Modified Newton Integration Algorithm With Noise Tolerance Applied to Robotics

Dongyang Fu, Haoen Huang, Lin Wei, Xiuchun Xiao, Long Jin, *Member, IEEE*, Shan Liao, Jialiang Fan, and Zhengtai Xie

***Abstract*—Currently, the Newton–Raphson iterative algorithm has been extensively employed in the fields of basic research and engineering. However, when noise components exist in a system, its performance is largely affected. To remedy shortcomings that the conventional computing methods have encountered in a noisy workspace, a novel modified Newton integration (MNI) algorithm is proposed in this article. In addition, the steady-state error of the proposed MNI algorithm is smaller than that of the Newton–Raphson algorithm under a noise-free or noisy workspace. To lay the foundations for the corresponding theoretical analyses, the proposed MNI algorithm is first converted into a homogeneous linear equation with a residual term. Then, the related theoretical analyses are carried out, which indicate that the MNI algorithm possesses noise-tolerance ability under various noisy environments. Finally, multiple computer simulations and physical experiments on robot control applications are performed to verify the feasibility and advantage of the proposed MNI algorithm.**

***Index Terms*—Dynamic system of linear equations, modified Newton integration (MNI) algorithm, noise tolerance, steady-state error.**

Manuscript received February 17, 2020; revised November 30, 2020; accepted December 27, 2020. This work was supported in part by the Fund of Southern Marine Science and Engineering Guangdong Laboratory of Zhanjiang, China, under Grant ZJW-2019-08; in part by the Key Projects of the Guangdong Education Department under Grant 2019KZDXM019; in part by the High-Level Marine Discipline Team Project of Guangdong Ocean University under Grant 002026002009; in part by the Guangdong Graduate Academic Forum Project under Grant 230420003; in part by the First Class Discipline Construction Platform Project in 2019 of Guangdong Ocean University under Grant 231419026; in part by the CAAI-Huawei MindSpore Open Fund under Grant CAAIXSJLJJ-2020-009A; in part by the CAS "Light of West China" Program; in part by the Innovation and Strength Project in Guangdong Province, China (Natural Science) under Grant 230419065; in part by the Key Lab of Digital Signal and Image Processing of Guangdong Province, China, under Grant 2019GDDSIPL-01; in part by the Industry-University-Research Cooperation Education Project of Ministry of Education under Grant 201801328005; in part by the Guangdong Graduate Education Innovation Project, Graduate Summer School under Grant 2020SQXX19; in part by the Special Project in Key Fields of Universities in Department of Education of Guangdong Province under Grant 2019KZDZX1036; in part by the Guangdong Graduate Education Innovation Project, Graduate Academic Forum under Grant 2020XSLT27; in part by the Doctoral Initiating Project of Guangdong Ocean University under Grant E13428; in part by the Special Project in Key Fields of Universities in Department of Education of Guangdong Province, China, under Grant 2019033; and in part by the Natural Science Foundation of Chongqing, China, under Grant cstc2020jcyj-zdxmX0028. This article was recommended by Associate Editor M. Zhou. *(Corresponding author: Long Jin.)*

Dongyang Fu, Haoen Huang, and Xiuchun Xiao are with the School of Electronics and Information Engineering, Guangdong Ocean University, Zhanjiang 524088, China, also with the Guangdong Provincial Engineering and Technology Research Center of Marine Remote Sensing and Information Technology, Guangdong Ocean University, Zhanjiang 524088, Guangdong, China, and also with the Shenzhen Institute of Guangdong Ocean University, Guangdong Ocean University, Shenzhen 518108, China (e-mail: fdy163@163.com; hh1848088380@gmail.com; xcxiao@hotmail.com).

Lin Wei, Long Jin, Jialiang Fan, and Zhengtai Xie are with the Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Beijing 100049, China, and also with the School of Information Science and Engineering, Lanzhou University, Lanzhou 730000, China (e-mail: jinlongsysu@foxmail.com; longjin@ieee.org).

Shan Liao is with the School of Cybersecurity, Sichuan University, Chengdu 610000, China (e-mail: liaoe1110@163.com).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TSMC.2021.3049386.

Digital Object Identifier 10.1109/TSMC.2021.3049386

I. Introduction

The solution of many problems can eventually be transformed into solving the solution of the system of linear equations in natural sciences and engineering applications [1]–[3]. For instance, the solution of the system of linear equations can be employed to robot motion planning [4], [5], image processing [6], robust track control [7], [8], model predictive control [9], [10], etc. Due to the importance of the system of linear equations in theoretical and practical engineering, plenty of researchers have developed extensive investigations in this area for many years [11]–[13]. According the previous researches, iterative algorithms are effective for solving the system of linear equations problems, including gradient-type algorithm, Newton-type algorithm, and their modifications [14], [15]. A modified gradient method is provided to solve the Sylvester equation [16], which can generate high-performance computing solutions with less convergent time. Besides, Ferreira *et al.* [17] exploited the gradient approach to handle an underdetermined system of linear equations, whose upper bound estimated is proved to converge in finite time. Based on the combination of traditional Newton method and damped Newton method, a new Newton method is employed to solve the underdetermined equations [18], whose complexity analyses and convergence proof are given. A static iterative method called Scale-Splitting method is presented to solve the complex symmetric system of linear equations, of which the numerical experiments show that its effectiveness is better than the rotate block triangular method [19].

Generally speaking, many valuable results have been obtained for solving the system of linear equations. However, most of the existing investigations are far from able to suppress noises. In other words, most of the existing algorithms are designed without considering the influence of noise in the stability of a system, which could bring about a series of negative consequences. Commonly, solving techniques are to restrict the noise components in a sufficiently ignorable range for ensuring the desired results. To this end, it needs to add filters to diminish the corresponding noise components in advance in the practical workspace. Nevertheless, this method can not handle the residual error caused by dynamic parameters in the computing process [17]–[19].

In this case, some researchers have a mind to eliminate or reduce the influence of the noise components by utilizing the control algorithms [20]–[22]. A novel recurrent neural network with noise-tolerant ability is designed for solving the time-varying underdetermined linear system, which can resist the influence of noises on the system [23]. Xiao *et al.* [24] established a dynamical neural model with the capability to deal with the unknown noise, which can be applied to the system of the linear equations. For solving real-time-varying matrix inversion, Jin *et al.* [25] proposed an integration-enhanced neural network model that converges to the theoretical solution in the presence of various kinds of noises. In another research [26], a distributed scheme presented for the cooperative motion generation is reformulated as quadratic programming, and then a noise-tolerant iterative neural network is constructed to solve this quadratic programming problem online. In addition, Guo *et al.* [27] investigated the noise-tolerant design formula to solve the system of time-varying nonlinear equations. However, the majority of the above mentioned algorithms are based on continuous-time systems and cannot be directly implemented in digital devices.

To conquer the weaknesses of the aforementioned algorithms, a modified Newton integration (MNI) algorithm is proposed in this article, inspired by Newton–Raphson iterative (NRI) with fast convergence [28]. In the light of control theory, the error-summation term is perceived to possess the remarkable effect on the noise-suppressing and can be used to improve the robustness of the algorithm. In other words, the error-summation feedback term can effectively eliminate the deviation between the computing solution and the theoretical one. Moreover, the proposed MNI algorithm not only shows the advantageous performance of noise tolerance, but also can generate the computing solution with high precision. Generally speaking, the proposed MNI algorithm possesses prominent robustness and stability.

The remainder sections of this article are arranged as follows. In Section II, the proposed MNI algorithm is deduced. The relevant proof on the convergence of the proposed MNI algorithm is provided in Section III. Via a series of numerical simulations, Section IV compares the properties among newly proposed MNI algorithm, the zeroing neural network (ZNN) algorithm [26] and the NRI algorithm [28] under the circumstances of four different noises. Next, in Section V, multiple computer simulations and physical experiments on robot control employed by the proposed MNI algorithm and other solving methods are performed under various noisy conditions, which further verify the availability and capability of the proposed MNI algorithm in noisy workspace. Naturally, Section VI summarizes the full article. Before ending this section, the main contributions of this article are concluded as follows.

1) Based on the control theory, the proposed MNI algorithm, being a major breakthrough in the conventional Newton algorithm for solving the dynamic system of linear equations with noise-suppressing ability, can be seen as a novel algorithm paradigm.
2) The hallmark of the proposed MNI algorithm is that it remarkably improves the accuracy of the steady-state error from $O(\Delta)$ to $O(\Delta^2)$ without any aid of the time-difference information, where Δ denotes the sampling period.
3) Differing from the continuous-time model, the proposed MNI algorithm can tackle the discrete-time linear systems directly with various kinds of noises, which is ready for hardware implementation.

II. Formula Derivation

Without loss of generality, a dynamic system of linear equations with discrete-time form is shown as

$$P_k\mathbf{x}_k = \mathbf{q}_k \tag{1}$$

where matrix $P_k \in \mathbb{R}^{m\times n}$ and vector $\mathbf{q}_k \in \mathbb{R}^m$ represent the system inputs measured at time instant $t = k\Delta$ with k denoting the updating index and Δ denoting the sampling period. Additionally, the solution of (1) is a future calculational model with unknown parameters. The system output $\mathbf{x}_k \in \mathbb{R}^n$ is calculated before the time instant $t = k\Delta$. In other words, the unknown vector $\mathbf{x}_k$ is solved during the time sampling interval $[t_{k-1}, t_k)$, whereas P_k and $\mathbf{q}_k$ are unknown.

