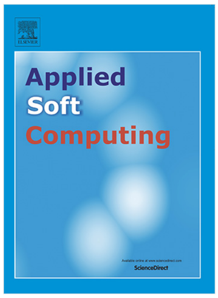
Journal Pre-proof



Direct derivation scheme of DT-RNN algorithm for discrete time-variant matrix pseudo-inversion with application to robotic manipulator

Yang Shi, Wenhan Zhao, Shuai Li, Bin Li, Xiaobing Sun

PII: S1568-4946(22)00910-3
DOI: https://doi.org/10.1016/j.asoc.2022.109861
Reference: ASOC 109861

To appear in: *Applied Soft Computing*

Received date : 23 May 2022
Revised date : 17 October 2022
Accepted date : 18 November 2022

Please cite this article as: Y. Shi, W. Zhao, S. Li et al., Direct derivation scheme of DT-RNN algorithm for discrete time-variant matrix pseudo-inversion with application to robotic manipulator, *Applied Soft Computing* (2022), doi: https://doi.org/10.1016/j.asoc.2022.109861.

Direct derivation scheme of DT-RNN algorithm for discrete time-variant matrix pseudo-inversion with application to robotic manipulator

Yang Shi[a,b,*], Wenhan Zhao[a,b], Shuai Li[c,*], Bin Li[a,b], Xiaobing Sun[a,b]

[a]*School of Information Engineering, Yangzhou University, Yangzhou 225127, China*
[b]*Jiangsu Province Engineering Research Center of Knowledge Management and Intelligent Service, Yangzhou University, Yangzhou 225127, China*
[c]*College of Engineering, Swansea University, Fabian Way, Swansea, UK*

Abstract

The improvement of recurrent neural network (RNN) algorithms is one of target of many researchers, and these algorithms are wieldy used to solve time-variant problems in a variety of domains. A novel direct derivation scheme of discrete time-variant RNN (DT-RNN) algorithm for addressing discrete time-variant matrix pseudo-inversion is discussed in this paper. To be more specific, firstly, a DT-RNN algorithm mathematically founded on the second-order Taylor expansion is proposed for dealing with discrete time-variant matrix pseudo-inversion, and it does not require the theoretical support of continuous time-variant RNN (CT-RNN) algorithm. Secondly, the results of theoretical analyses of the proposed DT-RNN algorithm are also presented in this paper. These results demonstrate that the novel DT-RNN algorithm has remarkable computing performance. The efficiency and applicability of the DT-RNN algorithm have been verified through one numerical experiment example and two robotic manipulator experiments.

Keywords: Direct derivation scheme, Discrete time-variant recurrent neural network (DT-RNN), Discrete time-variant matrix pseudo-inversion, Second-order Taylor expansion, Robotic manipulator.

*Corresponding author.
E-mail Addresses: shiy@yzu.edu.cn; shuaili@ieee.org

1. Introduction

For a matrix P, if there exists a matrix Q such that $PQ = QP = I$, in which I is an identity matrix of the same dimensions as P and Q, we can say that P is an invertible matrix, i.e., P is invertible; and say that Q is the inversion matrix of P, i.e., $Q = P^{-1}$. For example, we set up a system of linear equations as follows [1]:

$$\left.\begin{array}{c} q_{11}x_1 + q_{12}x_2 + \cdots + q_{1n}x_n = p_1 \\ q_{21}x_1 + q_{22}x_2 + \cdots + q_{2n}x_n = p_2 \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ q_{m1}x_1 + q_{m2}x_2 + \cdots + q_{mn}x_n = p_m \end{array}\right\},$$

in which x_1, x_2, ..., x_n represent the unknown quantities, q_{ij} $(1 \leq i \leq m, 1 \leq j \leq n)$ is the coefficient of above system of equations, and p_i $(1 \leq i \leq m)$ is the constant term. The coefficients and constant terms are all arbitrary complex numbers or elements of a domain. Then, we simplify the above system of equations into a form of the matrix and vector as

$$Q\mathbf{x} = \mathbf{p},$$

in which Q is a $m \times n$ matrix, $\mathbf{x}$ is an unknown n-dimensional vector, and $\mathbf{p}$ is a m-dimensional constant vector. Immediately after, we assume that

1. $m = n$.
2. Q is invertible (i.e., Q is a square and full rank matrix).

Thus, we have $\mathbf{x} = Q^{-1}\mathbf{p}$, which is a solution to the system of equations, in which Q^{-1} is a $m \times m$ $(n \times n)$ matrix satisfying $Q^{-1}Q = QQ^{-1} = I$. Nevertheless, when $m \neq n$ or Q is not invertible, the above assumption becomes meaningless. Because at this point, the Q^{-1} evidently does not exist. Consequently, as a natural extension of the concept of inversion, the pseudo-inversion becomes particularly important. Generally speaking, it not only occupies an important place in computational mathematics [2, 3], but also has a wide range of applications in other fields, for example, control theorey [4, 5], robotics [6, 7], machine learning [8, 9] and pattern recognition [10]. At the same time, many algorithms or models for pseudo-inversion solutions have been proposed and improved one after another [11–13]. They is beneficial to later researches; for instance, in [11], the authors presented a technique for solving the weighted Moore-Penrose inversion of rational or polynomial matrices with one variable; in [12], on the basis of Newton iteration, an optimization algorithm was proposed, which can be used to deal with Toeplitz matrix pseudo-inversion; in [13], for improving the accuracy of classifiers, author used pseudo-inversion sample covariance matrix. Nevertheless, most of these proposed algorithms are introduced to solve static matrix pseudo-inversion, i.e., time-invariant matrix pseudo-inversion [12, 14]. When we directly use these algorithms to solve the time-variant matrix pseudo-inversion, they become inefficient and limited due to the omission of the key information of the time derivative, which lacks application value to some extent in real life.

Consistently, many algorithms for solving time-variant matrix pseudo-inversion have been presented. In the past decades, the artificial intelligence has become a key force of the industrial development. Neural network, as a fundamental research direction in the field of artificial intelligence, gradually attracts attention of researchers [15–21]. With the development of relevant technologies, neural networks have become an important mathematical tool for handling various domain problems, such as distributed parallel information processing. Because of the research and promotion of previous researchers, recurrent neural networks (RNNs), as a branch of neural networks and with their powerful advantages [22–26], have developed in leaps and bounds. Afterwards, a new class of RNN algorithm is presented by Zhang and Ge. This is a continuous time-variant RNN (CT-RNN) algorithm for solving time-variant matrix inversion [27], which shows global and exponential convergence and is characterized by implicit dynamics. However, general speaking, continuous time-variant algorithms may be more difficult to implement in industry than discrete time-variant algorithms, such as on the digital circuits and on the digital computers.

Many researchers proposed improved RNN algorithms for solving discrete time-variant matrix pseudo-inversion after Zhang et al. introduced the algorithm [27], such as Wei et al. [28], Guo et al. [29], Liao et al. [30] and Petković et al. [31]. Besides, the study and processing of some mathematical problems can also be realized by RNNs [32–37].

However, when researchers solve such discrete time-variant problems, firstly, they need present discrete time-variant problem in the continuous time-variant form. Then they define the error function of the continuous time-variant problem through introducing the RNN design formula, immediately following that, the CT-RNN algorithm is developed. Whereafter, the CT-RNN algorithm in the discrete time-variant form is shown. Finally, to

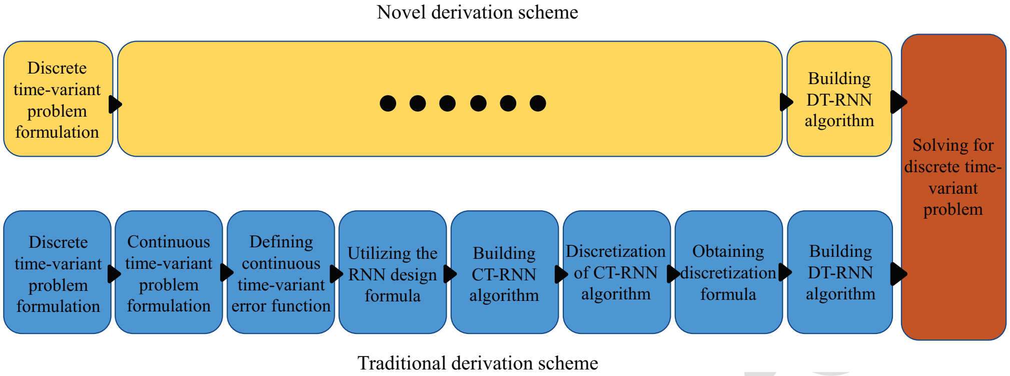


Figure 1: Comparison between novel derivation scheme and traditional derivation scheme.

solve the discrete time-variant problems, the corresponding discrete time-variant recurrent neural network (DT-RNN) algorithm is established. It is appropriate to note here that the above solving process is a indirect derivation scheme of DT-RNN algorithm, and the main procedure is shown in Fig. 1. Evidently, in this whole derivation scheme, researchers ignore the fact that repeated conversion between the discrete and continuous environment requires additional computational time, which may significantly reduce the real-time performance of algorithm.

