition, if p_2 is the terminal point $(s_e, \dot{s}_e)$, then starting from p_2 and switch points at the right side of o_2, all integrated backward α-$profiles$ must hit the purple dash-dot $o_4\overset{\frown}{o_2}$ or cyan dash $o_2\overset{\frown}{p_2}$ according to *Properties 1–3*, which also indicates that this step holds.

Step 4: To synthesize above analysis and indicate failure of trajectory planning tasks.

The above *Step 1–3* indicate that if conditions of *Property 6* hold, then the *NI* method using MVC^{*} fails to output a feasible trajectory: in the iterative process, α-$profiles$ and β-$profiles$ on two different sides of $MVC^{\ddagger}$ do not intersect in the admissible region, which causes that the final trajectory is incomplete. In summary, *Property 6* holds. ■

Remark 2: For other complex cases mentioned in *Step 1*, there may exist many parts of $MVC^{\dagger}$ below T. The trajectory T could be accelerating or decelerating at the first and last part as *Step 1*. In these cases, *NI* still searches switch points along $MVC^{\dagger}$ from the first part in the same reason as *Step 2*. The condition C_2 indicates that the $sp_{\alpha\to\beta}$ on $MVC^{\dagger}$ is nonexistent. Starting from switch points at the right side of the last part, all α-$profiles$ cannot intersect β-$profiles$ at the left side of the first part as *Step 3*. Therefore, in these complex cases, α-$profiles$ and β-$profiles$ on two different sides of $MVC^{\dagger}$ do not intersect in the admissible region. It indicates that *Property 6* still holds for complex cases. ■

In the presence of velocity and torque constraints, *Property 6* actually gives sufficient conditions for failure of the *NI* method in [10], which is important because it is theoretically shown that the failure cases indeed exists for this method. In the following, a necessary and sufficient failure condition is given by a numerical 'run-and-test' algorithm (*RT*):

RT-1. In the plane $(s, \dot{s})$, the curve $MVC^{*}(s)$ is obtained with (9). It is initialized that the point p is $(0, \dot{s}_0)$, boolean variable $isContinuous$ is $TRUE$, the longest continuous trajectory T^{*} from $(0, \dot{s}_0)$ is null and the scalar $sLast$ is zero. Then go to **RT-2**.

RT-2. Starting from the point p, a β-$profile$ is integrated forward until it hits the boundaries of the admissible region at $(s = s_h, \dot{s} = \dot{s}_h)$. If $isContinuous$ is $TRUE$, then *RT* will call the subfunction $addProfile$(T^{*}, β-$profile$), and $sLast$ is updated as s_h. And go to **RT-3**.

RT-3. From the hitting point, the first switch point found along MVC^{*} or terminal point is assigned to the point q. Starting from q, an α-$profile$ is integrated backward until it intersects one of following lines:

- The trajectory T^{*}, then *RT* will call the subfunction $addProfile$(T^{*}, α-$profile$) and set $isContinuous$=$TRUE$. Meanwhile, $sLast$ is updated as the path coordinate of q.
- The boundaries of the admissible region, then *RT* will set $isContinuous = FALSE$.

The point p is updated as q. If p is the terminal point, then go to **RT-4**, else go to **RT-2**.

RT-4. If $sLast < s_e$, this method fails to generate a feasible trajectory, else output a feasible trajectory T^{*}.

Remark 3: The subfunction $addProfile$ adds accelerating/decelerating curves to T^{*}. The core of this algorithm is running the *NI* method and detecting whether those generated accelerating and decelerating curves constitute a continuous trajectory on the whole path. The scalar $sLast$ indicates the path length of the trajectory T^{*}. Therefore, after the algorithm is completed, the inequality $sLast < s_e$ in **RT-4** is a necessary

and sufficient failure condition for the *NI* method in presence of velocity and torque constraints. ■

IV. Simulation Results

In order to verify *Properties 1–6*, some simulation results on a unicycle vehicle are provided in this section.

