11th International Conference on Intelligent Control and Information Processing
Dali, China, December 3-7, 2021

2021 11th International Conference on Intelligent Control and Information Processing (ICICIP) | 978-1-6654-2515-5/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICICIP53388.2021.9642176

Robot Manipulator Control via Solving Four-Layered Time-Variant Equations Including Linear, Nonlinear Equalities and Inequalities

Xinhui Zhu[1], Li Zhang[1], Yang Shi[3], Jing Wang[1,2], Jian Li[1,2]
1. School of Computer and Information Technology, Xinyang Normal University, Xinyang 464000, China
2. Henan Key Lab. of Analysis and Application of Education Big Data, Xinyang Normal University, Xinyang, 464000, China
3. School of Information Engineering, Yangzhou University, Yangzhou 225127, China

***Abstract*—Robot manipulator control is a complicated multi-tasking problem in reality. It includes not only basic tracking task, but also additional tasks, such as conquering joint angle limits, posture control. However, most existing works only consider the goal of tracking and formulate it as single-layered time-variant problems, which leads to impracticality. In this work, robot manipulator control problem is formulated as four-layered time-variant equations including linear, nonlinear equalities and inequalities. Each layer formulates one subtask: The first layer of nonlinear equality describes basic tracking task based on forward kinematics; The second layer and third layer are inequalities, which describe joint angle upper and lower limits; The last layer is a linear equality with respect to joint angle velocity, which could be designed by user to describe other task, such as posture control. To solve this complicated four-layered time-variant problem, it is converted as single-layered equation based on the zeroing neural dynamics method. Then, continuous-time solution is proposed. Furthermore, discrete-time algorithm is proposed based on a third-order time-discretization formula and continuous-time solution. Numerical experiments illustrate the effectiveness and superiority compared to existing work.**

***Index Terms*—Robot manipulator control, Zeroing neural dynamics, Multitasking problem, Four-layered time-variant equations**

I. Introduction

The problem of robot manipulator control is significant in robotics, which appears in most robotic applications [1]–[4]. Recent years, increasing attentions to this problem have been given because the requirements of real-time control and high accuracy increase urgently to satisfy needs of more intelligent industry. Besides, robot manipulator control is a complicated multitasking problem in reality. It includes not only basic tracking task, but also additional tasks, such as conquering joint angle limits, posture control. Every tasks have to be completed simultaneously during control process. Thus, it is rigorous to algorithm developing for satisfying such requirements.

This work was supported by the National Natural Science Foundation of China (with number 62006205, 61906164, 31900710); by the China Postdoctoral Science Foundation funded project (with number 2021TQ0299); by the Natural Science Foundation of Jiangsu Province of China (with number BK20190875) and by Nanhu Scholars Program for Young Scholars of XYNU.
Corresponding author is Jian Li (Email: lijcit@xynu.edu.cn)

Classical methods to develop robot control algorithms include PID method, sliding mode control, self-adaptive method and so on [1]–[6]. In [4], a universal motion model was developed to solve the manipulator control problem of photovoltaic cleaning robots by transforming it into the stabilization problem of the error system. In [5], a sliding mode controller was designed to track each joints of created two joint robot model, which is designed by combining the mathematical model and dynamic characteristics of the robot system. In [6], trajectory tracking problem of a spherical robot was investigated using adaptive PID controller, which is an attempt to the control problem of other similar underactuated mobile robots. These existing works using classical methods usually consider dynamics of robots, robust performances and so on, which perform well in some applications. However, these methods may have some disadvantages such as poor real-time performance and time-consuming parameters adjustment. Recent years, deep reinforcement learning method has been applied to the developments of robot control algorithms. In [7], a hybrid control strategy combined mode-based control and actor-critic based deep reinforcement learning method was presented to solve the tracking control problem of nonholonomic wheeled mobile robots. However, this method is based on deep neural networks, which exists uncertainty during the control process.

Recent years, more and more researchers have investigated the robot manipulator control problem via formulating it as time-variant problems [8]–[13]. Because time-variant problems could be solved systematically by zeroing neural dynamics and this kind of formulations make the corresponding control algorithms have great performance in many aspects. For example, in [8], the robot manipulator control was formulated as time-variant matrix inversion, which was solved by zeroing neural dynamics (or say, Zhang neural network) method as well as a second-order time-discretization formula. This method of formulating robot control as time-variant problems has advantages in control accuracy and real time. However, as limited by solution method, it is usually formulated as relatively easy-handled time-variant problems, such as time-variant linear and nonlinear system. It leads to the failure of multitasking descriptions.