Under these conditions, we need to find a straightforward and effective derivation scheme. By investigating second-order Taylor expansion, a novel direct derivation scheme of DT-RNN algorithm is proposed, which means that the derivation process nearly skips the continuous time-variant environment and the target problem can be solved in the discrete time-variant environment by a direct and efficient method. As shown in Fig. 1, compared with the traditional derivation scheme, the new direct derivation scheme omits some intermediate solving procedures, which can effectively improve the efficiency of solving the discrete time-variant matrix pseudo-inversion [38, 39].

Based on the above analyses, in this paper, a novel DT-RNN algorithm is proposed, which is founded on the direct derivation scheme. The remaining work is mainly divided into five parts. In the second section, we formulate the discrete time-variant matrix pseudo-inversion problem and show an application preliminary of robotic manipulator. The third section introduces the DT-RNN algorithm for handling discrete time-variant matrix pseudo-inversion, which is mathematically built on the second-order Taylor expan-

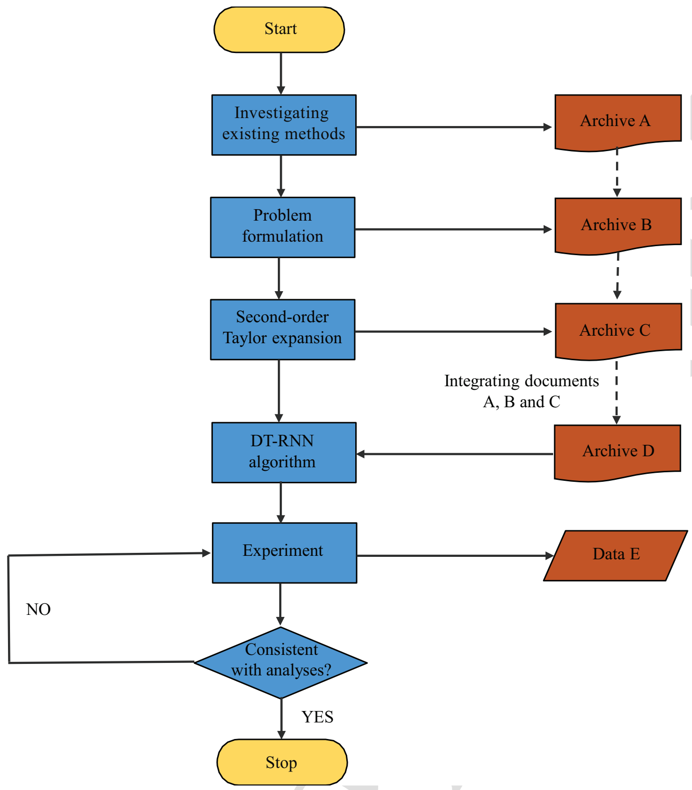


Figure 2: Workflow diagram of entire research work.

sion. Moreover, the comparison with other method and theoretical analyses of the DT-RNN algorithm are presented, and it is shown that the proposed algorithm has excellent convergence.

The validity of the theoretical analyses is tested in the fourth section, which includes a numerical experiment and corresponding comparison results. In addition, benefitted from [40–42], in the fifth section, we apply two robotic manipulator experiments to further demonstrate efficiency and applicability of the proposed DT-RNN algorithm, and the sixth section concludes the paper. The main contributions of this paper can be summarized by the following three items.

1. A direct derivation scheme founded on the second-order Taylor expansion is proposed to establish DT-RNN algorithm for solving discrete time-variant matrix pseudo-inversion, and the solving process no longer requires the theoretical support of continuous time-variant algorithm.

To the authors' knowledge, the proposed algorithm is quite different from the previous DT-RNN algorithms.

2. For the proposed DT-RNN algorithm, theoretical analyses have shown that such an algorithm is exactly convergent when it is exploited to solve discrete time-variant matrix pseudo-inversion.
3. The effectiveness of DT-RNN algorithm for solving discrete time-variant matrix pseudo-inversion is proved using numerical experiment results. In addition, two application experiments of robotic manipulator are shown to further validate the efficiency and practicability of the DT-RNN algorithm.

In addition, for improving the readability of entire research work, a workflow diagram is presented in Fig. 2.

2. Problem formulation and preliminary

In this section, the discrete time-variant matrix pseudo-inversion problem is formulated firstly. Then, an application preliminary of robotic manipulator is introduced.

2.1. Problem formulation

To begin with, benefitted from [7], we express the following definition:
Definition 1: For a matrix $Q \in \mathbb{R}^{m\times n}$, if $X \in \mathbb{R}^{n\times m}$ meets all the four Penrose equations below:

$$\begin{array}{ll} QXQ = Q, & XQX = X, \\ (QX)^{\mathrm{T}} = QX, & (XQ)^{\mathrm{T}} = XQ, \end{array}$$

in which $(\cdot)^{\mathrm{T}}$ represents the execution of matrix transpose, X is the pseudo-inversion of the matrix Q, which is represented by Q^{+}. Note that, generally speaking, Q^{+} is unique in this paper.

Consider a matrix $Q \in \mathbb{R}^{m\times n}$ is full rank, i.e., $\mathrm{rank}(Q) = \min\{m, n\}$, we can obtain the pseudo-inversion of Q by the lemma [32] below:
Lemma 1 : Consider a matrix $Q \in \mathbb{R}^{m\times n}$, if $\mathrm{rank}(Q) = \min\{m, n\}$, the unique pseudo-inversion Q^{+} can be expressed as

$$Q^{+} = \begin{cases} Q^{\mathrm{T}}(QQ^{\mathrm{T}})^{-1}, m < n, \\ Q^{-1} \qquad\qquad, m = n, \\ (Q^{\mathrm{T}}Q)^{-1}Q^{\mathrm{T}}, m > n. \end{cases} \tag{1}$$

The above three rows of formula represent the right pseudo-inversion of matrix Q, the inversion when Q is a square matrix and the left pseudo-inversion, respectively. Here, all the matrices in this paper are defined as full-rank matrices, and we only consider $m < n$.

Let us formulate the discrete time-variant matrix pseudo-inversion problem as

$$Q(t_k)X(t_k) = I_m \in \mathbb{R}^{m\times m}.$$

Note that we use the previous or current data at each computational time interval $[t_k, t_{k+1})$ in order to obtain the next data at the instant of time t_{k+1}. Therefore, actually, in the discrete time-variant environment, instead of $Q(t_k)X(t_k) = I_m$, we can express the target problem as

$$Q(t_{k+1})X(t_{k+1}) = I_m \in \mathbb{R}^{m\times m}, \tag{2}$$

in which $Q(t_{k+1}) \in \mathbb{R}^{m\times n}$, $X(t_{k+1}) \in \mathbb{R}^{n\times m}$, and $I_m \in \mathbb{R}^{m\times m}$ represent the coefficient matrix, the unsolved matrix pseudo-inversion, and the identity matrix, respectively. Obtaining time-variant solution $X(t_{k+1})$ is the purpose of this work, and $X(t_{k+1})$ holds true at any instant of time $[t_k, t_{k+1}) \subseteq [t_0, t_{\mathrm{f}}] \subseteq [0, +\infty)$, where k represents the number of updating.

2.2. Preliminary of robotic manipulator

A simplified robotic manipulator is considered in this subsection. For one robotic manipulator tracking control problem [43, 44], there exists a connection between the joint-angle vector

$$\theta(t_{k+1}) = [\theta_1(t_{k+1}); \theta_2(t_{k+1}); \theta_3(t_{k+1})] \in \mathbb{R}^3$$

and the Cartesian position vector $\mathbf{r}(t_{k+1}) \in \mathbb{R}^2$ of the end-effector

$$\mathbf{r}(t_{k+1}) = \phi(\theta(t_{k+1}), t_{k+1}).$$

For a specific manipulator, $\phi(\cdot)$ represents a nonlinear forward kinematics mapping function and we have known its parameters and structure [38]. Moreover, there exists a linear relationship between the joint velocity and the end-effector Cartesian velocity which can be presented as

$$\dot{\mathbf{r}}(t_{k+1}) = J(t_{k+1})\dot{\theta}(t_{k+1}),$$

where $J(t_{k+1})$ is the Jacobian matrix. For above equation, we can obtain

$$\dot{\theta}(t_{k+1}) = J^{+}(t_{k+1})\dot{\mathbf{r}}(t_{k+1}).$$

Remark 1. The related control rule can handle the robotic manipulator to accomplish the tracking control task, and it can also be solved and characterized by joint-angle or joint-velocity variables after establishing Jacobian matrix. Without a doubt, the matrix pseudo-inversion is linked to current development of information science. Finding a novel and high-efficiency approach for matrix pseudo-inversion is critical. Evidently, we need to obtain $J^{+}(t_{k+1})$ for solving $\dot{\theta}(t_{k+1})$ at any instant of time $[t_k, t_{k+1}) \subseteq [t_0, t_{\mathrm{f}}] \subseteq [0, +\infty)$.