In the first place, an error function is defined to measure the accuracy of the computing solution

$$\mathbf{e}_k = P_k\mathbf{x}_k - \mathbf{q}_k. \tag{2}$$

Then, exploiting the NRI algorithm [28] to solve the benchmark problem (1), the iterative formula is

$$\mathbf{x}_{k+1} = \mathbf{x}_k - P_k^{\dagger}(P_k\mathbf{x}_k - \mathbf{q}_k) \tag{3}$$

where $P_k^{\dagger}$ represents the pseudoinverse of matrix P_k. Subsequently, multiplying $1/\Delta$ by both sides of (3), it has

$$\frac{\mathbf{x}_{k+1} - \mathbf{x}_k}{\Delta} = -\frac{1}{\Delta}P_k^{\dagger}(P_k\mathbf{x}_k - \mathbf{q}_k). \tag{4}$$

According to the definition of the derivative, the above formula (3) can be converted into a continuous-time form (5) when Δ is small enough

$$\dot{\mathbf{x}}(t) = -\gamma P^{\dagger}(t)(P(t)\mathbf{x}(t) - \mathbf{q}(t)) \tag{5}$$

of which $\gamma = 1/\Delta$. An integration feedback term is utilized to enhance the robustness of model (5)

$$\dot{\mathbf{x}}(t) = -P^{\dagger}(t)\left(\gamma\mathbf{e}(t) + \beta\int_0^t \mathbf{e}(\alpha)d\alpha\right) \tag{6}$$

where $\beta > 0$ is a scaling factor of the integration term. For facilitating the derivation or evolution of the following context, the Euler forward difference formula is provided as [29]

$$\dot{\mathbf{x}}_k = \frac{\mathbf{x}_{k+1} - \mathbf{x}_k}{\Delta} + \mathbf{O}(\Delta). \tag{7}$$

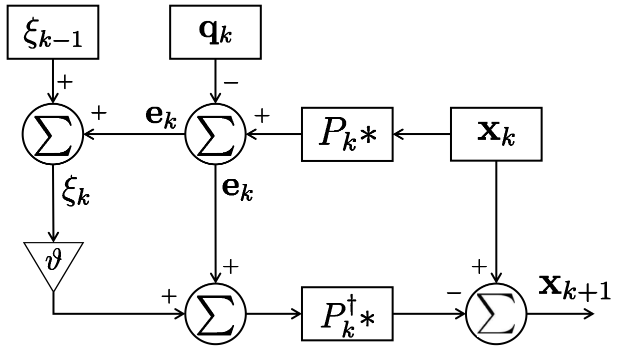


Fig. 1. Block diagrams of the proposed MNI algorithm (8) for the dynamic system of linear equations, where the left multiplication is signified by the symbol $*$, e.g., P_k* signifies $P_k * \mathbf{x}_k$.

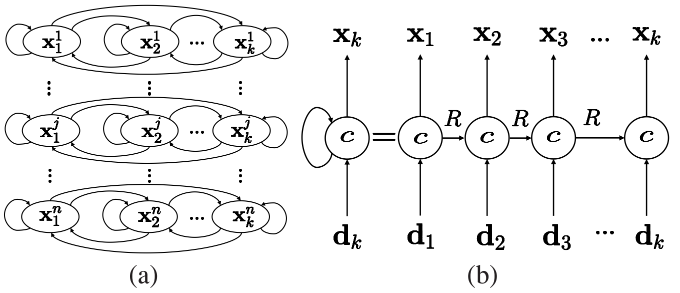

$\mathbf{x}_1^1$ $\mathbf{x}_2^1$ ... $\mathbf{x}_k^1$
$\mathbf{x}_1^j$ $\mathbf{x}_2^j$ ... $\mathbf{x}_k^j$
$\mathbf{x}_1^n$ $\mathbf{x}_2^n$ ... $\mathbf{x}_k^n$
(a)
$\mathbf{x}_k$ $\mathbf{x}_1$ $\mathbf{x}_2$ $\mathbf{x}_3$ ... $\mathbf{x}_k$
c = c R c R c R c
$\mathbf{d}_k$ $\mathbf{d}_1$ $\mathbf{d}_2$ $\mathbf{d}_3$ ... $\mathbf{d}_k$
(b)

Fig. 2. Network structure of the proposed MNI algorithm (8). (a) Whole network structure. (b) Computing procedure of an individual neuron at time instant $t = k\Delta$.

Exploiting the Euler forward difference formula (7), the proposed MNI algorithm for dynamic system of linear equations is generated as follows with its block diagrams presented in Fig. 1:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - P_k^{\dagger}\left(P_k\mathbf{x}_k - \mathbf{q}_k + \vartheta\sum_{i=0}^{k}(P_i\mathbf{x}_i - \mathbf{q}_i)\right) \quad (8)$$

where $\vartheta = \beta\Delta^2 (0 < \vartheta < 1)$.

For comparison, the ZNN algorithm performed in [26] is arranged as

$$\mathbf{x}_{k+1} = \mathbf{x}_k - P_k^{\dagger}\big(\Delta\dot{P}_k\mathbf{x}_k - \Delta\dot{\mathbf{q}}_k + \eta(P_k\mathbf{x}_k - \mathbf{q}_k)\big) \quad (9)$$

with $\eta \in \mathbb{R}^+$.

Remark 1: The network structure of the proposed MNI algorithm (8) is provided in Fig. 2. From the whole network structure depicted in Fig. 2(a), the future solution $\mathbf{x}_{k+1}$ can be obtained by using the previous data in the update rule. Fig. 2(b) presents the computing procedure of an individual neuron at time instant $t = k\Delta$, of which $\mathbf{d}$ denotes the input data; R represents the stored data; $c = 1$ is the individual neuron.

Remark 2: The computational complexity of the proposed MNI algorithm (8) can be analyzed as follows. First, the proposed MNI algorithm (8) is a discrete-time algorithm, meaning that the integral term of the proposed MNI algorithm (8) can be readily calculated with an accumulator. Additionally, the proposed MNI algorithm (8) can be presented with another form as

$$\begin{cases} \mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{g}_k \\ \mathbf{g}_k = -P_k^{\dagger}((P_k\mathbf{x}_k - \mathbf{q}_k) + \vartheta\boldsymbol{\varrho}_k) \\ \boldsymbol{\varrho}_{k+1} = \boldsymbol{\varrho}_k + \Delta(P_k\mathbf{x}_k - \mathbf{q}_k). \end{cases}$$

Simultaneously, the sizes of the above variables are retold, i.e., $\mathbf{x}_k$ and $\mathbf{g}_k \in \mathbb{R}^n$; $\mathbf{q}_k$ and $\boldsymbol{\varrho}_k \in \mathbb{R}^m$; $P_k \in \mathbb{R}^{m\times n}$; Δ and $\vartheta \in \mathbb{R}^+$. At the kth iterative interval of the MNI algorithm (8), $\boldsymbol{\varrho}_{k+1}$, $\mathbf{g}_k$, and $\mathbf{x}_k$ require to be obtained in advance and $\mathbf{x}_{k+1}$ would be output as the computing solution. Before analyzing the computational complexity of the MNI algorithm (8), some prerequisites need to be informed [30].

1) Multiplying a real number by an n-dimensional vector requires n flops.
2) Adding or subtracting two n-dimensional vectors requires n flops.
3) Multiplying a matrix of size $m \times n$ and a n-dimensional vector requires $m(2n - 1)$ flops.
4) The pseudoinversion of a matrix of size $m \times n$ requires $n^3 + 2m^2(n - 1) + 2n^2(m - 1)$ flops.

Especially, based on the above prerequisites, computing $\mathbf{g}_k$ requires $n^3 + 2m^2(n - 1) + 2n^2(m - 1) + 4mn + 3m - n$ flops; computing ϱ_{k+1} requires $2m(n + 1)$ flops; computing $\mathbf{x}_{k+1}$ requires n flops. Consequently, the proposed MNI algorithm (8) costs $n^3 + 2m^2(n - 1) + 2n^2(m - 1) + 6mn + 5m$ flops at the kth iterative interval. In the same way, the ZNN algorithm (9) costs $n^3 + 2m^2(n - 1) + 2n^2(m - 1) + 6mn + m$ flops at the kth iterative interval. The NRI algorithm (3) costs $n^3 + 2m^2(n - 1) + 2n^2(m - 1) + 4mn$ flops at the kth iterative interval. By comparison, it can be clearly found that the MNI algorithm (8) requires more $4m$ and $2mn + 5m$ flops than those of the ZNN algorithm (9) and the NRI algorithm (3) at each iterative interval, respectively. From the perspective of computational complexity, the MNI algorithm (8) is more complicated than the ZNN algorithm (9), and the ZNN algorithm (9) is more complicated than the NRI algorithm (3).

III. Theoretical Analyses and Proof

In this section, theoretical analyses and related proofs on the proposed MNI algorithm (8) are executed, which illustrate the high-precision performance of the proposed MNI algorithm (8) with various noise components.

