Applied Mathematical Modelling 96 (2021) 355–366

Contents lists available at ScienceDirect

Applied Mathematical Modelling

journal homepage: www.elsevier.com/locate/apm

Real-time robot manipulator tracking control as multilayered time-varying problem

Jian Li[a,c,*], Xinhui Zhu[a], Yang Shi[b], Jing Wang[a,c], Huaping Guo[a,c]

[a] School of Computer and Information Technology, Xinyang Normal University, Xinyang 464000, China
[b] School of Information Engineering, Yangzhou University, Yangzhou 225127, China
[c] Henan Key Laboratory of Analysis and Application of Education Big Data, Xinyang Normal University, Xinyang 464000, China

ARTICLE INFO

Article history:
Received 22 September 2020
Revised 4 March 2021
Accepted 9 March 2021
Available online 17 March 2021

Keywords:
Kinematic control
Joint angle limits
Multilayered time-varying problem
Zeroing neural dynamics
Discretization formula

ABSTRACT

When kinematic control of robot manipulator is modeled as time-varying problems, control algorithms generated by solving time-varying problems perform well in the aspect of real-time control and control precision. However, conventional algorithms are based on relatively simple time-varying problems modeling and solving, and thus, it is difficult for them to solve more other problems while completing tracking task. In this work, the kinematic control of robot manipulator is first modeled as more complicated multilayered time-varying problem, which simultaneously formulates the tracking task and joint angle limits. Then, zeroing neural dynamics method is future investigated to find the equivalence of multilayers. Finally, three explicit solutions based on three discretization formulas are presented to solve the multilayered time-varying problem. The presented solutions not only complete the tracking task in real time with different precision, but also solve the problem of joint angle limits during the control process.

1. Introduction

Real-time tracking control of robot manipulator is a fundamental topic in robotics, which has been investigated by numerous researchers [1,2]. The problem of joint angle limits must be considered during the control process. If not, the tracking task could not be completed and the hardware of robot manipulator could damage [3,4]. The kinematic tracking control problem is formulated as to find a joint angle trajectory $\theta(t)$ such that $\mathbf{f}(\theta(t)) = \mathbf{r}_\mathrm{d}(t)$ is satisfied for a given desired path in the task space $\mathbf{r}_\mathrm{d}(t)$ [5]. Classical methods include inverse Jacobi method [6], sliding mode method [7,8], Newton-based method [9] and optimal control [10] to solve this problem. For example, in [8], a suboptimal sliding mode control method was derived from combination of the sliding mode control and the state-dependent Riccati equation technique, applied for nonlinear robot systems. One of the distinguished features of this control method in [8] is its robustness towards uncertainty. However, these methods use the direct kinematic mappings for robot manipulators in a static/time-invariant perspective, i.e., $\mathbf{f}(\theta) = \mathbf{r}$ and $J(\theta)\dot{\theta} = \dot{\mathbf{r}}$, where $J(\theta)$ is the Jacobi matrix of robot manipulator. Most of them often implicitly assume that the start robot joint angle corresponds to the initial desired position of end-effector, which is not satisfied in practice generally [9]. Besides, the time-invariant modeling leads to lagging control and effects the real-time [11].

* Corresponding author at: School of Computer and Information Technology, Xinyang Normal University, Xinyang 464000, China.
E-mail address: lijcit@xynu.edu.cn (J. Li).

https://doi.org/10.1016/j.apm.2021.03.021
0307-904X/© 2021 Elsevier Inc. All rights reserved.

Table 1
Detailed differences between previous works and the present study.

Paper	A	B	C	D
[3,4]	Monolayer	No	Yes	Yes
[25,26]	Multilayers	No	No	Yes
[27–31]	Multilayers	Yes	Yes	No
This paper	Multilayers	Yes	Yes	Yes

A: Multilayers vs monolayer. B: Calculate joint angle predictively. C: Start with random joint angles. D: Consider all joint angle limits including upper and lower bounds.

Increasing researchers pay attention to the investigation of time-varying problems recent years and various kinds of time-varying problems have been investigated and solved [12,13]. Jin et al. [14] solved time-varying matrix inversion from a control-theoretical perspective. Zhang et al. [15] presented an integral recurrent neural network to solve time-varying Sylvester equation. Li et al. [16] presented a variable-gain finite-time convergent recurrent neural network to solve time-varying quadratic programming with unknown noises endured. Wei et al. [17] presented a noise-tolerant neural algorithm to solve time-varying nonlinear optimization with estimation on Hessian matrix inversion.

With the development of time-varying problems solving, researchers rethink the problem of robot manipulator tracking control and study it in a time-varying perspective [18]. The main idea of this method is modeling the tracking control problem as time-varying problems, and then solving the tracking control problem via solving the time-varying problems [19]. A biological-heuristic optimization model based on time-varying perspective was developed to control active sensing of robot arms in [20]. Robots' kinematic control was modeled as time-varying nonlinear equations and solved by utilizing a noise-tolerant zeroing dynamics design formula in [21]. Repetitive motion planning of robotic manipulators was solved in a time-varying perspective with guaranteed precision in [22]. The solution of time-varying generalized Sylvester equation was employed to robot manipulator control in [23]. A complex-valued discrete-time neural dynamics model for solving perturbed time-dependent complex quadratic programming was applied to robot manipulator control in [24].

The above mentioned studies of robot manipulator control based on time-varying problems solving have advantages on real-time computation and control precision. However, these studies formulate the complicated robot tracking control problems as relatively simple-layered time-varying problems, such as linear equations system, matrix inversion problem and quadratic programming. Recently, multilayered time-varying problems are investigated and applied to the robot manipulator control. For example, time-varying linear equation and inequality systems were investigated in [25], and perturbed multilayered linear systems were studied in [26]. They are employed to the robot manipulator control with all joint angle limits considered including upper and lower bounds. However, the presented solutions are continuous-time, which are implemented by ode solvers. Thus, the presented solutions do not calculate joint angle predictively, which leads to relatively weak real-time performance. Besides, the presented solutions assume the initial joint angle corresponds to initial desired path, which is hard to be satisfied in reality. Discrete-time solutions were presented to solve multilayered time-varying equalities and inequalities, and further control planar robot manipulator and mobile robot in [27,28]. Multilayered time-varying linear and nonlinear equations were investigated and applied to robot manipulator with partial joint angle limits conquered in [29]. The robot manipulator control was formulated as multilayered linear equations and nonlinear equations respectively in [30] and [31]. The performances of presented solutions and the proposed solution in this paper are listed and compared in Table 1.

In this work, the robot manipulator tracking control is firstly directly formulated as multilayered time-varying problem with all joint angle limits including upper and lower bounds considered. Then, the multilayered time-varying problem is equivalently converted into a relatively easily solved equation on the basis of zeroing neural dynamics method, and a continuous-time solution is developed. Finally, explicit solutions is proposed based on different discretization formulas and the continuous-time solution.

The remainder of this paper is organized into four sections. Section 2 formulates the robot manipulator control as multilayered time-varying problem. Section 3 solves this problem via zeroing neural dynamics method and discretization formulas. Section 4 presents some numerical experimental results to verify the effectiveness and superiority of proposed solution. Section 5 concludes this paper with final remarks. Before ending this section, the main contributions of this paper are listed in the following facts.

(1) Robot manipulator tracking control is directly formulated as multilayered time-varying problem with all joint angle limits considered including time-varying upper and lower bounds. Note that the constant bounds are special cases of time-varying bounds, which are investigated simultaneously.
(2) Three explicit four-step solutions are proposed to solve the multilayered time-varying problem based on the continuous-time solution and three discretization formulas. Note that the continuous-time solution is based on the study of zeroing neural dynamics method.

