This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI10.1109/TNNLS.2024.3354936,

Efficient Predefined-Time Adaptive Neural Networks for Computing Time-Varying Tensor Moore-Penrose Inverse

Zhaohui Qi, Yingqiang Ning, Lin Xiao, Zidong Wang, *Fellow, IEEE* and Yongjun He

***Abstract*—This paper proposes predefined-time adaptive neural network (PTANN) and event-triggered PTANN (ET-PTANN) models to efficiently compute time-varying tensor Moore-Penrose inverse. The PTANN model incorporates a novel adaptive parameter and activation function, enabling it to achieve strongly predefined-time convergence. Unlike traditional time-varying parameters that increase over time, the adaptive parameter is proportional to the error norm, thereby better allocating computational resources and improving efficiency. To further enhance efficiency, the ET-PTANN model combines an event trigger with the evolution formula, resulting in the adjustment of step size and reduction of computation frequency compared to the PTANN model. By conducting mathematical derivations, the paper derives the upper bound of convergence time for the proposed neural network models and determines the minimum execution interval for the event trigger. A simulation example demonstrates that the PTANN and ET-PTANN models outperform other related neural network models in terms of computational efficiency and convergence rate. Finally, the practicality of the PTANN and ET-PTANN models is demonstrated through their application to mobile sound source localization.**

***Index Terms*—Recurrent neural network, adaptive parameter, event-triggering mechanism, time-varying tensor Moore-Penrose inverse, predefined-time convergence, sound source localization.**

I. Introduction

The Moore-Penrose (MP) inverse is a fundamental concept in science and engineering. For example, there are many image-based applications, such as image fusion [1], image classification [2], and gesture recognition [3]. Most of them involve matrix inversion operations. Moreover, the MP inverse finds applications in various fields, including network learning [4], robotics [5], and big data analysis [5]. With the rapid growth of data volume due to advancements in internet and hardware technologies, the need to handle large-scale data has become crucial. In response to this challenge, researchers have extended the MP inverse from matrices to tensors and have explored the properties of tensor MP inverse [6], [7].

This work was supported in part by the Natural Science Foundation of Hunan Province of China under Grants 2022RC1103 and 2021JJ20005, and in part by the National Natural Science Foundation of China under Grant 61866013. (*Corresponding author: Lin Xiao*)

Z. Qi, Y. Ning, L. Xiao, and Y. He are with Hunan Provincial Key Laboratory of Intelligent Computing and Language Information Processing, Hunan Normal University, Changsha 410081, China, and also with MOE-LCSM, Hunan Normal University, Changsha 410081, China (e-mail: xiaolin5@hunnu.edu.cn).

Z. Wang is with Department of Computer Science, Brunel University London, Uxbridge, Middlesex, UB8 3PH, United Kingdom (e-mail: zidong.wang@brunel.ac.uk).

The importance of the MP inverse has sparked the interest of numerous scholars, leading to the development of various methods for its computation [8]–[10]. For instance, Stanimirovic *et al.* [9] proposed an iterative scheme based on error bounds to compute the MP inverse, while Zontini *et al.* [10] employed the generalized Schulz iterative method. However, iterative methods with a time complexity of $O(n^3)$ are not suitable for handling large-scale data. Consequently, recurrent neural networks (RNNs) with parallel computing capabilities have been utilized for computing matrix and tensor equations. Among RNNs, gradient neural networks (GNNs) have shown proficiency in addressing time-invariant MP inverse problems [11]–[13]. Nevertheless, GNNs overlook the derivative information of coefficients, rendering them insufficient for solving time-varying tensor MP inverse (TV-TMPI) problems.

Numerous studies have proven that zeroing neural networks (ZNN) can effectively solve time-varying problems [14]–[19]. To address the time-varying matrix inverse, Zhang *et al.* [14] introduced the zeroing neural network, a unique type of RNN. The ZNN combines the advantages of GNNs while effectively handling time-varying problems. The ZNN model achieves exponential rate convergence to the theoretical solution. Over time, researchers have made continuous improvements to the convergence performance of ZNN models [15]–[19]. Tan *et al.* [15] proposed two activation functions with varying parameters to achieve finite-time convergence and robustness in ZNN models. The finite-time convergence of ZNN models is influenced by the initial state, leading researchers to develop fixed-time convergence ZNN models by designing new activation functions [16]–[18]. For example, Jin *et al.* [18] introduced two novel activation functions and presented two fixed-time convergence ZNN models. Furthermore, Qi *et al.* [19] proposed a novel ZNN model with strong predefined-time convergence, where the upper bound on convergence time is independent of other model parameters.

The ZNN model has been utilized by some researchers to address time-varying MP inverses. Two novel ZNN models, characterized by their non-linear activation functions, were introduced by Chai *et al.* [20] and Sowmya *et al.* [21]. It has been demonstrated that both ZNN models converge to the theoretical solution of the time-varying MP inverse within a finite time scale. However, it is worth noting that the aforementioned ZNN models come with increased computational costs as they aim to improve the convergence rate. To address this issue, Xiao *et al.* [22], [23] proposed adaptive neural network (ANN) models. These models utilize parameters that adapt to

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI10.1109/TNNLS.2024.3354936,

changes in the error norm, thereby enhancing the convergence rate and computational efficiency. Similarly, Wang *et al.* [24] pointed out that adaptive dynamic programming technology significantly reduces the computational burden of neural networks.

The current research landscape indicates substantial advancements in addressing the MP inverse and in the development of ZNN models. Nevertheless, there remain significant gaps and limitations in the field. Primarily, research on the time-varying tensor MP inverse is scant, despite some studies extending the MP inverse to the time-varying or tensor domain. Secondly, the existing ANN models, which include the optimized ZNN models for computational efficiency, exhibit subpar convergence performance and do not attain optimal predefined-time convergence. Thirdly, as the tensor order increases, the computational cost of the evolution formula grows rapidly. Moreover, the value of the evolution formula between adjacent iterations does not change significantly, resulting in inefficient utilization of computing resources. Thus, the focal issue addressed in this paper is the time-varying tensor Moore-Penrose inverse, where the research is directed towards enhancing the convergence performance and computational efficiency of the ANN model.

The convergence performance of the ANN model can be enhanced by introducing novel adaptive parameters and activation functions. Additionally, to reduce the computational frequency of the evolution formula, event-triggered control can be employed. Event-triggered control, based on the system state, offers significant improvements in computational efficiency compared to periodic control [25]. Consequently, the concept of event triggering has attracted considerable interest among researchers. For example, within the scope of the consistent tracking problem in multi-agent systems, an event-based finite-time control method was introduced by Li *et al.* [26]. This method not only guarantees that the tracking error converges to zero in a finite time, but also reduces resource waste. However, event-triggered control is susceptible to measurement errors. To address this, researchers have devised threshold strategies [27]–[29]. Inspired by these approaches, and to ensure the accuracy of the upper bound on convergence time, we propose a hybrid threshold. Furthermore, to further reduce computational costs, we adjust the value of the evolution formula within the trigger interval, thereby influencing the step size of the ode45 solver.

In summary, this paper introduces two neural network models: the predefined-time adaptive neural network (PTANN) model and the event-triggered PTANN (ET-PTANN) model. These models utilize novel activation functions and adaptive parameters to achieve strongly predefined-time convergence. The adaptive parameters optimize the utilization of computing resources, while the event-triggered mechanism, combined with the evolution formula, reduces unnecessary computations. Therefore, when compared with conventional solving methods, the PTANN and ET-PTANN models demonstrate a faster rate of convergence, greater accuracy in convergence time, and improved computational efficiency.