3. Direct derivation scheme of DT-RNN algorithm

An innovative DT-RNN algorithm and corresponding theoretical analyses are investigated and studied in this section.

3.1. DT-RNN algorithm

To begin with, we present a theorem for introducing DT-RNN algorithm.

Theorem 1. *The DT-RNN algorithm is presented as*

$$\begin{aligned} X(t_{k+1}) = &- X(t_k)[(1-\omega)(Q(t_k)X(t_k) - I_m)] - \xi X(t_k)\dot{Q}(t_k)X(t_k) \\ &+ X(t_k) \end{aligned} \tag{3}$$

with truncation error $\mathbf{O}(\xi^2)$. *Here,* $X(t_{k+1})$ *represents the matrix pseudo-inversion to be solved at the instant of time* $t = (k+1)\xi$*;* ω *represents a design parameter;* $Q(t_k)$ *represents a discrete time-variant non-square matrix with respect to* t_k*;* $\dot{Q}(t_k)$ *represents the time derivative of* $Q(t_k)$ *at* t_k*;* I_m *represents identity matrix; the sampling period is represented by the variable* ξ*, where* $\xi > 0$.

Proof. Firstly, let us define matrix-valued error functions as

$$E(X(t_{k+1}), t_{k+1}) = Q(t_{k+1})X(t_{k+1}) - I_m \in \mathbb{R}^{m\times m} \tag{4}$$

and

$$E(X(t_k), t_k) = Q(t_k)X(t_k) - I_m \in \mathbb{R}^{m\times m}. \tag{5}$$

Then, based on (4) and (5), we have

$$\begin{aligned} E(X(t_{k+1}),&t_{k+1}) - E(X(t_k), t_k) \\ &= (Q(t_{k+1})X(t_{k+1}) - I_m) - (Q(t_k)X(t_k) - I_m) \\ &= Q(t_{k+1})X(t_{k+1}) - Q(t_k)X(t_k), \end{aligned} \tag{6}$$

when k is large enough, we set $E(X(t_{k+1}), t_{k+1}) = \omega E(X(t_k), t_k)$, where ω represents a design parameter. In view of the second-order Taylor expansion [38], we further have

$$\begin{aligned} Q(t_{k+1})X(t_{k+1}) &= Q(t_k)X(t_k) + Q(t_k)(X(t_{k+1}) - X(t_k)) \\ &\quad + (t_{k+1} - t_k)\dot{Q}(t_k)X(t_k) + \mathbf{O}(\xi^2) \\ &= Q(t_k)X(t_k) + Q(t_k)(X(t_{k+1}) - X(t_k)) \\ &\quad + \xi\dot{Q}(t_k)X(t_k) + \mathbf{O}(\xi^2). \end{aligned} \tag{7}$$

Here $\mathbf{O}(\xi^2)$ represents a matrix of order $O(\xi^2)$ for each of these elements, we can further obtain the following formula by substituting (7) into (6) and dropping the $\mathbf{O}(\xi^2)$ above:

$$(\omega - 1)(Q(t_k)X(t_k) - I_m) = Q(t_k)(X(t_{k+1}) - X(t_k)) + \xi\dot{Q}(t_k)X(t_k).$$

Next, the DT-RNN algorithm (3) can be represented by the following formula:

$$X(t_{k+1}) - X(t_k) = -X(t_k)[(1-\omega)(Q(t_k)X(t_k) - I_m)] - \xi X(t_k)\dot{Q}(t_k)X(t_k),$$

the above formula can be further transformed into

$$X(t_{k+1}) = -X(t_k)[(1-\omega)(Q(t_k)X(t_k) - I_m)] - \xi X(t_k)\dot{Q}(t_k)X(t_k) + X(t_k).$$

The proof is thus completed. □

For further comparison, we develop and present the existing Newton iteration [38, 40] as

$$X(t_{k+1}) = -X(t_k)(Q(t_k)X(t_k) - I_m) + X(t_k)$$

for solving discrete time-variant matrix pseudo-inversion. Generally speaking, the Newton iteration is a very common method for mathematical problems. Note that the Newton iteration can not process discrete time-variant problem with a higher computational precision.

3.2. Theoretical analyses and results

The proposed DT-RNN algorithm (3) is actually built to solve discrete time-variant problems, instead of solving continuous time-variant problems, which means that the derivative process of the proposed algorithm (3) does not need the theoretical support provided by continuous time-variant algorithm. Furthermore, the following theorems are presented in this subsection to show theoretical analyses and results of the DT-RNN algorithm (3) for solving discrete time-variant matrix pseudo-inversion (2).

Theorem 2. *The proposed DT-RNN algorithm (3) converges to the truncation error of order* $\mathbf{O}(\xi^2)$ *when the boundary of all partial derivatives of* $Q(t_{k+1})X(t_{k+1})$ *is continuous.*

Proof. As presented by Theorem 1, we have the following formula:

$$\begin{aligned}E(X(t_{k+1}),&t_{k+1}) - E(X(t_k), t_k)\\ &= (Q(t_{k+1})X(t_{k+1}) - I_m) - (Q(t_k)X(t_k) - I_m)\\ &= Q(t_{k+1})X(t_{k+1}) - Q(t_k)X(t_k).\end{aligned}$$

On the one hand, we define $Q(t_{k+1})X(t_{k+1})$ as one binary function with two variables (X and t); on the other hand, for $X(t_{k+1})$ and t_{k+1}, they are in close proximity to any fixed point $X(t_k)$ and t_k. It should be noted here that $Q(t_{k+1})$ is a variable with respect to t_{k+1}, and is affected by it. If all partial derivatives of $Q(t_{k+1})X(t_{k+1})$ exist, we have

$$\begin{aligned}Q(t_{k+1})X(t_{k+1}) = Q(t_k)X(t_k) &+ Q(t_k)(X(t_{k+1}) - X(t_k))\\ &+ (t_{k+1} - t_k)\dot{Q}(t_k)X(t_k) + \|\sigma\|_{\mathrm{F}},\end{aligned} \tag{8}$$

where

$$\|\sigma\|_{\mathrm{F}} \leq \frac{\lambda}{2}(\|X(t_{k+1}) - X(t_k)\|_{\mathrm{F}} + |t_{k+1} - t_k|)^2,$$

and the partial derivatives of $Q(t_{k+1})X(t_{k+1})$ all have a constant border, which is represented by λ. The sampling period is $\xi = t_{k+1} - t_k$, and $\xi > 0$. By using Euler discretization formula [38], we have

$$\dot{X}(t_k) = \frac{1}{\xi}X(t_{k+1}) - \frac{1}{\xi}X(t_k) + \mathbf{O}(\xi),$$

which can be rewritten as

$$X(t_{k+1}) - X(t_k) = \xi\dot{X}(t_k) + \mathbf{O}(\xi^2),$$

and we have

$$\begin{aligned}\|\sigma\|_{\mathrm{F}} \leq & \frac{\lambda}{2}\left(\left\|\xi\dot{X}(t_k)+\mathbf{O}\left(\xi^2\right)\right\|_{\mathrm{F}}+\xi\right)^2 \\ = & \frac{\lambda}{2}\left(\xi\left(\left\|\dot{X}(t_k)+\mathbf{O}\left(\xi\right)\right\|_{\mathrm{F}}+1\right)\right)^2 \\ = & \frac{\lambda}{2}\xi^2\left(\left\|\dot{X}(t_k)+\mathbf{O}\left(\xi\right)\right\|_{\mathrm{F}}+1\right)^2 \\ = & \frac{\lambda N}{2}\xi^2 \\ = & \mathbf{O}(\xi^2),\end{aligned}$$

in which $N = \left(\left\|\dot{X}(t_k)+\mathbf{O}(\xi)\right\|_{\mathrm{F}}+1\right)^2$. Then, equation (8) can be rewritten as

$$\begin{aligned}Q(t_{k+1})X(t_{k+1}) = & Q(t_k)X(t_k) + Q(t_k)(X(t_{k+1}) - X(t_k)) \\ & + (t_{k+1} - t_k)\dot{Q}(t_k)X(t_k) + \mathbf{O}(\xi^2),\end{aligned}$$

and we further obtain

$$\begin{aligned}Q(t_k)(X(t_{k+1}) - X(t_k)) = & Q(t_{k+1})X(t_{k+1}) - Q(t_k)X(t_k) - \xi\dot{Q}(t_k)X(t_k) \\ & + \mathbf{O}(\xi^2).\end{aligned}$$