2. Multilayered time-varying modeling

From a time-varying perspective, the kinematic equation of robot manipulator [1] is formulated as

$$\mathbf{f}(\theta(t)) = \mathbf{r}_{\mathrm{a}}(t), \tag{1}$$

where $\theta(t) \in \mathbb{R}^n$ is the joint angle vector; $\mathbf{r}_{\mathrm{a}}(t) \in \mathbb{R}^m$ is the actual trajectory of end-effector; and $\mathbf{f}(\cdot)$ is the map between the joint angle and the actual trajectory of end-effector. The inverse kinematic control of robot manipulator is to find a feasible solution $\theta(t)$ so that the actual trajectory of end-effector tracks the time-varying desired path quickly, i.e., $\mathbf{r}_{\mathrm{a}}(t) \to \mathbf{r}_{\mathrm{d}}(t)$, where $\mathbf{r}_{\mathrm{d}}(t)$ is the time-varying desired path. It is formulated as the following time-varying equation

$$\mathbf{r}_{\mathrm{a}}(t) - \mathbf{r}_{\mathrm{d}}(t) = 0.$$

Furthermore, based on the kinematic Eq. (1), we have the first layer equation to be solved as below:

$$\mathbf{f}(\theta(t)) - \mathbf{r}_{\mathrm{d}}(t) = 0. \tag{2}$$

Joint angle limits of robot manipulator are the range of joint angle during the control process, which is dynamically formulated as

$$\theta^-(t) \le \theta(t) \le \theta^+(t), \tag{3}$$

where $\theta^-(t)$ and $\theta^+(t)$ are time-varying upper and lower bounds of the controlled variable $\theta(t)$. (3) can be reformulated as the second and third time-varying layers as below:

$$\begin{cases} \theta(t) - \theta^+(t) \le 0, \\ -\theta(t) + \theta^-(t) \le 0, \end{cases} \tag{4}$$

where $\theta(t) - \theta^+(t) \le 0$ means each elements of vector $\theta(t) - \theta^+(t)$ are less than or equal to 0. Combining the first, second and third layers yields the following multilayered time-varying equation

$$\begin{cases} \mathbf{f}(\theta(t)) - \mathbf{r}_{\mathrm{d}}(t) = 0, \\ \theta(t) - \theta^+(t) \le 0, \\ -\theta(t) + \theta^-(t) \le 0, \end{cases} \tag{5}$$

which simultaneously formulates the time-varying tracking task and joint angle limits constraints.

The superiority of the model (5) to formula the robot manipulator control is shown compared with the presented models in [1,30] (i.e., model (9) in [1] and model (1)-(2) in [30]). Firstly, [1] formulas the robot manipulator control as single layered time-varying nonlinear equation as follows:

$$\mathbf{r}_{\mathrm{a}}(t) - \mathbf{r}_{\mathrm{d}}(t) = 0,$$

i.e.,

$$\mathbf{f}(\theta(t)) - \mathbf{r}_{\mathrm{d}}(t) = 0.$$

That equation only considers the tracking task, and thus the corresponding solution could not complete other task. Furthermore, [30] models the robot manipulator control as multilayered time-varying linear system, which is formulated as follows:

$$\begin{cases} A(t)\theta(t) = \mathbf{b}(t), \\ J(\theta(t))\dot{\theta}(t) = \dot{\mathbf{r}}_{\mathrm{d}}(t) - \lambda(\mathbf{f}(\theta(t)) - \mathbf{r}_{\mathrm{d}}(t)). \end{cases}$$

The second equality formulas the tacking task and the first equality could be constructed as needed to complete other task. However, that work only investigates equalities instead of inequalities, which is difficult to formulate joint angle limits. Specifically, please see Example 3 of [30], by constructing the matrix $A(t)$ as $[\sin(0.5t)+2, 0, 0]$, and the vector $\mathbf{b}(t)$ as $\pi/2$, only the first joint angle conquers the joint limits instead of all joint angle limits (When all joint angle limits are considered by equalities, the basic tracking task could not be formulated because of overdetermination), which is less practical in reality. In contrast, this work models the robot manipulator control as multilayered equality and inequality equations, which formulate the tracking task by equality and formulates all joint angle limits by inequalities. Thus, the corresponding solution proposed in this work to solve (5) could simultaneously solve the problem of tracking time-varying desired path and the problem of all joint angle limits.

3. Zeroing neural dynamics solving and time discretization

In this section, multilayered time-varying Eq. (5) is solved based on zeroing neural dynamics and a discretization formula.

3.1. Equivalency analyses and continuous-time solution

The three layers of multilayered time-varying Eq. (5) have to be equivalently converted into other forms so that they can be unified and solved together. Equivalency analyses based on zeroing neural dynamics [32] are presented by the following lemmas, corollary and theorems.

Lemma 1. *If the parameter $\lambda > 0$ and $t \to +\infty$, then the first layer of (5), i.e., $\mathbf{f}(\theta(t)) - \mathbf{r}_\mathrm{d}(t) = 0$, is equivalent to*

$$J(\theta(t))\dot{\theta}(t) = \dot{\mathbf{r}}_\mathrm{d}(t) - \lambda(\mathbf{f}(\theta(t)) - \mathbf{r}_\mathrm{d}(t)), \tag{6}$$

where $J(\theta(t))$ is the Jacobi matrix of the robot manipulator and $\dot{\mathbf{r}}_\mathrm{d}(t)$ is the time derivative of desired path.

Proof. The proof process is given in Appendix A. □

Lemma 2. *Define a vector $\gamma(t) \in \mathbb{R}^n$ as $[\gamma_1(t), \gamma_2(t), ..., \gamma_n(t)]^\mathrm{T}$, and a vector $\gamma^{\cdot 2}(t) \in \mathbb{R}^n$ as $[\gamma_1^2(t), \gamma_2^2(t), ..., \gamma_n^2(t)]^\mathrm{T}$. If the parameter $\lambda > 0$ and $t \to +\infty$, then the second layer of (5), i.e., $\theta(t) - \theta^+(t) \le 0$, is equivalent to*

$$\begin{bmatrix} I & 2\Lambda(t) \end{bmatrix} \begin{bmatrix} \dot{\theta}(t) \\ \dot{\gamma}(t) \end{bmatrix} = \dot{\theta}^+(t) - \lambda\big(\theta(t) - \theta^+(t) + \gamma^{\cdot 2}(t)\big) \tag{7}$$

where $I \in \mathbb{R}^{n\times n}$ is the identity matrix; and $\Lambda(t)$ is denoted as below:

$$\Lambda(t) = \begin{bmatrix} \gamma_1(t) & 0 & \cdots & 0 \\ 0 & \gamma_2(t) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \gamma_n(t) \end{bmatrix}.$$

Proof. The proof process is given in Appendix B. □

Corollary 1. *Define a vector $\eta(t) \in \mathbb{R}^n$ as $[\eta_1(t), \eta_2(t), ..., \eta_n(t)]^\mathrm{T}$, and a vector $\eta^{\cdot 2}(t) \in \mathbb{R}^n$ as $[\eta_1^2(t), \eta_2^2(t), ..., \eta_n^2(t)]^\mathrm{T}$. If the parameter $\lambda > 0$ and $t \to +\infty$, then the third layer of (5), i.e., $-\theta(t) + \theta^-(t) \le 0$, is equivalent to*

$$\begin{bmatrix} -I & 2\Gamma(t) \end{bmatrix} \begin{bmatrix} \dot{\theta}(t) \\ \dot{\eta}(t) \end{bmatrix} = -\dot{\theta}^-(t) - \lambda\big(\eta^{\cdot 2}(t) - \theta(t) + \theta^-(t)\big) \tag{8}$$

where $\Gamma(t)$ is denoted as below:

$$\Gamma(t) = \begin{bmatrix} \eta_1(t) & 0 & \cdots & 0 \\ 0 & \eta_2(t) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \eta_n(t) \end{bmatrix}.$$

Based on the above lemmas and corollary, we have the following theorem to convert the multilayered time-varying Eq. (5) into a more solvable equation.