Theorem 1: The proposed MNI algorithm (8) can be transformed into

$$\mathbf{e}_{k+1} + \vartheta\sum_{i=0}^{k}\mathbf{e}_i + \mathbf{O}\big(\Delta^2\big) = \mathbf{0} \quad (10)$$

where $\mathbf{O}(\Delta^2)$ stands for the vector generated by the second-order error $O(\Delta^2)$.

Proof: The proposed MNI algorithm (8) is rewritten as

$$\mathbf{x}_{k+1} = \mathbf{x}_k - P_k^{\dagger}\left(P_k\mathbf{x}_k - \mathbf{q}_k + \vartheta\sum_{i=0}^{k}(P_i\mathbf{x}_i - \mathbf{q}_i)\right) \quad (11)$$

which can be formulated as

$$P_k(\mathbf{x}_{k+1} - \mathbf{x}_k) = -\left(\mathbf{e}_k + \vartheta\sum_{i=0}^{k}\mathbf{e}_i\right). \quad (12)$$

Considering the error function (2), it is not difficult to find that error variable $\mathbf{e}_k$ is a dependent variable with respect to variable $\mathbf{x}_k$ at a the kth time instant in the procedure of solving the dynamic system of linear equations (1). Simultaneously, according to the derivative definition of the discrete-time function and combined with formula (2), it has

$$\begin{aligned}\dot{\mathbf{e}}_k &= \lim_{\Delta\to 0}\frac{\mathbf{e}_{k+1}-\mathbf{e}_k}{\Delta}\\ &= \lim_{\Delta\to 0}\frac{(P_k\mathbf{x}_{k+1}-\mathbf{q}_k)-(P_k\mathbf{x}_k-\mathbf{q}_k)}{\Delta}\\ &= P_k\dot{\mathbf{x}}_k. \end{aligned}\tag{13}$$

Therefore, based on the above evolution, (12) can be rearranged as

$$-\gamma\left(\mathbf{e}_k+\vartheta\sum_{i=0}^{k}\mathbf{e}_i\right)=P_k\dot{\mathbf{x}}_k=\dot{\mathbf{e}}_k.\tag{14}$$

Subsequently, via using the Euler forward difference formula (7) to discretize $\dot{\mathbf{e}}_k$, the above equation can be turned into as

$$\gamma(\mathbf{e}_{k+1}-\mathbf{e}_k)+\mathbf{O}(\Delta)=-\gamma\left(\mathbf{e}_k+\vartheta\sum_{i=0}^{k}\mathbf{e}_i\right).\tag{15}$$

Eventually, (15) can be formulated as

$$\mathbf{e}_{k+1}+\vartheta\sum_{i=0}^{k}\mathbf{e}_i+\mathbf{O}\left(\Delta^2\right)=\mathbf{0}.\tag{16}$$

The proof is thus completed. ■

Theorem 2: The steady-state error $\lim_{k\to\infty}\|\mathbf{e}_k\|_2$ of the proposed MNI algorithm (8) equals to $O(\Delta^2)$, where $\|\cdot\|_2$ signifies the two-norm operation.

Proof: The jth subsystem of formula (10) is written as

$$\mathbf{e}^j_{k+1}+\vartheta\sum_{i=0}^{k}\mathbf{e}^j_i+O\left(\Delta^2\right)=0\tag{17}$$

with the jth subsystem of term $\mathbf{e}_{k-1}$ being

$$\mathbf{e}^j_{k}+\vartheta\sum_{i=0}^{k-1}\mathbf{e}^j_i+O\left(\Delta^2\right)=0.\tag{18}$$

Letting formula (17) subtract formula (18) leads to

$$\mathbf{e}^j_{k+1}=h\mathbf{e}^j_k+O\left(\Delta^2\right)\tag{19}$$

where $h=1-\vartheta$ $(0<h<1)$. Naturally, formulating the above equation leads to

$$\begin{aligned}\mathbf{e}^j_{k+1}&=h\mathbf{e}^j_k+O\left(\Delta^2\right)\\ &=h^2\mathbf{e}^j_{k-1}+hO\left(\Delta^2\right)+O\left(\Delta^2\right)\\ &=h^2\mathbf{e}^j_{k-1}+O\left(\Delta^2\right)\\ &\ \vdots\\ &=h^k\mathbf{e}^j_1+O\left(\Delta^2\right).\end{aligned}$$

In addition, when k is toward to infinite, it has $\lim_{k\to\infty}\mathbf{e}^j_{k+1}=O(\Delta^2)$. Because of the relationship between $\mathbf{e}^j_{k+1}$ and $\mathbf{e}_k$, it can be found that the steady-state error $\lim_{k\to\infty}\|\mathbf{e}_k\|_2$ equals to $O(\Delta^2)$, which comes to the conclusion that the steady-state error of the proposed MNI algorithm (8) globally converges to $O(\Delta^2)$ under the noise-free condition. The proof is completed. ■

In order to further figure out the theoretical solution of the proposed MNI algorithm (8) in the cases of constant noises and linear noises, Theorem 3 is provided as follows.

Theorem 3: Supposing the linear noise components being $\omega(t)=\lambda t+v$, $\lambda\in\mathbb{R}^n$ being the coefficient vector and $v\in\mathbb{R}^n$ being the constant vector, the theoretical steady-state error $\lim_{k\to\infty}\|\mathbf{e}_k\|_2$ of the proposed MNI algorithm (8) is $\|\lambda/\beta\Delta\|_2+O(\Delta^2)$. Considering the constant noise $\omega(t)=v$ with $\lambda=\mathbf{0}$, the steady-state error of the proposed MNI algorithm (8) is $O(\Delta^2)$, no matter how large $\omega(t)=v$ is.

Proof: Since the proposed MNI algorithm (8) is a linear equation, the superposition theorem is suitable for it. When the linear or constant noise components exist in the system, it can be divided into two parts, i.e., the main noise part $\omega(t)=\lambda t+v$ and the residual error part $\mathbf{O}(\Delta^2)$. First, it can be depicted as the following formula (20), which originates from formula (10) when considering the $\omega(t)=\lambda t+v$ part

$$\mathbf{e}_{k+1}+\vartheta\sum_{i=0}^{k}\mathbf{e}_i+\lambda k\Delta+v=\mathbf{0}.\tag{20}$$

Taking the jth subsystem of $\mathbf{e}_k$ into account, formula (20) can be rewritten as

$$\mathbf{e}^j_{k+1}+\vartheta\sum_{i=0}^{k}\mathbf{e}^j_i+\lambda^j k\Delta+v^j=0.\tag{21}$$

Exploiting Z-transform to formula (21) could yield

$$z\mathbf{e}^j(z)+\vartheta\frac{z\mathbf{e}^j(z)}{z-1}+\frac{z\lambda^j\Delta}{(z-1)^2}+\frac{zv^j}{z-1}=0.\tag{22}$$

After arrangement and simplification, the above formula is reformulated as

$$\mathbf{e}^j(z)=-\frac{z(z-1)v^j+z\lambda^j\Delta}{(z-1)\left(z^2+z(\vartheta-1)\right)}.\tag{23}$$

It is clear to calculate that the poles of formula (23) are $z_1=1$, $z_2=0$ and $z_3=1-\vartheta$. Thus, in view of the final value theorem for formula (23), the result would be acquired as

$$\begin{aligned}\lim_{k\to\infty}\mathbf{e}^j_k&=\lim_{z\to 1}(z-1)\mathbf{e}^j(z)\\ &=\lim_{z\to 1}\frac{-z(z-1)v^j-z\lambda^j\Delta}{\vartheta z+z^2-1}\\ &=\frac{-\lambda^j}{\beta\Delta}.\end{aligned}$$

Then, taking the residual error part $\mathbf{O}(\Delta^2)$ into the proposed MNI algorithm (8), the result of $\lim_{k\to\infty}\|\mathbf{e}_k\|_2$ equals to $O(\Delta^2)$ by the corresponding proof as same as Theorem 2. Combining the two parts, the steady-state error $\lim_{k\to\infty}\|\mathbf{e}_k\|_2$ of the proposed MNI algorithm (8) equals to $\|\lambda/\beta\Delta\|_2+O(\Delta^2)$. This makes it simple to calculate that the steady-state error $\lim_{k\to\infty}\|\mathbf{e}_k\|_2$ of the proposed MNI algorithm (8) is $O(\Delta^2)$ in the condition of constant noise $\omega(t) = v$. Hereto, the proof is completed. ■

In the interest of revealing the completeness of the proposed MNI algorithm (8) encountering various noise components, the convergence of the proposed MNI algorithm (8) polluted with random noise component $\omega(t) = \delta$ is testified in Theorem 4.

Theorem 4: Assuming the noise component to be a random variable $\omega(t) = \delta$, the steady-state error $\lim_{k\to\infty} \|\mathbf{e}_k\|_2$ of the proposed MNI algorithm (8) is less than the upper bound $2n \sup_{1\le i\le k, 1\le j\le n} |\delta_i^j|/\vartheta + O(\Delta^2)$, which keeps the boundary output with the boundary input.