According to pervious section, we have

$$\begin{aligned}X(t_{k+1}) = & -X(t_k)[(1-\omega)(Q(t_k)X(t_k) - I_m)] - \xi X(t_k)\dot{Q}(t_k)X(t_k) \\ & + X(t_k) + X(t_k)\mathbf{O}(\xi^2),\end{aligned}$$

where $X(t_k)\mathbf{O}(\xi^2) = \mathbf{O}(\xi^2)$, and finally we have

$$\begin{aligned}X(t_{k+1}) = & -X(t_k)[(1-\omega)(Q(t_k)X(t_k) - I_m)] - \xi X(t_k)\dot{Q}(t_k)X(t_k) \\ & + X(t_k) + \mathbf{O}(\xi^2).\end{aligned}$$

It is clear from the aforementioned analyses that the DT-RNN algorithm (3) converges to the truncation error of order $\mathbf{O}(\xi^2)$. The proof is thus completed. □

Theorem 3. *The DT-RNN algorithm (3) has consistence and zero-stability.*

Proof. Refer to Appendix for details. □

Algorithm: Numerical Implementation of DT-RNN Algorithm (3)
1. **Input:** Non-square and discrete time-variant matrix $Q(t_k) \in \mathbb{R}^{m\times n}$;
2. **Input:** Sampling period ξ, design parameter ω, identity matrix I_m and instant of time $[t_k, t_{k+1}) \subseteq [t_0, t_{\mathrm{f}}] \subseteq [0, +\infty)$;
3. **Initialize:** $X(t_1), Q(t_0)$ and $\dot{Q}(t_0)$;
4. **Calculate:** $X(t_2), Q(t_1), \dot{Q}(t_1)$, $X(t_3), Q(t_2)$ and $\dot{Q}(t_2)$;
5. **For:** $t_3 \to t_{\mathrm{f}}$
6. **Calculate:** $Q(t_k)$ and $\dot{Q}(t_k)$;
7. **Calculate:** $X(t_{k+1})$ via the DT-RNN algorithm (3);
8. **Output:** Residual error of the DT-RNN algorithm (3) via $\\|E(t=(k+1)\xi)\\|_{\mathrm{F}}$;
9. **End for:**
10. **Stop:** Numerical Implementation of DT-RNN algorithm (3) is completed.

Theorem 4. *For handling discrete-time matrix pseudo-inversion (2), the steady-state residual error of DT-RNN algorithm (3) changes in an $O(\xi^2)$ pattern with k being large enough.*

Proof. Firstly, benefitted from the work of Jin and Guo et al. [7, 24], we have

$$\lim_{k\to\infty} \|E(t=(k+1)\xi)\|_{\mathrm{F}} = \lim_{k\to\infty} \|Q(t_{k+1})X(t_{k+1}) - I_m\|_{\mathrm{F}},$$

secondly, we define one theoretical solution for solving discrete time-variant matrix pseudo-inversion (2), that is,

$$X^*(t_{k+1}) = X^*(t=(k+1)\xi) \in \mathbb{R}^{n\times m}.$$

According to the aforementioned theorems, $X(t_{k+1}) = X^*(t_{k+1}) + \mathbf{O}(\xi^2)$ (k is large enough), then we can obtain

$$\begin{aligned}\|Q(t_{k+1})X(t_{k+1}) - I_m\|_{\mathrm{F}} &= \left\|Q(t_{k+1})(X^*(t_{k+1}) + \mathbf{O}(\xi^2)) - I_m\right\|_{\mathrm{F}} \\ &= \left\|Q(t_{k+1})X^*(t_{k+1}) - I_m + Q(t_{k+1})\mathbf{O}(\xi^2)\right\|_{\mathrm{F}}.\end{aligned}$$

Since $Q(t_{k+1})X^*(t_{+1}) - I_m = \mathbf{0}$, we can rewrite the above equation as

$$\|Q(t_{k+1})X(t_{k+1}) - I_m\|_{\mathrm{F}} = \left\|Q(t_{k+1})\mathbf{O}(\xi^2)\right\|_{\mathrm{F}} = O(\xi^2).$$

In summary, for handling discrete-time matrix pseudo-inversion (2), the steady-state residual error of DT-RNN algorithm (3) changes in an $O(\xi^2)$ pattern with k being large enough. The proof is thus completed. □

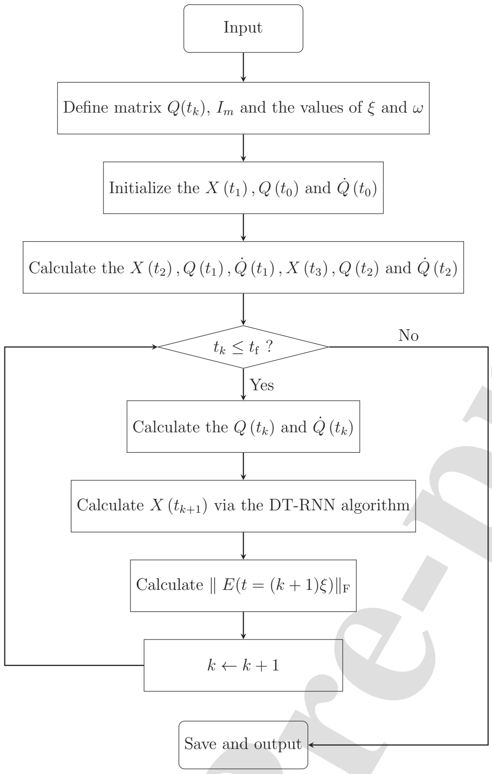


Figure 3: Flow chart of numerical implementation of proposed DT-RNN algorithm (3).

Remark 2. Theorems 2, 3 and 4 prove that the DT-RNN algorithm (3) not only has excellent convergence in the computational time interval $[t_k, t_{k+1}) \subseteq [t_0, t_{\mathrm{f}}] \subseteq [0, +\infty)$, but also shows the computational effectiveness simultaneously. Compared with previous DT-RNN algorithms, it can been seen as a significant improvement. The reason is that, for solving discrete time-variant problems, by using traditional DT-RNN algorithm, we need to perform several complex intermediate procedures.

4. Numerical experiment and verifications

In this section, we visually present the authenticity and validity of the proposed DT-RNN algorithm (3) through a numerical experiment and corresponding comparison results. Firstly, for the convenience of presentation and readability, the numerical implementation process of DT-RNN algorithm (3)

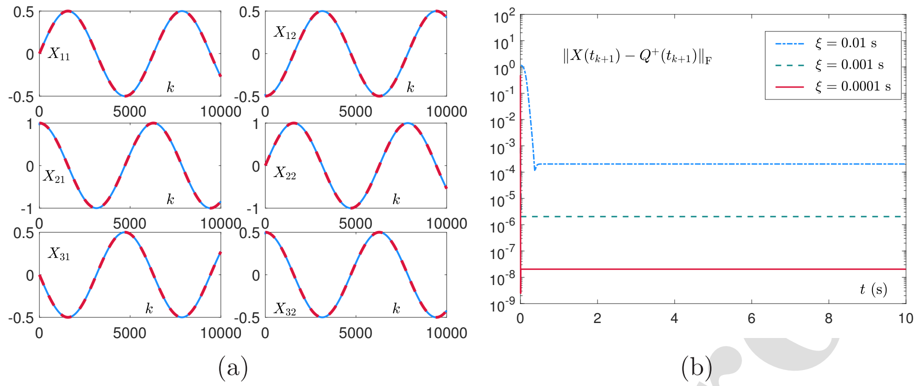


Figure 4: Numerical experimental results of proposed DT-RNN algorithm (3) to solve example 1 in Section 4 with $t = k\xi$. (a) State trajectories of DT-RNN algorithm (3) using $\xi = 0.001$ s and $\omega = 0.7$, where horizontal axis represents number of updates and vertical axis represents errors of two solutions. (b) Using different values of ξ and $\omega = 0.7$, where horizontal axis represents range of time and vertical axis represents residual errors.

Table 1: Maximum steady-state residual errors of DT-RNN algorithm in numerical experiment example 1 with different sampling periods.

	2 s	6 s	8 s	10 s
$\xi = 0.01\ s$	2.04×10^{-4}	2.04×10^{-4}	2.04×10^{-4}	2.04×10^{-4}
$\xi = 0.001\ s$	2.041×10^{-6}	2.041×10^{-6}	2.041×10^{-6}	2.041×10^{-6}
$\xi = 0.0001\ s$	2.041×10^{-8}	2.041×10^{-8}	2.041×10^{-8}	2.041×10^{-8}

is summarized and presented, and corresponding flow chart is also shown in Fig. 3.2.