Theorem 1. *If the parameter $\lambda > 0$ and $t \to +\infty$, then multilayered time-varying Eq. (5) is equivalent to*

$$W(\mathbf{y}(t))\dot{\mathbf{y}}(t) = \mathbf{d}(t), \tag{9}$$

where $\mathbf{y}(t) = [\theta^\mathrm{T}(t), \gamma^\mathrm{T}(t), \eta^\mathrm{T}(t)]^\mathrm{T}$,

$$W(\mathbf{y}(t)) = \begin{bmatrix} J(\theta(t)) & 0 & 0 \\ I & 2\Lambda(t) & 0 \\ -I & 0 & 2\Gamma(t) \end{bmatrix} \tag{10}$$

and

$$\mathbf{d}(t) = \begin{bmatrix} \dot{\mathbf{r}}_\mathrm{d}(t) - \lambda(\mathbf{f}(\theta(t)) - \mathbf{r}_\mathrm{d}(t)) \\ \dot{\theta}^+(t) - \lambda\big(\theta(t) - \theta^+(t) + \gamma^{\cdot 2}(t)\big) \\ -\dot{\theta}^-(t) - \lambda\big(\eta^{\cdot 2}(t) - \theta(t) + \theta^-(t)\big) \end{bmatrix}.$$

Proof. Based on Lemma 1, Lemma 2 and Corollary 1, if the parameter $\lambda > 0$ and $t \to +\infty$, then the first, second and third layer of multilayered time-varying Eq. (5) is equivalently converted into Eqs. (6), (7) and (8), respectively. Combining (6), (7) and (8) yields (9), and thus, if the parameter $\lambda > 0$ and $t \to +\infty$, then multilayered time-varying Eq. (5) is equivalent to (9). □

Based on the equivalency between (5) and (9), the continuous-time solution is obtained to solve multilayered time-varying Eq. (5) as below:

$$\dot{\mathbf{y}}(t) = W^{+}(\mathbf{y}(t))\mathbf{d}(t), \tag{11}$$

where $^{+}$ denotes pseudo-inverse operation. It is an equation of explicit neural dynamics. We assume that matrix $W(\mathbf{y}(t))$ is always non-singular.

Remark 1. continuous-time solution (11) is based on the three equivalent equations in Lemma 1, Lemma 2 and Corollary 1. In terms of tracking task, the first layer of (5), i.e., $\mathbf{f}(\theta(t)) - \mathbf{r}_\mathrm{d}(t) = 0$, is equivalent to (6). This equivalence is based on linear design formula of zeroing neural dynamics i.e., $\dot{\mathbf{e}}(t) = -\lambda\mathbf{e}(t)$ [1]. This design formula makes the equivalent Eq. (6) globally exponentially convergent. Thus, continuous-time solution (11) makes the actual trajectory $\mathbf{r}_\mathrm{a}(t)$ converge to desired path $\mathbf{r}_\mathrm{d}(t)$ globally and exponentially.

3.2. Solutions and theoretical analyses

Based on previous work [1], the continuous-time solution should be discretized by effective one-step-ahead discretization formulas, so that the corresponding solutions could calculate the control variable in predictive way, and then satisfy the requirement of real-time computation.

In this work, we utilize three one-step-ahead discretization formulas. Euler formula [33] is the first effective discretization formula, which is shown as

$$\dot{x}_k(t) = \frac{x(t_{k+1}) - x(t_k)}{\delta} + O(\delta),$$

where δ is the time sampling gap. Utilizing Euler formula to discretize continuous-time solution (11) generates an explicit one-step solution as below:

$$\mathbf{y}(t_{k+1}) = \mathbf{y}(t_k) + \delta\mathbf{g}(\mathbf{y}(t_k), t_k), \tag{12}$$

where

$$\mathbf{g}(\mathbf{y}(t_k), t_k) = W^{+}(\mathbf{y}(t_k))\mathbf{d}(t_k). \tag{13}$$

Formula in [34] is shown as

$$\dot{x}(t_k) = \frac{2x(t_{k+1}) - 3x(t_k) + 2x(t_{k-1}) - x(t_{k-2})}{2\delta} + O(\delta^2). \tag{14}$$

Utilizing formula (14) to discretize continuous-time solution (11) generates the following explicit three-step solution:

$$\mathbf{y}(t_{k+1}) = \frac{3}{2}\mathbf{y}(t_k) - \mathbf{y}(t_{k-1}) + \frac{1}{2}\mathbf{y}(t_{k-2}) + \delta\mathbf{g}(\mathbf{y}(t_k), t_k). \tag{15}$$

Another formula, which is obtained by Taylor expansion, is presented as

$$\dot{x}(t_k) = \frac{50}{111\delta}x(t_{k+1}) + \frac{7}{222\delta}x(t_k) - \frac{11}{37\delta}x(t_{k-1}) - \frac{67}{222\delta}x(t_{k-2}) + \frac{13}{111\delta}x(t_{k-3}) + O(\delta^3). \tag{16}$$

Utilizing formula (16) to discretize continuous-time solution (11) generates an explicit four-step solution as below:

$$\mathbf{y}(t_{k+1}) = -\frac{7}{100}\mathbf{y}(t_k) + \frac{33}{50}\mathbf{y}(t_{k-1}) + \frac{67}{100}\mathbf{y}(t_{k-2}) - \frac{13}{50}\mathbf{y}(t_{k-3}) + \delta\frac{111}{50}\mathbf{g}(\mathbf{y}(t_k), t_k). \tag{17}$$

Solution (17) is relatively universal, which can be implemented in any robot manipulator system when kinematic parameters are known. Based on the above process, multilayered time-varying problem including equalities and inequalities could be solved by using solution (17). Real-time implementation of explicit four-step solution (17) is shown in Algorithm. Besides, solution (17) has advantages in the aspect of real-time computation, convergence and joint angle limits conquering. The corresponding discussions are presented as below:

(1) Solution (17) satisfies the requirement of real-time computation via predicting future-instant solution. Specifically, solution (17) uses information at t_k e.g., $W(\mathbf{y}(t_k))$, $\tilde{\mathbf{d}}(t_k)$, and information before t_k, e.g., $\mathbf{y}(t_{k-1})$, $\mathbf{y}(t_{k-2})$, to predictively calculate future control variable $\theta(t_{k+1})$, which is exactly the first n elements of $\mathbf{y}(t_{k+1})$. In this way, the inevitable time consuming of solution computation does not lead to the lag of tracking control.
(2) For solution (17), initial position of end-effector could be away from desired position, and then it quickly tacks the desired path.
(3) Solution (17) solves the problem of joint angle limits, not just completes the basic tracking task. Solution (17) is to solve multilayered time-varying problem (5), which models both the tracking problem and the joint angle limits via three layers, and thus, solution (17) could simultaneously solve the problem of tracking time-varying desired path and the problem of joint angle limits.

Algorithm Real-time implementation of explicit four-step solution (17)

```
1. Initialize: Arbitrarily set $\theta(t_0)$, $\gamma(t_0)$, $\eta(t_0)$. Set sampling gap $\tau$, parameter $\kappa$ and task duration $t_f$. Updating index $k=0$.
2. While($t_k \le t_3$)
3.          Get current data at time $t_k$ including $\theta^+(t_k)$, $\theta^-(t_k)$, $\dot{\theta}^+(t_k)$, $\dot{\theta}^-(t_k)$, $\mathbf{r}_d(t_k)$ and $\dot{\mathbf{r}}_d(t_k)$.
4.          Calculate matrix $W(\mathbf{y}(t_k)$ by (10); Calculate vector $\tilde{\mathbf{d}}(t_k)$ by (13).
5.          Predict future value $\mathbf{y}(t_{k+1})$, which contains $\theta(t_{k+1})$, $\gamma(t_{k+1})$ and $\eta(t_{k+1})$, by solution (12).
6.          Convert the joint angle value $\theta(t_{k+1})$ to control signal and control the robot manipulator.
7.          When time $t_{k+1}$ comes, index $k=k+1$.
8. While($t_k \le t_f$)
9.          Get current data at time $t_k$ including $\theta^+(t_k)$, $\theta^-(t_k)$, $\dot{\theta}^+(t_k)$, $\dot{\theta}^-(t_k)$, $\mathbf{r}_d(t_k)$ and $\dot{\mathbf{r}}_d(t_k)$.
10.          If derivative data ($\dot{\theta}^+(t_k)$, $\dot{\theta}^-(t_k)$ and $\dot{\mathbf{r}}_d(t_k)$) are unknown
11.                  Calculate derivative data approximately by (18).
12.          Calculate matrix $W(\mathbf{y}(t_k)$ by (10); Calculate vector $\tilde{\mathbf{d}}(t_k)$ by (13).
13.          Predict future $\mathbf{y}(t_{k+1})$ by solution (17).
14.          Convert the joint angle value $\theta(t_{k+1})$ to control signal and control the robot manipulator.
15.          When time $t_{k+1}$ comes, index $k=k+1$.
16.~End
```