A. Model

A unicycle vehicle moves along a specified path, with the direction of the vehicle being always tangent to this given path from a initial pose to a target pose. The angular velocity $\omega \in \mathbb{R}$, the path velocity $v \in \mathbb{R}$, and the path curvature $\kappa \in \mathbb{R}$ have the following relationship [26]:

$$\omega = \kappa v. \tag{11}$$

For the specified path, the curvature could be computed as a function of the path coordinate as follows:

$$\kappa : s \in [0, s_e] \rightarrow \kappa(s) \in \mathbb{R}, \tag{12}$$

where the scalar s is the path coordinate.

According to (11) and (12), the kinematic model of the vehicle is described as

$$\boldsymbol{u} = \boldsymbol{M}(s)\dot{s}, \tag{13}$$

where $\boldsymbol{u} = [\omega \ v]^{\mathrm{T}}$, and $\boldsymbol{M}(s) = [\kappa(s) \ 1]^{\mathrm{T}}$. Taking the time derivative of (13) yields that

$$\dot{\boldsymbol{u}} = \boldsymbol{M}(s)\ddot{s} + \boldsymbol{M}_s(s)\dot{s}^2, \tag{14}$$

where $\dot{\boldsymbol{u}} = [\dot{w} \ \dot{v}]^{\mathrm{T}}$, $\boldsymbol{M}_s(s) = [\kappa_s \ 0]^{\mathrm{T}}$, $\kappa_s = d\kappa/ds$. The scalars $\dot{v}, \dot{w}$ are the path acceleration and angular acceleration, respectively. Velocity and acceleration constraints are given as

$$-\boldsymbol{v}_{\max} \leq \boldsymbol{u} \leq \boldsymbol{v}_{\max}, \tag{15}$$

$$-\boldsymbol{a}_{\max} \leq \dot{\boldsymbol{u}} \leq \boldsymbol{a}_{\max}, \tag{16}$$

where $\boldsymbol{v}_{\max} \in \mathbb{R}^2$ and $\boldsymbol{a}_{\max} \in \mathbb{R}^2$ are constant vectors representing the velocity boundary and the acceleration boundary, respectively. These vector inequalities (15)–(16) should be interpreted componentwise.

In order to guarantee acceleration constraints, substituting (14) into (16) yields that

$$\boldsymbol{A}(s)\ddot{s} + \boldsymbol{B}(s)\dot{s}^2 + \boldsymbol{C}(s) \leq \boldsymbol{0}, \tag{17}$$

where $\boldsymbol{A}(s) = [\boldsymbol{M}(s)^{\mathrm{T}} - \boldsymbol{M}(s)^{\mathrm{T}}]^{\mathrm{T}}$, $\boldsymbol{B}(s) = [\boldsymbol{M}_s(s)^{\mathrm{T}} - \boldsymbol{M}_s(s)^{\mathrm{T}}]^{\mathrm{T}}$ and $\boldsymbol{C}(s) = -[\boldsymbol{a}_{\max}^{\mathrm{T}} \ \boldsymbol{a}_{\max}^{\mathrm{T}}]^{\mathrm{T}}$, which are all 4×1 vectors.

In order to guarantee velocity constraints, substituting (13) into (15) yields that

$$\boldsymbol{A}(s)\dot{s} + \boldsymbol{D}(s) \leq \boldsymbol{0}, \tag{18}$$

where $\boldsymbol{A}(s) = [\boldsymbol{M}(s)^{\mathrm{T}} - \boldsymbol{M}(s)^{\mathrm{T}}]^{\mathrm{T}}$ and $\boldsymbol{D}(s) = -[\boldsymbol{v}_{\max}^{\mathrm{T}} \ \boldsymbol{v}_{\max}^{\mathrm{T}}]^{\mathrm{T}}$, which are all 4×1 vectors.

The velocity limit curve $MVC(s)$ considering acceleration constraints is computed with (7) and (17), and the limit curve $V(s)$ considering velocity constraints is computed with (8) and (18). When both acceleration and velocity constraints are taken into account, the maximum velocity curve is obtained as MVC^* by fusing $MVC(s)$ and $V(s)$ with (9).

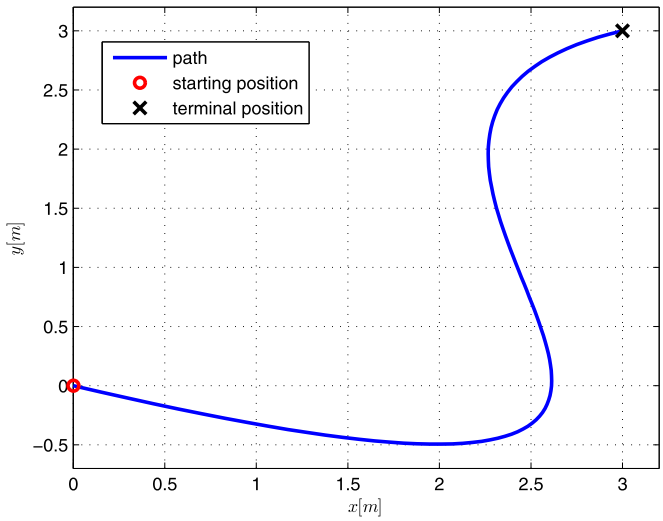

Fig. 8. The specified path: cubic Bézier curve

B. Results

As shown in Fig. 8, the specified blue path is the following cubic Bézier curve:

$$x = (1-\lambda)^3 x_0 + 3(1-\lambda)^2\lambda x_1 + 3(\lambda^2 - \lambda^3)x_2 + \lambda^3 x_3,$$

$$y = (1-\lambda)^3 y_0 + 3(1-\lambda)^2\lambda y_1 + 3(\lambda^2 - \lambda^3)y_2 + \lambda^3 y_3,$$

where the position of the vehicle is $(x[m], y[m])$, the points $(x_i[m], y_i[m])$, $i \in [0, 3]$ are path control points, and the scalar $\lambda \in [0, 1]$ is the path parameter. The λ and s obey a nonlinear scaling relation. The starting and terminal path velocity $\dot{s}_0 = \dot{s}_e = 0[m/s]$. This cubic Bézier curve is used to find a path from a initial pose to a target pose so that the orientation of the unicycle vehicle is always tangent to the path, and thus the nonholonomic constraint is satisfied. The following cases show that the *NI* method assigns velocity profiles to the path.

Case 1: This case shows that, when the velocity constraints are moderate, the *NI* method [10] using MVC^* generates the time-optimal trajectory. In this simulation, the velocity and acceleration constraints are set as $\boldsymbol{v}_{\max} = [0.5 \text{ rad/s} 1.3 \text{ m/s}]^{\mathrm{T}}$, $\boldsymbol{a}_{\max} = [0.05 \text{ rad/s}^2 \ 0.1 \text{ m/s}^2]^{\mathrm{T}}$.

In Fig. 9, the cyan dash line represents $MVC(s)$ with acceleration constraints. When considering velocity constraints corresponding to the purple dash-dot line $V(s)$ in Fig. 9, the maximum velocity curve is altered from MVC to MVC^*, which is represented as the boundary between the gray (inadmissible) and blank (admissible) regions. To facilitate subsequent analysis, symbols $\#_1, \#_2$ are used to represent the two closed areas bounded by $MVC(s)$ and $V(s)$. The red and green solid lines are accelerating curves (β-$profiles$) and decelerating curves (α-$profiles$), respectively, which comply with *Properties 1–3*. The red $\triangleright$ and green $\triangleleft$ denote the points $sp_{\beta\rightarrow\alpha}$ and $sp_{\alpha\rightarrow\beta}$ respectively (see Section II-B).

As shown in Fig. 9, under the curve MVC^*, the *NI* method outputs the optimal trajectory as follows. The accelerating curve β_0, starting from $(0, \dot{s}_0)$, hits MVC^* at p_1. Searching forward along MVC^* from p_1, the first $sp_{\alpha\rightarrow\beta}$ found is p_2. Starting from p_2, the integrated backward decelerating curve α_1 intersects β_0 at o_1, and the integrated forward accelerating curve

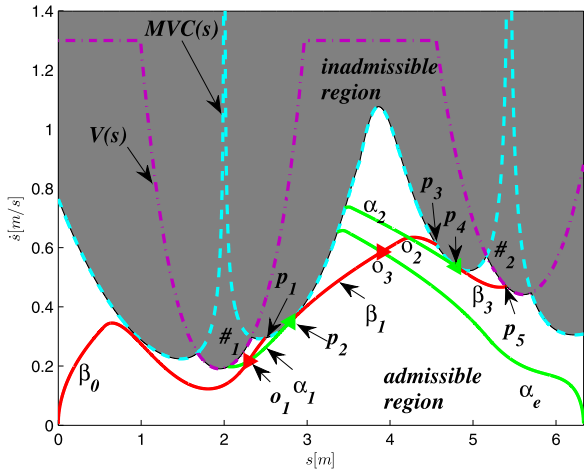

Fig. 9. Case 1: The *NI* method outputs the optimal trajectory with constraints $\boldsymbol{v}_{\max} = [0.5 \text{ rad/s } 1.3 \text{ m/s}]^{\mathrm{T}}$, $\boldsymbol{a}_{\max} = [0.05 \text{ rad/s}^2 \ 0.1 \text{ m/s}^2]^{\mathrm{T}}$.

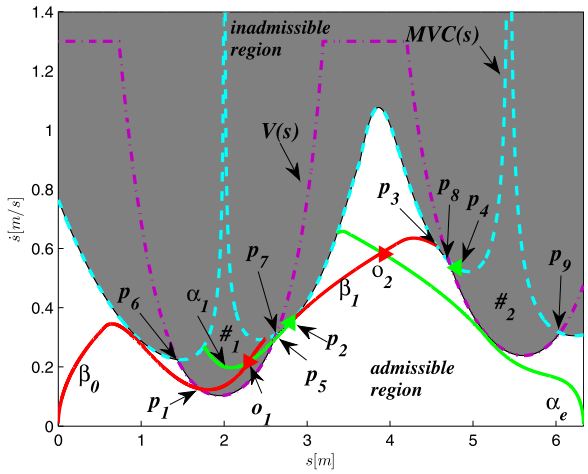

Fig. 10. Case 2: The *NI* method fails to output a feasible trajectory with constraints $\boldsymbol{v}_{\max} = [0.2 \text{ rad/s } 1.3 \text{ m/s}]^{\mathrm{T}}$, $\boldsymbol{a}_{\max} = [0.05 rad/s^2 \ 0.1 m/s^2]^{\mathrm{T}}$.

β_1 hits MVC^* at p_3. Then, searching forward along MVC^* from p_3, the first $sp_{\alpha\to\beta}$ found is p_4. Starting from p_4, the integrated backward decelerating curve α_2 intersects β_1 at o_2, and the accelerating curve β_3 hits the MVC^* at p_5. No $sp_{\alpha\to\beta}$ is found along MVC^* from p_5 when $s \le s_e$. Therefore, the integrated backward α_e, starting from $(s_e, \dot{s}_e)$, intersects β_1 at o_3. The $sp_{\beta\to\alpha}$ on β_1 is updated from o_2 to o_3, which verifies *Property 4*. Finally, the *NI* method outputs the feasible and optimal trajectory: $\beta_0 - \alpha_1 - \beta_1 - \alpha_e$.

Case 2: This case shows that, when the velocity constraints are too restrictive, the conditions in *Property 6* will be satisfied, and thus the *NI* method [10] using MVC^* fails to output a feasible trajectory. In this simulation, the velocity constraint $\boldsymbol{v}_{\max}$ is modified as $[0.2 \text{ rad/s } 1.3 \text{ m/s}]^{\mathrm{T}}$ and the acceleration constraint $\boldsymbol{a}_{\max}$ remains the same as that of *Case 1*. Therefore, in Fig. 10, the velocity limit curve $V(s)$ due to the velocity constraint becomes lower, and the areas of two inadmissible regions $\#_1, \#_2$ increase. Meanwhile, the trajectory $T : \beta_0 - \alpha_1 - \beta_1 - \alpha_e$, obtained by the *NI* method using MVC, is greater than MVC^* at the region $\#_1$, and the $sp_{\alpha\to\beta}$ on $MVC^{\dagger}$ is nonexistent, which

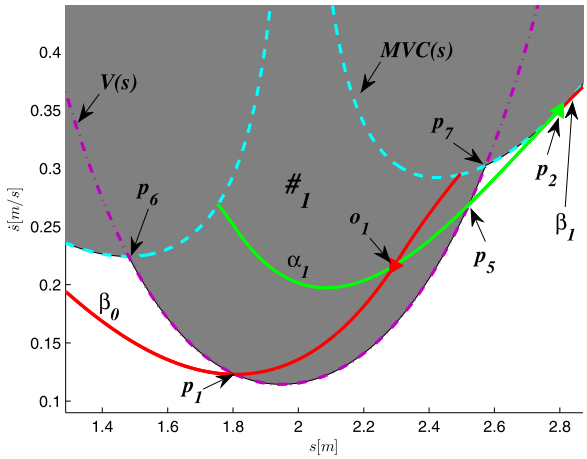

Fig. 11. The enlarged view of the region $\#_1$ in Fig. 10.

indicates that the conditions of *Property 6* hold and also verifies *Property 5*.

Under the curve MVC^*, the *NI* method fails to output a feasible trajectory, which is described as follows. For clarity, the region $\#_1$ is enlarged as shown in Fig. 11. The integrated forward curve β_0 from $(0, \dot{s}_0)$ hits the MVC^* at p_1. Then, searching forward from p_1 along MVC^*, the first $sp_{\alpha\to\beta}$ found is the point p_2. However, starting from p_2, the integrated backward α_1 hits the MVC^* at p_5 before intersecting β_0 (the point o_1 is in the inadmissible region). Then, the integrated forward β_1 from p_2 hits the MVC^* at p_3. No $sp_{\alpha\to\beta}$ is found along MVC^* from p_3 when $s \le s_e$. Therefore, the integrated backward α_e, starting from $(s_e, \dot{s}_e)$, intersects β_1 at o_2. In all these procedures, the $\alpha\text{-}profiles$ on the right side of the purple dash-dot curve $MVC^{\dagger}$ ($\widehat{p_6 p_7}$), cannot intersect β_0. This $MVC^{\dagger}$ ($\widehat{p_6 p_7}$) breaks the intersection between the accelerating and decelerating curves, and causes that the final trajectory is blank between p_1 and p_5, which indicates that the *NI* method fails and verifies *Property 6*.

V. Conclusion

This letter revisits the original version of *NI* method for time-optimal trajectory planning along specified paths. On this basis, we first summarize several known and new properties regarding switch points and accelerating/decelerating curves of the *NI* method, and give corresponding mathematical proofs. Then, we provide concrete failure conditions and rigorous proofs for the property, which indicates that, in the presence of velocity constraints, the original version of *NI* which only considers torque constraints may result in failure of trajectory planning tasks. Accordingly, a failure detection algorithm is given in a 'run-and-test' manner. Simulation results on a unicycle vehicle are provided to verify these presented properties.

Appendix

A. Proof of Property 2

Proof: In the $AREM$ region, an $\alpha\text{-}profile$ intersects an $\beta\text{-}profile$ at a point $(s = s_c, \dot{s} = \dot{s}_c)$. At the neighborhood of s_c, the slopes of the $\alpha\text{-}profile$ and $\beta\text{-}profile$ satisfy the

inequality $k_\alpha < k_\beta$ and the Lipschitz condition [11]. Therefore, based on *Comparison Theorem* [27] (Let y, z be solutions of the differential equations $\dot{y} = F(x, y), \dot{z} = G(x, z)$. If $F(x, y) < G(x, z), x \in [a, b]$, the function F or G satisfies a Lipschitz condition, and $y(a) = z(a)$, then $y(x) < z(x), x \in (a, b]$), it is proven that the $\alpha\text{-}profile$ is greater than the $\beta\text{-}profile$ in the left neighborhood of s_c, but less than the $\beta\text{-}profile$ in the right neighborhood of s_c. ■

B. Proof of Property 3

Proof: This property is proven by contradiction. Assume that an $\alpha\text{-}profile$ is tangent to another $\beta\text{-}profile$ in the $AREM$ region. Then, on the tangent point, the slope k_α of the $\alpha\text{-}profile$ is equal to k_β of the $\beta\text{-}profile$, which contradicts with the inequality $k_\alpha < k_\beta$ in the $AREM$ region. Thus, the assumption is invalid and the property is proven. ■

C. Proof of Property 4

Proof: In terms of the number m of intersection points, there are totally two cases: $m = 1, m > 1$.

Case 1: $m = 1$. There is only one intersection point, so the point $sp_{\beta\to\alpha}$ on the $\beta\text{-}profile$ is X_1. This property holds for $m = 1$.

Case 2: $m > 1$. There are m intersection points as Fig. 2. In terms of path coordinate, the intersection point X_i is less than $X_j, 1 \le i < j \le m$. Each X_i has one corresponding decelerating curve α_i and one switch point Y_i. According to *Property 1*, Y_i is at the right side of Y_j, $i < j$. If X_i, $i > 1$ is chosen as $sp_{\beta\to\alpha}$ on β^*, then, starting from X_i, the trajectory consisting of $\alpha\text{-}profiles$ and $\beta\text{-}profiles$ cannot leave the region D, which is enclosed by α_1, β^* and MVC, across α_1 due to *Properties 1-3*. Thus, X_1 is chosen as $sp_{\beta\to\alpha}$ on β^*, which can aid the trajectory to leave the region D along α_1 and go on extending to the right side of Y_1 with β_1. This property holds for $m > 1$. In summary, *Property 4* holds. ■

D. Proof of Property 5

Proof: Due to (10), $MVC^\dagger$ is less than MVC. According to the definition of AR, the inequality $\alpha(s, \dot{s}) < \beta(s, \dot{s})$ holds on $MVC^\dagger$. Then, based on the facts $k_\alpha = \alpha(s, \dot{s})/\dot{s}, k_\beta = \beta(s, \dot{s})/\dot{s}$, the inequality $k_\alpha < k_\beta$ also holds on $MVC^\dagger$, which violates $k_\alpha = k_\beta$ in the definition of tangent switch points (see $sp_{\alpha\to\beta}$ in Section II-B). Therefore, tangent switch points on $MVC^\dagger$ are nonexistent. ■

REFERENCES

[1] E. Barnett and C. Gosselin, "Time-optimal trajectory planning of cable-driven parallel mechanisms for fully-specified paths with G1 discontinuities," *J. Dyn. Syst. Meas. Control*, vol. 137, no. 7, pp. 603–617, 2013.

[2] K. Kim and B. Kim, "Minimum-time trajectory for three-wheeled omnidirectional mobile robots following a bounded-curvature path with a referenced heading profile," *IEEE Trans. Robot.*, vol. 27, no. 4, pp. 800–808, Aug. 2011.

[3] S. Macfarlane and E. Croft, "Jerk-bounded manipulator trajectory planning: Design for real-time applications," *IEEE Trans. Robot. Autom.*, vol. 19, no. 1, pp. 42–52, Feb. 2003.

[4] K. Shin and N. McKay, "Selection of near-minimum time geometric paths for robotic manipulators," *IEEE Trans. Autom. Control*, vol. 31, no. 6, pp. 501–511, Jun. 1986.

[5] K. Shin and N. McKay, "A dynamic programming approach to trajectory planning of robotic manipulators," *IEEE Trans. Autom. Control*, vol. 31, no. 6, pp. 491–500, Jun. 1986.

[6] S. Singh and M. Leu, "Optimal trajectory generation for robotic manipulators using dynamic programming," *J. Dyn. Syst. Meas. Control*, vol. 109, no. 2, pp. 88–96, 1987.

[7] D. Verscheure, B. Demeulenaere, J. Swevers, J. Schutter, and M. Diehl, "Time-optimal path tracking for robots: A convex optimization approach," *IEEE Trans. Autom. Control*, vol. 54, no. 10, pp. 2318–2327, Oct. 2009.

[8] K. Hauser, "Fast interpolation and time-optimization with contact," *Int. J. Robot. Res.*, vol. 33, no. 9, pp. 1231–1250, 2014.

[9] F. Debrouwere *et al.*, "Time-optimal path following for robots with convex-concave constraints using sequential convex programming," *IEEE Trans. Robot.*, vol. 29, no. 6, pp. 1485–1495, Dec. 2013.

[10] K. Shin and N. Mckay, "Minimum-time control of robotic manipulators with geometric path constraints," *IEEE Trans. Autom. Control*, vol. 30, no. 6, pp. 531–541, Jun. 1985.

[11] J. Bobrow, S. Dubowsky, and J. Gibson, "Time-optimal control of robotic manipulators along specified paths," *Int. J. Robot. Res.*, vol. 4, no. 3, pp. 3–17, 1985.

[12] J. Slotine and H. Yang, "Improving the efficiency of time-optimal path-following algorithms," *IEEE Trans. Robot. Autom.*, vol. 5, no. 1, pp. 118–124, Feb. 1989.

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

[14] F. Pfeiffer and R. Johanni, "A concept for manipulator trajectory planning," *IEEE J. Robot. Autom.*, vol. 3, no. 2, pp. 115–123, Apr. 1987.

[15] Z. Shiller, "On singular time-optimal control along specified paths," *IEEE Trans. Robot. Autom.*, vol. 10, no. 4, pp. 561–566, Aug. 1994.

[16] Q. Pham and O. Stasse, "Time-optimal path parameterization for redundantly actuated robots: A numerical integration approach," *IEEE/ASME Trans. Mechatronics*, vol. 20, no. 6, pp. 3257–3263, Dec. 2015.

[17] S. Behzadipour and A. Khajepour, "Time-optimal trajectory planning in cable-based manipulators," *IEEE Trans. Robot.*, vol. 22, no. 3, pp. 559–563, Jun. 2006.

[18] T. Kunz and M. Stilman, "Time-optimal trajectory generation for path following with bounded acceleration and velocity," *Robot., Sci. Syst.*, vol. 8, no. 12, pp. 9–13, 2012.

[19] H. Nguyen and Q. Pham, "Time-optimal path parameterization of rigid-body motions: Applications to spacecraft reorientation," *J. Guid., Control Dyn.*, vol. 39, no. 7, pp. 1667–1671, 2016.

[20] Q. Pham and Y. Nakamura, "Time-optimal path parameterization for critically dynamic motions of humanoid robots," in *Proc. 2012 IEEE Int. Conf. Humanoid Robots*, 2012, pp. 165–170.

[21] L. Žlajpah, "On time optimal path control of manipulators with bounded joint velocities and torques," in *Proc. 1996 IEEE Int. Conf. Robot. Autom.*, 1996, pp. 1572–1577.

[22] F. Lamiraux and J. Laumond, "From paths to trajectories for multi-body mobile robots," in *Proc. 5th Int. Symp. Exp. Robot.*, 1998, pp. 301–309.

[23] T. Kalmár-Nagy, "Real-time trajectory generation for omni-directional vehicles by constrained dynamic inversion," *Mechatronics*, vol. 35, pp. 44–53, 2016.

[24] Q. Pham, S. Caron, and Y. Nakamura, "Kinodynamic planning in the configuration space via velocity interval propagation," in *Proc. Robot., Sci. Syst.*, 2013, pp. 8–15.

[25] H. Sussman, "The Markov-Dubins problem with angular acceleration control," in *Proc. 36th IEEE Conf. Decision Control*, 1997, pp. 2639–2643.

[26] C. Bianco and M. Romano, "Optimal velocity planning for autonomous vehicles considering curvature constraints," in *Proc. 2007 IEEE Int. Conf. Robot. Autom.*, 2007, pp. 2706–2711.

[27] G. Birkhoff and G. Rota, "Ordinary differential equations," *USSR Comput. Math. Phys.*, vol. 4, no. 4, pp. 230–232, 1964.