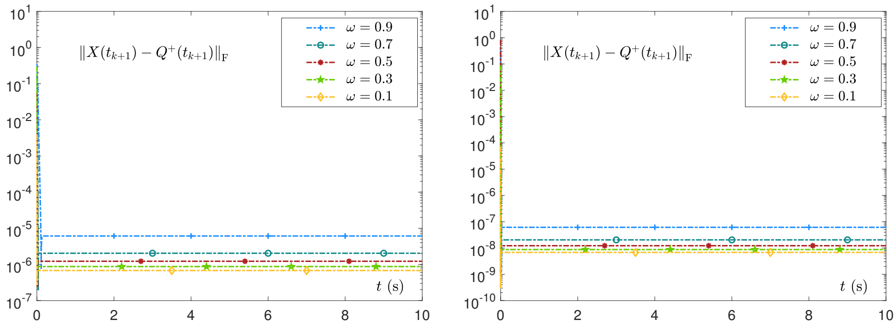
4.1. Example 1

Here, we present the following discrete time-variant matrix as an example to authenticate the efficiency of the DT-RNN algorithm (3):

$$Q(t_k) = \begin{bmatrix} \sin(t_k) & \cos(t_k) & -\sin(t_k) \\ -\cos(t_k) & \sin(t_k) & \cos(t_k) \end{bmatrix} \in \mathbb{R}^{2\times 3}, \tag{9}$$

and the theoretical solution can be presented as

$$X^*(t_k) = \begin{bmatrix} 1/2\sin(t_k) & -1/2\cos(t_k) \\ \cos(t_k) & \sin(t_k) \\ -1/2\sin(t_k) & 1/2\cos(t_k) \end{bmatrix}.$$



(a) (b)

Figure 5: Numerical experimental results of proposed DT-RNN algorithm (3) to solve example 1 in Section 4 with $t = k\xi$. (a) Using different values of ω and $\xi = 0.001$ s, where horizontal axis represents range of time and vertical axis represents residual errors. (b) Using different values of ω and $\xi = 0.0001$ s, where horizontal axis represents range of time and vertical axis represents residual errors.

Table 2: Maximum steady-state residual errors of DT-RNN algorithm in numerical experiment example 1 using different values of ω and $\xi = 0.001$ s.

	2 s	6 s	8 s	10 s
$\omega = 0.9$	6.123×10^{-6}	6.123×10^{-6}	6.123×10^{-6}	6.123×10^{-6}
$\omega = 0.7$	2.041×10^{-6}	2.041×10^{-6}	2.041×10^{-6}	2.041×10^{-6}
$\omega = 0.5$	1.225×10^{-6}	1.225×10^{-6}	1.225×10^{-6}	1.225×10^{-6}
$\omega = 0.3$	8.748×10^{-7}	8.748×10^{-7}	8.748×10^{-7}	8.748×10^{-7}
$\omega = 0.1$	6.804×10^{-7}	6.804×10^{-7}	6.804×10^{-7}	6.804×10^{-7}

Then we obtain matrix pseudo-inversion of (9) through the DT-RNN algorithm (3). It is shown in Fig. 4 that the numerical experiment results are generated by the DT-RNN algorithm (3) when solving example 1. Specifically, starting with one random value, Fig. 4(a) illustrates that the actual experimental results (represented by the blue dashed curve in the figure) generated by the DT-RNN algorithm (3) using sampling period $\xi = 0.001$ s and $\omega = 0.7$ can be well fitted the theoretical solution (represented by the red dash curve in the figure). Then, in Fig. 4(b) and Table 1, we can see that the residual errors synthesized by DT-RNN algorithm (3) with $\omega = 0.7$ and different values of sampling period ξ, which shows that the residual errors change in an $O(\xi^2)$ pattern approximatively, in other words, when the sampling period value ξ decreases tenfold, the residual error synthesized by DT-RNN algorithm (3) reduces hundredfold, which complies with the theo-

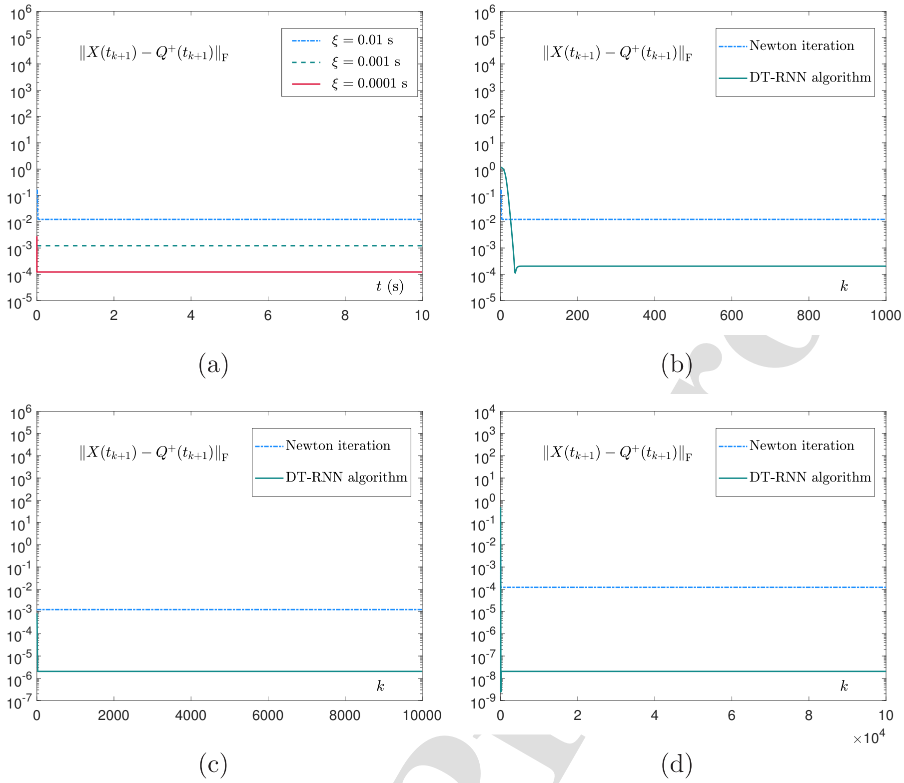


Figure 6: Numerical experimental results of Newton iteration and comparisons between DT-RNN algorithm and Newton iteration with $t = k\xi$. (a) Newton iteration using different values of ξ and $\omega = 0.7$. (b) Residual errors using $\xi = 0.01$ s. (c) Residual errors using $\xi = 0.001$ s. (d) Residual errors using $\xi = 0.0001$ s.

retical analyses of Theorem 4.

To further investigate the effect of change of design parameter ω on the computational performance of the proposed algorithm, we select five different values from the range greater than 0 to less than 1 as variables. As shown in Fig. 5, Table 2 and Table 3, the residual errors synthesized by DT-RNN algorithm (3) with two different values of sampling period ξ can quickly converge to a relatively small value, respectively, which authenticates the accuracy and convergence of DT-RNN algorithm (3). Generally speaking, for the DT-RNN algorithm (3), choosing different values of sampling period ξ can obtain different computational performances. That is, the smaller sampling period ξ value is, the more significant the computational performance improvement

Table 3: Maximum steady-state residual errors of DT-RNN algorithm in numerical experiment example 1 using different values of ω and $\xi = 0.0001$ s.

	2 s	6 s	8 s	10 s
$\omega = 0.9$	6.124×10^{-8}	6.124×10^{-8}	6.124×10^{-8}	6.124×10^{-8}
$\omega = 0.7$	2.041×10^{-8}	2.041×10^{-8}	2.041×10^{-8}	2.041×10^{-8}
$\omega = 0.5$	1.225×10^{-8}	1.225×10^{-8}	1.225×10^{-8}	1.225×10^{-8}
$\omega = 0.3$	8.748×10^{-9}	8.748×10^{-9}	8.748×10^{-9}	8.748×10^{-9}
$\omega = 0.1$	6.804×10^{-9}	6.804×10^{-9}	6.804×10^{-9}	6.804×10^{-9}

Table 4: Position errors of elliptical path example using $\omega = 0.3$ and $\xi = 0.0001$ s.

	5 s	15 s	25 s	30 s
e_X	1.214×10^{-9}	4.982×10^{-9}	4.811×10^{-9}	5.22×10^{-9}
e_Y	5.12×10^{-9}	1.297×10^{-9}	1.155×10^{-9}	5.048×10^{-9}

of the proposed DT-RNN algorithm (3) is.

Furthermore, we present Fig. 6 to show the experimental comparison between DT-RNN algorithm (3) and Newton iteration. Through illustrative figures, the efficiency and superiority of the proposed DT-RNN algorithm have been validated for solving discrete time-variant matrix pseudo-inversion. Here, we can see that the steady-state residual error of the DT-RNN algorithm changes in an $O(\xi^2)$ pattern, and the steady-state residual error of the Newton iteration changes in an $O(\xi)$ pattern.