Proof: Similar to the proof of Theorem 3, the derivation procedure should be divided into random noise part $\omega(t) = \delta$ and the residual error part $\mathbf{O}(\Delta^2)$, respectively. With the same method for obtaining formula (19), the equation can be formulated as

$$\mathbf{e}_{k+1}^j = h\mathbf{e}_k^j + \delta_k^j - \delta_{k-1}^j. \tag{24}$$

Taking the identical manipulation in Theorem 2, a concise equation can be obtained by formulating the above equation

$$\mathbf{e}_{k+1}^j = h\mathbf{e}_k^j + \varphi_k^j \tag{25}$$

of which $\varphi_k^j = \delta_k^j - \delta_{k-1}^j$. Then, it can be evolved as below

$$\begin{aligned}\mathbf{e}_{k+1}^j &= h\mathbf{e}_k^j + \varphi_k^j \\ &= h^2\mathbf{e}_{k-1}^j + h\varphi_{k-1}^j + \varphi_k^j \\ &\ \ \vdots \\ &= h^k\mathbf{e}_1^j + h^{k-1}\varphi_1^j + \cdots + h\varphi_{k-1}^j + \varphi_k^j.\end{aligned}$$

When k tends to infinity, it has $\lim_{k\to\infty} h^k = 0$. Therefore, the aforementioned formula is expressed as

$$\begin{aligned}\lim_{k\to\infty} \mathbf{e}_{k+1}^j &= h^{k-1}\varphi_1^j + \cdots + h\varphi_{k-1}^j + \varphi_k^j \\ &< \max_{1\le i\le k} \varphi_i^j\left(h^{k-1} + h^{k-2} + \cdots + h^2 + h + 1\right) \\ &= \max_{1\le i\le k} \varphi_i^j \frac{1-h^{k-1}}{1-h} \\ &< \max_{1\le i\le k} \varphi_i^j/\vartheta \\ &< 2\max_{1\le i\le k} |\delta_i^j|/\vartheta.\end{aligned}$$

Then, taking the relationship between $\mathbf{e}_k^j$ and $\mathbf{e}_k$ into consideration, $\|\mathbf{e}_k\|_2$ can be derived as

$$\lim_{k\to\infty} \|\mathbf{e}_k\|_2 < 2m \sup_{1\le i\le k, 1\le j\le m} |\delta_i^j|/\vartheta. \tag{26}$$

After adding the residual part $O(\Delta^2)$, the value of $\|\mathbf{e}_k\|_2$ is described as

$$\lim_{k\to\infty} \|\mathbf{e}_k\|_2 < 2m \sup_{1\le i\le k, 1\le j\le m} |\delta_i^j|/\vartheta + O\left(\Delta^2\right). \tag{27}$$

The proof is thus completed. ■

IV. Illustrative Examples

In this section, the corresponding numerical simulations are conducted to indicate that the proposed MNI algorithm (8) possesses the outstanding robustness and stability when solving the dynamic systems of linear equations.

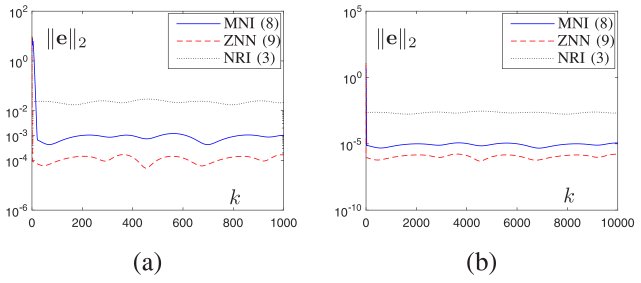


Fig. 3. Convergence performance of the steady-state error of three different algorithms, the proposed MNI algorithm (8), ZNN algorithm (9), and NRI algorithm (3) under zero noise condition. (a) $\Delta = 0.01$ s. (b) $\Delta = 0.001$ s.

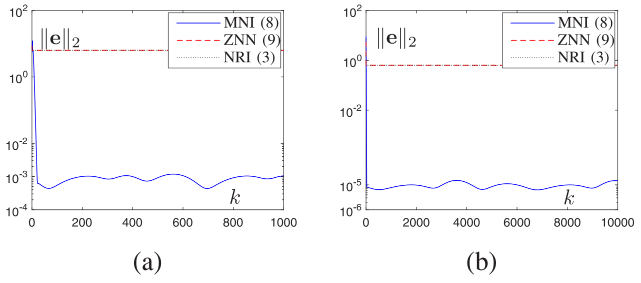


Fig. 4. Convergence performance of the steady-state error of three different algorithms, the proposed MNI algorithm (8), ZNN algorithm (9) and NRI algorithm (3) under constant noise condition $\omega(t) = 200$. (a) $\Delta = 0.01$ s. (b) $\Delta = 0.001$ s.

A. Four Types of Numerical Experiments

Above all, the unknown variable $\mathbf{x}(t)$ is calculated in the computing time interval $[t_0, t_f] \subseteq [0,10)$, and $P(t), \mathbf{q}(t)$ matrices are defined as

$$\begin{cases} P(t) = \begin{bmatrix} P_{11}(t) & P_{12}(t) & \cdots & P_{1n}(t) \\ P_{21}(t) & P_{22}(t) & \cdots & P_{2n}(t) \\ \vdots & \vdots & \ddots & \vdots \\ P_{m1}(t) & P_{m2}(t) & \cdots & P_{mn}(t) \end{bmatrix} \in \mathbb{R}^{m\times n} \\ \mathbf{q}(t) = \left[q_1(t) \quad q_2(t) \quad \cdots \quad q_m(t)\right]^{\mathrm{T}} \in \mathbb{R}^m \end{cases} \tag{28}$$

where P_{ab} and q_a indicate the abth argument of $P(t)$ and the ath argument of $\mathbf{q}(t)$, respectively. For ensuring the feasibility of the numerical simulations, matrix $P(t)$ should be designed as $P_{ab}(t) = a + \sin(t)$, when $a = b$; $P_{ab}(t) = \cos(t)/(a-b)$, when $a > b$; $P_{ab}(t) = \sin(t)/(a-b)$, when $a < b$. Meanwhile, vector $\mathbf{q}(t)$ is set as $\mathbf{q}(t) = \cos(t)$, when the subscript of argument a is odd; and $\mathbf{q}(t) = \sin(t)$, when the subscript of argument a is even. Before the numerical simulation, m and n are set to 10 and 11. Thus, the comparative experimental results are furnished in Fig. 3 through Fig. 6 and Table I, where the NRI algorithm (3) and ZNN algorithm (9) are selected as the comparison solving methods. Besides, the steady-state error $\|\mathbf{e}\|_2$ is chosen as the index performance for convergence in the visualization results. Moreover, the maximal steady-state residual error (MSSRE) and the average computing time per updating (ACTPU) are recorded in Table I as joint support evaluating index.

1) Zero Noise Condition: As plotted in Fig. 3, the convergence of the steady-state error of NRI algorithm (3) is

TABLE I
MSSRE AND ACTPU IN BRACKETS GENERATED BY THE PROPOSED MNI ALGORITHM (8), ZNN ALGORITHM (9), AND NRI ALGORITHM (3) FOR SOLVING THE DYNAMIC SYSTEMS OF LINEAR EQUATIONS (28) UNDER DIFFERENT SAMPLING PERIODS AND NOISE CONDITIONS

τ(s)	Algorithm	MSSRE and ACTPU (s) in brackets with different kinds of noises			
		Zero noise $\omega(t)=0$	Constant noise $\omega(t)=200$	Linear noise $\omega(t)=200*(t+1)$	Random noise $\omega(t)\in 2*[-1,1]+200$
0.01	MNI (8)	$1.054\times10^{-3}(4.690\times10^{-5})$	$1.046\times10^{-3}(4.439\times10^{-5})$	$2.536\times10^{-1}(4.318\times10^{-5})$	$5.716\times10^{-2}(4.895\times10^{-5})$
	ZNN (9)	$1.676\times10^{-4}(3.321\times10^{-5})$	$6.324\times10^{0}(3.371\times10^{-5})$	$6.944\times10^{1}(3.315\times10^{-5})$	$6.352\times10^{0}(3.330\times10^{-5})$
	NRI (3)	$2.487\times10^{-2}(1.433\times10^{-5})$	$6.337\times10^{0}(1.441\times10^{-5})$	$6.957\times10^{1}(1.439\times10^{-5})$	$6.356\times10^{0}(1.435\times10^{-5})$
0.001	MNI (8)	$1.159\times10^{-5}(4.276\times10^{-5})$	$1.472\times10^{-5}(4.362\times10^{-5})$	$2.536\times10^{-2}(4.529\times10^{-5})$	$5.212\times10^{-3}(4.126\times10^{-5})$
	ZNN (9)	$1.648\times10^{-6}(3.626\times10^{-5})$	$6.324\times10^{-1}(3.401\times10^{-5})$	$6.955\times10^{0}(3.876\times10^{-5})$	$6.361\times10^{-1}(3.387\times10^{-5})$
	NRI (3)	$2.490\times10^{-3}(1.537\times10^{-5})$	$6.346\times10^{-1}(1.456\times10^{-5})$	$6.957\times10^{0}(1.448\times10^{-5})$	$6.381\times10^{-1}(1.443\times10^{-5})$

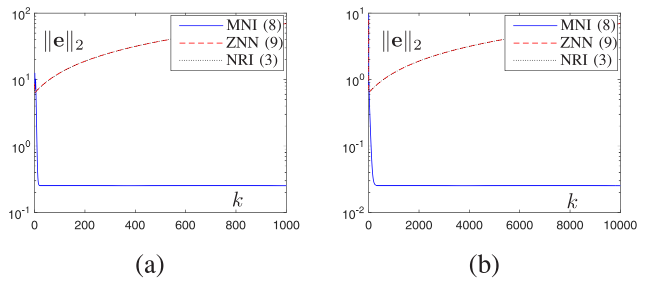


Fig. 5. Convergence performance of the steady-state error of three different algorithms, the proposed MNI algorithm (8), ZNN algorithm (9), and NRI algorithm (3) under linear noise condition $\omega(t) = 200*(t+1)$. (a) $\Delta = 0.01$ s. (b) $\Delta = 0.001$ s.