In this work, we formulate robot manipulator control problem as four-layered time-variant equations including linear, nonlinear equalities and inequalities. Each layer formulates one subtask: The first layer of nonlinear equality describes basic tracking task based on forward kinematics; The second layer and third layer are inequalities, which describe joint angle upper and lower limits; The last layer is a linear equality with respect to joint angle velocity, which could be designed by user to describe other task, such as posture control. To solve this complicated four-layered time-variant problem, it is converted as single-layered equation based on the zeroing neural dynamics method. Then, continuous-time solution is proposed. Furthermore, discrete-time algorithm is proposed based on a third-order time-discretization formula and continuous-time solution. Finally, numerical experiments are conducted to illustrate the effectiveness of our algorithm.

The remainder of this paper is organized into five sections. Section II is problem formulation, which describes robot manipulator problem as four-layered time-variant equations. Section III presents continuous-time solution via zeroing neural dynamics method. Section IV presents our discrete-time algorithm. Section V shows many numerical results, which substantiate the effectiveness and superiority of proposed algorithm. Section VI is the conclusion of this work. The main contributions are listed as below.

1) Robot manipulator control problem is described as four-layered time-variant equations including linear, nonlinear equalities and inequalities.
2) Continuous-time solution and discrete-time algorithm are proposed on a basis of zeroing neural dynamics as well as a third-order time-discretization formula.

II. Problem formulation of robot manipulator control

The problem of robot manipulator control is formulated as the following four-layered time-variant equations (FLTVE):

$$\begin{cases} \mathbf{h}(\eta(t)) = \mathbf{l}_{\mathrm{d}}(t), \\ \theta \, \eta(t) - \eta^{+}(t) \leq 0, \\ \quad -\eta(t) + \eta^{-}(t) \leq 0, \\ A(t)\dot{\eta}(t) + B(t)\eta(t) = \mathbf{d}(t), \end{cases} \tag{1}$$

which includes linear, nonlinear equalities and inequalities. $\mathbf{h}(\cdot) \in \mathbb{R}^m$ is forward kinematic map of robot; $\eta(t) \in \mathbb{R}^n$ is joint angle vector, which is time-variant control variant; $\mathbf{l}_{\mathrm{d}}(t)$ is time-variant desired path, which could only get online (i.e., future information of desired path is unknown at current instant); $\eta^{+}(t) \in \mathbb{R}^n$ and $\eta^{-}(t) \in \mathbb{R}^n$ are joint angle upper and lower bounds, respectively. $\dot{\eta}(t)$ is the joint velocity. Time-variant matrix $A(t) \in \mathbb{R}^{j\times n}$, $B(t) \in \mathbb{R}^{j\times n}$ and vector $\mathbf{d}(t) \in \mathbb{R}^j$ could be constructed as needed to satisfy additional requirements during robot control process.

It is evident that the problem of FLTVE (1) includes four layers. The first layer is to formulate the forward kinematic constraint such that the basic tracking task could be completed. The second layer and third layer is to formulate joint angle limits including upper and lower bounds, respectively. The fourth layer is to formulate additional constraints during the control process based on the constructed time-variant matrix and vector. We take the planar serial robot manipulator as an example, we could construct the matrix $A(t)$ as $[1, 1, ..., 1]$ with 1 row and n columns, the matrix $B(t)$ as zero and the vector $\mathbf{b}(t)$ as zero such that the posture of last link of robot manipulator remains fixed.

III. Continuous-time solution

The problem of FLTVE (1) includes four different layers containing linear, nonlinear equalities and inequalities. Thus, it is difficult to solve FLTVE (1) directly. In this work, we solve it by transforming each layers into a uniform layer and combine them together such that FLTVE (1) is equivalently converted as a simple form. For more concise presentation, notation t denoting time is omitted sometimes in this section.

Firstly, the first equation in FLTVE (1), i.e.,

$$\mathbf{h}(\eta) = \mathbf{l}_{\mathrm{d}} \tag{2}$$

is equivalently converted as

$$J(\eta)\dot{\eta} = \dot{\mathbf{l}}_{\mathrm{d}} - \lambda(\mathbf{h}(\eta) - \mathbf{l}_{\mathrm{d}}), \tag{3}$$

where $J(\eta)$ is Jacobian matrix of the robot and defined as

$$J(\eta) = \frac{\partial \mathbf{h}(\eta)}{\partial \eta}.$$

$\dot{\mathbf{l}}_{\mathrm{d}}$ is the time derivative of $\mathbf{l}_{\mathrm{d}}$. The derivation process of this equivalency could be found in [13].