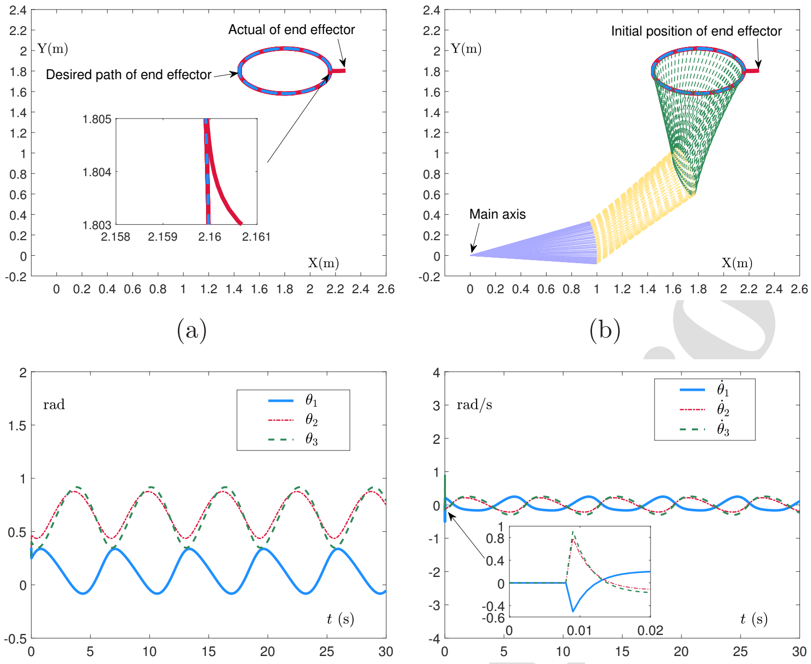
5. Robotic manipulator application

In this section two robotic manipulator experiments are shown to further verified the efficiency and applicability of the DT-RNN algorithm (3).

5.1. Example 1

In this subsection, benefitted from [6, 7, 41, 42], we handle a robotic manipulator to prove potential of the DT-RNN algorithm (3) in practical applications by performing a tracking control task.

Specifically, for such a robotic manipulator, there is a task duration t_d at instant of time $t_k \in [0, t_d]$ while performing a task. Through computing the pseudo-inversion of a discrete time-variant Jacobian matrix, the proposed DT-RNN algorithm (3) is applied to handle a robotic manipulator to draw a designed path. When performing a task, the Jacobian matrix $J \in \mathbb{R}^{2\times 3}$ for



(a) (b)

(c) (d)

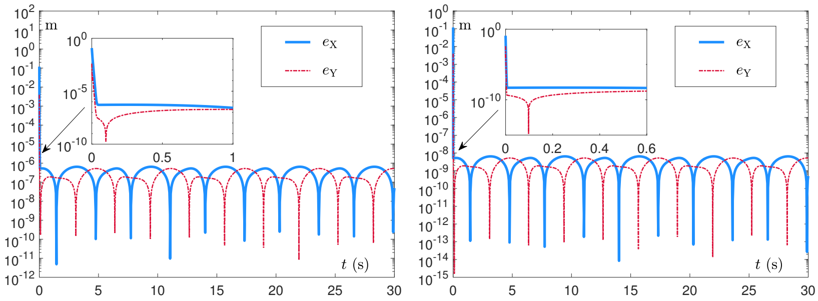
Figure 7: Application of DT-RNN algorithm (3) with $\omega = 0.3$ and sampling period $\xi = 0.001$ s for tracking path (11). (a) Motion trajectory of end-effector, where horizontal axis represents horizontal displacement and vertical axis represents longitudinal displacement. (b) Motion trajectory of whole robotic manipulator, where horizontal axis represents horizontal displacement and vertical axis represents longitudinal displacement. (c) Joint-angle profiles, where horizontal axis represents range of time and vertical axis represents joint-angle change. (d) Joint-velocity profiles, where horizontal axis represents range of time and vertical axis represents joint-velocity change.

such a robotic manipulator is provided as follows:

$$J = \begin{bmatrix} -l_1 s\theta_1 - l_2 s\theta_{12} - l_3 s\theta_{123} & -l_2 s\theta_{12} - l_3 s\theta_{123} & -l_3 s\theta_{123} \\ l_1 c\theta_1 + l_2 c\theta_{12} + l_3 s\theta_{123} & l_2 c\theta_{12} + l_3 s\theta_{123} & l_3 s\theta_{123} \end{bmatrix}, \quad (10)$$

in which $l_1, l_2, and l_3$ represent the length of manipulator rod $1, 2$ and 3, respectively; $s\theta_{123} = \sin(\theta_1 + \theta_2 + \theta_3)$, $c\theta_{123} = \cos(\theta_1 + \theta_2 + \theta_3)$, $s\theta_{12} = \sin(\theta_1 + \theta_2)$ and $c\theta_{12} = \cos(\theta_1 + \theta_2)$. Then the tracking control task is an elliptical path that can be defined as

$$\mathbf{r}(t_k) = \begin{bmatrix} (9/25)\cos(t_k) + 9/5 \\ (11/50)\sin(t_k) + 9/5 \end{bmatrix}. \quad (11)$$



(a) (b)

Figure 8: End-effector position errors. (a) Using $\xi = 0.001$ s, where horizontal axis represents range of time and vertical axis represents end-effector position errors. (b) Using $\xi = 0.0001$ s, where horizontal axis represents range of time and vertical axis represents end-effector position errors.

Table 5: Position errors of cardioid path example using $\omega = 0.3$ and $\xi = 0.0001$ s.

	5 s	15 s	25 s	30 s
e_X	1.159×10^{-8}	7.467×10^{-9}	6.546×10^{-9}	1.194×10^{-8}
e_Y	1.681×10^{-13}	2.691×10^{-9}	4.121×10^{-13}	3.000×10^{-9}

Here, we assign value of sampling period ξ in the proposed DT-RNN algorithm (3) being 0.001 s. In addition to these assignments, we set these parameters: each rod length of robotic manipulator is 1 meter; the initial state of three joints is set to $\theta_0 = [\pi/9; \pi/9; \pi/12]$; the task duration $t_\mathrm{d} = 30$ s. As we can see, Fig. 7(a) shows that the experimental trajectory (denoted by red solid line) is very close to the desired elliptical path (denoted by blue dashed line). Fig. 7(b) represents the simplified diagram of robotic manipulator for drawing an elliptical path. Following that, Fig. 7(c) and Fig. 7(d) illustrate the joint-angle θ and joint-velocity $\dot{\theta}$, respectively.

From Fig. 8 and Table 4, we can see the end-effector position errors synthesized by the DT-RNN algorithm (3) using different values of sampling period ξ. Compared with $\xi = 0.001$ s, when $\xi = 0.0001$ s, the maximum position error synthesized by the DT-RNN algorithm (3) is of order 10^{-9}.

5.2. Example 2

Based on the previous regular elliptical path, in this example, we choose another complicated geometry as the second path for tracking control task of robotic manipulator. The Jacobian matrix setting is the same as in example

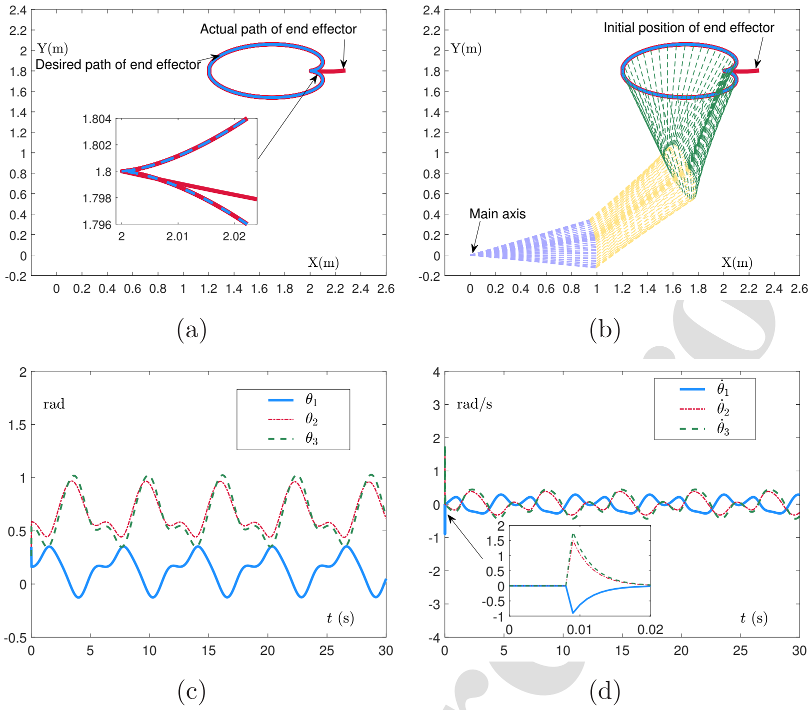
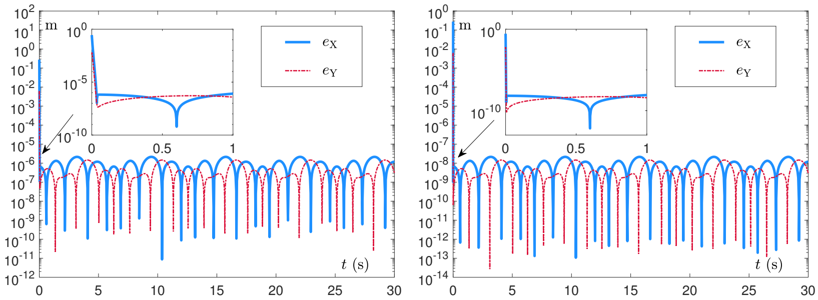