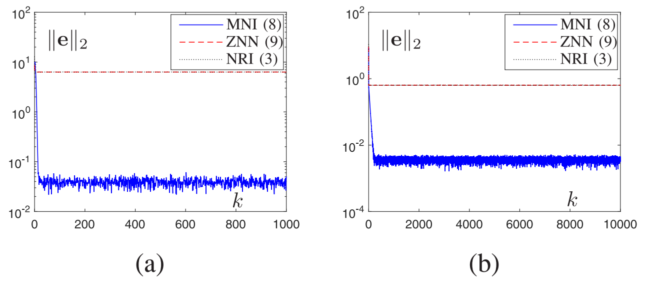


Fig. 6. Convergence performance of the steady-state error of three different algorithms, the proposed MNI algorithm (8), ZNN algorithm (9), and NRI algorithm (3) under random noise condition $\omega(t) \in 2*[-1, 1]+200$. (a) $\Delta = 0.01$ s. (b) $\Delta = 0.001$ s.

the worst among the three algorithms, whose MSSRE are 2.487×10^{-2} and 2.487×10^{-3} with the sampling period $\Delta = 0.01$ s and $\Delta = 0.001$ s, respectively. Then, exploiting the time-difference information, ZNN algorithm (9) can get a more accurate computing solution with the velocity compensation for dynamic parameters, whose MSSRE are the smallest than others from Table I. Without the aid of time-difference term, the proposed MNI algorithm (8) can also provide a pleasurable computing solution. As Fig. 3 and Table I depicted, when the sampling period Δ changes from 0.01 s to 0.001 s, the MSSREs generated by the MNI algorithm (8) changes from 1.054×10^{-3} to 1.159×10^{-5} with an $O(\Delta^2)$ changing manner, which confirms the correctness of Theorem 2. In addition, in terms of computational efficiency, the ACTPU of the three algorithm are all at 10^{-5} s order, while the NRI algorithm (3) performs the best.

2) Constant Noise Condition: As demonstrated in Fig. 4(a), the steady-state error of the proposed MNI algorithm (8) is able to rapidly converge to 10^{-3}. On the contrary, the accuracy of the computing solutions of ZNN algorithm (9) as well as NRI algorithm (3) tremendously drop to 10^0. In comparison with Figs. 3(a) and 4(a), it can be found that the proposed MNI algorithm (8) still possesses a superior performance, which is attributed to the error-summation term. Unfortunately, ZNN algorithm (9) and NRI algorithm (3) cannot attain a promising result, which reflects that constant noise has huge influence on their computing solutions. Besides, Fig. 4(b) also exhibits the advantage of the proposed MNI algorithm (8) under the sampling period 0.001 s. Not only that, from Table I, the MSSREs generated by the MNI algorithm also maintain an $O(\Delta^2)$ changing manner when perturbed by the constant noise, which also verifies the correctness of Theorem 3.

3) Linear Noise Condition: Beyond the above numerical experiments, the linear noise condition should be considered for the integrity of the proposed MNI algorithm (8). As can be viewed from Fig. 5, the proposed MNI algorithm (8) also possesses outstanding effectiveness under the linear noise condition $\omega(t) = 200 * (t + 1)$, while those of the NRI algorithm (3) and ZNN algorithm (9) are of divergence. The simulation results of Table I demonstrate that the MSSREs of the proposed MNI algorithm (8) are 2.536×10^{-1} for $\Delta = 0.01$ s and 2.536×10^{-2} for $\Delta = 0.001$ s with an $O(\Delta)$ changing manner, which is consistent with the conclusion given by Theorem 4. Instead, the ZNN algorithm (9) and NRI algorithm (3) fail to overcome the effects of the linear noise components, whose MSSREs are both infinite.

4) Random Noise Condition: In any system, the random noise components are inevitable. Thus, it is extremely necessary to contemplate the random noise impacting on the system for further comparing the effectiveness of the three different algorithms. As Fig. 6 exemplifies, the steady-state error of the proposed MNI algorithm (8) is much small than that of ZNN algorithm (9) and NRI algorithm (3) in different sampling period, i.e., 0.01 s and 0.001 s. From Table I, the MSSREs obtained by the MNI algorithm (8) are 5.716×10^{-2} and 5.212×10^{-3} when the sampling periods are $\Delta = 0.01$ s and $\Delta = 0.001$ s, while those of the NRI algorithm (3) and ZNN algorithm (9) are of the order 10^0 and 10^{-1}, respectively. Briefly speaking, observing the random noise numerical experiments, it can further verify that the proposed MNI algorithm (8) performs better effectiveness and superiority.

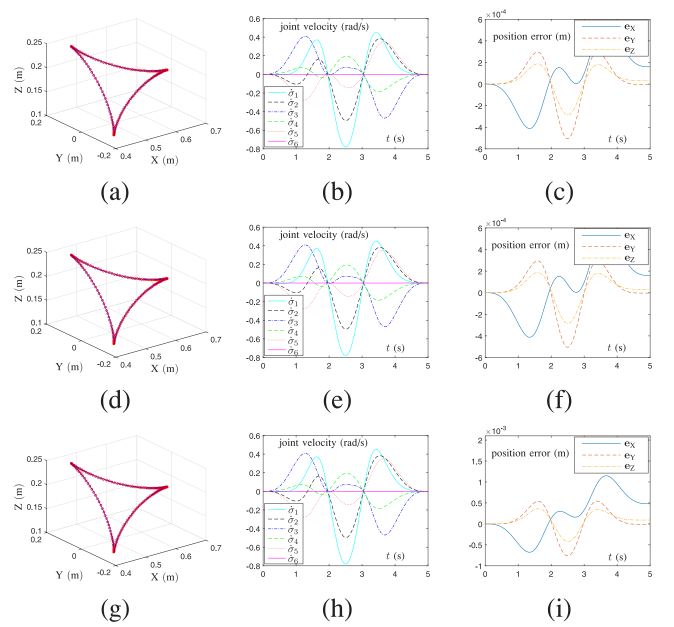


Fig. 7. Computer simulation results of PUMA 560 manipulator tracking a tricuspid valve path by employing the proposed MNI algorithm (8), ZNN algorithm (9), and NRI algorithm (3) under zero noise condition. (a) Desired path with blue lines and actual trajectory with red dash-dot lines generated by the MNI algorithm (8). (b) Historical states of joint velocity generated by the MNI algorithm (8). (c) Historical states of position error generated by the MNI algorithm (8). (d) Desired path with blue lines and actual trajectory with red dash-dot lines generated by the ZNN algorithm (9). (e) Historical states of joint velocity generated by the ZNN algorithm (9). (f) Historical states of position error generated by the ZNN algorithm (9). (g) Desired path with blue lines and actual trajectory with red dash-dot lines generated by the NRI algorithm (3). (h) Historical states of joint velocity generated by the NRI algorithm (3). (i) Historical states of position error generated by the NRI algorithm (3).

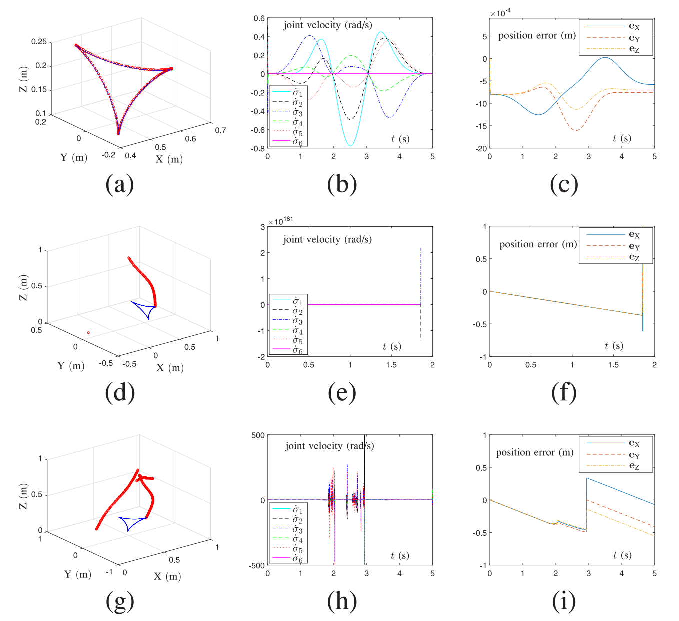


Fig. 8. Computer simulation results of PUMA 560 manipulator tracking a tricuspid valve path by employing the proposed MNI algorithm (8), ZNN algorithm (9), and NRI algorithm (3) under constant noise $\omega(t) = 200$ condition. (a) Desired path with blue lines and actual trajectory with red dash-dot lines generated by the MNI algorithm (8). (b) Historical states of joint velocity generated by the MNI algorithm (8). (c) Historical states of position error generated by the MNI algorithm (8). (d) Desired path with blue lines and actual trajectory with red dash-dot lines generated by the ZNN algorithm (9). (e) Historical states of joint velocity generated by the ZNN algorithm (9). (f) Historical states of position error generated by the ZNN algorithm (9). (g) Desired path with blue lines and actual trajectory with red dash-dot lines generated by the NRI algorithm (3). (h) Historical states of joint velocity generated by the NRI algorithm (3). (i) Historical states of position error generated by the NRI algorithm (3).

TABLE II
TOTAL COMPUTING TIME GENERATED BY THE MNI ALGORITHM (8), ZNN ALGORITHM (9), AND NRI ALGORITHM (3) FOR MANIPULATING PUMA 560 MANIPULATOR TO TRACK A TRICUSPID VALVE PATH UNDER DIFFERENT NOISE CONDITIONS

Algorithm	Total computing time (s)			
	Zero noise $\omega(t) = 0$	Constant noise $\omega(t) = 200$	Linear noise $\omega(t) = 200 * (t + 1)$	Random noise $\omega(t) = 2 * [-1, 1] + 200$
MNI (8)	3.168	3.203	3.272	3.451
ZNN (9)	3.119	8.733	11.028	9.286
NRI (3)	2.623	3.108	3.065	2.989

V. Robot Control Application

Robot manipulators have been widely investigated and expanded in industrial production and practical applications [31] and [35] . Therefore, this section furnishes the applications of the proposed MNI algorithm (8) on robot control, including multiple computer simulations and physical experiments under different noisy conditions. First of all, on the foundation of mechanical kinematics [32], the motion tracking formula can be received as

$$J(\sigma(t))\dot{\sigma}(t) = \dot{\mathbf{r}}_d(t) \tag{29}$$

where $J(\sigma(t)) \in \mathbb{R}^{3\times 6}$ is the Jacobian matrix [36]; $\dot{\sigma}(t) \in \mathbb{R}^6$ denotes the joint velocity of six-axis joints; $\dot{\mathbf{r}}_d(t) \in \mathbb{R}^3$ means the end-effector velocity. The above kinematics formula (29) can be deduced in a discrete form with the MNI algorithm (8).

$$\dot{\sigma}_{k+1} = J_k^{\dagger}\dot{\mathbf{r}}_k - \beta J_k^{\dagger}\sum_{i=0}^{k}(J_k\dot{\sigma}_k - \dot{\mathbf{r}}_k). \tag{30}$$

In the following tracking control application, the used robot manipulators involve the PUMA 560 for computer simulations and Franka Emika Panda redundant manipulator for physical experiments, both of which the structure and dynamics parameters could refer to [33], [34]. As far as the fair comparisons are concerned, it is fixed as $\vartheta = \eta = 0.25$ with the sampling period $\Delta = 0.01$ s and task execution time $T = 5$ s for the following computer simulations and physical experiments, respectively.

A. Computer Simulations

In this part, a classical PUMA 560 robot arm manipulation is exploited to carry out a tracking control application with a tricuspid valve path. Simulations are conducted on the MindSpore framework, and the correlative simulation results are presented in Fig. 7 through Fig. 10 and Table II with comparisons among the proposed MNI algorithm (8), ZNN algorithm (9), and NRI algorithm (3) under zero noise, constant noise, linear noise, and random noise conditions.

Specifically, Fig. 7 shows the simulation results generated by three algorithms under a zero noise $\omega(t) = 0$ condition, where the produced actual trajectories are well consistent with

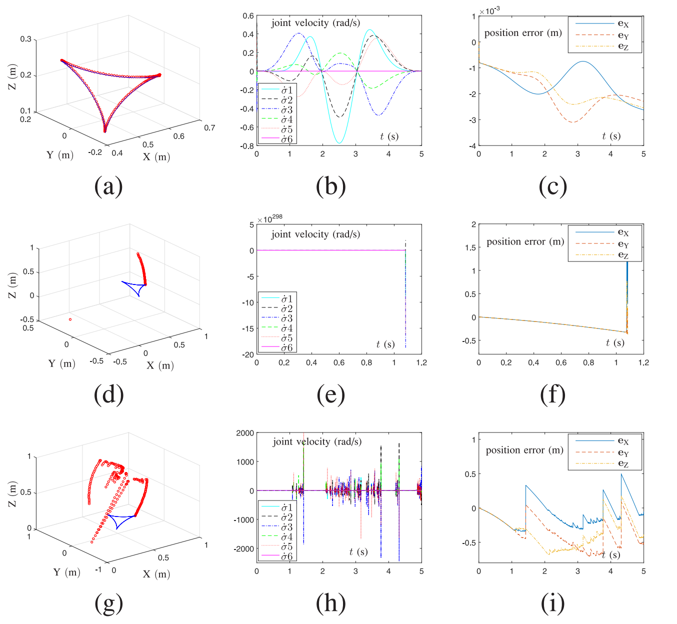


Fig. 9. Computer simulation results of PUMA 560 manipulator tracking a tricuspid valve path by employing the proposed MNI algorithm (8), ZNN algorithm (9), and NRI algorithm (3) under linear noise $\omega(t) = 200 * (t+1)$ condition. (a) Desired path with blue lines and actual trajectory with red dash-dot lines generated by the MNI algorithm (8). (b) Historical states of joint velocity generated by the MNI algorithm (8). (c) Historical states of position error generated by the MNI algorithm (8). (d) Desired path with blue lines and actual trajectory with red dash-dot lines generated by the ZNN algorithm (9). (e) Historical states of joint velocity generated by the ZNN algorithm (9). (f) Historical states of position error generated by the ZNN algorithm (9). (g) Desired path with blue lines and actual trajectory with red dash-dot lines generated by the NRI algorithm (3). (h) Historical states of joint velocity generated by the NRI algorithm (3). (i) Historical states of position error generated by the NRI algorithm (3).

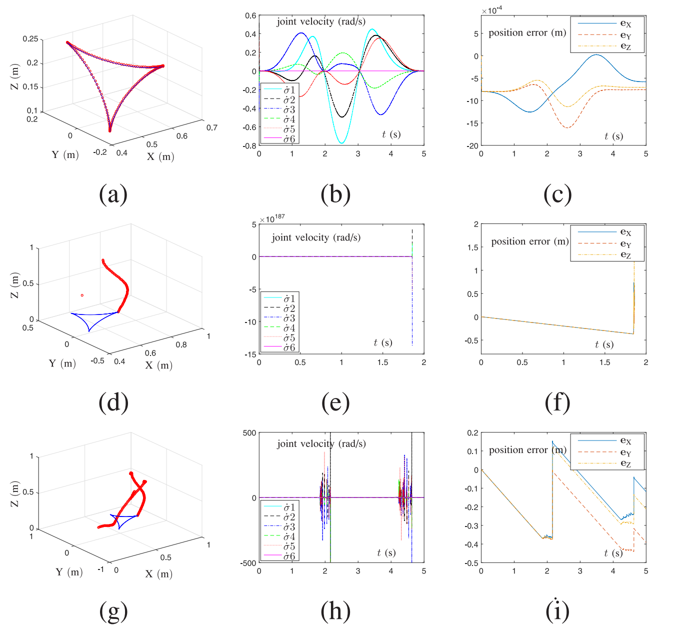
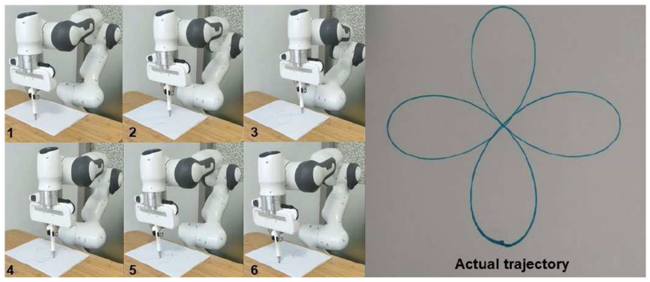


Fig. 10. Computer simulation results of PUMA 560 manipulator tracking a tricuspid valve path by employing the proposed MNI algorithm (8), ZNN algorithm (9), and NRI algorithm (3) under random noise $\omega(t) = 2*[-1, 1]+200$ condition. (a) Desired path with blue lines and actual trajectory with red dash-dot lines generated by the MNI algorithm (8). (b) Historical states of joint velocity generated by the MNI algorithm (8). (c) Historical states of position error generated by the MNI algorithm (8). (d) Desired path with blue lines and actual trajectory with red dash-dot lines generated by the ZNN algorithm (9). (e) Historical states of joint velocity generated by the ZNN algorithm (9). (f) Historical states of position error generated by the ZNN algorithm (9). (g) Desired path with blue lines and actual trajectory with red dash-dot lines generated by the NRI algorithm (3). (h) Historical states of joint velocity generated by the NRI algorithm (3). (i) Historical states of position error generated by the NRI algorithm (3).

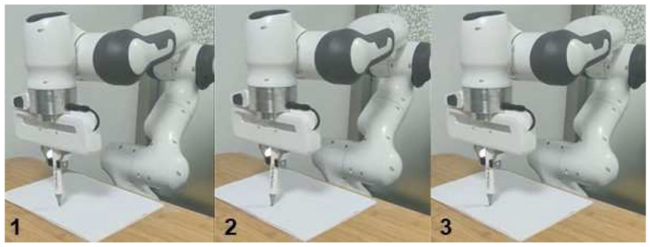
the desired path and their historical states of position error are all order of 10^{-4} m. In addition, as Fig. 7(b), (e), and (h) present, the joint velocity generated by all algorithms returns to their own original states. As a continuation, Figs. 8, 9, and 10 provide the corresponding simulation results generated by these algorithms with respect to the constant noise $\omega(t) = 200$, linear noise $\omega(t) = 200 * (t + 1)$, and random noise $\omega(t) = 2 * [-1, 1] + 200$ conditions, respectively. From Fig. 8 through Fig. 10, it is clear that the ZNN algorithm (9) and NRI algorithm (3) cannot handle the tracking control tasks in any noisy environment, where the generated actual trajectories are grossly different from the desired ones. Besides, the resulting historical states of joint velocity and position error are turbulent and messy. In contrast, the corresponding results of the MNI algorithm (8) are performing-well, as shown in subgraphs (a)–(c) of Figs. 8–10, where the historical states of velocity are basically the same as under zero noise condition and the position error maintains the maximum error less than 3.0×10^{-3} m. Afterwards, Table II records the total computing time consumed by various algorithms to complete the tracking control task under various noise conditions. Under zero noise condition, the total computing time of these algorithms are not much different, i.e., 3.168, 3.119, and 2.623 s, respectively, all which are shorter than the task execution time $T = 5$ s. In addition, the total computational time of the MNI algorithm (8) is 0.049 s longer than that of the ZNN algorithm (9) and 0.545 s longer than that of the NRI algorithm (3). Under other noisy conditions, the total computing time of the proposed MNI algorithm (8) still maintains an excellent performance without any case exceeding 3.5 s. Additionally, the total computational time of the NRI algorithm (3) is 0.095 s, 0.207 s, and 0.462 s shorter than those of the MNI algorithm (8) under constant noise, linear noise, and random noise workspaces, respectively. However, the NRI algorithm (3) fails to complete track control with noise perturbations, which can be confirmed from subgraphs (g)–(i) of Figs. 8–10. Further, the total computational time of the ZNN algorithm (9) is over the task duration 5 s under any noise perturbations, and also cannot complete the track control task, whose corresponding results are presented in subgraphs (d)–(f) of Figs. 8–10. In general, the MNI algorithm (8) sacrifices computational efficiency to obtain the excellent noise-suppressing ability that is quite feasible and efficient in the practical industrial environments.

B. Physical Experiments

In this part, to illustrate the noise tolerance capability of the proposed MNI algorithm (8) vividly and convincingly, physical experiments on the Franka Emika Panda redundant manipulator with seven degrees of freedom and a marking pen attached to its end effector are executed driven by the



(a)



(b)

Fig. 11. Snapshots of the physical experiments on the Franka Emika Panda redundant manipulator for tracking a four-leaf clover path under linear noise $\omega(t) = k\Delta/T$ workspace. (a) Physical experiments generated by the proposed MNI algorithm (8). (b) Physical experiments generated by the NRI algorithm (3).

proposed MNI algorithm (8) and NRI algorithm (3) for tracking a four-leaf clover path under linear noise $\omega(t) = k\Delta/T$ workspace. In addition, note that the desired trajectory data ($\mathbf{r}_d$, $\dot{\mathbf{r}}_d$, and $\ddot{\mathbf{r}}_d$) are default in advance; the status information of the Franka Emika Panda redundant manipulator (σ, $\dot{\sigma}$, $\mathbf{r}$, and $\dot{\mathbf{r}}$) are obtained in real-time measurement; the real-time Jacobian matrix J can be obtained directly during program running from interface of manipulator. On top of that, the physical experiments on the Franka Emika Panda redundant manipulator are controlled by C++ in the ROS system.

After completing the above statement about the workspace and special content of the physical experiments, the snapshots of the corresponding experimental results generated by the proposed MNI algorithm (8) and NRI algorithm (3) and the video are presented in Fig. 11 and at https://youtu.be/WVLJU9Pz0LM and https://youtu.be/27bEdItiVRM, respectively. It is worth pointing out that as demonstrated in Fig. 11(a), even under the linear noise $\omega(t) = k\Delta/T$ workspace, the Franka Emika Panda redundant manipulator driven by the proposed MNI algorithm (8) presents excellent noise-tolerance ability with tiny position error. Unfortunately, due to the lack of effective noise-tolerance ability, the position error generated by the NRI algorithm (3) between the end effector of the Franka Emika Panda redundant manipulator and the desired task is too large, leading to a strong press force between the end effector and the desktop, which can be well confirmed by the oblique marking pen from Fig. 11(b). At the same time, the error information "motion aborted by reflex" promptly appears in the ROS system of the host computer, and immediately triggers the self-protection mechanism of the Franka Emika Panda redundant manipulator operation system, which causes its end effector to stop functioning. This also means that the Franka Emika Panda redundant manipulator driven by the NRI algorithm (3) cannot complete the trajectory tracking task under linear noise $\omega(t) = k\Delta/T$ workspace.

C. Summary

In general, all the results of computer simulation and physical experiments reveal that the proposed MNI algorithm (8) possesses the high-precision tracking control performance under noisy conditions, which accords with conclusions drawn in the part of numerical experiments. In comparison with the other existing methods, the proposed MNI algorithm (8) not only has advantages in theoretical computer simulation, but also exhibits its availability and capability in physical experiments, both of which verify the strong robustness and potential prospects of the proposed MNI algorithm (8).

VI. Conclusion

In this article, a novel MNI algorithm has been proposed on the perspective of control theory referred to as the MNI algorithm (8). Different from the conventional continuous-time algorithms, the proposed MNI algorithm has been directly designed in the discrete-time form. Through the related theoretical analyses and proofs, it is concluded that the proposed MNI algorithm possesses the superior noise tolerance. In view of the results of the numerical simulations, compared with the ZNN algorithm (9) and the NRI algorithm (3), the steady-state error of the proposed MNI algorithm is the smallest under the noisy circumstances. Furthermore, multiple computer simulations and physical experiments on robot control applications have been carried out, which further substantiates that the proposed MNI algorithm is capable of withstanding the disturbance brought by external and internal noises.

References

[1] W. He, Z. Li, Y. Dong, and T. Zhao, "Design and adaptive control for an upper limb robotic exoskeleton in presence of input saturation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 1, pp. 97–108, Jan. 2019.

[2] D. Guo, Z. Nie, and L. Yan, "Novel discrete-time Zhang neural network for time-varying matrix inversion," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 8, pp. 2301–2310, Aug. 2017.

[3] S. Al-Wais, R. Mohajerpoor, L. Shanmugam, H. Abdi, and S. Nahavandi, "Improved delay-dependent stability criteria for telerobotic systems with time-varying delays," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 48, no. 12, pp. 2470–2484, Dec. 2018.

[4] M. Van, M. Mavrovouniotis, and S. S. Ge, "An adaptive backstepping nonsingular fast terminal sliding mode control for robust fault tolerant control of robot manipulators," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 49, no. 7, pp. 1448–1458, Jul. 2019.

[5] D. Nakhaeinia, P. Payeur, and R. Laganiere, "A mode-switching motion control system for reactive interaction and surface following using industrial robots," *IEEE/CAA J. Autom Sinica*, vol. 5, no. 3, pp. 670–682, May 2018.

[6] X. Xiao, N. N. Xiong, J. Lai, C.-D. Wang, Z. Sun, and J. Yan, "A local consensus index scheme for random-valued impulse noise detection systems," *IEEE Trans. Syst., Man, Cybern., Syst.*, early access, Jul. 23, 2019, doi: 10.1109/TSMC.2019.2925886.

[7] Z. Cao, Q. Xiao, R. Huang, and M. Zhou, "Robust neuro-optimal control of underactuated snake robots with experience replay," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 1, pp. 208–217, Jan. 2018.

[8] M. Chen, "Robust tracking control for self-balancing mobile robots using disturbance observer," *IEEE/CAA J. Autom Sinica*, vol. 4, no. 3, pp. 458–465, Jul. 2017.

[9] X. Luo, M. Zhou, Y. Xia, Q. Zhu, A. C. Ammari, and A. Alabdulwahab, "Generating highly accurate predictions for missing QoS data via aggregating nonnegative latent factor models," *IEEE Trans. Neural Netw. Learn. Syst.*, vol 27, no. 3, pp. 524–537, Mar. 2016.

[10] X. Luo, M. Zhou, S. Li, Y. Xia, Z.-H. You, Q. Zhu, and H. Leung, "Incorporation of efficient second-order solvers into latent factor models for accurate prediction of missing QoS data," *IEEE Trans. Cybern.*, vol. 48, no. 4, pp. 1216–1228, Apr. 2018.

[11] H. Asadi, S. Mohamed, C. P. Lim, and S. Nahavandi, "Robust optimal motion cueing algorithm based on the linear quadratic regulator method and a genetic algorithm," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 2, pp. 238–254, Feb. 2017.

[12] C. Chen, H. Modares, K. Xie, F. L. Lewis, Y. Wan, and S. Xie, "Reinforcement learning-based adaptive optimal exponential tracking control of linear systems with unknown dynamics," *IEEE Trans. Autom. Control.*, vol. 64, no. 11, pp. 4423–4438, Nov. 2019.

[13] S. Jahandari, A. Kalhor, and B. N. Araabi, "Online forecasting of synchronous time series based on evolving linear models," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 50, no. 5, pp. 1865–1876, May 2020, doi: 10.1109/TSMC.2018.2789936.

[14] Y. Saad and H. A. van der Vorst, "Iterative solution of linear systems in the 20th century," *J. Comput. Appl. Math.*, vol. 123, nos. 1–2, pp. 1–33, Nov. 2000.

[15] W. W. Hager and H. Zhang, "A survey of nonlinear conjugate gradient methods," *Pac. J. Optim.*, vol. 2, no. 1, pp. 35–58, 2006.

[16] S. Liao, J. Liu, X. Xiao, D. Fu, G. Wang, and L. Jin, "Modified gradient neural networks for solving the time-varying Sylvester equation with adaptive coefficients and elimination of matrix inversion," *Neurocomputing*, vol. 379, pp. 1–11, Feb. 2020.

[17] L. V. Ferreira, E. Kaszkurewicz, and A. Bhaya, "Solving systems of linear equations via gradient systems with discontinuous righthand sides: Application to LS-SVM," *IEEE Trans. Neural Netw.*, vol. 16, no. 2, pp. 501–505, Mar. 2005.

[18] B. Polyak and A. Tremba, "New versions of Newton method: Step-size choice, convergence domain and under-determined equations," *Optim. Methods Softw.*, vol. 35, no. 6, pp. 1272–1303, 2020.

[19] D. Hezari, D. K. Salkuyeh, and V. Edalatpour, "A new iterative method for solving a class of complex symmetric system of linear equations," *Numer. Algorithms*, vol. 73, pp. 927–955, Mar. 2016.

[20] L. Wei, L. Jin, C. Yang, K. Chen, and W. Li, "New noise-tolerant neural algorithms for future dynamic nonlinear optimization with estimation on Hessian matrix inversion," *IEEE Trans. Syst., Man, Cybern., Syst.*, early access, May 27, 2019, doi: 10.1109/TSMC.2019.2916892.

[21] S. Li, M. Zhou, and X. Luo, "Modified primal-dual neural networks for motion control of redundant manipulators with dynamic rejection of harmonic noises," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 10, pp. 4791–4801, Oct. 2018.

[22] D. Chen and Y. Zhang, "Robust zeroing neural-dynamics and its time-varying disturbances suppression model applied to mobile robot manipulators," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 9, pp. 4385–4397, Sep. 2018.

[23] H. Lu, L. Jin, X. Luo, B. Liao, D. Guo, and 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.*, vol. 15, no. 11, pp. 5931–5942, Nov. 2019.

[24] L. Xiao, S. Li, K. Li, L. Jin, and B. Liao, "Co-design of finite-time convergence and noise suppression: A unified neural model for time varying linear equations with robotic applications," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 50, no. 12, pp. 5233–5243, Dec. 2020, doi: 10.1109/TSMC.2018.2870489.

[25] L. Jin, Y. Zhang, and S. Li, "Integration-enhanced Zhang neural network for real-time-varying matrix inversion in the presence of various kinds of noises," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 12, pp. 2615–2627, Dec. 2016.

[26] L. Jin, Y. Zhang, and B. Qiu, "Neural network-based discrete-time Z-type model of high accuracy in noisy environments for solving dynamic system of linear equations," *Neural Comput. Appl.*, vol. 29, no. 11, pp. 1217–1232, 2018.

[27] D. Guo, Z. Nie, and 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.*, vol. 48, no. 12, pp. 2188–2197, Dec. 2018.

[28] W. E. Leithead and Y. Zhang, "$O(N^2)$-operation approximation of covariance matrix inverse in Gaussian process regression based on quasi-Newton BFGS method," *Commun. Stat. Simulat. Comput.*, vol. 36, no. 2, pp. 367–380, Mar. 2007.

[29] Y. Zhang, D. Guo, Y. Yin, L. Jin, and Y. Chou, "Taylor-type 1-step-ahead numerical differentiation rule for first-order derivative approximation and ZNN discretization," *J. Comput. Appl. Math.*, vol. 273, pp. 29–40, Jan. 2015.

[30] R. Hunger, "Floating point operations in matrix-vector calculus," Assoc. Inst. Signal Process., Technische Universität München, München, Germany, Rep., 2007.

[31] W. He, X. He, M. Zou, and H. Li, "PDE model-based boundary control design for a flexible robotic manipulator with input backlash," *IEEE Trans. Control Syst. Technol.*, vol. 27, no. 2, pp. 790–797, Mar. 2019.

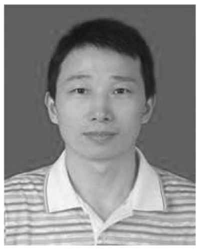
[32] D. Chen, S. Li, F.-J. Lin, and Q. Wu, "New super-twisting zeroing neural-dynamics model for tracking control of parallel robots: A finite-time and robust solution," *IEEE Trans. Cybern.*, vol. 50, no. 6, pp. 2651–2660, Jun. 2020, doi: 10.1109/TCYB.2019.2930662.

[33] Z. Xie, L. Jin, X. Du, X. Xiao, H. Li, and S. Li, "On generalized RMP scheme for redundant robot manipulators aided with dynamic neural networks and nonconvex bound constraints," *IEEE Trans. Ind. Informat.*, vol 15, no. 9, pp. 5172–5181, Sep. 2019.

[34] C. Gaz, M. Cognetti, A. Oliva, P. R. Giordano, and A. D. Luca, "Dynamic identification of the Franka Emika Panda robot with retrieval of feasible parameters using penalty-based optimization," *IEEE Robot. Autom. Lett.*, vol. 4, no. 4, pp. 4147–4154, Oct. 2019.

[35] Z. Xie, L. Jin, X. Luo, Z. Sun, and M. Liu, "RNN for repetitive motion generation of redundant robot manipulators: An orthogonal projection based," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Oct. 20, 2020, doi: 10.1109/TNNLS.2020.3028304.

[36] L. Jin, Z. Xie, M. Liu, C. Ke, C. Li, and C. Yang, "Novel joint-drift-free scheme at acceleration level for robotic redundancy resolution with tracking error theoretically eliminated," *IEEE/ASME Trans. Mechatronics*, early access, Jun. 11, 2020, doi: 10.1109/TMECH.2020.3001624.



Dongyang Fu received the Ph.D. degree in physical oceanography from the South China Sea Institute of Oceanology, Chinese Academy of Sciences, Beijing, China, in 2010.

He was a Postdoctoral Fellow with the State Key Laboratory of Satellite Ocean Environment Dynamics, Second Institute of Oceanography, State Oceanic Administration, Guangzhou, in 2010. He is currently a Professor with the School of Electronics and Information Engineering, Guangdong Ocean University, Zhanjiang, China. His current research interests include ocean color remote sensing and its application, remote sensing in off-shore water quality, response of upper ocean to typhoon, and neural networks.

Haoen Huang received the B.E. degree in electrical engineering and automation from Guangdong Ocean University, Zhanjiang, China, in 2019, where he is currently pursuing the M.Agr. degree in agricultural engineering and information technology with the School of Electronics and Information Engineering.

His current research interests include Newton algorithm, neural networks, and robotics.

Lin Wei received the B.E. degree in electrical engineering and information from the Beijing Institute of Technology, Beijing, China, in 2018. She is currently pursuing the Ph.D. degree in computer application technology with the School of Information Science and Engineering, Lanzhou University, Lanzhou, China.

Her research interests include neural networks and robotics.

Xiuchun Xiao received the Ph.D. degree in communication and information system from Sun Yat-sen University, Guangzhou, China, in 2013.

He is currently a Full Professor with the School of Electronics and Information Engineering, Guangdong Ocean University, Zhanjiang, China. His current research interests include image processing, artificial neural networks, and computer vision.

Long Jin (Member, IEEE) received the B.E. degree in automation and the Ph.D. degree in information and communication engineering from Sun Yat-sen University, Guangzhou, China, in 2011 and 2016, respectively.

He had completed the Postdoctoral training with the Department of Computing, Hong Kong Polytechnic University, Hong Kong, from 2016 to 2017. His current research interests include neural networks, robotics, optimization, and intelligent computing.

Dr. Jin received the Excellent Doctoral Dissertation Award of the Chinese Association for Artificial Intelligence.

Shan Liao received the master's degree in software engineering from the Guangdong University of Technology, Guangzhou, China, in 2012. She is currently pursuing the Doctoral degree in cybersecurity with Graduate School, Sichuan University, Chengdu, China.

Her current research interests include remote sensing signal processing and algorithms, artificial neural network, and communication technology.

Jialiang Fan received the B.E. degree in software engineer from Shandong University, Jinan, China, in 2019. He is currently pursuing the M.E. degree in computer technology with the School of Information Science and Engineering, Lanzhou University, Lanzhou, China.

His main research interests include robotics, neural networks, intelligent information processing, artificial intelligence, and optimization theory.

Zhengtai Xie received the B.E. degree in electronic information science and technology from Lanzhou University, Lanzhou, China, in 2019, where he is currently pursuing the M.E. degree in communication and information systems.

His main research interests include robotics, neural networks, intelligent information processing, artificial intelligence, and optimization theory.