Secondly, the second equation of FLTVE (1), i.e.,

$$\eta - \eta^{+} \leq 0 \tag{4}$$

is equivalently converted as

$$\begin{bmatrix} I & 2\Lambda \end{bmatrix} \begin{bmatrix} \dot{\eta} \\ \dot{\gamma} \end{bmatrix} = \dot{\eta}^{+} - \lambda\left(\eta - \eta^{+} + \gamma^{\cdot 2}\right), \tag{5}$$

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

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

Besides, we define $\gamma \in \mathbb{R}^n$ as $[\gamma_1, \gamma_2, ..., \gamma_n]^{\mathrm{T}}$, and $\gamma^{\cdot 2} \in \mathbb{R}^n$ as $[\gamma_1^2, \gamma_2^2, ..., \gamma_n^2]^{\mathrm{T}}$. The derivation process of this equivalency can be found in [14].

Thirdly, the third equation of FLTVE (1), i.e.,

$$-\eta + \eta^{-} \leq 0 \tag{6}$$

is equivalently converted as

$$\begin{bmatrix} -I & 2\Gamma \end{bmatrix} \begin{bmatrix} \dot{\eta} \\ \dot{\theta} \end{bmatrix} = -\dot{\eta}^{-} - \lambda\left(\theta^{\cdot 2} - \eta + \eta^{-}\right) \tag{7}$$

where Γ is denoted as below:

$$\Gamma = \begin{bmatrix} \theta_1 & 0 & \cdots & 0 \\ 0 & \theta_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \theta_n \end{bmatrix}.$$

Besides, we define $\theta \in \mathbb{R}^n$ as $[\theta_1, \theta_2, ..., \theta_n]^{\mathrm{T}}$, and $\theta^{\cdot 2} \in \mathbb{R}^n$ as $[\theta_1^2, \theta_2^2, ..., \theta_n^2]^{\mathrm{T}}$. The derivation process of this equivalency is similar to the second equivalency, which is omitted for compact presentation.

Based on the above analyses, we combine the above three equivalent equations (i.e., equation (3), equation (5) and equation (7)) into the fourth layer of FLTVE (1), which correspond to the first layer, second layer and third layer. Then, FLTVE (1) is equivalently converted as below:

$$W(\mathbf{z})\dot{\mathbf{z}} = \nu, \tag{8}$$

where $\mathbf{z} = [\eta^{\mathrm{T}}, \gamma^{\mathrm{T}}, \theta^{\mathrm{T}}]^{\mathrm{T}}$,

$$W(\mathbf{z}) = \begin{bmatrix} J(\eta) & 0 & 0 \\ I & 2\Lambda & 0 \\ -I & 0 & 2\Gamma \\ A & 0 & 0 \end{bmatrix} \tag{9}$$

and

$$\nu = \begin{bmatrix} \dot{\mathbf{l}}_{\mathrm{d}} - \lambda(\mathbf{h}(\eta) - \mathbf{l}_{\mathrm{d}}) \\ \dot{\eta}^{+} - \lambda\left(\eta - \eta^{+} + \gamma^{\cdot 2}\right) \\ -\dot{\eta}^{-} - \lambda\left(\theta^{\cdot 2} - \eta + \eta^{-}\right) \\ \mathbf{d} - B\eta \end{bmatrix}.$$

Thus, to solve complicated FLTVE (1) with four different layers is converted to solve relatively simple equation (8), which has single layer and linear form. Finally, the continuous-time solution is derived as

$$\dot{\mathbf{z}} = W^{+}(\mathbf{z})\nu, \tag{10}$$

where $^{+}$ denotes pseudo-inverse operation. Note that continuous-time solution (10) is one feasible solution among infinite solutions of equation (8).

IV. Discrete-time algorithm

In this section, a time-discretization formula is developed firstly, and then a discrete-time algorithm is developed based on this formula and continuous-time solution (10).

A. Time-discretization formula

Continuous-time solution (10) could be implemented by ODE solvers directly. However, it may fail to satisfy the requirement of real-time computation, which is quite important in robot control. Based on previous work, real-time discrete-time algorithms could be developed by using a special kind of time-discretization formulas to discretize continuous-time solution. This kind of time-discretization formulas have a form of finite difference and must be one step ahead, i.e., $\dot{x}(t_k) = p_1 x(t_{k+1}) + p_2 x(t_k) + p_3 x(t_{k-1}) + p_4 x(t_{k-2})....$ Besides, the time-discretization formulas must satisfy the constraint of zero-stability, such that the corresponding algorithms are convergent. In this work, a third-order time-discretization formula is developed as below:

$$\begin{aligned} \dot{x}(t_k) =& \frac{50}{111\iota}x(t_{k+1}) + \frac{7}{222\iota}x(t_k) - \frac{11}{37\iota}x(t_{k-1}) \\ & - \frac{67}{222\iota}x(t_{k-2}) + \frac{13}{111\iota}x(t_{k-3}) + O(\iota^3), \end{aligned} \tag{11}$$