Figure 9: Application of DT-RNN algorithm (3) with $\omega = 0.3$ and sampling period $\xi = 0.001$ s for tracking path (12). (a) Motion trajectory of end-effector, where horizontal axis represents horizontal displacement and vertical axis represents longitudinal displacement. (b) Motion trajectory of whole robotic manipulator, where horizontal axis represents horizontal displacement and vertical axis represents longitudinal displacement. (c) Joint-angle profiles, where horizontal axis represents range of time and vertical axis represents joint-angle change. (d) Joint-velocity profiles, where horizontal axis represents range of time and vertical axis represents joint-velocity change.

1. Then the tracking control task is a cardioid path that can be defined as

$$\mathbf{r}(t_k) = \begin{bmatrix} (1/5)(2\cos(t_k) - \cos(2t_k)) + 9/5 \\ (1/10)(2\sin(t_k) - \sin(2t_k)) + 9/5 \end{bmatrix}. \tag{12}$$

Whereafter, we set the same parameter as those used in example 1: the value of sampling period ξ is 0.001 s; each rod length is 1 meter; the initial state of three joints are $\theta_1 = [\pi/9]$, $\theta_2 = [\pi/9]$ and $\theta_3 = [\pi/12]$, respectively; the task duration $t_\mathrm{d} = 30$ s. Firstly, from Fig. 9(a) we can see two different colored lines, which represent the experimental trajectory of robotic manipulator and the desired path. Then Fig. 9(b) shows a simplified diagram of



(a)

(b)

Figure 10: End-effector position errors. (a) Using $\xi = 0.001$ s, where horizontal axis represents range of time and vertical axis represents end-effector position errors. (b) Using $\xi = 0.0001$ s, where horizontal axis represents range of time and vertical axis represents end-effector position errors.

robotic manipulator similar to that in example 1 when performing a tracking cardioid path task. Afterwards the joint-angle θ and joint-velocity $\dot{\theta}$ of robotic manipulator are depicted in Fig. 9(c) and Fig. 9(d), respectively. Furthermore, the maximum position error synthesized by the DT-RNN algorithm (3) with $\xi = 0.0001$ s is of order 10^{-8}, which is about 100 times less than that with $\xi = 0.001$ s on the whole, as shown in Fig. 10 and Table 5.

As shown in the aforementioned robotic manipulator applications as well as the results of numerical experiment above, the DT-RNN algorithm (3), such a novel algorithm from the direct derivation scheme, can not only have excellent performance, but also omit some intermediate procedures compared with the traditional derivation scheme.

6. Conclusion

The DT-RNN algorithm (3) has been proposed from the technical scheme of direct derivation in this paper. Firstly, we have formulated the discrete time-variant matrix pseudo-inversion problem and an application preliminary of robotic manipulator has also been introduced. Secondly, the DT-RNN algorithm (3) is founded on the second-order Taylor expansion. Then the DT-RNN algorithm (3) has been analyzed theoretically, and the zero-stability, consistence and convergence of the proposed DT-RNN algorithm(3) have been proved. Finally, numerical experiment and comparison results have verified the validity of the DT-RNN algorithm (3). Besides, inspired by previous work, two applications of robotic manipulator through computing the discrete time-variant Jacobian matrix pseudo-inversion further validate the efficiency and applicability for industry application of the proposed DT-RNN algorithm (3). Due to the limitation of laboratory hardware equipment, the implementation of the proposed DT-RNN algorithm on a physical application would be our future research direction. In addition, more researches about advanced statistical method and computational time are also meaningful and worthwhile.

Appendix

For the purpose of theoretical analyses and explanations, based on the [40], the following results are presented.

Result 1: The zero-stability of an N-step discrete time-variant method $\sum_{k=0}^{N} x_k\sigma_{a+k} = \xi \sum_{k=0}^{N} y_k\omega_{a+k}$ can be checked by determining the roots of its characteristic polynomial $\eta(t) = \sum_{k=0}^{N} x_k t^k$. If all roots of $\eta(t) = 0$ meet the requirements: each root has a modulus of less than 1, and a root whose modulus equals 1 is a simple root, the N-step discrete time-variant method has zero-stability.

Result 2: In general, we consider that if the truncation error of a smooth exact solution of an N-step discrete time-variant method is $O(\xi^q)$ $(q > 0)$, the N-step method has consistence of q-order and the method is convergent to the same order as its truncation error.

Referring to Result 1 and inspired by [40], we can know about the solution of characteristic polynomial that belongs to the DT-RNN algorithm (3) is $\delta = 1$. In other words, the DT-RNN algorithm (3) shows zero-stability when there is just one root on the unit circle; referring to the previous theorems and considering about the structural form of the DT-RNN algorithm (3), it converges to a truncation error of order $\mathbf{O}(\xi^2)$. Therefore, the DT-RNN algorithm (3) is considered to have consistence according to Result 2. From the above analyses, the DT-RNN algorithm (3) has zero-stability and consistence. The proof is thus completed.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (with numbers 61906164 and 61972335), by the Natural Science Foundation of Jiangsu Province of China (with number BK20190875), by the Six Talent Peaks Project in Jiangsu Province (with number RJFW-053), by Jiangsu "333" Project, by Qinglan project of Yangzhou University, by High-end Talent Support Program of Yangzhou University, by the Cross-Disciplinary Project of the Animal Science Special Discipline of Yangzhou University, and by the Postgraduate Research & Practice Innovation Program of Jiangsu Province (with numbers KYCX21_3234 and SJCX22_1709).

CRediT authorship contribution statement

Yang Shi: Writing - original draft, Formal analyses, Conceptualization, Methodology. Wenhan Zhao: Formal analyses, Software, Writing - review & editing. Shuai Li: Resources, Conceptualization, Methodology, Writing - review & editing. Bin Li: Resources, Writing - review & editing, Supervision. Xiaobing Sun: Validation, Formal analyses, Visualization, Software.

Declaration of competing interest

The authors declare that they have no known competing nancial interests or personal relationships that could have appeared to inuence the work reported in this paper.

References

[1] Y. Shi, Y. Zhang, New discrete-time models of zeroing neural network solving systems of time-variant linear and nonlinear inequalities, IEEE Trans. Syst. Man Cybern. Syst. 50 (2) (2020) 565–576.

[2] L. Jin, S. Li, H. Wang, Z. Zhang, Nonconvex projection activated zeroing neurodynamic models for time-varying matrix pseudoinversion with accelerated finite-time convergence, Appl. Soft Comput. 62 (2018) 840–850.

[3] V. Chauhan, A. Tiwari, Randomized neural networks for multilabel classification, Appl. Soft Comput. 115 (2022) 108184.

[4] C. A. Lightcap, S. A. Banks, An extended Kalman filter for real-rime estimation and control of a rigid-link flexible-joint manipulator, IEEE Trans. Control Syst. Technol. 18 (1) (2010) 91–103.

[5] K. Hauser, S. Emmons, Global redundancy resolution via continuous pseudoinversion of the forward kinematic map, IEEE Trans. Autom. Sci. Eng. 15 (3) (2018) 932–944.

[6] Z. Hu, L. Xiao, K. Li, K. Li, J. Li, Performance analysis of nonlinear activated zeroing neural networks for time-varying matrix pseudoinversion with application, Appl. Soft Comput. 98 (2021) 106735.

[7] 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.

[8] C. M. Wong, C. M. Vong, P. K. Wong, J. Cao, Kernel-based multilayer extreme learning machines for representation learning, IEEE Trans. Neural Netw. Learn. Syst. 29 (3) (2018) 757–762.

[9] Y. Zhang, J. Zhang, J. Weng, Dynamic Moore-Penrose inversion with unknown derivatives: Gradient neural network approach, IEEE Trans. Neural Netw. Learn. Syst. (2022) doi: 10.1109/TNNLS.2022.3171715. In Press.

[10] M. Hanmandlu, S. Singhal, Face recognition under pose and illumination variations using the combination of information set and PLPP features, Appl. Soft Comput. 53 (2017) 396–406.