Remark 2. To complement explicit four-step solution (17), the values of $\dot{\mathbf{r}}_d(t_k)$, $\dot{\theta}^+(t_k)$ and $\dot{\theta}^-(t_k)$ is necessary, which are unknown usually. Backward finite difference formula is employed to approximate these first-order derivatives because future information $\theta(t_{k+1})$ is unknown. To make sure the precision and simplicity, the truncation error of backward finite difference formula should be equal to that of our discretization formula (16). Thus, these first-order derivatives in solution (17) can be approximated by the following backward formula [33]:

$$\dot{u}(t_k) = \frac{11}{6\delta}u(t_k) - \frac{3}{\delta}u(t_{k-1}) + \frac{3}{2\delta}u(t_{k-2}) - \frac{1}{3\delta}u(t_{k-3}) + O(\delta^3). \tag{18}$$

Theoretical analyses on 0-stability, consistency and convergence, which are based on four results in Appendix of [1] for an N-step method [35], are presented.

Theorem 2. *With $\mathbf{O}(\delta^4)$ denoting a vector and every element being $O(\delta^4)$, explicit four-step solution (17) is 0-stable, consistent and convergent, which converges with the truncation error of order $\mathbf{O}(\delta^4)$. Besides, explicit one-step solution (12) and explicit three-step solution (15) are also 0-stable, consistent and convergent, which converge with the truncation errors of order $\mathbf{O}(\delta^2)$ and $\mathbf{O}(\delta^3)$, respectively.*

Proof. The first characteristic polynomial of explicit four-step solution (17) is

$$P_4(\varsigma) = \varsigma^4 + \frac{17}{100}\varsigma^3 - \frac{33}{50}\varsigma^2 - \frac{67}{100}\varsigma + \frac{13}{50},$$

of which the roots are

$$\begin{cases} \varsigma_1 = 1, \\ \varsigma_2 = 0.3102, \\ \varsigma_3 = -0.6901 + 0.6016\mathrm{i}, \\ \varsigma_4 = -0.6901 - 0.6016\mathrm{i}. \end{cases}$$

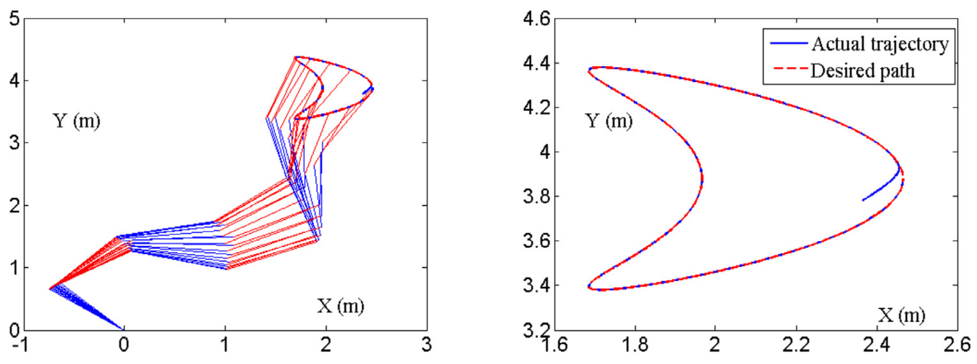
The four roots satisfy the 0-stability condition, and thus, solution (17) is 0-stable. The second characteristic polynomial of explicit four-step solution (17) **is** $\sigma(\varsigma) = 111/50\varsigma^3$. Let us observe that $P_4(1) = 0$ and $P_4'(1) = \sigma(1) \ne 0$. Thus, solution (17) is consistent based on the result on page 338 of [33]. Furthermore, discretization formula (16) can be rewritten as

$$x(t_{k+1}) = \frac{111\delta}{50}\dot{x}(t_k) - \frac{7}{100}x(t_k) + \frac{33}{50}x(t_{k-1}) + \frac{67}{100}x(t_{k-2}) - \frac{13}{50}x(t_{k-3}) + O(\delta^4). \tag{19}$$

Thus, solution (17) generated by formula (16) has an truncation error of $\mathbf{O}(\delta^4)$. Finally, according to Dahlquist equivalence theorem on page 340 of [33], solution (17) is convergent, which converges with the truncation error of order $\mathbf{O}(\delta^4)$. In terms of one-step solution (12) and three-step solution (15), the 0-stability and consistency are guaranteed by similar analyses with above. Their truncation errors are $\mathbf{O}(\delta^2)$ and $\mathbf{O}(\delta^3)$, respectively. Thus, solution (12) and solution (15) are also 0-stable, consistent and convergent, which converge with the truncation errors of order $\mathbf{O}(\delta^2)$ and $\mathbf{O}(\delta^3)$, respectively. □

4. Numerical experiments and verification

In this section, some comparative numerical experimental results are presented. For comparisons, conventional solution in formula (17) of [1] and solution in formula (22) of [30] are shown, which are based on the models shown in Section 2.



(a) Robot trajectories (b) End-effector trajectories

Fig. 1. Robot trajectories and end-effector trajectories of planar robot system synthesized by solution (17). The end-effector from a relatively arbitrary initial position quickly tracks the desired path.

Table 2
Maximal steady-state tracking error (MSSTE) defined as $\lim_{k\to+\infty}\sup\|\mathbf{r}_\mathrm{a}(t_{k+1})-\mathbf{r}_\mathrm{d}(t_{k+1})\|$ and average computing time per updating (ACTPU) data when using solution (12), solution (15), solution (17) and solution in [1] with different δ values for planar robot manipulator system with constant joint angle limits as shown in Section 4.1.

δ (ms)	Solution (12)		Solution (15)		Solution in [1]		Solution (17)	
	MSSTE (m)	ACTPU (ms)	MSSTE (m)	ACTPU (ms)	MSSTE (m)	ACTPU (ms)	MSSTE (m)	ACTPU (ms)
100	5.05×10^{-3}	0.3594	3.69×10^{-4}	0.3681	5.93×10^{-6}	0.1394	6.45×10^{-6}	0.5184
50	1.36×10^{-3}	0.3473	5.04×10^{-5}	0.3482	4.42×10^{-7}	0.1376	4.40×10^{-7}	0.5410
10	5.66×10^{-5}	0.3335	4.17×10^{-7}	0.3609	8.86×10^{-10}	0.1310	9.16×10^{-10}	0.4082
5	1.42×10^{-5}	0.3364	5.22×10^{-7}	0.3369	5.60×10^{-11}	0.1332	5.82×10^{-11}	0.3841
1	5.69×10^{-7}	0.3255	4.17×10^{-10}	0.3243	9.49×10^{-14}	0.1333	9.74×10^{-14}	0.3886

Specifically, the conventional solution in formula (17) of [1] is

$$\begin{aligned}\theta(t_{k+1}) = {} & \frac{9}{4}J^{+}(\theta(t_k))(\delta\dot{\mathbf{r}}_\mathrm{d}(t_k)-\kappa(\mathbf{f}(\theta(t_k))-\mathbf{r}_\mathrm{d}(t_k)))\\ & -\frac{1}{8}\theta(t_k)+\frac{3}{4}\theta(t_{k-1})+\frac{5}{8}\theta(t_{k-2})-\frac{1}{4}\theta(t_{k-3}).\end{aligned}\tag{20}$$

The solution in formula (22) of [30] is

$$\theta(t_{k+1}) = \frac{16}{7}F^{+}(t_k)\mathbf{p}(t_k)-\frac{4}{21}\theta(t_k)+\frac{6}{7}\theta(t_{k-1})+\frac{4}{7}\theta(t_{k-2})-\frac{5}{21}\theta(t_{k-3}),\tag{21}$$

where

$$F(t_k)=\begin{bmatrix}A(t_k)\\ J(t_k)\end{bmatrix}$$

and

$$\mathbf{p}(t_k)=\begin{bmatrix}-\delta(\dot{A}(t_k)\theta(t_k)-\dot{\mathbf{b}}(t_k))-\kappa(A(t_k)\theta(t_k)-\mathbf{b}(t_k))\\ \delta\dot{\mathbf{r}}_\mathrm{d}(t_k)+\kappa(\mathbf{r}_\mathrm{d}(t_k)-\mathbf{f}(\theta(t_k)))\end{bmatrix}.$$

4.1. Planar robot manipulator with constant joint angle limits

Some numerical experiments are conducted in a system of six-link planar robot manipulator. Its kinematic mapping and Jacobi matrix could be found and derived by D-H method [36]. Solution (17) is employed to control the tracking process. Each link of the planar robot manipulator is 1 m; initial joint angle is $\theta(t_0)=[3\pi/4,-\pi/2,-\pi/4,\pi/6,\pi/3,-\pi/6]$ rad; the joint angle limits are $[\theta(t_0)-\pi/15,\theta(t_0)+\pi/9]$ rad; task duration is 20 s; sampling gap $\delta=0.01$ s. The task is to track a desired path shown in Fig. 1, which has bounded fourth-order derivative. This setting guarantees the effectiveness of discretization formula and corresponding proposed solution. In the on-line control process, the inputs are current data at time t_k including $\theta^{+}(t_k)$, $\theta^{-}(t_k)$, $\dot{\theta}^{+}(t_k)$, $\dot{\theta}^{-}(t_k)$, $\mathbf{r}_\mathrm{d}(t_k)$ and $\dot{\mathbf{r}}_\mathrm{d}(t_k)$, in which derivative data ($\dot{\theta}^{+}(t_k)$, $\dot{\theta}^{-}(t_k)$ and $\dot{\mathbf{r}}_\mathrm{d}(t_k)$) are optional. The output is future value $\theta(t_{k+1})$, which could be converted as control signal to control the robot manipulator. Numerical results are shown in Figs. 1 2 and Table 2.

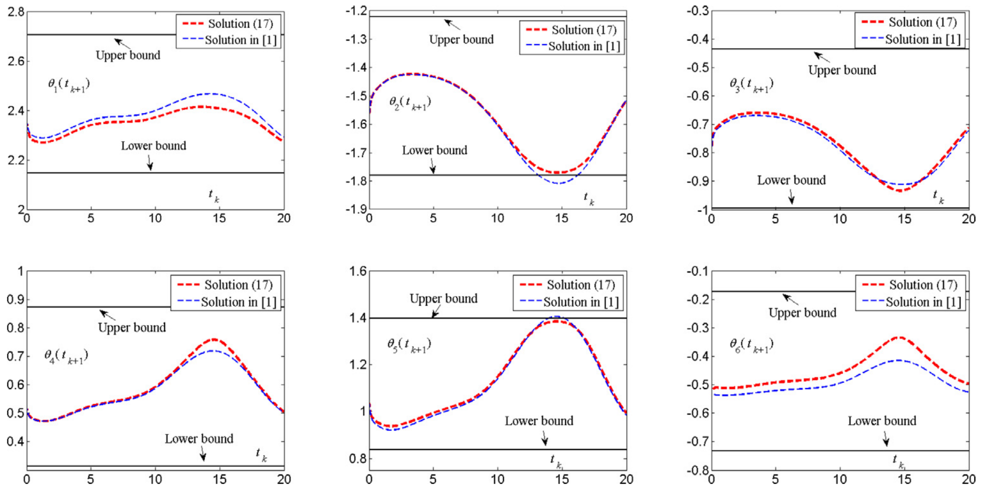


Fig. 2. Joint angle trajectories of planar robot system synthesized by solution in [1] and solution (17) with constant joint angle limits. Trajectories of every joint angles synthesized by solution (17) always stay in the joint angle limits. In contrast, joint angles $\theta_2(t_{k+1})$ and $\theta_5(t_{k+1})$ synthesized by solution in [1] are sometimes beyond the joint angle upper bound or lower bound.

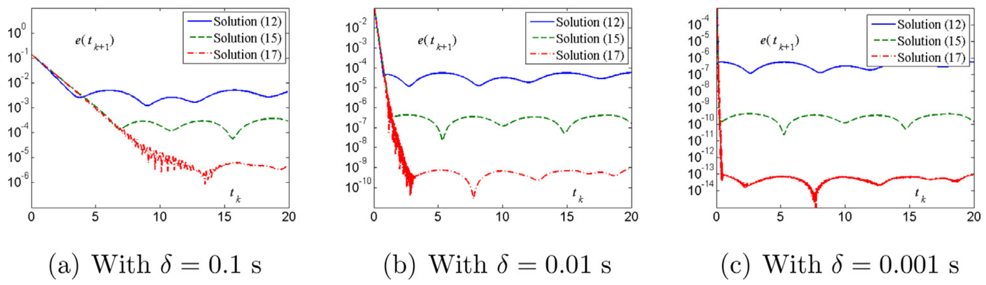


Fig. 3. Tracking errors of planar robot with constant joint angle limits defined as $e(t_{k+1}) = \|\mathbf{r}_a(t_{k+1}) - \mathbf{r}_d(t_{k+1})\|$ synthesized by solution (12), solution (15) and solution (17) with $\delta = 0.1$ s, $\delta = 0.01$ s and $\delta = 0.001$ s. When sampling gap δ reduces by a tenth, tracking errors of solutions (12), (15) and (17) reduce by 10^{-2}, 10^{-3} and 10^{-4}, respectively.

Fig. 1 (a) shows the trajectories of robot manipulator and Fig. 1(b) shows the actual trajectories of end-effector as well as the time-varying desired path. It can be observed that initial position of end-effector is away from desired position, and then it quickly tacks the desired path, which means that solution (17) successfully completes the tracking task.

Furthermore, Fig. 2 shows the trajectories of every joint angles as well as the constant upper and lower bounds. For comparisons, conventional solution in [1] is also employed. It is observed that trajectories of every joint angles synthesized by solution (17) always stay in the joint angle limits. However, as shown in Fig. 2(b) and (e), joint angles $\theta_2(t_{k+1})$ and $\theta_5(t_{k+1})$ synthesized by solution in [1] are sometimes beyond the joint angle upper bound or lower bound. Therefore, it is verified that solution (17) successfully solves the problem of joint angle limits, while the solution in [1] fails to do it, which shows the superiority of solution (17).

Some numerical experiments are implemented to verify the control precision of solution (17). For comparisons, solution (12), solution (15) and solution (17) are employed simultaneously. Tracking error is defined as $e(t_{k+1}) = \|\mathbf{r}_a(t_{k+1}) - \mathbf{r}_d(t_{k+1})\|$. Numerical results are shown in Fig. 3. It is observed that tracking errors synthesized by solution (17) are of order 10^{-6}, 10^{-10} and 10^{-14} with sampling gap $\delta = 0.1$ s, 0.01 s and 0.001 s, respectively, which substantiates that the control precision of solution (17) is δ^4. However, tracking errors synthesized by solution (12) are of order 10^{-3}, 10^{-5} and 10^{-7} with sampling gap $\delta = 0.1$ s, 0.01 s and 0.001 s, respectively. Tracking errors synthesized by solution (15) are of order 10^{-4}, 10^{-7} and 10^{-10} with sampling gap $\delta = 0.1$ s, 0.01 s and 0.001 s, respectively. The superiority of solution (17) in terms of control precision compared to solutions (12) and (15) is thus substantiated.

More numerical experimental results are shown in Table 2, which presents maximal steady-state tracking error (MSSTE) defined as $\lim_{k\to+\infty} \sup \|\mathbf{r}_a(t_{k+1}) - \mathbf{r}_d(t_{k+1})\|$ and average computing time per updating (ACTPU) data when using solution (12), solution (15), solution (17) and solution in [1] for planar robot manipulator system with constant joint angle limits. Different situations with sampling gap $\delta = 0.1$, 0.05, 0.01, 0.005 and 0.001 s are considered. It is observed that their ACTPU data in different situations are smaller than 0.6 ms, which means that the control process is real time even when sampling gap δ is up to 0.6 ms. ACTPU data of solutions (12), (15) and (17) are smaller than those of solution in [1] because of relatively complicated structure. Besides, their MSSTE data are consistent with the results in Fig. 3.

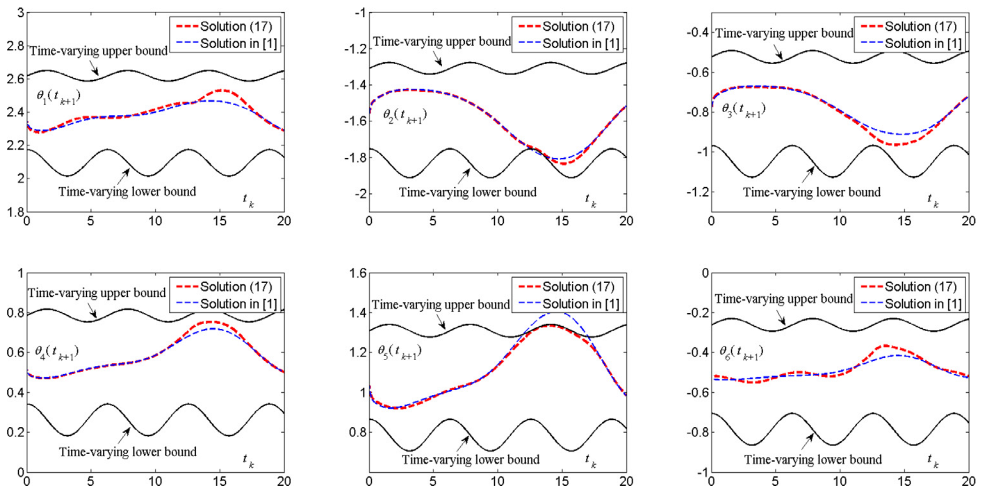


Fig. 4. Joint angle of planar robot trajectories synthesized by solution in [1] and solution (17) with time-varying joint angle limits. Trajectories of every joint angles synthesized by solution (17) always stay in the joint angle limits. In contrast, some joint angles synthesized by solution in [1] are sometimes beyond the joint angle upper bound or lower bound.

Table 3
MSSTE and ACTPU data when using solution (12), solution (15), solution (17) and solution in [1] with different δ values for planar robot manipulator system with time-varying joint angle limits as shown in Section 4.2.

δ (ms)	Solution (12)		Solution (15)		Solution in [1]		Solution (17)	
	MSSTE (m)	ACTPU (ms)	MSSTE (m)	ACTPU (ms)	MSSTE (m)	ACTPU (ms)	MSSTE (m)	ACTPU (ms)
100	5.04×10^{-3}	0.3953	3.51×10^{-4}	0.3894	5.93×10^{-6}	0.1585	6.89×10^{-6}	0.5690
50	1.36×10^{-3}	0.3756	4.94×10^{-5}	0.4176	4.42×10^{-7}	0.1401	4.71×10^{-7}	0.5337
10	5.61×10^{-5}	0.3576	4.78×10^{-7}	0.3609	8.86×10^{-10}	0.1287	8.26×10^{-10}	0.4843
5	1.41×10^{-5}	0.3682	5.23×10^{-7}	0.3626	1.56×10^{-11}	0.1567	5.18×10^{-11}	0.4411
1	5.63×10^{-7}	0.3680	4.19×10^{-10}	0.3809	9.49×10^{-14}	0.1434	8.97×10^{-14}	0.4265

4.2. Planar robot manipulator with time-varying joint angle limits

Some numerical experiments are also conducted for six-link planar robot manipulator system with time-varying joint angle limits. The time-varying lower bound is $\theta(t_0) - \pi/12 + 0.25\cos(t)/\pi$ and the time-varying upper bound is $\theta(t_0) + \pi/12 + 0.1\sin(t)/\pi$. The other settings are the same as those in Section 4.1. Numerical results are shown in Fig. 4 and Table 3. It is observed from Fig. 4 that trajectories of every joint angles synthesized by solution (17) always stay in the joint angle limits. However, joint angles $\theta_2(t_{k+1})$ and $\theta_5(t_{k+1})$ synthesized by solution in [1] are sometimes beyond the time-varying joint angle upper bound or lower bound. Table 3 presents MSSTE and ACTPU data when using solutions (12), (15), (17) and solution in [1] with sampling gap $\delta = 0.1$, 0.05, 0.01, 0.005 and 0.001 s. Please refer to Section 4.1 for detailed explanations.

4.3. PUMA560 with joint angle limits

In this subsection, a more realistic robotic system, i.e., PUMA560 robot manipulator, is investigated. It is a serial robot manipulator with 6 degree of freedom. The parameters of PUMA560 could be found in [37], in which joint angle limits (The units are angles) are listed as

$$\begin{cases} -160 \le \theta_1(t) \le 160, \\ -225 \le \theta_2(t) \le 45, \\ -45 \le \theta_3(t) \le 225, \\ -110 \le \theta_4(t) \le 170, \\ -100 \le \theta_5(t) \le 100, \\ -266 \le \theta_6(t) \le 266. \end{cases}$$

The task is to track a desired path shown in Fig. 5 with joint angle limits conquered. Initial joint angle is $\theta(0) = [0; -\pi/2; 0; \pi/2; \pi/2; -\pi/4]$ (rad). Task duration is 40 s. The actual trajectories generated by solution (17) are also shown in Fig. 5. It is observed from this figure that the initial position of end-effector is away from desired position. It successfully tacks the desired path in a short time and then keeps the overlap with desired path.

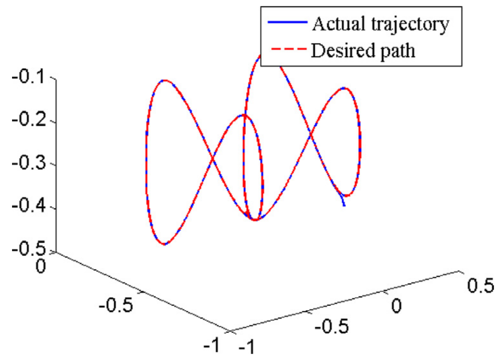


Fig. 5. End-effector trajectories of PUMA560 robot system synthesized by solution (17). The end-effector is in keeping with desired path all the while as the initial state of end-effector agrees with desired path.

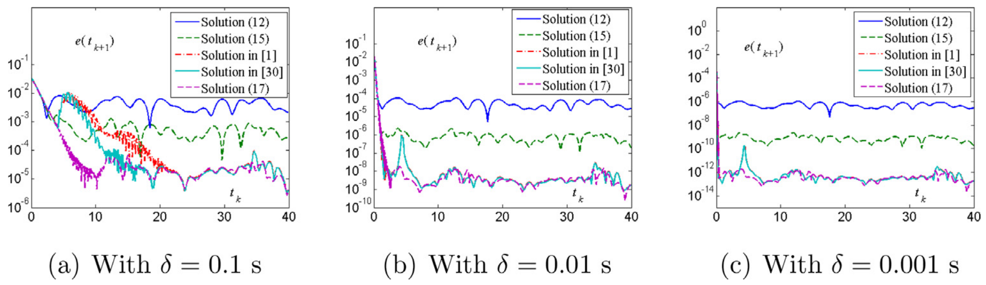


Fig. 6. Tracking errors of PUMA560 robot system defined synthesized by solution (12), solution (15), solution (17), solution in [1] and solution in [30] with $\delta = 0.1$ s, 0.01 s and 0.001 s. Solution (17), solution in [1] and solution in [30] have similar tracking errors. Solutions (12) and (15) have relatively lower accuracy.

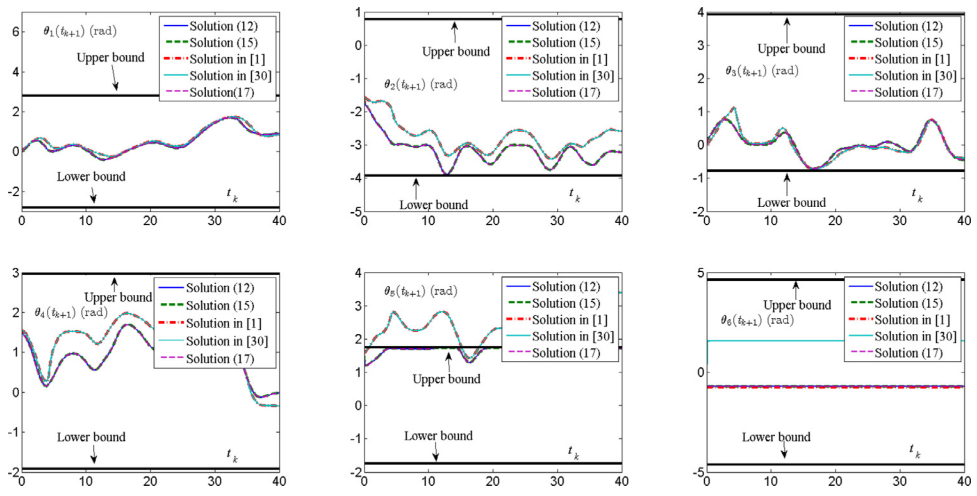


Fig. 7. Joint angle trajectories of PUMA560 robot system synthesized by solution (12), solution (15), solution (17), solution in [1] and solution in [30] with joint angle limits. Trajectories of every joint angles synthesized by solution (12), solution (15) and solution (17) always stay in the joint angle limits. In contrast, some joint angles synthesized by solutions in [1] and [30] are sometimes beyond the joint angle upper bound or lower bound.

To illustrate the superiority of solution (17), solutions in [1] and [30] as well as solutions with traditional discretization formulas (i.e., solutions (12) and (15)) are investigated. Comparative results are shown in Figs. 6, 7 and Table 4. From Fig. 6, it can be observed that all solutions perform well without considering joint angle limits. Solutions (12) and (15) have relatively lower accuracy compared with other solutions. Detailed results are shown in Table 4, which shows MSSTE and ACTPU data when using solutions (12), (15), (17) and solutions in [1] and [30] with sampling gap $\delta = 0.1$, 0.05, 0.01, 0.005 and 0.001 s. Fig. 7 shows variation of all joint angle values during the control process, from which it can be observed that joint angle values generated by solutions (12), (15) and (17) vary in the joint angle limits. In contrast, solutions in [1] and [30] fail

Table 4
MSSTE and ACTPU data when using solution (12), solution (15), solution (17), solution in [1] and solution in [30] with different δ values for PUMA560 robot manipulator system with joint angle limits as shown in Section 4.3.

δ (ms)	Solution (12)		Solution (15)		Solution in [1]		Solution in [30]		Solution (17)	
	MSSTE (m)	ACTPU (ms)	MSSTE (m)	ACTPU (ms)	MSSTE (m)	ACTPU (ms)	MSSTE (m)	ACTPU (ms)	MSSTE (m)	ACTPU (ms)
100	6.77×10^{-3}	0.6006	8.87×10^{-4}	0.6344	9.38×10^{-5}	0.4583	9.04×10^{-5}	0.4575	4.08×10^{-5}	0.7035
50	1.98×10^{-3}	0.6743	1.45×10^{-4}	0.5865	9.87×10^{-6}	0.3621	9.53×10^{-6}	0.4155	4.48×10^{-6}	0.6692
10	8.43×10^{-5}	0.5826	1.31×10^{-6}	0.5827	2.81×10^{-8}	0.3766	2.72×10^{-8}	0.4119	1.48×10^{-8}	0.6713
5	2.11×10^{-5}	0.5780	1.64×10^{-7}	0.5959	1.87×10^{-9}	0.3630	1.80×10^{-9}	0.3886	9.95×10^{-10}	0.6474
1	8.45×10^{-7}	0.5769	1.32×10^{-9}	0.5763	3.07×10^{-12}	0.3637	2.96×10^{-12}	0.3920	1.64×10^{-12}	0.6238

to conquer joint angle limits. In a word, solution (17) has high accuracy and has the ability to conquer joint angle limits simultaneously while the other solutions do not.

5. Conclusion

To solve the problem that requirements of real time, high precision and conquering joint angle limits cannot be satisfied simultaneously for conventional methods, the control of robot manipulator system is directly formulated as multilayered time-varying problem (5) including equalities and inequalities, in which equalities formulate tacking task and inequalities formulate joint angle limits. This new scheme to formulate robot manipulator control makes the corresponding proposed solution (17) more practical in real-word. As illustrated, future joint angle values $\theta(t_{k+1})$ could be obtained before time t_{k+1} comes, and average computing time per updating of solution (17) is less than 1 ms. The precision could reach 10^{-13} when sampling gap $\delta = 1$ ms, which is less than average computing time per updating. Besides, all joint angle limits could be conquered. The weakness of this work is that the dynamics of robot system is not considered, which could be our future research direction.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (with number 62006205, 31900710, 61906164), by the Natural Science Foundation of Jiangsu Province of China (with number BK20190875), and by Nanhu Scholars Program for Young Scholars of XYNU.

Appendix A

Based on zeroing neural dynamics method [32], to solve $\mathbf{f}(\theta(t)) - \mathbf{r}_\mathrm{d}(t) = 0$, a vector-form time-varying error is defined as

$$\mathbf{e}(t) = \mathbf{f}(\theta(t)) - \mathbf{r}_\mathrm{d}(t). \tag{A.1}$$

Then, using linear design formula of zeroing neural dynamics i.e., $\dot{\mathbf{e}}(t) = -\lambda\mathbf{e}(t)$, to zero out error function (A.1) [1]. Finally, the following equation is obtained as

$$J(\theta(t))\dot{\theta}(t) = \dot{\mathbf{r}}_\mathrm{d}(t) - \lambda(\mathbf{f}(\theta(t)) - \mathbf{r}_\mathrm{d}(t)). \tag{A.2}$$

Due to the utilization of linear design formula $\dot{\mathbf{e}}(t) = -\lambda\mathbf{e}(t)$, every element of $\mathbf{e}(t)$ exponentially converges to zero if the parameter $\lambda > 0$ and $t \to +\infty$. Thus, if the parameter $\lambda > 0$ and $t \to +\infty$, then the first layer of (5), i.e., $\mathbf{f}(\theta(t)) - \mathbf{r}_\mathrm{d}(t) = 0$, is equivalent to (A.2). The proof is completed. □

Appendix B

We define a vector $\gamma^{.2}(t) \in \mathbb{R}^n$ as $[\gamma_1^2(t), \gamma_2^2(t), ..., \gamma_n^2(t)]^\mathrm{T}$. As each elements of $\gamma^{.2}(t)$ are non-negative, the inequality $\theta(t) - \theta^+(t) \le 0$ is equivalent to

$$\theta(t) - \theta^+(t) + \gamma^{.2}(t) = 0. \tag{B.1}$$

To solve (B.1), based on zeroing neural dynamics method [32], an error function is defined as

$$\mathbf{e}(t) = \theta(t) - \theta^+(t) + \gamma^{.2}(t). \tag{B.2}$$

Then, to zero out error function (B.2), linear design formula of zeroing neural dynamics, i.e., $\dot{\mathbf{e}}(t) = -\lambda\mathbf{e}(t)$ is utilized [1], and Eq. (7) is obtained. Thus, if the parameter $\lambda > 0$ and $t \to +\infty$, then the second layer of (5), i.e., $\theta(t) - \theta^+(t) \le 0$, is equivalent to (7). The proof is completed. □

References

[1] J. Li, Y. Zhang, S. Li, M. Mao, New discretization-formula-based zeroing dynamics for real-time tracking control of serial and parallel manipulators, IEEE Trans. Ind. Informat. 14 (8) (2018) 3416–3425.
[2] M.H. Korayem, S.R. Nekoo, State-dependent differential Riccati equation to track control of time-varying systems with state and control nonlinearities, ISA Trans. 57 (2015) 117–135.
[3] Y. Zhang, W. Li, Z. Zhang, Physical-limits-constrained minimum velocity norm coordinating scheme for wheeled mobile redundant manipulators, Robotica 33 (6) (2015) 1325–1350.
[4] Z. Zhang, Y. Zhang, Variable joint-velocity limits of redundant robot manipulators handled by quadratic programming, IEEE/ASME Mechatron. 18 (2) (2013) 679–686.
[5] B. Siciliano, Kinematic control of redundant robot manipulators: a tutorial, J. Intell. Rob. Syst. 3 (1990) 201–212.
[6] G.Z. Grudic, P.D. Lawrence, Iterative inverse kinematics with manipulator configuration control, IEEE Trans. Rob. Autom. 9 (4) (1993) 476–483.
[7] G. Chen, B. Jin, Y. Chen, Accurate and robust body position trajectory tracking of six-legged walking robots with nonsingular terminal sliding mode control method, Appl. Math. Modell. 77 (2) (2020) 1348–1372.
[8] A.H. Korayem, S.R. Nekoo, M.H. Korayem, Sliding mode control design based on the state-dependent Riccati equation: theoretical and experimental implementation, Int. J. Control 92 (9) (2019) 2136–2149.
[9] J.M. Ahuactzin, K.K. Gupta, The kinematic roadmap: a motion planning based global approach for inverse kinematics of redundant robots, IEEE Trans. Rob. Autom. 15 (4) (1999) 653–669.
[10] M.H. Korayem, H.R. Nohooji, A. Nikoobin, Path planning of mobile elastic robotic arms by indirect approach of optimal control, Int. J. Adv. Rob. Syst. 8 (1) (2011) 10–20.
[11] X. Li, G. Zhao, B. Li, Generating optimal path by level set approach for a mobile robot moving in static/dynamic environments, Appl. Math. Modell. 85 (2020) 210–230.
[12] L. Jin, S. Li, B. Hu, M. Liu, J. Yu, A noise-suppressing neural algorithm for solving the time-varying system of linear equations: a control-based approach, IEEE Trans. Ind. Informat. 15 (1) (2019) 236–246.
[13] J. Li, Y. Zhang, M. Mao, General square-pattern discretization formulas via second-order derivative elimination for zeroing neural network illustrated by future optimization, IEEE Trans. Neural Netw. Learn. Syst. 30 (3) (2019) 891–901.
[14] L. Jin, S. Li, B. Hu, RNN models for dynamic matrix inversion: a control-theoretical perspective, IEEE Trans. Ind. Informat. 14 (1) (2018) 189–199.
[15] Z. Zhang, L. Zheng, H. Yang, X. Qu, Design and analysis of a novel integral recurrent neural network for solving time-varying Sylvester equation, IEEE Trans. Cybern. (2019), doi:10.1109/TCYB.2019.2939350.
[16] W. Li, Z. Su, Z. Tan, A variable-gain finite-time convergent recurrent neural network for time-variant quadratic programming with unknown noises endured, IEEE Trans. Ind. Informat. 15 (9) (2019) 5330–5340.
[17] L. Wei, L. Jin, C. Yang, K. Chen, W. Li, New noise-tolerant neural algorithms for future dynamic nonlinear optimization with estimation on Hessian matrix inversion, IEEE Trans. Syst. Man Cybern. Syst. (2019), doi:10.1109/TSMC.2019.2916892. To be published
[18] J. Li, M. Mao, F. Uhlig, Y. Zhang, Z-type neural-dynamics for time-varying nonlinear optimization under a linear equality constraint with robot application, J. Comput. Appl. Math. 327 (2018) 155–166.
[19] H. Lu, L. Jin, J. Zhang, Z. Sun, S. Li, Z. Zhang, New joint-drift-free scheme aided with projected ZNN for motion generation of redundant robot manipulators perturbed by disturbances, IEEE Trans. Syst. Man Cybern. Syst. (2019), doi:10.1109/TSMC.2019.2956961. To be published
[20] W. Gong, D. Chen, S. Li, Active sensing of robot arms based on zeroing neural networks: a biological-heuristic optimization model, IEEE Access 8 (2020) 25976–25989.
[21] D. Guo, Z. Nie, L. Yan, The application of noise-tolerant ZD design formula to robots' kinematic control via time-varying nonlinear equations solving, IEEE Trans. Syst. Man Cybern. Syst. 48 (12) (2018) 2188–2197.
[22] D. Guo, Z. Xi, A.H. Khan, Q. Feng, J. Cai, Repetitive motion planning of robotic manipulators with guaranteed precision, IEEE Trans. Ind. Informat. 17 (2021) 356–366.
[23] L. Jin, J. Yan, X. Du, X. Xiao, D. Fu, RNN for solving time-variant generalized Sylvester equation with applications to robots and acoustic source localization, IEEE Trans. Ind. Informat. 16 (10) (2020) 6359–6369.
[24] Y. Qi, L. Jin, Y. Wang, L. Xiao, J. Zhang, Complex-valued discrete-time neural dynamics for perturbed time-dependent complex quadratic programming with applications, IEEE Trans. Neural Netw. Learn. Syst. 31 (9) (2020) 3555–3569.
[25] F. Xu, Z. Li, Z. Nie, H. Shao, D. Guo, Zeroing neural network for solving time-varying linear equation and inequality systems, IEEE Trans. Neural Netw. Learn. Syst. 30 (8) (2019) 2346–2357.
[26] H. Lu, L. Jin, X. Luo, B. Liao, D. Guo, L. Xiao, RNN for solving perturbed time-varying underdetermined linear system with double bound limits on residual errors and state variables, IEEE Trans. Ind. Informat. 15 (11) (2019) 5931–5942.
[27] Y. Zhang, M. Yang, H. Huang, M. Xiao, H. Hu, New discrete-solution model for solving future different-level linear inequality and equality with robot manipulator control, IEEE Trans. Ind. Informat. 15 (4) (2019) 1975–1984.
[28] M. Yang, Y. Zhang, H. Hu, B. Qiu, General 7-instant DCZNN model solving future different-level system of nonlinear inequality and linear equation, IEEE Trans. Neural Netw. Learn. Syst. 31 (9) (2020) 3204–3214.
[29] J. Guo, B. Qiu, J. Chen, Y. Zhang, Solving future different-layer nonlinear and linear equation system using new eight-node DZNN model, IEEE Trans. Ind. Informat. 16 (4) (2020) 2280–2289.
[30] J. Li, Y. Zhang, M. Mao, Continuous and discrete zeroing neural network for different-level dynamic linear system with robot manipulator control, IEEE Trans. Syst. Man Cybern. Syst. 50 (2020) 4633–4642.
[31] J. Li, M. Mao, Y. Zhang, B. Qiu, Different-level algorithms for control of robotic systems, Appl. Math. Modell. 77 (2020) 922–933.
[32] Y. Zhang, C. Yi, Zhang Neural Networks and Neural-Dynamic Method, Nova, New York, USA, 2011.
[33] E. Suli, D.F. Mayers, An Introduction to Numerical Analysis, Cambridge University Press, Oxford, UK, 2003.
[34] L. Jin, Y. Zhang, Discrete-time zhang neural network of $o(\tau^3)$ pattern for time-varying matrix pseudoinversion with application to manipulator motion generation, Neurocomputing 142 (2014) 165–173.
[35] D.F. Griffiths, D.J. Higham, Numerical Methods for Ordinary Differential Equations: Initial Value Problems, Springer, England, 2010.
[36] S.B. Niku, Introduction To Robotics: Analysis, Control, Applications, 2nd, Wiley, 2010.
[37] P.V. Nagy, Dynamic characteristics of a PUMA 560 manipulator, and a complementary tracking control strategy, CAD/CAM Robotics and Factories of the Future, Springer, Berlin, Heidelberg, 1989.