where ι denotes the sampling gap, i.e., $\iota = t_{k+1} - t_k = t_k - t_{k-1} = t_{k-1} - t_{k-2}...$ and $O(\iota^3)$ is the truncation error. We assume that $x(t)$ has bounded fourth-order derivative. Specifically, the following equations are obtained on the basis of Taylor expansion because $x(t)$ has bounded fourth-order derivative:

$$\begin{aligned} x(t_{k+1}) =& x(t_k) + \dot{x}(t_k)\iota + \frac{x^{(2)}(t_k)}{2!}\iota^2 \\ & + \frac{x^{(3)}(t_k)}{3!}\iota^3 + \frac{x^{(4)}(c_1)}{4!}\iota^4, \end{aligned} \tag{12}$$

$$\begin{aligned} x(t_{k-1}) =& x(t_k) - \dot{x}(t_k)\iota + \frac{x^{(2)}(t_k)}{2!}\iota^2 \\ & - \frac{x^{(3)}(t_k)}{3!}\iota^3 + \frac{x^{(4)}(c_2)}{4!}\iota^4, \end{aligned} \tag{13}$$

$$\begin{aligned} x(t_{k-2}) =& x(t_k) - 2\dot{x}(t_k)\iota + \frac{x^{(2)}(t_k)}{2!}(2\iota)^2 \\ & - \frac{x^{(3)}(t_k)}{3!}(2\iota)^3 + \frac{x^{(4)}(c_3)}{4!}\iota^4, \end{aligned} \tag{14}$$

and

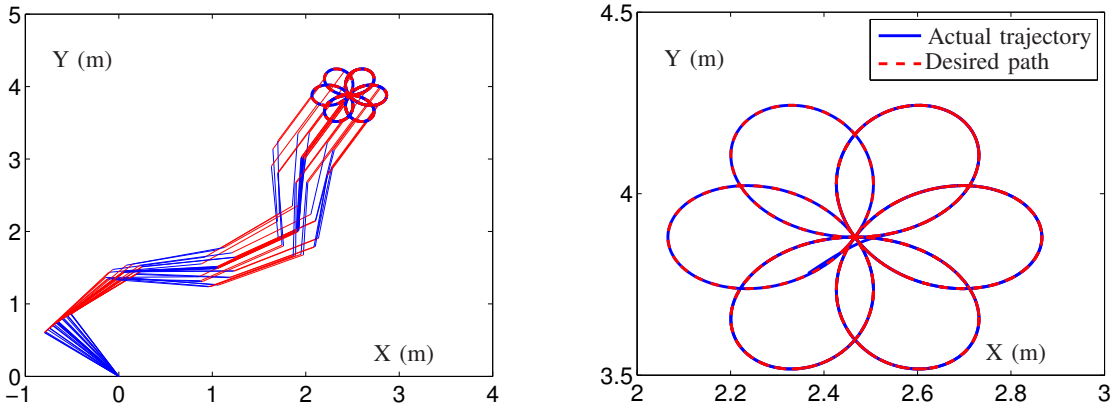
$$\begin{aligned} x(t_{k-3}) =& x(t_k) - 3\dot{x}(t_k)\iota + \frac{x^{(2)}(t_k)}{2!}(3\iota)^2 \\ & - \frac{x^{(3)}(t_k)}{3!}(3\iota)^3 + + \frac{x^{(4)}(c_4)}{4!}\iota^4, \end{aligned} \tag{15}$$

where $x^{(2)}(t)$, $x^{(3)}(t)$ and $x^{(4)}(t)$ denote the second-order, third-order and fourth-order derivatives of $x(t)$ with respect to t, respectively; c_1, c_2, c_3 and c_4 lie in (t_k, t_{k+1}), (t_{k-1}, t_k), (t_{k-2}, t_k) and (t_{k-3}, t_k), respectively; ! denotes the factorial operator. Let (12) multiply 100; let (13) multiply -66; let (14) multiply -67; and let (15) multiply 26. Adding together the results yields new five-instant time-discretization formula (11) with $O(\iota^3)$ truncation error. The coefficients in formula (11) is derived by four fundamental operations of above Taylor expansions by trying. The readers may obtain general form formulas based on previous work [15].