[11] M. B. Tasić, P. S. Stanimirović, M. D. Petković, Symbolic computation of weighted Moore-Penrose inverse using partitioning method, Appl. Math. Comput. 189 (1) (2007) 615–640.

[12] Y. Wei, J. Cai, M. K. Ng, Computing Moore-Penrose inverses of Toeplitz matrices by Newton's iteration, Math. Comput. Model. 40 (1) (2004) 181–191.

[13] D. C. Hoyle, Accuracy of pseudo-inverse covariance learning - a random matrix theory analysis, IEEE Trans. Pattern Anal. Mach. Intel. 33 (7) (2011) 1470–1481.

[14] V. Y. Pan, F. Soleymani, L. Zhao, An efficient computation of generalized inverse of a matrix, Appl. Math. Comput. 316 (2018) 89–101.

[15] L. Xiao, J. Dai, R. Lu, S. Li, J. Li, S. Wang, Design and comprehensive analysis of a noise-tolerant ZNN model with limited-time convergence for time-dependent nonlinear minimization, IEEE Trans. Neural Netw. Learn. Syst. 31 (12) (2020) 5339–5348.

[16] B. Liao, Y. Wang, W. Li, C. Peng, Q. Xiang, Prescribed-time convergent and noise-tolerant Z-type neural dynamics for calculating time-dependent quadratic programming, Neural. Comput. Appl. 33 (10) (2021) 5327–5337.

[17] L. Xiao, Y. Zhang, Q. Zuo, J. Dai, J. Li, W. Tang, A noise-tolerant zeroing neural network for time-dependent complex matrix inversion under various kinds of noises, IEEE Trans. Ind. Informat. 16 (6) (2020) 3757–3766.

[18] L. Jin, L. Wei, S. Li, Gradient-based differential neural-solution to time-dependent nonlinear optimization, IEEE Trans. Automat. Contr. (2022) doi: 10.1109/TAC.2022.3144135. In Press.

[19] M. Liu, L. Chen, X. Du, L. Jin, M. Shang, Activated gradients for deep neural networks, IEEE Trans. Neural Netw. Learn. Syst. (2021) doi: 10.1109/TNNLS.2021.3106044. In Press.

[20] Y. Shi, J. Wang, S. Li, B. Li, and X. Sun, Tracking control of cable-driven planar robot based on discrete-time recurrent neural network with immediate discretization method, IEEE Trans. Ind. Inf. to be published. doi: 10.1109/TII.2022.3210255. In Press.

[21] Y. Shi, Z. Pan, J. Li, B. Li, and X. Sun, Recurrent neural dynamics for handling linear equation system with rank-deficient coefficient and disturbance existence, J. Franklin Inst. 359 (7) (2022) 3090–3102.

[22] J. Guo, B. Qiu, Y. Zhang, Future different-layer linear equation and bounded inequality solved by combining adams-bashforth methods with CZNN model, IEEE Trans. Ind. Electron. 68 (2) (2021) 1515–1524.

[23] Y. Shi, C. Mou, Y. Qi, B. Li, S. Li, and B. Yang, Design, analysis and verification of recurrent neural dynamics for handling time-variant augmented Sylvester linear system, Neurocomputing 426 (2021) 274–284

[24] O. Barron, M. Raison, G. Gaudet, S. Achiche, Recurrent neural network for electromyographic gesture recognition in transhumeral amputees, Appl. Soft Comput. 96 (2020) 106616.

[25] Y. Zhang, S. Li, J. Weng, Learning and near-optimal control of underactuated surface vessels with periodic disturbances, IEEE Trans. Cybern. (2020) doi: 10.1109/TCYB.2020.3041368. In Press.

[26] Y. Shi, L. Jin, S. Li, and J. Qiang, Proposing, developing and verification of a novel discrete-time zeroing neural network for solving future augmented Sylvester matrix equation, J. Franklin Inst. 357 (6) 2020 3636–3655

[27] Y. Zhang, S. S. Ge, Design and analysis of a general recurrent neural network model for time-varying matrix inversion, IEEE Trans. Neural Netw. 16 (6) (2005) 1477–1490.

[28] W. Wei, B. Zheng, Improved recurrent neural networks for solving Moore-Penrose inverse of real-time full-rank matrix, Neurocomputing 418 (2020) 221–231.

[29] D. Guo, Z. Nie, L. Yan, Novel discrete-time Zhang neural network for time-varying matrix inversion, IEEE Trans. Syst. Man Cybern. Syst. 47 (8) (2017) 2301–2310.

[30] B. Liao, Y. Zhang, From different ZFs to different ZNN models accelerated via Li activation functions to finite-time convergence for time-varying matrix pseudoinversion, Neurocomputing 133 (2014) 512–522.

[31] M. D. Petković, P. S. Stanimirović, V. N. Katsikis, Modified discrete iterations for computing the inverse and pseudoinverse of the time-varying matrix, Neurocomputing 289 (2018) 155–165.

[32] B. Liao, Y. Zhang, Different complex ZFs leading to different complex ZNN models for time-varying complex generalized inverse matrices, IEEE Trans. Neural Netw. Learn. Syst. 25 (9) (2014) 1621–1631.

[33] 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.

[34] P. S. Stanimirović, I. S. Živković, Y. Wei, Recurrent neural network for computing the Drazin inverse, IEEE Trans. Neural Netw. Learn. Syst. 26 (11) (2015) 2830–2843.

[35] 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. 51 (9) (2021) 5639–5651.

[36] Y. Zhang, S. Li, G. Geng, Initialization-based k-winners-take-all neural network model using modified gradient descent, IEEE Trans. Neural Netw. Learn. Syst. (2021) doi: 10.1109/TNNLS.2021.3123240. In Press.

[37] Z. Zhang, X. Deng, M. He, T. Chen, J. Liang, Runge-Kutta type discrete circadian RNN for resolving tri-criteria optimization scheme of noises perturbed redundant robot manipulators, IEEE Trans. Syst. Man Cybern. Syst. 52 (3) (2022) 1405–1416.

[38] Y. Shi, W. Zhao, S. Li, B. Li, X. Sun, Novel discrete-time recurrent neural network for robot manipulator: a direct discretization technical route, IEEE Trans. Neural Netw. Learn. Syst. (2021) doi: 10.1109/TNNLS.2021.3108050. In Press.

[39] Y. Shi, L. Jin, S. Li, J. Li, J. Qiang, D. K. Gerontitis, Novel discrete-time recurrent neural networks handling discrete-form time-variant multi-augmented Sylvester matrix problems and manipulator application, IEEE Trans. Neural Netw. Learn. Syst. 33 (2) (2022) 587–599.

[40] Y. Shi, B. Qiu, D. Chen, J. Li, Y. Zhang, Proposing and validation of a new four-point finite-difference formula with manipulator application, IEEE Trans. Ind. Informat. 14 (4) (2018) 1323–1333.

[41] L. Jin, Y. Zhang, G2-type SRMPC scheme for synchronous manipulation of two redundant robot arms, IEEE Trans. Cybern. 45 (2) (2015) 153–164.

[42] D. Chen, S. Li, L. Liao, A recurrent neural network applied to optimal motion control of mobile robots with physical constraints, Appl. Soft Comput. 85 (2019) 105880.

[43] Z. Xie, L. Jin, X. Luo, Z. Sun, M. Liu, RNN for repetitive motion generation of redundant robot manipulators: an orthogonal projection-based scheme, IEEE Trans. Neural Netw. Learn. Syst. 33 (2) (2022) 615–628.

[44] L. Xiao, Z. Zhang, S. Li, Solving time-varying system of nonlinear equations by finite-time recurrent neural networks with application to motion tracking of robot manipulators, IEEE Trans. Syst. Man Cybern. Syst. 49 (11) (2019) 2210–2220.

Highlights

First of all, a direct derivation scheme founded on the second-order Taylor expansion has been proposed to establish discrete time-variant recurrent neural network algorithm for discrete time-variant matrix pseudo-inversion, and the solving process has no longer required the theoretical support of continuous time-variant background.

Secondly, for the proposed discrete time-variant recurrent neural network algorithm, theoretical analyses have been shown that such algorithm could be exactly convergent.

Finally, the effectiveness of discrete time-variant recurrent neural network algorithm for discrete time-variant matrix pseudo-inversion has been proved using numerical experiment results. In addition, two application experiments of robotic manipulator have been shown to further validate the efficiency and practicability of the discrete time-variant recurrent neural network algorithm.

Credit authorship statement

Yang Shi: Writing - original draft, Formal analysis, Conceptualization, Methodology.

Wenhan Zhao: Formal analysis, Software, Writing – review & editing.

Shuai Li: Resources, Conceptualization, Methodology, Writing - review & editing.

Bin Li: Resources, Writing - review & editing, Supervision.

Xiaobing Sun: Validation, Formal analysis, Visualization, Software.

Declaration of interests

☒ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐ The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: