Int. J. Appl. Math. Comput. Sci., 2022, Vol. 32, No. 1, 11–21
DOI: 10.34768/amcs-2022-0002

FAST AND SMOOTH TRAJECTORY PLANNING FOR A CLASS OF LINEAR SYSTEMS BASED ON PARAMETER AND CONSTRAINT REDUCTION

GUANGYU LIU [a], SHANGLIANG WU [a], LING ZHU [b,*], JIAJUN WANG [a], QIANG LV [a]

[a]School of Automation Engineering
Hangzhou Dianzi University
Xiasha Higher Education Zone, Hangzhou, Zhejiang Province, China
e-mail: guangyu_liu@hotmail.com

[b]School of Information Management and Artificial Intelligence
Zhejiang University of Finance and Economics
18 Xueyuan Street, Xiasha Higher Education Zone, Hangzhou City, Zhejiang Province, China
e-mail: zhuling@zufe.du.cn

Fast and smooth trajectory planning is crucial for modern control systems, e.g., missiles, aircraft, robots and AGVs. However, classical spline based trajectory planning tools introduce redundant constraints and parameters, leading to high costs of computation and complicating fast and smooth execution of trajectory planning tasks. A new tool is proposed that employs truncated power functions to annihilate some constraints and reduce the number of parameters in the optimal model. It enables solving a simplified optimal problem in a shorter time while keeping the trajectory sufficiently smooth. With an engineering background, our case studies show that the proposed method has advantages over other solutions. It is promising in regard to the demanding tasks of trajectory planning.

Keywords: constraint reduction, parameter reduction, fast calculation, trajectory planning.

1. Introduction

Trajectory planning plays a significant role in guidance and control of dynamical systems, such as robots (Qian *et al.*, 2020; Yu *et al.*, 2020; Li *et al.*, 2021; Tatematsu and Ohnishi, 2003; Cheon and Kim, 2019) missiles (Liu *et al.*, 2018; 2020), autonomous underwater vehicles (Sun and Liu, 2021) and flights (Muscio *et al.*, 2018; Spedicato and Notarstefano, 2018; Heidari and Saska, 2020; Park and Kim, 2021). There are two ways to fulfill the task. Indirect methods of trajectory planning resort to the necessary conditions derived from the minimum principle, which usually ends up with boundary value problems (Wang *et al.*, 2013). They yield good accuracies. However, they suffer from some difficulties in deriving necessary conditions (Wang *et al.*, 2013; Gong *et al.*, 2006) and the sensitivities to initial guesses of boundary value problems (Gong *et al.*, 2006). Direct methods use a strategy to discretize an optimal control problem into a parameterized optimization problem that can be solved by applying the techniques of nonlinear programming (NLP) (Fahroo and Ross, 2000). The parameterized methods include S-curves (Tho *et al.*, 2020; Wang *et al.*, 2020; Aguilar-Ibanez and Suarez-Castanon, 2019), polynomial functions (Desai *et al.*, 2019; Cui *et al.*, 2020) , B-splines (Schoenberg, 1969; Boor and Fix, 1973; Mercy *et al.*, 2018; Rousseau *et al.*, 2019) and sinusoidal functions (Liu *et al.*, 2021; Fang *et al.*, 2020). Among them, trajectory planning via spline approximation attracts more attentions (Berselli *et al.*, 2016; Kroger and Wahl, 2010).

Spline approximation for trajectory planning has different forms, which contains different groups of parameters. Spline functions have been used extensively; they include the piece-wise polynomial form (PPF) and classical the cubic spline form (CCF). The forms for the spline with some nodes of a fixed degree are given in Table 1.

When using the PPF, splines are expressed via piecewise polynomial functions under certain conditions of smoothness (Berselli *et al.*, 2016) and coefficients

*Corresponding author

Table 1. Formulas for the PPF and CCF.

Name	Formula
PPF	$s_{n,\,N}(t) = \sum\limits_{j=0}^{n} K_{i,\,j} t^j,\ t \in [t_i,\ t_{i+1}]$ ①
CCF	$s_{n,\,N}(t) = \dfrac{s_i^{(2)}(t_{i+1}-t)^3 + s_{i+1}^{(2)}(t-t_i)^3 + \left(6s_i - s_i^{(2)}\delta t_i^2\right)(t_{i+1}-t) + \left(6s_{i+1} - s_{i+1}^{(2)}\delta t_i^2\right)(t-t_i)}{6\delta t_i},\ t \in [t_i,\ t_{i+1}]$ ②

① $t_1,\ t_2,\ \ldots,\ t_{N+1}$ are spline nodes, which are sorted from small to large.
② $\delta t_i = t_i - t_{i-1}$, $s_i = s_{n,\,N}(t_i)$ and $s_i^{(2)} = s_{n\,N}^{(2)}(t_i)$, for $i = 1,\ 2,\ \ldots,N$.

of piece-wise polynomials are regarded as parameters to be optimized. Therefore, the PPF introduces a large number of optimization variables and equality constraints. On the one hand, a large number of variables and constraints would cost computational time in each iterative step when solving optimization problems. On the other hand, equality constraints tend to generate sparse feasible regions which increase the number of iterative steps and create possibly infeasible solutions. When using the CCF, the functional and derivative values at nodes are considered as parameters. In addition, spline expressions and smoothness conditions are obtained through integration and differentiation, respectively. However, some issues are unsolved. For example, there are some equality constraints for boundary conditions and smoothness, which makes the feasible solutions sparse. Furthermore, the CCF can only guarantee the second-order smoothness, which restricts the scope of application. In conclusion, the computing speed of these classical forms is slow for generating a smooth trajectory.

Currently, the computational efficiency is important to the performance index of trajectory planning method. This is because many industrial applications imply dynamically changed environments where unpredictable situations would affect the trajectories to be planned. Very often, the trajectories should be altered on-line to cope with unforeseen situations on site to avoid any damage (Zhang *et al.*, 2020; Cheon and Kim, 2019). Therefore, how to improve the computational efficiency is a hot topic in academia and industry, which needs to be investigated further.

Admittedly, some advanced methods are developed to alleviate the computational load. An artificial intelligence method is used to decouple the multiple joints of robots to divide a complex NLP into simple ones, mitigating the computational load (Zhang *et al.*, 2020). Heuristic techniques are applied to minimize the computational cost for finding optimal trajectories of mobile robots (Kim, 2020). Unfortunately, artificial intelligence tools rely on large databases and require tedious off-line training, making the modeling process complicated.

Here, we have two research objectives. First, we focus on modifying the truncated power form (TPF) to alleviate the computational load; another objective is to propose a fast and smooth trajectory planning tool for a class of linear systems with some normal form. The modified truncated power form (MTPF) annihilates original equality constraints and eliminates unnecessary parameters, improving the computational efficiency. The tool is incorporated into the process of optimal trajectory planning. Experimental results show some advantages of the MTPF over the existing tools including the PPF, DIF, and TPF, e.g., the computational time.

The paper is organized as follows. Section 2 introduces a linear system with a certain normal form and defines the TPF. Section 3 defines a problem of trajectory parameterization. The MTPF-based method is proposed and a simplified model of optimization is derived in Section 4. In Section 5, some case studies are investigated to evaluate the method's performance. Final conclusion is given in Section 6.

2. Preliminaries

2.1. System model. Consider the class of linear systems described by

$$\begin{cases} \dot{x}(t) = Ax(t) + Bu(t), \\ y(t) = Cx(t), \end{cases} \tag{1}$$

where $x(t) = \big(x_1(t), x_2(t), \ldots, x_n(t)\big)^{\mathrm{T}}$ is the state vector,

$$A = \begin{pmatrix} 0 & 1 & 0 & \ldots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ -a_0 & -a_1 & -a_2 & \cdots & -a_{n-1} \end{pmatrix},$$

is the state matrix in $\mathbb{R}^{n\times n}$, $B = \big(0, 0, \ldots, b\big)^{\mathrm{T}}$ $(b \neq 0)$ is the input matrix in $\mathbb{R}^{n\times 1}$, and $C = \big(1, 0, \ldots, 0\big)$ is the output matrix in $\mathbb{R}^{1\times n}$. In addition, $\dot{x}_i(t)$ for $i = 1,\ 2, \ldots, n$ is bounded and Riemann integrable.

2.2. Splines of the TPF. In trajectory planning, the PPF is the most fundamental to express a spline via piecewise polynomial functions under certain conditions of smoothness (Berselli *et al.*, 2016). However, it introduces equality constraints and a large number of parameters. To alleviate the problem, truncated power functions are developed, which make these polynomial functions simple (Powell, 1981).

Definition 1. The *truncated power function* is defined by

$$t_+^l = \begin{cases} t^l, & t \geq 0, \\ 0, & t < 0, \end{cases} \tag{2}$$

where t_+^l is simplified as t_+ for $l = 1$ and becomes $0^0 = 1$ for $t = 0$ and $l = 0$.

Definition 2. The TPF is used to express a spline such that $s_{n,N} : [t_1, t_{N+1}] \to \mathbb{R}$ is the spline function of the n-th degree with its nodes $t_1, t_2, \dots, t_{N+1}$ satisfying $t_1 < t_2 < \dots < t_{N+1}$, if $s_{n,N}(t)$ can be expressed as

$$s_{n,N}(t) = \sum_{i=0}^{n} K_i t^i + \sum_{i=2}^{N} \alpha_i (t - t_i)_+^n , \tag{3}$$

where K_i, $\alpha_j \in \mathbb{R}$, for $i = 0, 1, \dots, n$ and $j = 2, 3, \dots, N$.

Given $t_+^l \in C^{l-1}$, the smoothness is guaranteed.

3. Problem formulation

3.1. Some constraints of trajectory planning. In control applications, the output variable is usually steady about some value at the initial stage and about another value at its final stage. For example, gantry cranes need to be stable at the beginning and the ending of a conveying task in order to avoid collisions or swings. This leads to some conditions.

Condition 1. *One of boundary conditions is the initial condition given by*

$$x(t_s) = (y_s, 0, \dots, 0)^{\mathrm{T}} , \tag{4}$$

where y_s is the initial output.

Condition 2. *Another boundary condition is the final condition*

$$x(t_f) = (y_f, 0, \dots, 0)^{\mathrm{T}} , \tag{5}$$

where y_f is the target output.

Moreover, many related physical variables which can be expressed linearly by the state variables need to be in numerical intervals.

Condition 3. *The constraints on the state and control are represented by*

$$G\Big((x(t))^{\mathrm{T}} , u(t) \Big)^{\mathrm{T}} \leq \overline{V}, \tag{6}$$

$$G\Big((x(t))^{\mathrm{T}} , u(t) \Big)^{\mathrm{T}} \geq \underline{V}, \tag{7}$$

for $G \in \mathbb{R}^{m \times (n+1)}$, $\overline{V} \in \mathbb{R}^m$ and $\underline{V} \in \mathbb{R}^m$.

3.2. Optimal trajectory planning model. We consider a task of trajectory planning for system (1) subject to Conditions 1, 2 and 3. A general optimization problem associated with trajectory planning is given by,

$$(x^*(t), u^*(t)) = \arg\min J(x(t), u(t)) \tag{8}$$

subject to

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t), \\ x(t_s) &= (y_s, 0, \dots, 0)^{\mathrm{T}} , \\ x(t_f) &= (y_f, 0, \dots, 0)^{\mathrm{T}} , \\ G\Big((x(t))^{\mathrm{T}} , u(t) \Big)^{\mathrm{T}} &\leq \overline{V}, \\ G\Big((x(t))^{\mathrm{T}} , u(t) \Big)^{\mathrm{T}} &\geq \underline{V}, \end{aligned}$$

where $J(x(t), u(t))$ is a cost function.

3.3. Problem statement. With respect to system (1), we consider a novel problem of spline approximation associated with the generic constrained optimization problem (8).

Problem 1. The problem is to seek a parameterized form $y(P, t) : \mathbb{R}^{N_P} \times \mathbb{R} \to \mathbb{R}$ for the actual output $y(t)$ such that for any fixed P, $y(P, t)$ is in C^{n-1} with respect to t.

Remark 1. In many fields (e.g., robots and mechanical control systems), the smoothness of the tracking trajectory is desirable, which aids the tracking controller to perform better.

Remark 2. To improve the computational speed, it is desirable to reduce the number of variables (i.e., the parameters) and the number of the constraints of the form $y(P, t)$, because an increase in these numbers may create sparse feasible regions, which aggravates the computational load of optimization.

4. Main results

4.1. Proposed method of the MTPF. We consider a classical TPF for spline approximation $y(P, t)$ about the output trajectory $y(t)$. Its starting node t_1 is fixed at the initial time t_s of $y(t)$ and its ending node t_{N+1} is

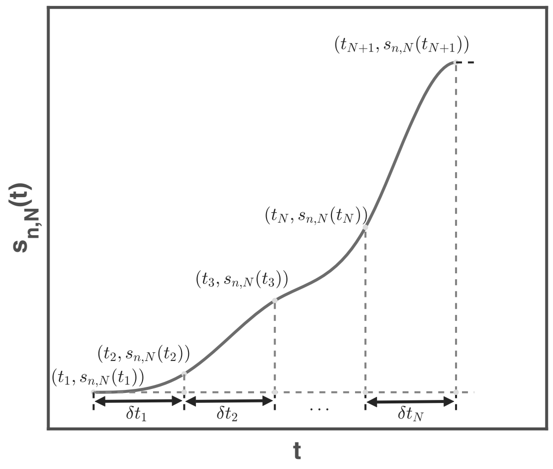


Fig. 1. Illustration for parameter variables.

fixed at the final time t_f of $y(t)$. The existing methods of the TPF take $n + 2N$ parameters into account, i.e., $\{t_i,\ i = 2, 3 \dots, N + 1\}$, $\{K_i,\ i = 0, 1, \dots, n\}$, and $\{\alpha_i,\ i = 2, 3, \dots, N\}$.

To begin with, the TPF is modified to decrease the number of K_i under Condition 1.

Lemma 1. *Consider boundary conditions $s_{n,N}(t_1) = y_s$ and $s^{(i)}_{n,N}(t_1) = 0$, for $i = 1, 2, \dots, n-1$ for the spline $s_{n,N}(t)$. This implies*

$$s_{n,N}(t) = y_s + \big((t - t_1)^n_+,\ (t - t_2)^n_+,\ \dots,\ (t - t_N)^n_+\big) \times \alpha \tag{9}$$

where $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_N)^{\mathrm{T}} \in \mathbb{R}^N$. Conversely, the spline (9) implies also the given boundary condition.

Proof. See Appendix for details (A1). ■

Remark 3. In comparison with $n + 2N$ parameters of the TPF, the proposed MTPF decreases the number of parameters by n.

Secondly, our further treatment of the TPF is going to reduce redundant variables in α_i, for $i = 1, 2, \dots, N$ under Condition 2.

Definition 3. The element of a vector $\Delta t = (\delta t_1, \delta t_2, \dots, \delta t_N)^{\mathrm{T}}$ is defined by $\delta t_i = t_{i+1} - t_i$. See Fig. 1 for the definition.

For the convenience of calculation, we apply the vector of parameters Δt instead of t_i, for $i = 2, 3, \dots, N+1$. In this case, $t_j > t_i$ for $j > i$ is equivalent to $\Delta t > 0$.

Definition 4. Define $D : \mathbb{R}^N \times \mathbb{R} \to \mathbb{R}^{n\times N}$ as

$$D(\Delta t,\ t) = \begin{pmatrix} (t-t_1)^n_+ & (t-t_2)^n_+ & \dots & (t-t_N)^n_+ \\ (t-t_1)^{n-1}_+ & (t-t_2)^{n-1}_+ & \dots & (t-t_N)^{n-1}_+ \\ \vdots & \vdots & \ddots & \vdots \\ (t-t_1)_+ & (t-t_2)_+ & \dots & (t-t_N)_+ \end{pmatrix}. \tag{10}$$

It is assumed that $s_{n,\,N}(t)$ is a spline in Eqn. (9). Final conditions, $s_{n,\,N}(t_{N+1}) = y_f$ and $s^{(i)}_{n,\,N}(t_{N+1}) = 0$, $i = 1, 2, \dots, n-1$, are equivalent to

$$D(\Delta t, t_{N+1})\alpha = \beta. \tag{11}$$

for $\beta = (y_f - y_s,\ 0, \dots,\ 0)^{\mathrm{T}}$.

Lemma 2. *Given $y_f \neq y_s$, the necessary and sufficient condition for the existence of $\Delta t \in \mathbb{R}^N_+$, and $\alpha \in \mathbb{R}^N$ is given by*

$$N \geq n\,, \tag{12}$$

which satisfies (11)

Proof. See Appendix for details (A2). ■

Remark 4. According to Lemma 2, if $N \geq n$, there exists a pair of variables $(\Delta t,\ \alpha)$ in $\mathbb{R}^N_+ \times \mathbb{R}^N$ satisfying (11). Otherwise, there is no pair in $\mathbb{R}^N_+ \times \mathbb{R}^N$ satisfying (11).

Definition 5. Let $D_1(\Delta t) \in \mathbb{R}^{n\times(N-n)}$ be a matrix consisting of the first $N - n$ columns of $D(\Delta t,\ t_{N+1})$, i.e.,

$$D_1(\Delta t) = \begin{pmatrix} \left(\sum\limits_{i=1}^{N} \delta t_i\right)^n & \left(\sum\limits_{i=2}^{N} \delta t_i\right)^n & \dots & \left(\sum\limits_{i=N-n}^{N} \delta t_i\right)^n \\ \left(\sum\limits_{i=1}^{N} \delta t_i\right)^{n-1} & \left(\sum\limits_{i=2}^{N} \delta t_i\right)^{n-1} & \dots & \left(\sum\limits_{i=N-n}^{N} \delta t_i\right)^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ \sum\limits_{i=1}^{N} \delta t_i & \sum\limits_{i=2}^{N} \delta t_i & \dots & \sum\limits_{i=N-n}^{N} \delta t_i \end{pmatrix}. \tag{13}$$

Furthermore, let $D_2(\Delta t) \in \mathbb{R}^{n\times n}$ be a matrix consisting of the last n columns of $D(\Delta t,\, t_{N+1})$, i.e.,

$$D_2(\Delta t) = \begin{pmatrix} \Big(\sum\limits_{i=N-n+1}^{N} \delta t_i\Big)^n & \Big(\sum\limits_{i=N-n+2}^{N} \delta t_i\Big)^n & \cdots & (\delta t_N)^n \\ \Big(\sum\limits_{i=N-n+1}^{N} \delta t_i\Big)^{n-1} & \Big(\sum\limits_{i=N-n+2}^{N} \delta t_i\Big)^{n-1} & \cdots & (\delta t_N)^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ \sum\limits_{i=N-n+1}^{N} \delta t_i & \sum\limits_{i=N-n+2}^{N} \delta t_i & \cdots & \delta t_N \end{pmatrix}. \tag{14}$$

$D_1(\Delta t)$ is not defined but $D(\Delta t,\, t_{N+1}) = D_2(\Delta t)$ holds for $N = n$.

Definition 6. Let $\alpha^1 \in \mathbb{R}^{N-n}$ be a vector consisting of initial elements of α from 1 to $N-n$ that leads to $\alpha^1 = (\alpha_1,\, \alpha_2,\, \dots,\, \alpha_{N-n})^{\mathrm{T}}$. Let $\alpha^2 \in \mathbb{R}^n$ be a vector consisting of final elements of α from $N-n+1$ to N that gives $\alpha^2 = (\alpha_{N-n+1},\, \alpha_{N-n+2},\, \dots,\, \alpha_N)^{\mathrm{T}}$.

According to Definitions 5 and 6, Eqn. (11) has a form $\beta = D_1(\Delta t)\alpha^1 + D_2(\Delta t)\alpha^2$, which implies

$$\alpha^2 = (D_2(\Delta t))^{-1}(\beta - D_1(\Delta t)\alpha^1), \tag{15}$$

where $D_2(\Delta t)$ is an invertible matrix for all $\Delta t > 0$. It leads to a revised definition of α.

Definition 7. For brevity,

$$\alpha = \begin{pmatrix} \alpha^1 \\ (D_2(\Delta t))^{-1}(\beta - D_1(\Delta t)\alpha^1) \end{pmatrix} \tag{16}$$

is used to represent $\alpha = \alpha(\Delta t,\, \alpha^1)$.

An update parameterized form for $y(t)$ with $P := (\Delta t^{\mathrm{T}},\, \alpha^{1\mathrm{T}})^{\mathrm{T}}$ is obtained by

$$\begin{aligned} y(P,\, t) =& y_s + \big((t-t_1)_+^n,\, (t-t_2)_+^n,\, \dots, \\ & (t-t_N)_+^n\big) \times \alpha(\Delta t,\, \alpha^1). \end{aligned} \tag{17}$$

Furthermore, the state vector $x(t)$ and the control input $u(t)$, derived from $y(P,t)$, have parameterized forms

$$x(P,\, t) = \left(y(P,t),\, \frac{\partial^1 y(P,t)}{\partial t^1}, \dots, \frac{\partial^{n-1} y(P,t)}{\partial t^{n-1}}\right)^{\mathrm{T}}, \tag{18}$$

$$u(P,\, t) = \sum_{i=0}^{n} \frac{a_i}{b} \frac{\partial^i y(P,t)}{\partial t^i}, \tag{19}$$

where b and a_i ,$i = 0, \dots, n-1$ are as given in Section 2.1, and $a_n = 1$. They can also be expressed by

$$x(P,\, t) = (y_s,\, 0, \dots,\, 0)^{\mathrm{T}} + \Lambda D(\Delta t,\, t)\alpha(\Delta t,\, \alpha^1), \tag{20}$$

$$\begin{aligned} u(P,\, t) =& \frac{a_0 y_s}{b} + \Big(\frac{a_0}{b}, \dots, \frac{a_n}{b}\Big)\overline{\Lambda} & (21) \\ & \times \overline{D}(\Delta t, t)\alpha(\Delta t,\, \alpha^1), & (22) \end{aligned}$$

where

$$\Lambda = \operatorname{diag}\Big\{1, \textstyle\prod_{i=N}^{N} i, \dots, \prod_{i=2}^{N} i\Big\},$$
$$\overline{\Lambda} = \operatorname{diag}\Big\{\Lambda, \textstyle\prod_{i=1}^{N} i\Big\},$$

$$\overline{D}(\Delta t,\, t) = \begin{pmatrix} (t-t_1)_+^n & (t-t_2)_+^n & \cdots & (t-t_N)_+^n \\ (t-t_1)_+^{n-1} & (t-t_2)_+^{n-1} & \cdots & (t-t_N)_+^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ (t-t_1)_+^0 & (t-t_2)_+^0 & \cdots & (t-t_N)_+^0 \end{pmatrix}.$$

Remark 5. For any $P \in \mathbb{R}^{2N-n}$, the parameterized form $x(P,\, t)$ must satisfy Conditions 1 and 2, i.e., $x(P,\, t_s) = (y_s,\, 0,\, \dots,\, 0)^T$ and $x(P,\, t_{N+1}) = (y_f,\, 0,\, \dots,\, 0)^{\mathrm{T}}$. This implies that Conditions 1 and 2 can be eliminated.

In summary, the proposed MTPF has fewer parameters and constraints than the same number associated with the original TPF.

4.2. Trajectory planning with the MTPF.

4.2.1. Revised Condition 3.

Inequalities (6) and (7) in Condition 3 are transformed to

$$g_i(P) \le 0, \quad i = 1, 2, \dots, 2m, \tag{23}$$

where the group of functions $g_i(P)$ is defined by

$$g_i(P) = \sup_{t\in[t_1, t_{N+1}]} \Big(G_i\left((x(t))^{\mathrm{T}},\, u(t)\right)^{\mathrm{T}} - \overline{V}_i\Big), \tag{24}$$

for $i = 1,\, 2,\, \dots,\, m$, and as

$$\begin{aligned} g_i(P) = \sup_{t\in[t_1, t_{N+1}]} \Big(& \underline{V}_{i-m} - G_{i-m} \\ & \times \left((x(t))^{\mathrm{T}},\, u(t)\right)^{\mathrm{T}}\Big), \end{aligned} \tag{25}$$

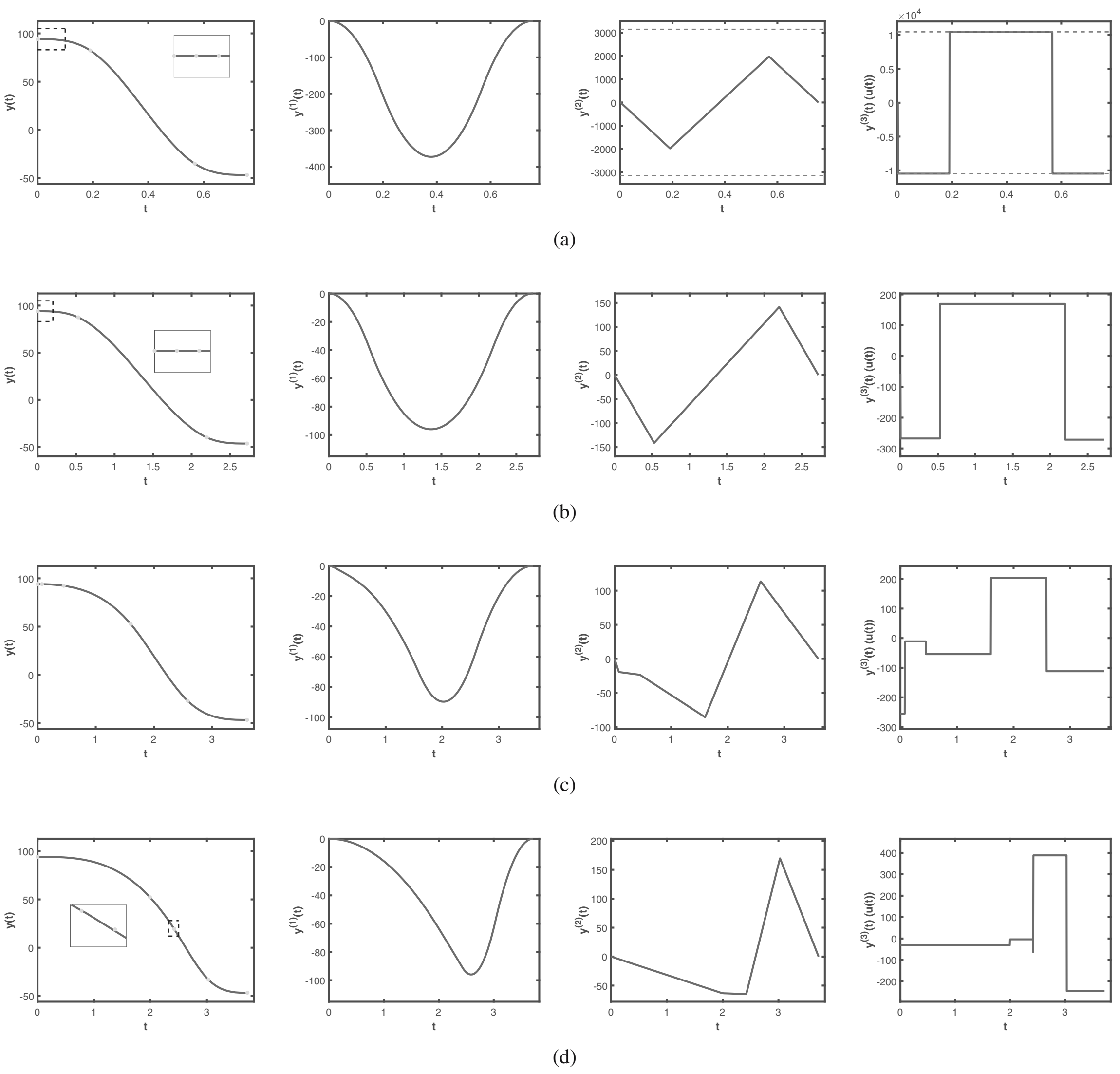


Fig. 2. Results with P^1_{init} obtained by different spline form based methods: MTPF (a), TPF (b), CCF (c), PPF (d).

for $i = m+1, m+2, \ldots, 2m$, with G_i being the i-th row vector of G, $\overline{V}_i$ being the i-th element of $\overline{V}$, and $\underline{V}_i$ being the i-th element of $\underline{V}$.

It is noted that revised Condition 3 is equivalent to Condition 3, when output, state and control are given by (17), (20) and (21), respectively. In this case, the inequality constraints on trajectories are reduced to the constraints on parameters.

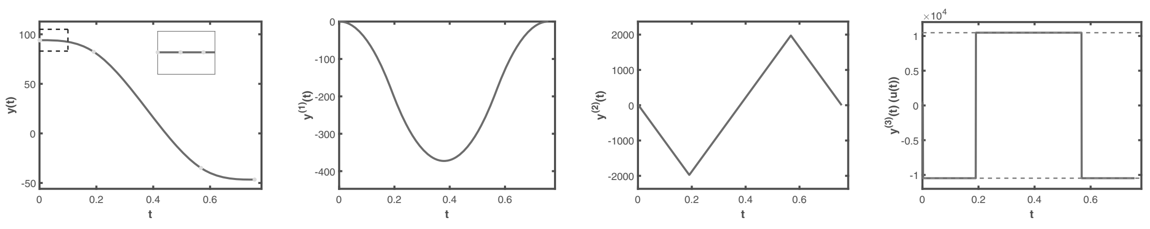
4.2.2. Optimal trajectory planning algorithm. If the state and control are parameterized as (20) and (21), respectively, the trajectory planing problem (8) with respect to $P = (\Delta t^T, \alpha^{1\,\mathrm{T}})^{\mathrm{T}}$ can be rewritten as

$$P^* = \arg\min J(x(P,t), u(P,t)) \tag{26}$$

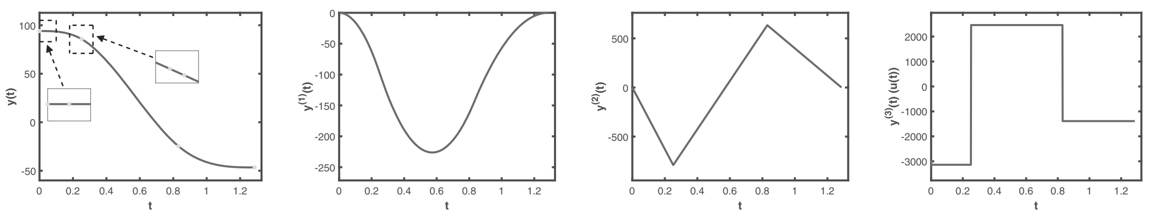
subject to

$$\begin{aligned} \Delta t &> 0, \\ g_i(P) &\leq 0, \quad i = 1, 2, \ldots, 2m, \end{aligned}$$

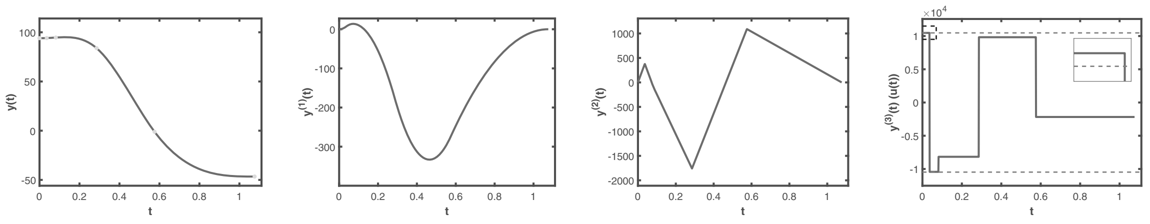
which takes advantage of the MTPF to construct an optimization problem for trajectory planning. Then, one can revoke some optimization tool to solve the optimization problem and generate a set of desirable trajectories that include $y^*(t) = y(P^*, t)$, $x^*(t) = x(P^*, t)$ and $u^*(t) = u(P^*, t)$, where the smoothness is guaranteed. The novel approach employs fewer parameters and constraints than those of the TPF. The optimal trajectory planning algorithm is described in Algorithm 1.



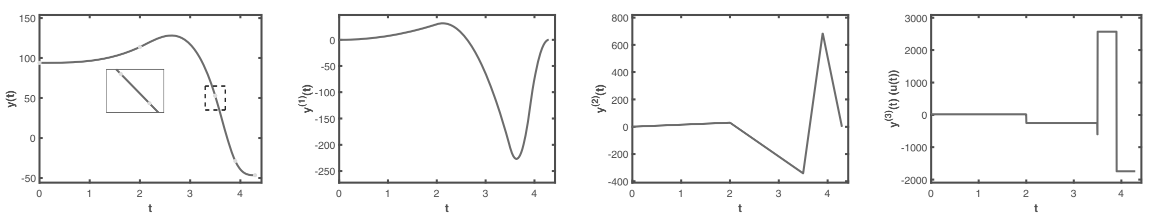
(a)



(b)



(c)



(d)

Fig. 3. Results with P^2_{init} obtained by different spline form based methods: MTPF (a), TPF (b), CCF (c), PPF (d).

5. Case study

5.1. Trajectory planning of an ideal motor. We take an ideal motor as a case study to evaluate the proposed trajectory planning method, where the trajectory planing of the motor's rotational variable $y(t)$ is formulated as the optimization problem (8) with the state and input matrices given by

$$A = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}, \quad B = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix},$$

and the corresponding constraints given by

$$G = \operatorname{diag}\{1, 1, 1, 1\},$$
$$\overline{V} = \left(418.88,\, 1.05 \times 10^3,\, 3.14 \times 10^3,\, 1.05 \times 10^4\right)^{\mathrm{T}},$$
$$\underline{V} = -\left(418.88,\, 1.05 \times 10^3,\, 3.14 \times 10^3,\, 1.05 \times 10^4\right)^{\mathrm{T}},$$

$$y_s = 94.02,$$
$$y_f = -46.60.$$

In the experiments, the optimization problem (8) refers to a time-optimal performance index $J(x(t),\, u(t)) = \int_{t_s}^{t_f},\mathrm{d}t$ with the segment number $N = 5$ and the degree $n = 3$. Sequential quadratic programming (SQP) is used to solve the constrained optimization problem to design some desirable trajectories, where two initial points are set in Table 2.

Algorithm 1. Trajectory planning algorithm with the MTPF.

```
Require:
    $n, a_i, i = 0, 1, \dots, n-1, b$: the system parameters
    $m, G, \overline{V}, \underline{V}$: the parameters of inequality constraints
    $t_s, y_s, y_f$: the parameters of boundary conditions
    $J$: the cost function
Ensure:
    $y^*(t)$: the optimal output trajectory
    $x^*(t)$: the optimal state trajectory
    $u^*(t)$: the optimal control trajectory
    Step 1. Set $N$ for $N \geq n$.
    Step 2. $y(P,t) \leftarrow$ Eqn. (17) with $N$, $n$, $y_s$ and $y_f$.
    Step 3. $x(P,t) \leftarrow$ Eqn. (20) with $N$ and $n$.
    Step 4. $u(P,t) \leftarrow$ Eqn. (21) with $N$, $n$, $y_s$ and $a_i$ for $i = 0, 1, \dots, n-1$.
    Step 5. $J(x(P,t), u(P,t)) \leftarrow J$ with $x(P,t)$ and $u(P,t)$.
    Step 6. Set inequality constraints by Eqns. (24), (25) and $\Delta t > 0$ for $m, n, G, \overline{V}, \underline{V}, x(P,t)$ and $u(P,t)$.
    Step 7. Derive $P^*$ via an iterative optimization procedure.
    Step 8. $y^*(t) \leftarrow y(P^*,t)$.
    Step 9. $x^*(t) \leftarrow x(P^*,t)$.
    Step 10. $u^*(t) \leftarrow u(P^*,t)$.
```

For a fair comparison, P^1_{init} and P^2_{init} in Table 2 are used for the tools that include the MTPF, TPF, CCF and PPF. All experiments are executed in the same environment of MATLAB 2021a operated on a laptop computer with an Intel (R) core (TM) i7-7700HQ CPU @ 2.80GHz, a 8.00GB RAM (7.89GB available) and a 64 bit operating system.

5.2. Experimental results for an ideal motor. The trajectory planing problem of the ideal motor is solved by the MTPF, TPF, CCF and PPF respectively, where both P^1_{init} and P^2_{init} are used in the iterative processes. As a result, two groups of the planned $y(t)$, the motor's rotational variable, are generated, which are shown in Figs. 2 and 3 respectively. It is to be noted from $y^{(3)}(t)$ that the results of the MTPF took full advantage of the constraints, -1.05×10^4 rad/s^3 and 1.05×10^4 rad/s^3, while those of the TPF, CCF and PPF are a far cry from these constraints. These advantages the time-optimal performance index arising improve from the MTPF defined by Eqn. (8).

To draw a fair comparison, the performance details associated with the computational time, the minimal cost, the boundary conditions, the smoothness conditions and the inequality conditions are listed in Table 3. All

Table 2. Setting-up of initial points.

	Δt	α^1	Feasibility
P^1_{init}	$(1,1,1,1,1)^{\mathrm{T}}$	$(-1,-1)^{\mathrm{T}}$	Yes
P^2_{init}	$(1,1,1,1,1)^{\mathrm{T}}$	$(80,-150)^{\mathrm{T}}$	No

Table 3. Performance comparison of different methods.

IP[1]	Form	CT[2]	MC[3]	BC[4]	SC[5]	IC[6]
P^1_{init}	MTPF	0.24	0.76	Y[7]	Y	Y
	TPF	0.95	2.72	N[8]	Y	Y
	CCF	0.91	3.60	N	N	Y
	PPF	1.79	3.72	N	N	Y
P^2_{init}	MTPF	0.31	0.76	Y	Y	Y
	TPF	0.81	1.29	Y	Y	Y
	CCF	0.88	1.08	N	N	N
	PPF	0.98	4.29	Y	Y	Y

[1] the initial point
[2] the computational time
[3] the minimal cost
[4] the boundary conditions
[5] the smoothness conditions
[6] the inequality conditions
[7] the meeting of the tolerance 1.5×10^{-6}
[8] the violation of the tolerance 1.5×10^{-6}

conditions are satisfied by the MTPF while they cannot be met totally by the TPF, CCF and PPF although the performance of the TPF is better than those of the CCF and PPF. Furthermore, the MTPF consumes the least time, either 0.24 s or 0.31 s, in the derivation of the optimal trajectories and the PPF consumes the most, either 1.79 s or 0.98 s. Another observation is that the MTPF leads to the minimal cost by 0.76 because it takes sufficiently the advantages of the constraints.

5.3. Trajectory planning for a UVW robot. We take a UVW robot as an application of the ideal motor's trajectories. The UVW robot has three degrees of freedom, two translation variables and one rotational variable, which are driven by three motor driven systems. Figure 4 shows the UVW robot's initial and final settings due to two translation variables and one rotation variable. These variables are derived by the time-optimal trajectories of three motors through the MTPF shown in Fig. 5. It is clear that the designed trajectories have met the constraints in the time-optimal problem of three motors that include the constrains on $y^{(3)}(t)$ with -1.05×10^4 rad/s^3 and 1.05×10^4 rad/s^3, those on $y^{(2)}(t)$ with -3.14×10^3 rad/s^2 and 3.14×10^3 rad/s^2, those on $y^{(1)}(t)$ with -1.05×10^3 rad/s and 1.05×10^3 rad/s, and those on $y(t)$ with -418.88 rad and 418.88 rad. All trajectories of motors meet the boundary conditions $(94.02,\ 91.08,\ 112.02)^{\mathrm{T}}$ and $(-46.60,\ -2.94,\ 331.57)^{\mathrm{T}}$.

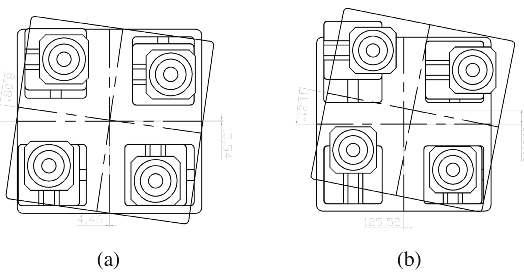


Fig. 4. UVW robot with two translation variables and one rotation variable: initial setting (a), final setting (b).

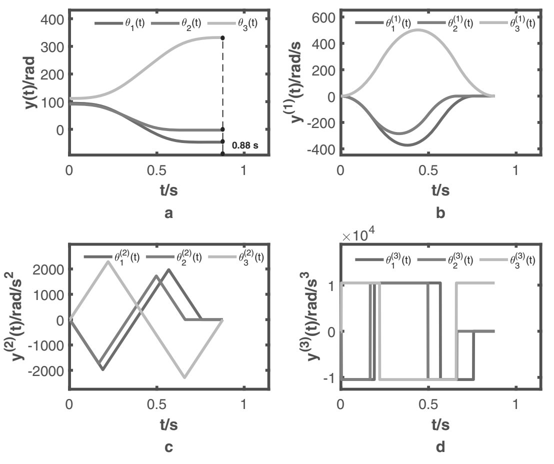


Fig. 5. Time optimal trajectories of three ideal motors of the UVW robot.

6. Conclusions

The proposed tool of the MTPF that takes advantage of truncated power functions could annihilate some original constraints and reduce parameters for a class of linear systems. It reduces the redundant parameters and eliminates the equality constraints in the TPF so that some sparse distribution of feasible solutions is avoided. By contrast, sparse distribution of feasible solutions is a common trouble in the existing methods. As a result, the performance of the MTPF is fast and efficient for the task of smooth trajectory planning. The experimental results show the advantages of the MTPF over the other methods. The tool of the MTPF is promising in trajectory planning for practical applications.

Acknowledgment

The work is funded by the National Natural Science Foundation of China (grants no. 62002315, 61174074 and 62073108) and the Key Research and Development Plan of Zhejiang Province (grant no. 2021C03034).

References

Aguilar-Ibanez, C. and Suarez-Castanon, M.S. (2019). A trajectory planning based controller to regulate an uncertain 3D overhead crane system, *International Journal of Applied Mathematics and Computer Science* **29**(4): 693–702, DOI: 10.2478/amcs-2019-0051.

Berselli, G., Balugani, F., Pellicciari, M. and Gadaleta, M. (2016). Energy-optimal motions for servo-systems: A comparison of spline interpolants and performance indexes using a CAD-based approach, *Robotics and Computer-Integrated Manufacturing* **40**: 55–65.

Boor, C.D. and Fix, G.J. (1973). Spline approximation by quasi-interpolants, *Journal of Approximation Theory* **8**(1): 19–45.

Cheon, H. and Kim, B.K. (2019). Online bidirectional trajectory planning for mobile robots in state-time space, *IEEE Transactions on Industrial Electronics* **66**(6): 4555–4565.

Cui, L., Wang, H. and Chen, W. (2020). Trajectory planning of a spatial flexible manipulator for vibration suppression, *Robotics and Autonomous Systems* **123**: 1–11, Paper no. 103316.

Desai, A., Collins, M. and Michael, N. (2019). Efficient kinodynamic multi-robot replanning in known workspaces, *2019 International Conference on Robotics and Automation (ICRA), Montreal, Canada*, pp. 1021–1027.

Fahroo, F. and Ross, I.M. (2000). Direct trajectory optimization by a Chebyshev pseudospectral method, *Proceedings of the 2000 American Control Conference, Chicago, USA*, Vol. 6, pp. 3860–3864.

Fang, Y., Qi, J., Hu, J., Wang, W. and Peng, Y. (2020). An approach for jerk-continuous trajectory generation of robotic manipulators with kinematical constraints, *Mechanism and Machine Theory* **153**: 1–22, Paper no. 103957.

Gong, Q., Kang, W. and Ross, I.M. (2006). A pseudospectral method for the optimal control of constrained feedback linearizable systems, *IEEE Transactions on Automatic Control* **51**(7): 1115–1129.

Heidari, H. and Saska, M. (2020). Trajectory planning of quadrotor systems for various objective functions, *Robotica* **39**(1): 137–152.

Kim, J. (2020). Trajectory generation of a two-wheeled mobile robot in an uncertain environment, *IEEE Transactions on Industrial Electronics* **67**(7): 5586–5594.

Kroger, T. and Wahl, F.M. (2010). Online trajectory generation: Basic concepts for instantaneous reactions to unforeseen events, *IEEE Transactions on Robotics* **26**(1): 94–111.

Li, J., Ran, M. and Xie, L. (2021). Efficient trajectory planning for multiple non-holonomic mobile robots via prioritized trajectory optimization, *IEEE Robotics and Automation Letters* **6**(2): 405–412.

Liu, C., Zhang, C. and Xiong, F. (2020). Multistage cooperative trajectory planning for multimissile formation via bi-level sequential convex programming, *IEEE Access* **8**: 22834–22853.

Liu, P., Han, Y., Wang, W., Liu, X. and Liu, J. (2018). Maneuvering trajectory planning during the whole phase based on piecewise Radau pseudospectral method, *Proceedings of the 37th Chinese Control Conference, Wuhan, China*, pp. 4627–4632.

Liu, Y., Zhang, Z., Wu, Z., Liu, F. and Li, X. (2021). Multiobjective preimpact trajectory-planning of space manipulator for self-assembling a heavy payload, *International Journal of Advanced Robotic Systems* **18**(1): 1–22, Paper no. 1729881421990285.

Mercy, T., Hostens, E. and Pipeleers, G. (2018). Online motion planning for autonomous vehicles in vast environments, *2018 IEEE 15th International Workshop on Advanced Motion Control (AMC), Tokyo, Japan*, pp. 114–119.

Muscio, G., Pierri, F., Trujillo, M.A., Cataldi, E., Antonelli, G., Caccavale, F., Viguria, A., Chiaverini, S., and Ollero, A. (2018). Coordinated control of aerial robotic manipulators: Theory and experiments, *IEEE Transactions on Control Systems Technology* **26**(4): 1406–1413.

Park, J. and Kim, H.J. (2021). Online trajectory planning for multiple quadrotors in dynamic environments using relative safe flight corridor, *IEEE Robotics and Automation Letters* **6**(2): 659–666.

Powell, M.J.D. (1981). *Approximation Theory and Methods*, Cambridge University Press, Cambridge.

Qian, Y., Yuan, J. and Wan, W. (2020). Improved trajectory planning method for space robot-system with collision prediction, *Journal of Intelligent and Robotic System* **99**(11): 289–302.

Rousseau, G., Maniu, C.S., Tebbani, S., Babel, M. and Martin, N. (2019). Minimum-time B-spline trajectories with corridor constraints: Application to cinematographic quadrotor flight plans, *Control Engineering Practice* **89**: 190–203.

Schoenberg, I.J. (1969). *Approximations with Special Emphasis on Spline Functions*, Academic Press, Madison.

Spedicato, S. and Notarstefano, G. (2018). Minimum-time trajectory generation for quadrotors in constrained environments, *IEEE Transactions on Control Systems Technology* **26**(4): 1335–1344.

Sun, K. and Liu, X. (2021). Path planning for an autonomous underwater vehicle in a cluttered underwater environment based on the heat method, *International Journal of Applied Mathematics and Computer Science* **31**(2): 289–301, DOI: 10.34768/amcs-2021-0020.

Tatematsu, N. and Ohnishi, K. (2003). Tracking motion of mobile robot for moving target using NURBS curve, *IEEE International Conference on Industrial Technology, Maribor, Slovenia*, Vol. 1, pp. 245–249.

Tho, H.D., Kaneshige, A. and Terashima, K. (2020). Minimum-time s-curve commands for vibration-free transportation of an overhead crane with actuator limits, *Control Engineering Practice* **98**: 1–12, Paper no. 104390.

Wang, M., Xiao, J., Zeng, F. and Wang, G. (2020). Research on optimized time-synchronous online trajectory generation method for a robot arm, *Robotics and Autonomous Systems* **126**: 1–12, Paper no. 103453, DOI: 10.1016/j.robot.2020.103453.

Wang, Y., Ueda, K. and Bortoff, S.A. (2013). A Hamiltonian approach to compute an energy efficient trajectory for a servomotor system, *Automatica* **49**(12): 3550–3561.

Yu, L., Wang, K., Zhang, Q. and Zhang, J. (2020). Trajectory planning of a redundant planar manipulator based on joint classification and particle swarm optimization algorithm, *Multibody System Dynamic* **50**(4): 25–43.

Zhang, S., Zanchettin, A.M., Villa, R. and and Dai, S. (2020). Real-time trajectory planning based on joint-decoupled optimization in human-robot interaction, *Mechanism and Machine Theory* **144**, Paper no. 103664, DOI: 10.1016/j.mechmachtheory.2019.103664.

Guangyu Liu received a BE degree in mechanical engineering from Jilin University, Changchun, China, in 1996, and a PhD degree in electrical engineering from the Department of Electrical Engineering, University of Melbourne, Australia, in 2006. He was an engineer at the Dong-Feng Motor (NISSAN) Corporation from 1997 to 2001, a research fellow at the Swinburne University of Technology, Melbourne, Australia, from 2001 to 2002, and at NICTA, Melbourne, from 2006 to 2008. He was appointed as an assistant professor at the University of Auckland, New Zealand, from 2008 to 2010. He is currently a distinguished professor at Hangzhou Dianzi University, China, and the founder of the Hangzhou Acme Automation Cooperation. He has published over 100 internationally refereed journal and conference papers. He holds over 110 patents. His current research interests include nonlinear optimization and control, machine learning, smart energy and industrial robots.

Shangliang Wu received his BE degree in electrical engineering from Jiaxing Nanhu University, China, in 2016, and his ME degree in control science from Hangzhou Dianzi University, China, in 2019. He is currently a PhD candidate in control systems at Hangzhou Dianzi University.

Jiajun Wang, PhD, graduated from Tianjin University in 2003, majoring in power electronics and power transmission. In 2005, he left the post-doctoral station of Zhejiang University in electrical engineering. He is now a professor and a doctoral supervisor of the College of Automation (College of Artificial Intelligence) of the Hangzhou University of Electronic Science and Technology.

Ling Zhu received her BE and ME degrees in computer science from Jiangxi Normal University, Nan Chang, China, in 2002 and 2005, respectively, and her PhD degree in information sciences from Tongji University, Shanghai, China, in 2018. She was a computer engineer at a company in Hong Kong, from 2003 to 2005, and a visiting scholar at Virginia Tech, Blacksburg, USA, in 2014. She is currently an associate professor at the Zhejiang University of Finance and Economics, Hangzhou, China. Her present research interests include game theory, machine learning, and optimization.

Qiang Lv received his bachelor's and PhD degrees from the East China University of Science and Technology in 2000 and 2007, respectively. He is a professor and deputy dean of the School of Automation Engineering, Hangzhou Dianzi University. He has published over 40 refereed journal papers and holds a dozen of patents. His research interests include multiple robot cooperative control, swarm intelligence and distributed optimization.

Appendix

A1. Proof of Lemma 1

The boundary conditions are equivalent to

$$\frac{\mathrm{d}^i}{\mathrm{d}t^i}(\sum_{i=0}^{n} K_i t^i - y_s)|_{t=t_1} = 0,$$

for $i = 0, 1, \dots, n-1$. It is easily checked that $\sum_{i=0}^{n} K_i t^i - y_s = \alpha_1 (t - t_1)^n$, where $\alpha_1 \in \mathbb{R}$. Therefore, (3) can be rewritten as (9) under the boundary conditions.

A2. Proof of Lemma 2

Note that $\operatorname{rank}(D(\Delta t, t_{N+1})) = \min\{n, N\}$ for $\Delta t > 0$, because

$$\begin{vmatrix} \lambda_1^k & \lambda_2^k, & \dots, & \lambda_k^k \\ \lambda_1^{k-1} & \lambda_2^{k-1} & \dots & \lambda_k^{k-1} \\ \vdots & \vdots & \dots & \vdots \\ \lambda_1 & \lambda_2 & \dots & \lambda_k \end{vmatrix} = \left(\prod_{i<j} (\lambda_i - \lambda_j) \right) \times \left(\prod_{i=1}^{k} \lambda_i \right). \quad \text{(A1)}$$

Firstly, the sufficiency is proved. When $N \geq n$, $\operatorname{rank}(D_{n,N}(\Delta t, t_{N+1})) = n$, which implies that there exists $\alpha \in \mathbb{R}^N$ such that $D(\Delta t, t_{N+1})\alpha = (y_f - y_s, 0, \dots, 0)^{\mathrm{T}}$.

In the second place, the necessity is proved. Suppose that there exists $\Delta t \in \mathbb{R}_+^N$, $\alpha \in \mathbb{R}^N$ with $N \leq n-1$ for $D(\Delta t, t_{N+1})\alpha = (y_f - y_s, 0, \dots, 0)^T$. This equation can be divided into $d_1 \alpha = y_f - y_s$ by

$$d_1 = \left(\left(\sum_{i=1}^{N} \delta t_i \right)^n, \left(\sum_{i=2}^{N} \delta t_i \right)^n, \dots, (\delta t_N)^n \right)$$

and $\underline{D}\alpha = 0$ by

$$\underline{D} = \begin{pmatrix} \left(\sum_{i=1}^{N} \delta t_i \right)^{n-1} & \left(\sum_{i=2}^{N} \delta t_i \right)^{n-1} & \dots & (\delta t_N)^{n-1} \\ \left(\sum_{i=1}^{N} \delta t_i \right)^{n-2} & \left(\sum_{i=2}^{N} \delta t_i \right)^{n-2} & \dots & (\delta t_N)^{n-2} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=1}^{N} \delta t_i & \sum_{i=2}^{N} \delta t_i & \dots & \delta t_N \end{pmatrix}.$$

It is to be noted that $\operatorname{rank}(\underline{D}) = N$ leads to $\alpha = 0$. Substituting $\alpha = 0$ into $d_1 \alpha = y_f - y_s$ gives $y_f = y_s$ which contradicts the condition $y_f \neq y_s$.

Received: 28 July 2021
Revised: 24 November 2021
Accepted: 6 December 2021