B. Development and analysis of discrete-time algorithm

Utilizing time-discretization formula (11) to discretize continuous-time solution (10), we obtain discrete-time algorithm. Specifically, based on time-discretization formula (11), we have

$$\begin{aligned} \dot{\mathbf{z}}(t_k) =& \frac{50}{111\iota}\mathbf{z}(t_{k+1}) + \frac{7}{222\iota}\mathbf{z}(t_k) - \frac{11}{37\iota}\mathbf{z}(t_{k-1}) \\ & - \frac{67}{222\iota}\mathbf{z}(t_{k-2}) + \frac{13}{111\iota}\mathbf{z}(t_{k-3}) + \mathbf{O}(\iota^3), \end{aligned} \tag{16}$$



(a) (b)

Fig. 1. (a) Robot trajectory and (b) end-effector trajectory generated by our algorithm (18)

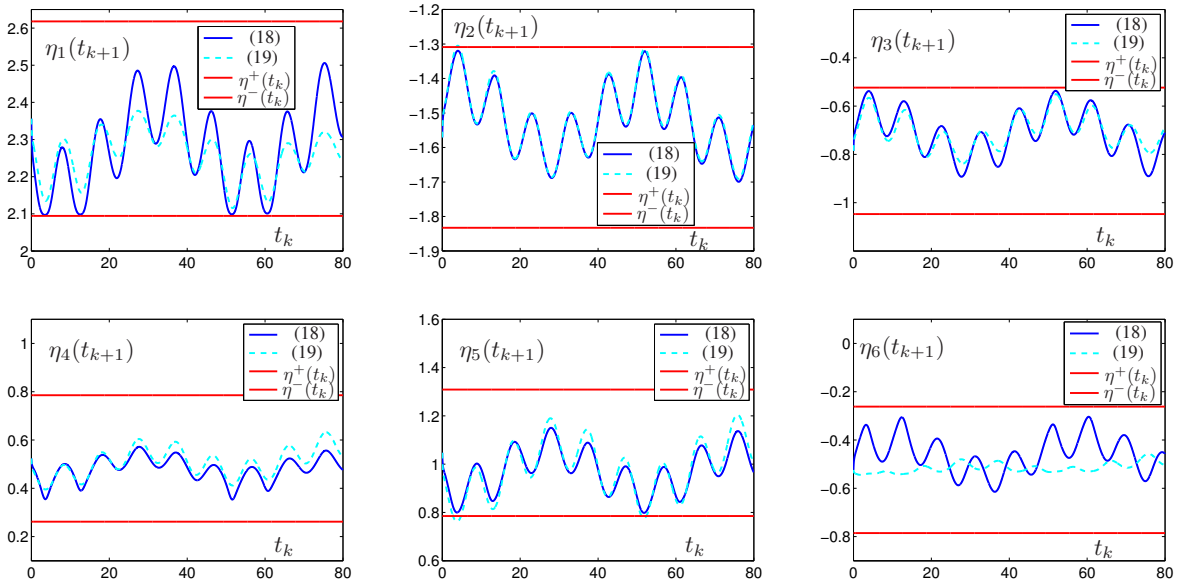


Fig. 2. Joint angle trajectories synthesized by our algorithm (18) and conventional algorithm (19) with constant joint angle limits.

where $\mathbf{O}(\iota^3)$ denotes a vector, in which each elements are $O(\iota^3)$. Combining equation (16) and continuous-time solution (10) yields

$$W^+(\mathbf{z}(t_k))\nu(t_k) = \frac{50}{111\iota}\mathbf{z}(t_{k+1}) + \frac{7}{222\iota}\mathbf{z}(t_k) - \frac{11}{37\iota}\mathbf{z}(t_{k-1}) - \frac{67}{222\iota}\mathbf{z}(t_{k-2}) + \frac{13}{111\iota}\mathbf{z}(t_{k-3}) + \mathbf{O}(\iota^3).$$

It is exactly

$$\mathbf{z}(t_{k+1}) = -\frac{7}{100}\mathbf{z}(t_k) + \frac{33}{50}\mathbf{z}(t_{k-1}) + \frac{67}{100}\mathbf{z}(t_{k-2}) - \frac{13}{50}\mathbf{z}(t_{k-3}) + \frac{111}{50}W^+(\mathbf{z}(t_k))\tilde{\nu}(t_k) + \mathbf{O}(\iota^4), \quad (17)$$

where $\tilde{\nu}(t_k) = \iota\nu(t_k)$ shown as

$$\begin{bmatrix} \iota\dot{\mathbf{l}}_{\mathrm{d}}(t_k) - h(\mathbf{h}(\eta(t_k)) - \mathbf{l}_{\mathrm{d}}(t_k)) \\ \iota\dot{\eta}^+(t_k) - h\left(\eta(t_k) - \eta^+(t_k) + \gamma^{\cdot 2}(t_k)\right) \\ -\iota\dot{\eta}^-(t_k) - h\left(\theta^{\cdot 2}(t_k) - \eta(t_k) + \eta^-(t_k)\right) \\ \iota\mathbf{d}(t_k) - \iota B(t_k)\eta(t_k) \end{bmatrix}$$

with $h = \lambda\iota$. Omitting the truncation error of (17), we obtain discrete-time algorithm as below:

$$\mathbf{z}(t_{k+1}) = -\frac{7}{100}\mathbf{z}(t_k) + \frac{33}{50}\mathbf{z}(t_{k-1}) + \frac{67}{100}\mathbf{z}(t_{k-2}) - \frac{13}{50}\mathbf{z}(t_{k-3}) + \frac{111}{50}W^+(\mathbf{z}(t_k))\tilde{\nu}(t_k). \quad (18)$$

Theorem 1: Discrete-time algorithm (18) is 0-stable, consistent and convergent, which converges with the truncation error of order $\mathbf{O}(\delta^4)$.

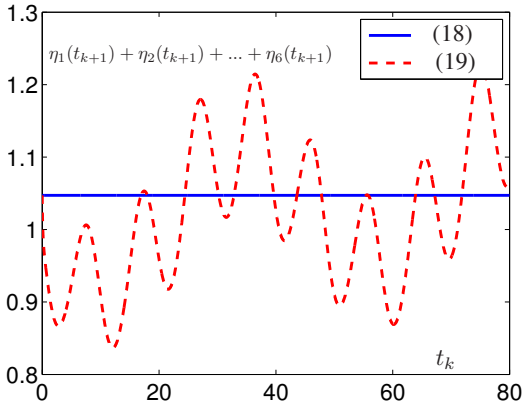
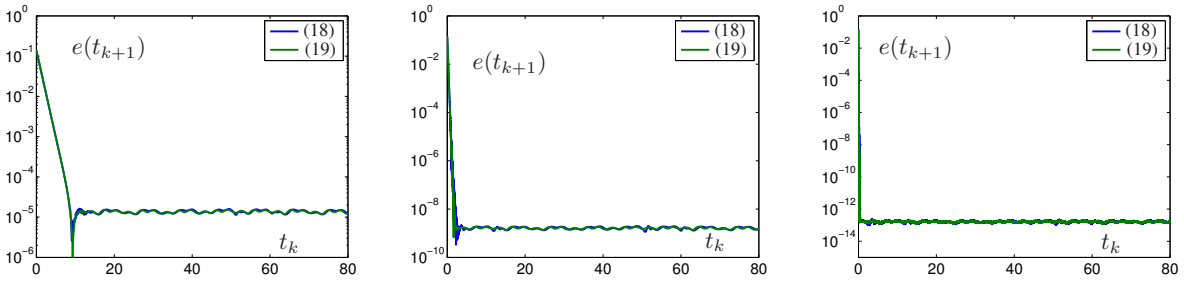


Fig. 3. Trajectories of sum of joint angles synthesized by our algorithm (18) and conventional algorithm (19) with constant joint angle limits.



(a) With $\delta = 0.1$ s (b) With $\delta = 0.01$ s (c) With $\delta = 0.001$ s

Fig. 4. Tracking errors defined as $e(t_{k+1}) = \|\mathbf{l}_{\mathrm{a}}(t_{k+1}) - \mathbf{l}_{\mathrm{d}}(t_{k+1})\|$ generated by our algorithm (18) and conventional algorithm (19) with (a) $\iota = 0.1$ s, (b) $\iota = 0.01$ s and (c) $\iota = 0.001$ s.

Proof: The 1st characteristic polynomial of discrete-time algorithm (18) is

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

Its four roots satisfy the 0-stability condition [16]. Thus, discrete-time algorithm (18) is zero-stable. The second characteristic polynomial of discrete-time algorithm (18) is $\sigma(\varsigma) = 111/50\varsigma^3$. It is evident that $P_4(1) = 0$ and $P_4'(1) = \sigma(1) \neq 0$. Thus, discrete-time algorithm (18) is consistent. Furthermore, discrete-time algorithm (18) has an truncation error of $\mathbf{O}(\iota^4)$. Finally, according to Dahlquist equivalence theorem [16], discrete-time algorithm (18) is convergent, which converges with the truncation error of order $\mathbf{O}(\delta^4)$. ■

Note that discretization formulas significantly influence the control precision, which may be a way to improve the tracking performance of the proposed control scheme.

V. Numerical Experiments

In this section, we employ six-link planar robot manipulator and do some numerical experiments to illustrate the effectiveness of our algorithm (18). Each links of the robot manipulator are 1 m. The forward kinematics and Jacobi matrix of robot manipulator is assumed to be known. Initial joint angle is set as $[3\pi/4, -\pi/2, -\pi/4, \pi/6, \pi/3, -\pi/6]^{\mathrm{T}}$ (rad). Initial value of intermediate variable vectors γ and θ are set as $[1, 1, 1, 1, 1, 1]^{\mathrm{T}}$. Upper bound of joint angle is set as $\eta(t_0) + \pi/12$ and lower bound of joint angle is set as $\eta(t_0) - \pi/12$. Task duration is 80 s. Parameter h is set as 0.1. Desired path is shown as

$$\mathbf{l}_{\mathrm{d}}(t_k) = \begin{bmatrix} (3\sin(\pi t_k/8) + 3\sin(\pi t_k/8))\cos(\pi t_k/12)/10 \\ 3(\sin(\pi t_k/8)/3 + \sin(\pi t_k/8))\sin(\pi t_k/12)/10 \end{bmatrix} + \mathbf{c},$$

where $\mathbf{c}$ is constant vector, such that the initial position of end-effector of robot manipulator is near the initial position of desired path. The time derivative of desired path is approximated by backward finite difference formula as below:

$$\dot{\mathbf{l}}_{\mathrm{d}}(t_k) = \frac{11}{6\iota}\mathbf{l}_{\mathrm{d}}(t_k) - \frac{3}{\iota}\mathbf{l}_{\mathrm{d}}(t_{k-1}) + \frac{3}{2\iota}\mathbf{l}_{\mathrm{d}}(t_{k-2}) - \frac{1}{3\iota}\mathbf{l}_{\mathrm{d}}(t_{k-3}).$$

Note that the data of desired path and time derivative are get online. We do not use future information at current instant during the implementing of our algorithm. Firstly, to verify the effectiveness of our algorithm, sampling gap is set as 0.01 s and employ our algorithm to complete the robot manipulator control tasks, which includes basic tracking task, the task of conquering joint angle limits and the task of posture control of the last link. Here, matrix $A(t_k)$ is constructed as $[1, 1, ..., 1]$ and vector $\mathbf{b}(t_k)$ is constructed as 0, such that the third subtask

could be completed. For comparison, a conventional algorithm [13] is shown as below:

$$\theta(t_{k+1}) = \frac{9}{4}J^{+}(\theta(t_k))\left(\iota\dot{\mathbf{r}}_{\mathrm{d}}(t_k) - h\left(\mathbf{f}(\theta(t_k)) - \mathbf{r}_{\mathrm{d}}(t_k)\right)\right) - \frac{1}{8}\theta(t_k) + \frac{3}{4}\theta(t_{k-1}) + \frac{5}{8}\theta(t_{k-2}) - \frac{1}{4}\theta(t_{k-3}). \tag{19}$$

Numerical experimental results are shown in Fig. 1, Fig. 2 and Fig. 3. Fig. 1 shows the robot trajectory and end-effector trajectory generated by our algorithm (18). Fig. 2 shows joint angle trajectories synthesized by our algorithm (18) and conventional algorithm (19) with constant joint angle limits. Fig. 3 shows trajectories of sum of joint angles synthesized by our algorithm (18) and conventional algorithm (19). One can observe from Fig. 1 that the actual trajectory generated by our algorithm (18) tracks the desired path online quickly, which shows that the basic tacking task is completed. From Fig. 2, one can observe that all joint angle values are always in the joint angle limits during the whole process, which shows that the task of conquering joint angle limits is completed. In contrast, conventional algorithm (19) fails to complete this task as its joint angle values are sometimes out of the limits. From Fig. 3, it is observed that the sum of all joint angle values generated by our algorithm (18) is invariant, which shows that the task of posture control of the last link is completed. In contrast, conventional algorithm (19) fails to do it.

To illustrate the precision of our algorithm (18) as well as conventional algorithm (19), we employ different sampling gap values and illustrate the tracking errors for different sampling gap values. The other settings are the same as above. Numerical results are shown in Fig. 4. Specifically, Fig. 4 shows tracking errors defined as $e(t_{k+1}) = \|\mathbf{l}_{\mathrm{a}}(t_{k+1}) - \mathbf{l}_{\mathrm{d}}(t_{k+1})\|$ generated by our algorithm (18) and conventional algorithm (19) with $\iota = 0.1$ s, $\iota = 0.01$ s and $\iota = 0.001$ s. When the tracking errors become steady states, the tracking errors are around 10^{-5}, 10^{-9} and 10^{-13} for $\iota = 0.1$ s, $\iota = 0.01$ s and $\iota = 0.001$ s, respectively. Thus, tracking error is $O(\iota^4)$. Bedsides, it can be observed that tracking errors generated by conventional algorithm (19) is similar with those of algorithm (18), which means that conventional algorithm (19) could also complete the basic tracking task if no other tasks to be completed simultaneously.

VI. Conclusion

In this paper, robot manipulator control has been formulated as four-layered time-variant equations to describe complicated multitasks of robot manipulator control, which makes the control algorithm more practical. The four-layered time-variant equations has been solved by converting it as single-layered equation via zeroing neural dynamics. Furthermore, based on a third-order time-discretization formula, our control algorithm has been proposed. Finally, comparative numerical experiments have been conducted to illustrate the effectiveness of our algorithm. Besides, our algorithm has been compared with another conventional algorithm, which shows the superiority of our algorithm. Further studies include the evaluation on the performance of the proposed method with stochastic disturbances and noises and the performance with physical experimental platform.

References

[1] A. Andreev and O. Peregudova, "On the trajectory tracking control of a wheeled mobile robot based on a dynamic model with slip," in Proceedings of 15th International Conference on Stability and Oscillations of Nonlinear Control Systems (Pyatnitskiy's Conference) (STAB), 2020, pp. 1–4.

[2] L. Ren, J. K. Mills, and D. Sun, "Experimental comparison of control approaches on trajectory tracking control of a 3-DOF parallel robot," *IEEE Transactions on Control Systems Technology*, vol. 15, no. 5, pp. 982-988, Sept. 2007.

[3] X. Liang, H. Wang, W. Chen, D. Guo, and T. Liu, "Adaptive image-based trajectory tracking control of wheeled mobile robots with an uncalibrated fixed camera," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 6, pp. 2266–2282, Nov. 2015.

[4] J. Li, P. Miao, L. Shao, H. Liu, and X. Chen, "Trajectory tracking control of photovoltaic cleaning robot based on Lyapunov theory and Barbalat lemma," in Proceedings of Chinese Control And Decision Conference (CCDC), 2018, pp. 2705-2709.

[5] C. Jun and W. Lin, "Track tracking of double joint robot based on sliding mode control," in Proceedings of 3rd International Conference on Information Systems and Computer Aided Education (ICISCAE), 2020, pp. 626–631.

[6] S. Zihao, W. Bin, and Z. Ting, "Trajectory tracking control of a spherical robot based on adaptive PID algorithm," in Proceedings of Chinese Control And Decision Conference (CCDC), 2019, pp. 5171–5175.

[7] X. Gao, R. Gao, P. Liang, Q. Zhang, R. Deng, and W. Zhu, "A hybrid tracking control strategy for nonholonomic wheeled mobile robot incorporating deep reinforcement learning approach," *IEEE Access*, vol. 9, pp. 15592–15602, 2021.

[8] L. Jin and Y. Zhang, "Discrete-time Zhang neural network of $O(\tau^3)$ pattern for time-varying matrix pseudoinversion with application to manipulator motion generation," *Neurocomputing*, vol. 142, pp. 165–173, Oct. 2014.

[9] J. Li, Y. Zhang, and M. Mao, "General square-pattern discretization formulas via second-order derivative elimination for zeroing neural network illustrated by future optimization," *IEEE Transactions on Neural Networks and Learning System*, vol. 30, no. 3, pp. 891–901, 2019.

[10] J. Li, M. Mao, F. Uhlig, and Y. Zhang, "Z-type neural-dynamics for time-varying nonlinear optimization under a linear equality constraint with robot application," *Journal of Computational and Applied Mathematics*, vol. 327, pp. 155–166, 2018.

[11] J. Li, Y. Zhang, and M. Mao, "Continuous and discrete zeroing neural network for different-level dynamic linear system with robot manipulator control," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 50, pp. 4633–4642, 2020.

[12] Y. Shi and Y. Zhang, "Solving future equation systems using integral-type error function and using twice ZNN formula with disturbances suppressed," *Journal of The Franklin Institute*, vol. 356, pp. 2130–2152, Mar. 2019.

[13] J. Li, Y. Zhang, S. Li, and M. Mao, "New discretization-formula-based zeroing dynamics for real-time tracking control of serial and parallel manipulators," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 8, pp. 3416–3425, Oug. 2018.

[14] J. Li, X. Zhu, Y. Shi, J. Wang, and H. Guo. "Real-time robot manipulator tracking control as multilayered time-varying problem," *Applied Mathematical Modelling*, vol. 96, pp. 355–366, 2021.

[15] H. Xuan, X. Zhu, J. Li, H. Guo, and Y. Li, "General third-order-accuracy formulas for time discretization applied to time-varying optimization," *IEEE Access*, vol. 8, pp. 224235–224245.

[16] D. F. Griffiths and D. J. Higham, *Numerical Methods for Ordinary Differential Equations: Initial Value Problems*. London, UK: Springer, 2010.