The main contributions of this paper are highlighted as follows:

1) The PTANN and ET-PTANN models are proposed to compute time-varying tensor MP inverse, both achieving strongly predefined-time convergence.
2) A novel adaptive parameter is introduced to allow for the flexible allocation of computing resources, thereby enhancing the computational efficiency of the ANN models.
3) A design of a hybrid threshold event trigger is presented, which integrates, for the first time, the evolution formula with the event triggering mechanism. This innovation is an improvement in computational efficiency with no loss of convergence rate.
4) The upper bound of convergence time for the proposed models and the minimum execution interval of the event trigger are calculated.

This paper is organized into six sections. Section II provides necessary preliminaries and presents the problem formulation. Section III describes the design process of the PTANN and ET-PTANN models. Theoretical analysis and mathematical proofs of the two proposed models are presented in Section IV. To validate the aforementioned theory, simulation examples were conducted in Section V. The application of the PTANN and ET-PTANN models to mobile sound source localization is demonstrated in Section VI. Section VII concludes the full paper and expresses future research direction.

II. Preliminaries and Problem Formulation

In this section, the relevant definitions of nonlinear systems and the knowledge of tensors will be introduced, with purpose to enhance the understanding of readers and provide an introduction to the problem formulation.

A. Nonlinear System

Consider a nonlinear system:

$$\dot{\boldsymbol{x}}(t) = -\lambda\Phi\left(\boldsymbol{x}(t), t\right) \in \mathbb{R}^n,\ \boldsymbol{x}(0) = \boldsymbol{x}_0,\ t \in [0, +\infty), \quad (1)$$

where $\lambda > 0$ is a tunable parameter; $\Phi(\cdot,\cdot) : \mathbb{R}^n \times \mathbb{R}_0^+ \to \mathbb{R}^n$ is a continuous monotonically increasing odd function; and $\mathbb{R}_0^+ = \{0\} \bigcup \mathbb{R}^+$. The equilibrium point of system (1) is $\boldsymbol{x}(t) = \boldsymbol{0}$.

Definition 1 ([30]). *If there exists a parameter $0 \leqslant \tau < +\infty$ such that $\boldsymbol{x}(t) = \boldsymbol{0}$ for all $t \geqslant \tau$, then system* (1) *is said to reach a stable state at time τ. In this case, the function $T(\boldsymbol{x_0}) = \inf\{\tau \geqslant 0 : \|\boldsymbol{x}(t)\|_2 = 0, \forall t > \tau\}$ is referred to as the settling-time function of system* (1)*.*

Definition 2 ([31]). *For system* (1)*, if there exists an independent parameter T satisfying*

$$T(\boldsymbol{x_0}) \leqslant T, \forall \boldsymbol{x_0} \in \mathbb{R}^n, \quad (2)$$

then system (1) *is said to be predefined-time stable.*

Definition 3 ([19]). *For $\forall \boldsymbol{x_0} \in \mathbb{R}^n$, if $T = \sup\{T(\boldsymbol{x_0})\}$, then T is non-conservative, and system* (1) *is said to be strongly predefined-time stable.*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI10.1109/TNNLS.2024.3354936,

B. Tensor Knowledge

For N positive integers $H_1, H_2, \ldots, H_N$, the expression of N-order tensor is as follows:

$$\mathcal{B} = (b_{h_1 h_2 \ldots h_N})_{1 \leqslant h_i \leqslant H_i}, i = \{1, 2, \ldots, N\}. \tag{3}$$

Here, $\mathcal{B} \in \mathbb{R}^{H_1 \times H_2 \times \cdots \times H_N}$ is a multidimensional array with N entries.

Definition 4 ([7]). *For tensor $\mathcal{B} \in \mathbb{R}^{H_1 \times \cdots \times H_N \times K_1 \times \cdots \times K_M}$, if there exists a tensor $\mathcal{A} \in \mathbb{R}^{K_1 \times \cdots \times K_M \times H_1 \times \cdots \times H_N}$ satisfying*

$$\begin{aligned} &\mathcal{A}_{k_1 \ldots k_M h_1 \ldots h_N} = \mathcal{B}_{h_1 \ldots h_N k_1 \ldots k_M}, \\ &1 \leqslant k_j \leqslant K_j, \quad j = \{1, 2, \ldots, M\}, \end{aligned} \tag{4}$$

then $\mathcal{A}$ is said to be the transpose of $\mathcal{B}$, denoted as $\mathcal{B}^{\mathrm{T}}$.

Definition 5 ([32]). *The Moore-Penrose inverse of tensor $\mathcal{B} \in \mathbb{R}^{H_1 \times \cdots \times H_N \times K_1 \times \cdots \times K_N}$ exists and is unique, denoted as $\mathcal{B}^{\dagger} \in \mathbb{R}^{K_1 \times \cdots \times K_N \times H_1 \times \cdots \times H_N}$, where $\mathcal{B}^{\dagger}$ satisfies*

$$\begin{cases} \mathcal{B} *_N \mathcal{B}^{\dagger} *_N \mathcal{B} = \mathcal{B}, \\ \mathcal{B}^{\dagger} *_N \mathcal{B} *_N \mathcal{B}^{\dagger} = \mathcal{B}^{\dagger}, \\ (\mathcal{B} *_N \mathcal{B}^{\dagger})^{\mathrm{T}} = \mathcal{B} *_N \mathcal{B}^{\dagger}, \\ (\mathcal{B}^{\dagger} *_N \mathcal{B})^{\mathrm{T}} = \mathcal{B}^{\dagger} *_N \mathcal{B}. \end{cases} \tag{5}$$

with $*_N$ standing for the Einstein product.

Definition 6 ([33]). *An invertible mapping that converts tensors to matrices is mathematically expressed as $\Theta(\mathcal{B}) = B \in \mathbb{R}^{H \times K}$, where $\mathcal{B} \in \mathbb{R}^{H_1 \times \cdots \times H_N \times K_1 \times \cdots \times K_M}$, $H = \prod_{y=1}^{N} H_y$ and $K = \prod_{y=1}^{M} K_y$. Here, the specific mapping rules are as follows:*

$$\mathcal{B}_{h_1 h_2 \ldots h_N k_1 k_2 \ldots k_M} \to B_{ij} \tag{6}$$

where i and j are positive integers satisfying

$$i = h_N + \sum_{\alpha=1}^{N-1} \left((h_\alpha - 1) \prod_{y=\alpha+1}^{N} H_y \right),$$

$$j = k_M + \sum_{\alpha=1}^{M-1} \left((k_\alpha - 1) \prod_{y=\alpha+1}^{M} K_y \right).$$

C. Problem Formulation

Extending **Definition 5** to the time-varying domain, the TV-TMPI is obtained, i.e., the MP inverse of $\mathcal{B}(t) \in \mathbb{R}^{H_1 \times \cdots \times H_N \times K_1 \times \cdots \times K_N}$ is $\mathcal{B}^{\dagger}(t) \in \mathbb{R}^{K_1 \times \cdots \times K_N \times H_1 \times \cdots \times H_N}$. Then, convert tensors $\mathcal{B}(t)$ and $\mathcal{B}^{\dagger}(t)$ into matrices $B(t)$ and $B^{\dagger}(t)$ by **Definition 6**, where $B(t)$ and $B^{\dagger}(t)$ satisfy

$$\begin{cases} B(t)B^{\dagger}(t)B(t) = B(t), \\ B^{\dagger}(t)B(t)B^{\dagger}(t) = B^{\dagger}(t), \\ (B(t)B^{\dagger}(t))^{\mathrm{T}} = B(t)B^{\dagger}(t), \\ (B^{\dagger}(t)B(t))^{\mathrm{T}} = B^{\dagger}(t)B(t) \end{cases} \tag{7}$$

with $B(t) \in \mathbb{R}^{H \times K}$, $B^{\dagger}(t) \in \mathbb{R}^{K \times H}$, $H = \prod_{y=1}^{N} H_y$ and $K = \prod_{y=1}^{M} K_y$.

According to the rank of $B(t)$, there are two ways to calculate $B^{\dagger}(t)$.

1) If $\mathrm{rank}(B(t)) = K$, then $A(t) = B^{\mathrm{T}}(t)B(t)$ is invertible, and $B^{\dagger}(t)$ is called the left MP inverse calculated by

$$\begin{aligned} &B^{\dagger}(t) = A^{-1}(t)B^{\mathrm{T}}(t), \\ &B^{\dagger}(t)B(t) = I \in \mathbb{R}^{K \times K}. \end{aligned} \tag{8}$$

2) If $\mathrm{rank}(B(t)) = H$, then $C(t) = B(t)B^{\mathrm{T}}(t)$ is invertible, and $B^{\dagger}(t)$ is called the right MP inverse calculated by

$$\begin{aligned} &B^{\dagger}(t) = B^{\mathrm{T}}(t)C^{-1}(t), \\ &B(t)B^{\dagger}(t) = I \in \mathbb{R}^{H \times H}. \end{aligned} \tag{9}$$

III. The Design of The Neural Network

In this section, the design process of the traditional ZNN model is introduced. The PTANN model is proposed by combining a novel adaptive parameter and activation function. Furthermore, the ET-PTANN model is proposed by combining an event trigger and evolution formulation.

A. Standard ZNN Model

The standard design process of the ZNN model consists of three steps: proposing the error equation, designing the evolution formula, and constructing the ZNN model. Equation (9) is taken as an example, where $X(t)$ represents the matrix to be solved. The following error function is obtained:

$$E(t) = X(t)C(t) - B^{\mathrm{T}}(t) \in \mathbb{R}^{K \times H}. \tag{10}$$

Then, an evolution formula is used to drive $E(t)$ to converge to zero [14]:

$$\dot{E}(t) = -\lambda \Phi\left(E(t)\right). \tag{11}$$

Here, the evolution formula (11) satisfies the definition of a nonlinear system (1), where λ represents the convergence parameter. The activation function array $\Phi(\cdot)$ is composed of continuous, monotonically increasing odd functions denoted as $\phi(\cdot)$.

Substituting (10) into (11), we obtain the ZNN-R model about the right MP inverse:

$$\dot{X}(t)C(t) = -\lambda \Phi\left(X(t)C(t) - B^{\mathrm{T}}(t)\right) - X(t)\dot{C}(t) + \dot{B}^{\mathrm{T}}(t) \tag{12}$$

and, similarly, we obtain the ZNN-L model for the left MP inverse based on (8):

$$A(t)\dot{X}(t) = -\lambda \Phi\left(A(t)X(t) - B^{\mathrm{T}}(t)\right) - \dot{A}(t)X(t) + \dot{B}^{\mathrm{T}}(t). \tag{13}$$

B. PTANN Model

The simulation of the ZNN model is commonly performed using the ode45 solver to evaluate its performance. Therefore, to improve computational efficiency within the ode45 solver, we propose the PTANN model. Before introducing the PTANN model, it is necessary to have a basic understanding of ode45.

The ode45 solver is a variable-step solver that combines the 4th-order and 5th-order Runge-Kutta (RK) methods [34]. The step size in ode45 represents the increment of time t between adjacent iterations [35], which dynamically adjusts the step size based on the change rate of the model state:

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI10.1109/TNNLS.2024.3354936,

when the state changes rapidly, the step size is reduced to improve accuracy, and when the state changes slowly, the step size is increased to avoid unnecessary computations [36]. In the case of formulas (12) and (13), the state matrix $X(t)$ and the magnitude of $\dot{E}(t)$ are directly proportional. Therefore, when using the ode45 solver to solve the state matrix $X(t)$ through the neural network model, the step size is inversely proportional to $\left\|\dot{E}(t)\right\|_{\mathrm{F}}$.

In previous studies, conventional time-varying parameters were designed to accelerate convergence [37], [38]. These parameters increase over time, reducing the step size of ode45 and resulting in increased computational costs for the neural network model. To reduce the computational cost, we propose a novel adaptive parameter:

$$\lambda_{\mathrm{dp}}(t) = \frac{w\|E(t)\|_{\mathrm{F}}^{p}}{T}, \tag{14}$$

where $0 < p < 0.5$, $w = (HK)^{-\frac{p}{2}}$, $T > 0$ is the only parameter that can affect the upper bound of convergence time, and $\|\cdot\|_{\mathrm{F}}$ stands for the Frobenius norm.

Remark 1. *The analysis above makes it clear that the convergence parameter $\lambda_{\mathrm{dp}}(t)$ has a direct impact on the model's change rate, subsequently influencing the step size of the ode45 solver. This step size is closely associated with the computational cost of the model. In particular, the convergence parameter $\lambda_{\mathrm{dp}}(t)$ is inversely proportional to the step size, while it is directly proportional to the convergence rate and the computational cost.*

During the convergence phase, larger values of $\|E(t)\|_{\mathrm{F}}$ and $\lambda_{\mathrm{dp}}(t)$ can accelerate the convergence rate of the ANN model. This allocation of computing resources speeds up the convergence process. During the stable phase, smaller values of $\|E(t)\|_{\mathrm{F}}$ and $\lambda_{\mathrm{dp}}(t)$ can reduce the computational cost of the ANN model. This allocation of computing resources minimizes unnecessary computations. Therefore, the adaptive parameter $\lambda_{\mathrm{dp}}(t)$ improves computational efficiency by appropriately allocating computing resources based on the magnitude of $\|E(t)\|_{\mathrm{F}}$.

The PTANN model is proposed by incorporating the adaptive parameter $\lambda_{\mathrm{dp}}(t)$ and the activation function $\Phi_{\mathrm{dp}}(\cdot)$. The element expression of the activation function $\Phi_{\mathrm{dp}}(\cdot)$ is given by:

$$\phi_{\mathrm{dp}}(x) = \frac{1}{p}\exp(|x|^{p})|x|^{1-2p}\mathrm{sign}(x), \tag{15}$$

where $0 < p < 1/2$ and $\mathrm{sign}(\cdot)$ is a function defined by

$$\mathrm{sign}(x) = \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{if } x = 0 \\ -1, & \text{if } x < 0. \end{cases} \tag{16}$$

Contrary to traditional activation functions, incorporating formula (15) with adaptive parameter $\lambda_{\mathrm{dp}}(t)$ equips the PTANN model with the capability to achieve strong predefined-time convergence and an accelerated convergence rate. The benefits of this approach will be showcased in subsequent simulation experiments.

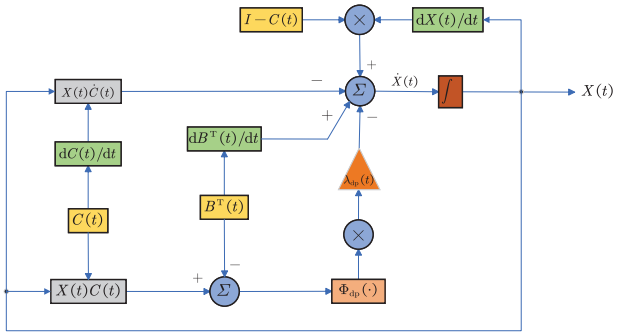


Fig. 1. Module diagram of the PTANN-R model (18).

Substituting the adaptive parameter $\lambda_{\mathrm{dp}}(t)$ and activation function $\Phi_{\mathrm{dp}}(x)$ into the evolution formula (11), we obtain

$$\begin{aligned} \dot{E}_{\mathrm{dp}}(t) =& -\lambda_{\mathrm{dp}}(t)\Phi_{\mathrm{dp}}(E(t)), \\ \dot{e}_{\mathrm{dp}}(t) =& -\frac{w\|E(t)\|_{\mathrm{F}}^{p}}{Tp}\exp(|e(t)|^{p})|e(t)|^{1-2p}\mathrm{sign}(e(t)), \end{aligned} \tag{17}$$

where $\dot{e}_{\mathrm{dp}}(t)$ represents the elements of matrix $\dot{E}_{\mathrm{dp}}(t)$.

Subsequently, the PTANN-R model about the right MP inverse is

$$\begin{aligned} \dot{X}(t)C(t) =& -\lambda_{\mathrm{dp}}(t)\Phi_{\mathrm{dp}}\left(X(t)C(t) - B^{\mathrm{T}}(t)\right) \\ & - X(t)\dot{C}(t) + \dot{B}^{\mathrm{T}}(t), \end{aligned} \tag{18}$$

and the PTANN-L model about the left MP inverse is

$$\begin{aligned} A(t)\dot{X}(t) =& -\lambda_{\mathrm{dp}}(t)\Phi_{\mathrm{dp}}\left(A(t)X(t) - B^{\mathrm{T}}(t)\right) \\ & - \dot{A}(t)X(t) + \dot{B}^{\mathrm{T}}(t). \end{aligned} \tag{19}$$

Considering the structural similarity between the PTANN-R and PTANN-L models, a module diagram has been created exclusively for the PTANN-R. As illustrated in Fig. 1, the PTANN model operates using a series of components, namely, accumulators, multipliers, amplifiers, integrators, and differential modules, with $X(t)$ denoting the state variable.

C. ET-PTANN Model with Event Trigger

As discussed in **Remark 1**, the adaptive parameter $\lambda_{\mathrm{dp}}(t)$ can regulate the allocation of computing resources in the PTANN model and improve computational efficiency. However, during the convergence phase, this adaptive parameter has little impact on the step size of ode45. To further enhance computational efficiency, we introduce an event-triggered evolution formula that reduces the computational frequency.

The design process for the ET-PTANN model with event triggering is described as follows. Building upon (17), an event-triggered mechanism is added to the evolution formula, leading to the following new evolution formula:

$$\dot{E}_{\mathrm{et}}(t) = -\lambda_{\mathrm{et}}(t)\Phi_{\mathrm{et}}(E(t)), \tag{20}$$

where $t \in [t_k, t_{k+1})$, $k \in \mathbb{N}$, and the sequence $\{t_k\}$ represents the moment at which $\dot{E}_{\mathrm{et}}(t)$ is recomputed.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI10.1109/TNNLS.2024.3354936,

Compared to the conventional periodic control, the $\dot{E}_{\rm et}(t)$ in the event-triggered ET-PTANN model contains measurement error $\varepsilon(t)$ obeying

$$\begin{aligned}\varepsilon(t) &= -\lambda_{\rm et}(t)\Phi_{\rm et}(E(t)) + \lambda_{\rm dp}(t)\Phi_{\rm dp}(E(t)) \\ &= \dot{E}_{\rm et}(t) - \dot{E}_{\rm dp}(t).\end{aligned} \tag{21}$$

Here, $\varepsilon(t)$ is used to quantify the error in $\dot{E}(t)$ resulting from the hysteresis of the event trigger.

To ensure the convergence performance of the ET-PTANN model, it is important to keep the error $\varepsilon(t)$ within a small range. The value of $\varepsilon(t)$ depends on the difference between $\dot{E}_{\rm dp}(t)$ and $\dot{E}_{\rm dp}(t_k)$. If this difference is too large, the value of $\dot{E}_{\rm et}(t)$ is updated. However, due to the hysteresis in $\dot{E}_{\rm et}(t)$, there is a possibility that $\dot{E}_{\rm et}(t)$ and $E(t)$ have the same sign, which can cause $E(t)$ to diverge, even though $E(t)$ and $\dot{E}_{\rm dp}(t)$ have different signs. Therefore, the trigger condition for updating $\dot{E}_{\rm et}(t)$ is defined as follows:

$$\left|\dot{e}_{\rm dp}(t_k) - \dot{e}_{\rm dp}(t)\right| \geqslant \gamma \ \text{ or } \ \dot{e}_{\rm et}(t)\dot{e}_{\rm dp}(t) < 0, t \geqslant t_k, \tag{22}$$

where $\gamma > 0$ represents the threshold of the event trigger, $\dot{e}_{\rm et}(t)$ represents the elements of matrix $\dot{E}_{\rm et}(t)$.

The threshold γ allows us to control the measurement error $\varepsilon(t)$ and the triggering frequency. A smaller threshold leads to a smaller measurement error and a higher triggering frequency, while a larger threshold results in a larger measurement error and a lower triggering frequency. Depending on the relationship between the threshold and the trigger condition, we can classify the threshold into two types: fixed threshold and relative threshold. Each type has its own advantages and disadvantages.

The relative threshold is proportional to $\left|\dot{e}_{\rm dp}(t_k)\right|$ according to (22). As $\left|\dot{e}_{\rm dp}(t_k)\right|$ approaches 0, the relative threshold controls $\varepsilon(t)$ within a small range. In contrast, the fixed threshold remains constant, resulting in a small $\varepsilon(t)$ for large $\left|\dot{e}_{\rm dp}(t_k)\right|$. To reduce measurement error and ensure the convergence rate of the model, we design a hybrid threshold using a piecewise function. The expression of the hybrid threshold is as follows:

$$\gamma = \begin{cases} \eta_1 & \text{if } \left|\dot{e}_{\rm dp}(t_k)\right| \geqslant \dfrac{\eta_1 - \eta_2}{\sigma}, \\ \sigma\left|\dot{e}_{\rm dp}(t_k)\right| + \eta_2 & \text{if } \left|\dot{e}_{\rm dp}(t_k)\right| < \dfrac{\eta_1 - \eta_2}{\sigma}, \end{cases} \tag{23}$$

where $\eta_1 > 0$, $\eta_2 > 0$ and $0 < \sigma < 1$. Here, η_2 is used to ensure a lower limit for γ. The specific range of values for η_1 and η_2 will be analyzed later.

Note that $|e(t)|$ is inversely proportional to time t, and both $\lambda_{\rm dp}(\cdot)$ and $\phi_{\rm dp}(\cdot)$ are monotonically increasing functions. Therefore, the following inequality holds:

$$\left|\dot{e}_{\rm dp}(t_{k+1})\right| \leqslant \left|\dot{e}_{\rm dp}(t)\right|, t \in [t_k, t_{k+1}). \tag{24}$$

From the previous analysis, it is evident that $\left|\dot{e}(t)\right|$ can be reduced to enhance computational efficiency. Letting $\dot{e}_{\rm et}(t) = \dot{e}_{\rm dp}(t_{k+1})$, its expression is as follows:

$$\begin{aligned}\dot{e}_{\rm et}(t) =& \dot{e}_{\rm dp}(t_{k+1}) \\ =& \dot{e}_{\rm dp}(t_k) - \gamma \text{sign}\left(\dot{e}_{\rm dp}(t_k)\right),\end{aligned} \tag{25}$$

TABLE I
EFFECTS OF MAIN DESIGN PARAMETERS OF PTANN AND ET-PTANN MODELS.

Model	Parameter	Source	Effect
PTANN, ET-PTANN	$\|E(t)\|_{\rm F}$	$\lambda_{\rm dp}$	allocating computing resources
	T	$\lambda_{\rm dp}$	determining convergence time
	p	$\lambda_{\rm dp}$, $\Phi_{\rm dp}$	controlling convergence rate
ET-PTANN	γ	$\dot{E}_{\rm et}$	controlling measurement error

where $t \in [t_k, t_{k+1})$. When $t = t_k$, $e_{\rm dp}(t_k) = e_{\rm et}(t_k) =: e(t_k)$. It follows from (17) that $\text{sign}\left(\dot{e}_{\rm dp}(t_k)\right) = -\text{sign}(e(t_k))$, and therefore

$$\begin{aligned}\dot{e}_{\rm et}(t) =& -\lambda_{\rm dp}(t_k)\phi_{\rm dp}(e(t_k)) + \gamma\text{sign}(e(t_k)) \\ =& -\frac{w\|E(t_k)\|_{\rm F}^p}{Tp}\exp(|e(t_k)|^p)|e(t_k)|^{1-2p}\text{sign}(e(t_k)) \\ &+ \gamma\text{sign}\left(e(t_k)\right).\end{aligned} \tag{26}$$

It is known from (22) that $\dot{E}_{\rm et}(t)$ and $\dot{E}_{\rm dp}(t)$ have the same sign, and

$$\begin{aligned}\dot{e}_{\rm et}(t)\dot{e}_{\rm dp}(t) &\geqslant 0, \\ \left|\dot{e}_{\rm dp}(t_k)\right|\left(\left|\dot{e}_{\rm dp}(t_k)\right| - \gamma\right) &\geqslant 0.\end{aligned} \tag{27}$$

Substituting (23) into (27), we have $\eta_1 \geqslant \eta_2/(1-\sigma)$, $(1-\sigma)e_{\min} \geqslant \eta_2 > 0$, where $e_{\min}$ represents the minimum value of $|e(t)|$ in the stable phase.

Replacing the evolution formula $\dot{E}_{\rm dp}(t)$ of PTANN models with $\dot{E}_{\rm et}(t)$, we obtain the ET-PTANN model. The ET-PTANN-R model about the right MP inverse is

$$\begin{aligned}\dot{X}(t)C(t) =& -\lambda_{\rm et}(t)\Phi_{\rm et}\left(E(t)\right) - X(t)\dot{C}(t) + \dot{B}^{\rm T}(t) \\ =& -\lambda_{\rm dp}(t_k)\Phi_{\rm dp}\left(X(t_k)C(t_k) - B^{\rm T}(t_k)\right) \\ &+ \gamma\text{sign}(E(t_k)) - X(t)\dot{C}(t) + \dot{B}^{\rm T}(t),\end{aligned} \tag{28}$$

and the ET-PTANN-L model about the left MP inverse is

$$\begin{aligned}A(t)\dot{X}(t) =& -\lambda_{\rm et}(t)\Phi_{\rm et}\left(E(t)\right) - \dot{A}(t)X(t) + \dot{B}^{\rm T}(t) \\ =& -\lambda_{\rm dp}(t_k)\Phi_{\rm dp}\left(A(t_k)X(t_k) - B^{\rm T}(t_k)\right) \\ &+ \gamma\text{sign}(E(t_k)) - \dot{A}(t)X(t) + \dot{B}^{\rm T}(t).\end{aligned} \tag{29}$$

Remark 2. *The PTANN and ET-PTANN models have four main parameters: $\|E(t)\|_{\rm F}$, T, p, and γ. As shown in Table I, $\|E(t)\|_{\rm F}$, T and p are common parameters for the two models. $\|E(t)\|_{\rm F}$ is desigen to allocate computing resources. p is used to control the convergence rate. T determines the upper bound of the convergence time. γ is a special parameter of the ET-PTANN model, which acts on the event trigger. Its role is to control the measurement error caused by the event trigger, which affects the convergence rate and convergence time of the model.*

IV. THEORETICAL ANALYSIS

A. Stability Analysis

Let us now show that the PTANN and ET-PTANN models can achieve convergence in the sense of Lyapunov.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI10.1109/TNNLS.2024.3354936,

Theorem 1. *Given a tensor $\mathcal{B}(t) \in \mathbb{R}^{H_1 \times \cdots \times H_N \times K_1 \times \cdots \times K_N}$, if it is mapped to a matrix $B(t) \in \mathbb{R}^{H \times K}$, the state matrix $X(t) \in \mathbb{R}^{K \times H}$ of the PTANN and ET-PTANN models can globally converge to the MP inverse of $B(t)$, i.e., $B^{\dagger}(t)$.*

Proof: For the PTANN model, we consider an element $e(t)$ of error matrix $E(t)$, which evolves as

$$\dot{e}(t) = -\lambda_{\mathrm{dp}}(t)\phi_{\mathrm{dp}}(e(t)). \tag{30}$$

Let a Lyapunov candidate function be $d(t) = e^2(t)$ and take its time derivative to obtain

$$\begin{aligned} \dot{d}(t) &= 2e(t)\dot{e}_{\mathrm{dp}}(t) \\ &= -2\lambda_{\mathrm{dp}}(t)e(t)\phi_{\mathrm{dp}}(e(t)). \end{aligned} \tag{31}$$

From (14) and (15), we know that $\lambda_{\mathrm{dp}}(t) > 0$ and $e(t)\phi_{\mathrm{dp}}(e(t)) > 0$. Therefore, we obtain

$$\begin{cases} d(t) \geqslant 0 & d(t) = 0 \text{ if } e(t) = 0, \\ \dot{d}(t) \leqslant 0 & \dot{d}(t) = 0 \text{ if } e(t) = 0. \end{cases} \tag{32}$$

According to the Lyapunov stability theorem, $E(t)$ of the PTANN model converges to 0 over time, and the state matrix $X(t)$ globally converges to the MP inverse of $B(t)$. Similarly, we can prove that $E(t)$ of the ET-PTANN model converges to 0 over time, and the state matrix $X(t)$ globally converges to the MP inverse of $B(t)$ as well. ■

B. Convergence Analysis

Theoretical analysis will be provided in this subsection to support the fact that the parameter T is the only factor that affects the upper bound of convergence time.

Theorem 2. *Given a tensor $\mathcal{B}(t) \in \mathbb{R}^{H_1 \times \cdots \times H_N \times K_1 \times \cdots \times K_N}$, if it is mapped to a matrix $B(t) \in \mathbb{R}^{H \times K}$, then for any initial state $X(0) \in \mathbb{R}^{K \times H}$, the state matrix $X(t)$ of the PTANN model will converge to the MP inverse of $B(t)$ within a predefined-time T.*

Proof: Let $e_{ij}(t)$ denotes the ij-th element of error matrix $E(t)$. Then, we define a new variable $v(t)$:

$$v(t) = \sqrt{\frac{1}{HK}\sum_{i=1}^{K}\sum_{j=1}^{H} e_{ij}^2(t)}, \tag{33}$$

$$\|E(t)\|_{\mathrm{F}} = \sqrt{HK}v(t). \tag{34}$$

From the above definition, $v(t)$ can be regarded as the element of $E(t)$ to a certain extent. Therefore, it follows that

$$\begin{aligned} \dot{v}(t) =& -\lambda_{\mathrm{dp}}(t)\phi_{\mathrm{dp}}(v(t)) \\ =& -\frac{w\|E(t)\|_{\mathrm{F}}^p}{Tp}\exp(v^p(t))(v(t))^{1-2p} \\ =& -\frac{1}{Tp}\exp(v^p(t))(v(t))^{1-p}, \end{aligned} \tag{35}$$

which can be transformed into

$$\mathrm{d}t = -Tp\exp(-v^p(t))(v(t))^{p-1}\mathrm{d}v(t). \tag{36}$$

Letting the convergence time of the PTANN model be T_{c}, its integral equation is

$$T_{\mathrm{c}} = \int_0^{T_{\mathrm{c}}} 1\mathrm{d}t. \tag{37}$$

When t tends to 0, $v(t) = v(0)$, and when t tends to T_{c}, $v(t) = 0$. By replacing $\mathrm{d}t$ with $\mathrm{d}v(t)$, we have

$$\begin{aligned} T_{\mathrm{c}} =& \int_{v(0)}^{0} -Tp\exp(-v^p(t))(v(t))^{p-1}\mathrm{d}v(t) \\ \leqslant& T\int_0^{+\infty}\exp(-v^p(t))\mathrm{d}v^p(t) \\ =& T, \end{aligned} \tag{38}$$

where the parameter T of $\lambda_{\mathrm{dp}}(t)$ serves the upper bound of convergence time of the PTANN model. The proof is now complete. ■

Remark 3. *To improve the computational efficiency of the ET-PTANN model, we set $\left|\dot{e}_{\mathrm{et}}(t)\right| \leqslant \left|\dot{e}_{\mathrm{dp}}(t)\right|$. However, this poses some difficulties in determining the upper bound of convergence time of the ET-PTANN model. By setting a small threshold γ, the measurement error $\varepsilon(t)$ is controlled within a small range, which leads to a similar convergence rate between the ET-PTANN and PTANN models. As a result, they can share the upper bound of convergence time.*

C. Execution Interval Analysis

In order to avoid Zeno behavior in the event trigger, it is necessary to prove that the minimum execution interval is greater than zero.

Theorem 3. *In the ET-PTANN model, there exists a certain time $t_\tau > 0$ such that, for $\forall k \in \mathbb{N}$, the execution interval $t_{k+1} - t_k$ is greater than t_τ.*

Proof: Let $m(t)$ represent the element with the largest absolute value in $E(t)$, whose evolution is governed by

$$\begin{aligned} \dot{m}_{\mathrm{dp}}(t) &= -\lambda_{\mathrm{dp}}(t)\phi_{\mathrm{dp}}(m(t)) \\ &= -\frac{w\|E(t)\|_{\mathrm{F}}^p}{Tp}\exp(|m(t)|^p)|m(t)|^{1-2p}\mathrm{sign}(m(t)). \end{aligned} \tag{39}$$

It follows from $\|E(t)\|_{\mathrm{F}}^p \leqslant (HK)^{\frac{p}{2}}|m(t)|^p$ that

$$\left|\frac{\mathrm{d}\dot{m}_{\mathrm{dp}}(t)}{\mathrm{d}t}\right| \leqslant \frac{1}{Tp}\exp(|m(t)|^p)(p + (1-p)|m(t)|^{-p}). \tag{40}$$

As shown by (22), there is a derivative error $\epsilon(t)$ in the trigger gap:

$$\epsilon(t) = \dot{m}_{\mathrm{dp}}(t_k) - \dot{m}_{\mathrm{dp}}(t) \quad t \in [t_k, t_{k+1}) \tag{41}$$

with the following time derivative:

$$\frac{\mathrm{d}|\epsilon(t)|}{\mathrm{d}t} = \mathrm{sign}(\epsilon(t))\dot{\epsilon}(t). \tag{42}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI10.1109/TNNLS.2024.3354936,

TABLE II
EXPERIMENTAL HARDWARE AND SOFTWARE INFORMATION.

Processor	RAM	Software
Intel(R) Core(TM) i5-1035G1 CPU @ 1.00 GHz 1.19 GHz	16.0 GB	MATLAB R2022a

Since $\dot{m}_{\mathrm{dp}}(t_k)$ is a constant, $\dot{\epsilon}(t) = -\mathrm{d}\dot{m}_{\mathrm{dp}}(t)/\mathrm{d}t$, and we have that

$$\begin{aligned}\frac{\mathrm{d}|\epsilon(t)|}{\mathrm{d}t} &= -\operatorname{sign}(\epsilon(t))\frac{\mathrm{d}\dot{m}_{\mathrm{dp}}(t)}{\mathrm{d}t}\\ &\leqslant \left|\frac{\mathrm{d}\dot{m}_{\mathrm{dp}}(t)}{\mathrm{d}t}\right| \\ &\leqslant \frac{1}{Tp}\exp(|m(t)|^p)(p+(1-p)|m(t)|^{-p}).\end{aligned} \tag{43}$$

Let $\kappa = \frac{1}{Tp}\exp(m_{\mathrm{a}}^p)(p+(1-p)m_{\mathrm{a}}^{-p})$, where $m_{\mathrm{a}} = \max\{|m(t)|\}$. When $t \to t_{k+1}$, it follows from (22) that $|\epsilon(t)| = \gamma$, and then

$$\begin{aligned}|\epsilon(t_{k+1})| - |\epsilon(t_k)| &= \frac{\mathrm{d}|\varepsilon(t)|}{\mathrm{d}t}(t_{k+1}-t_k),\\ \gamma - 0 &\leqslant \kappa(t_{k+1}-t_k).\end{aligned} \tag{44}$$

Letting $t_\tau = t_{k+1} - t_k$, it follows from (23) that

$$t_\tau \geqslant \frac{\gamma}{\kappa} \geqslant \frac{\eta_2}{\kappa}, \tag{45}$$

which ends the proof. ■

V. SIMULATION EXAMPLE

To validate the theoretical findings, a simulation example is conducted in this paper. Furthermore, to showcase the advantages of the PTANN and ET-PTANN models, several ZNN models are introduced for comparison. The hardware and software information of the experiment is shown in Table II.

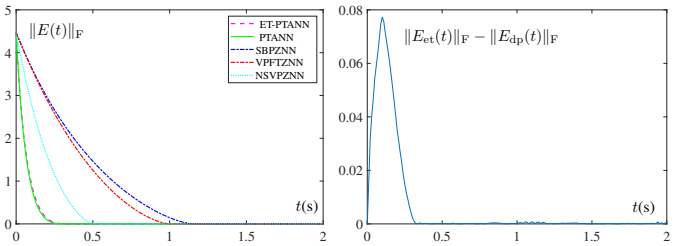
A. The ZNN Models for Comparison

This subsection provides a brief introduction to the ZNN models used for comparison, namely, the sign-bi-power ZNN (SBPZNN) [39], noise-suppression variable-parameter ZNN (NSVPZNN) [38], and varying-parameter finite-time ZNN (VPFTZNN) [37]. The information of the compared ZNN models and the novel ANN models is presented in Table III. The parameters $k_1, k_2, k_3, \lambda, \beta_1, \beta_2 > 0$ and $q > 1$; h and ϖ are odd integers and $\varpi > h$; p, w, γ, and T have been defined in Section III.

B. Example

Consider a 4th-order tensor $\mathcal{B}(t) \in \mathbb{R}^{2\times2\times2\times3}$ whose expression is given as follows:

$$\mathcal{B}(:,:,1,1) = \begin{bmatrix}\cos\left(\frac{3t}{2}\right) & 0\\ 0 & 0\end{bmatrix}, \quad \mathcal{B}(:,:,1,2) = \begin{bmatrix}\sin\left(\frac{3t}{2}\right) & \cos\left(\frac{3t}{2}\right)\\ 0 & 0\end{bmatrix},$$
$$\mathcal{B}(:,:,1,3) = \begin{bmatrix}0 & \sin\left(\frac{3t}{2}\right)\\ \cos\left(\frac{3t}{2}\right) & 0\end{bmatrix}, \quad \mathcal{B}(:,:,2,1) = \begin{bmatrix}0 & 0\\ \sin\left(\frac{3t}{2}\right) & \cos\left(\frac{3t}{2}\right)\end{bmatrix},$$
$$\mathcal{B}(:,:,2,2) = \begin{bmatrix}0 & 0\\ 0 & \sin\left(\frac{3t}{2}\right)\end{bmatrix}, \quad \mathcal{B}(:,:,2,3) = \begin{bmatrix}0 & 0\\ 0 & 0\end{bmatrix}.$$



(a) error norm (b) error norm difference

Fig. 2. Error norm obtained by neural network models and error norm difference between ET-PTANN and PTANN models.

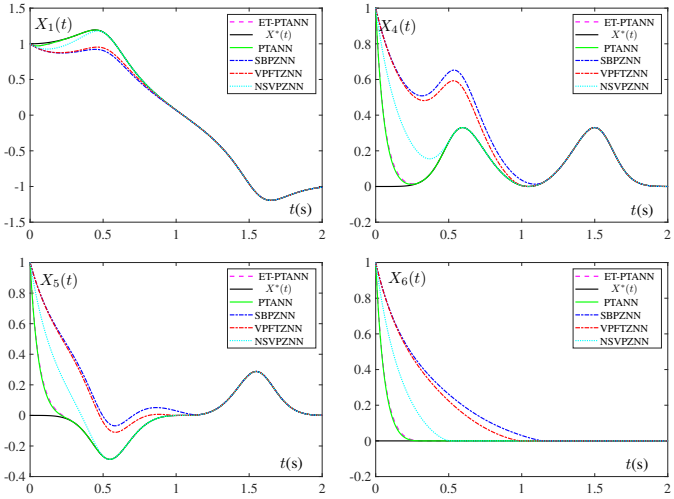


Fig. 3. Partial theoretical solution $X^*(t)$ and partial state solution $X(t)$ of neural network models.

Clearly, $\mathcal{B}(t)$ has a right MP inverse, i.e., the rank of matrix $B(t) \in \mathbb{R}^{4\times6}$ is 4, where $B(t)$ is a mapping of tensor $\mathcal{B}(t)$.

To compute the MP inverse of $\mathcal{B}(t)$, we set initial values for the parameters as $k_1 = 1, k_2 = 1, \lambda = 1, p = 1/3, q = 3/2, h = 1, \varpi = 3, k_3 = 0.5, \beta_1 = 0.5, \beta_2 = 0.2, w = 24^{-1/6}, \eta_1 = 0.5, \eta_2 = 0.001, \sigma = 0.1, T = 0.5$. All elements of initial state $X(0)$ are 1.

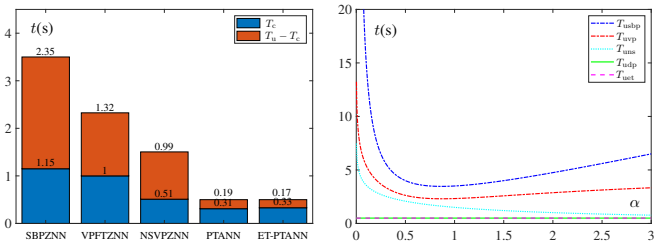
In the simulation example, we obtain Fig. 2 and Fig. 3 to evaluate the performance of the proposed models. Fig. 2(a) illustrates the dynamic trajectory of $\|E(t)\|_{\mathrm{F}}$. We observe that both the ET-PTANN and PTANN models exhibit the fastest convergence rate, with their error curves almost overlapping. To further compare the differences between the two ANN models, we present Fig. 2(b), which shows the difference in $\|E(t)\|_{\mathrm{F}}$ for the PTANN and ET-PTANN models. The maximum difference is only 0.078, and it quickly converges to 0 over time. Fig. 3 displays the dynamic trajectories of partial state solution $X(t)$ and theoretical solution $X^*(t)$. It can be seen that state solutions $X(t)$ of PTANN and ET-PTANN models converge to theoretical solution $X^*(t)$ in 0.3 s, which is mutually confirmed with Fig. 2(a).

Let T_{u} represent the upper bound of convergence time, and T_{c} represent the actual convergence time. T_{u} can be calculated using Table III, and the corresponding T_{c} is derived from Fig. 2(a). Based on this, we obtain Fig. 4(a), where the blue bars represent the T_{c} of the neural network models, while the red

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI10.1109/TNNLS.2024.3354936,

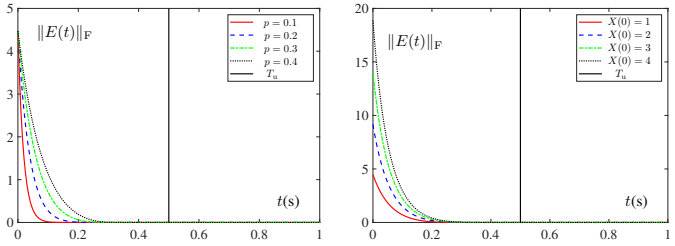
TABLE III
COMPARISON OF SBPZNN [39], VPFTZNN [37], NSVPZNN [38], PTANN AND ET-PTANN MODELS.

Model	Activation function	Parameter	Upper bound of convergence time
SBPZNN [39]	$\phi_{\rm sbp}(x)=(k_1\lvert x\rvert^p+k_2\lvert x\rvert^q)\text{sign}(x)$	λ	$\frac{1}{\lambda k_1(1-p)}+\frac{1}{\lambda k_2(q-1)}$
VPFTZNN [37]	$\phi_{\rm vp}(x)=(k_1x+k_2x^{\frac{h}{\varpi}})$	$\lambda\cosh(k_3t)$	$\frac{1}{k_3}\text{arcsinh}\left(\frac{k_3\varpi}{k_1h(\varpi-h)}\ln\left(\frac{k_1}{k_2}e_{\max}^{1-\frac{h}{\varpi}}(0)+1\right)\right)$
NSVPZNN [38]	$\phi_{\rm ns}(x)=\begin{cases}k_1\lvert x\rvert^q\text{sign}(x)+k_2x, & \text{if } \lvert x\rvert>1\\ k_1\lvert x\rvert^p\text{sign}(x)+k_2x, & \text{otherwise.}\end{cases}$	$\lambda\exp(\beta_1\text{arccot}(t)+\beta_2t)$	$\frac{1}{\beta_2}\ln\left(1+\frac{\beta_2(q-p)}{\lambda k_1(1-p)(q-1)}\right)$
PTANN	$\phi_{\rm dp}(x)=\frac{1}{p}\exp(\lvert x\rvert^p)\lvert x\rvert^{1-2p}\text{sign}(x)$	$w\lVert E(t)\rVert_{\rm F}^p/T$	T
ET-PTANN	$\phi_{et}(x)=\frac{1}{p}\exp(\lvert x_k\rvert^p)\lvert x_k\rvert^{1-2p}\text{sign}(x_k)$	$w\lVert E(t_k)\rVert_{\rm F}^p/T$	T



(a) (b)

Fig. 4. Upper bound $T_{\rm u}$ of convergence time of neural network models.



(a) PTANN $X(0)=1$ (b) PTANN $p=0.3$

(c) ET-PTANN $X(0)=1$ (d) ET-PTANN $p=0.3$

Fig. 5. Convergence performance of PTANN and ET-PTANN models with varying initial parameter and initial state.

bars represent the difference between $T_{\rm u}$ and $T_{\rm c}$, i.e., $T_{\rm u}-T_{\rm c}$. We observe that the $T_{\rm u}$ of the PTANN and ET-PTANN models is the most accurate, which reduces unnecessary computations and indirectly improves computational efficiency. Moreover, the $T_{\rm u}$ of the PTANN and ET-PTANN models are independent of other parameters. By setting $k_1=\alpha$ and $k_2=1/\alpha$, we obtain the following equations:

$$T_{\rm usbp}=\frac{1}{\lambda\alpha(1-p)}+\frac{\alpha}{\lambda(q-1)},$$
$$T_{\rm uvp}=\frac{1}{k_3}\text{Arcsinh}\left(\frac{k_3\varpi}{\alpha h(\varpi-h)}\ln\left(\alpha^2e_{\max}^{1-\frac{h}{\varpi}}(0)+1\right)\right),$$
$$T_{\rm uns}=\frac{1}{\beta_1}\ln\left(1+\frac{\beta_2(q-p)}{\lambda\alpha(1-p)(q-1)}\right),$$

in which $T_{\rm usbp}$, $T_{\rm uvp}$, and $T_{\rm uns}$ represent the upper bounds of convergence time of the SBPZNN, VPFTZNN, and NSVPZNN models, respectively. When $\alpha\to 0$, they will all tend to infinity, but the upper bounds $T_{\rm udp}$ and $T_{\rm uet}$ of convergence time of the PTANN and ET-PTANN models are not affected by α. Fig. 4(b) verifies the above conclusion.

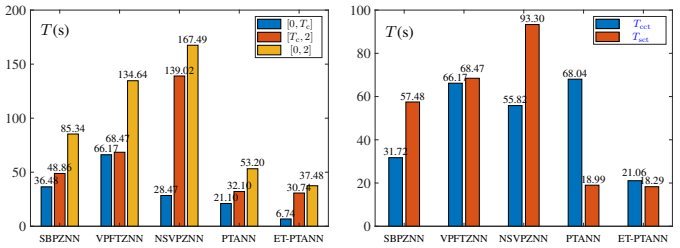
To maintain generality, we conduct tests to examine the influence of parameter p and initial state $X(0)$ on the PTANN and ET-PTANN models. As shown in Fig. 5, we observe that as p and $X(0)$ increase, the convergence time of both ANN models also increases. Nevertheless, such an increase is small and does not exceed the predefined upper bound of convergence time $T_{\rm u}$ which is represented by the black straight line in the figure.

To demonstrate the computational efficiency of the PTANN and ET-PTANN models, we conducted further research with the same parameter settings as in Fig. 2. The results of the research are presented in Table IV. The computation time of the neural network models is proportional to the number of iterations of ode45. Compared with the SBPZNN model with time-invariant parameters, the VPFTZNN and NSVPZNN models exhibit shorter convergence time, but the number of iterations also increases accordingly. However, both the ET-PTANN and PTANN models have shorter convergence time and computation time, that is, they achieve improvements in both convergence rate and computational efficiency. Their

TABLE IV
COMPUTATIONAL EFFICIENCY OF NEURAL NETWORK MODELS.

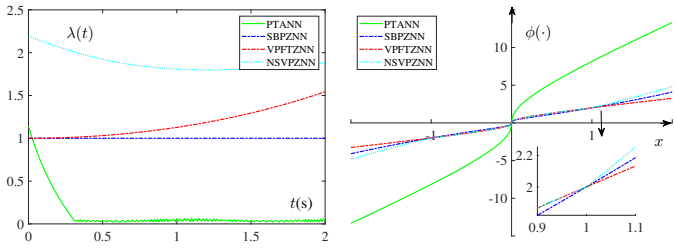
Model	Convergence time	Computation time	Iterations	Computation counts of evolution formula
SBPZNN	1.15 s	85.34 s	1736	1736
VPFTZNN	1 s	134.64 s	2096	2096
NSVPZNN	0.51 s	167.49 s	3368	3368
PTANN	0.31 s	53.20 s	1094	1094
ET-PTANN	0.33 s	37.48 s	740	501

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI10.1109/TNNLS.2024.3354936,



(a) computation time (b) average computation time

Fig. 6. Computation time and average computation time of neural network models at different phases.



(a) convergence parameter (b) activation function

Fig. 7. Activation function and convergence parameter of neural network models.

computation time are significantly lower than those of other neural network models, at 37.48 s and 53.20 s, respectively. Comparing the number of iterations and the computation counts of the evolution formula, we can observe that the computation counts of the evolution formula for the ET-PTANN model are lower. The event trigger reduces the computation counts of the evolution formula by 239, accounting for approximately 32.3% of the total iterations.

Next, we discuss the computation time of the neural network models in different phases. In Fig. 6, the blue bar signifies the computation time for the convergence phase, the red bar indicates the computation time for the stable phase, and the yellow bar represents computation time for the entire operational phase. As shown in Fig. 6(a), the computation time of the PTANN and ET-PTANN models is significantly lower than that of other models, both in the convergence phase and the stable phase. To account for the differences in convergence time among the neural network models, we normalized the computation time. The results are shown in Fig. 6(b). In the stable phase, the traditional time-varying ZNN models, namely the VPFTZNN and NSVPZNN models, exhibit larger average computation time ($T_{\rm sct}$) compared to the SBPZNN model with time-invariant parameters. The proposed adaptive parameters $\lambda_{\rm dp}(t)$ and $\lambda_{\rm et}(t)$ greatly reduce $T_{\rm sct}$. Furthermore, the event trigger reduces the average computation time ($T_{\rm cct}$) of the ET-PTANN model in the convergence phase.

To demonstrate the advantages of adaptive parameter $\lambda_{\rm dp}(t)$ and activation function $\phi_{\rm dp}(\cdot)$, we compared the changing trends of $\lambda(t)$ and $\phi(\cdot)$ from different models and obtained Fig. 7. Observing Fig. 7(a), it is evident that $\lambda_{\rm dp}(t)$ of the PTANN model diminishes as time progresses, ultimately converging to near 0 at 0.3 s. As depicted in Fig. 2(a), this is consistent with the time for $\|E(t)\|_{\rm F}$ to converge to 0 for the PTANN model. Conversely, the $\lambda(t)$ of other models does not decrease over time, and even $\lambda(t)$ of the VPFTZNN increases over time. By comparing Fig. 7(a) and Fig. 6(b), it can be observed that the order of the average computation time ($T_{\rm sct}$) of the stable phase is consistent with their $\lambda(t)$ order, thereby validating the conclusion in **Remark 1**, that is, $\lambda(t)$ is proportional to the computational cost. Moreover, comparing Fig. 7(b) and Fig. 2(a), we notice that the sequence of convergence times for the models aligns with the sequence of slopes of $\phi(\cdot)$. This further confirms that the slope of $\phi(\cdot)$ is proportional to the convergence rate, and the novel $\phi_{\rm dp}(\cdot)$ has more advantages in convergence performance.

VI. Application

In this section, we apply the PTANN and ET-PTANN models to mobile sound source localization, specifically using the Time-Difference-of-Arrival (TDOA) algorithm. The TDOA algorithm relies on measuring the time difference between the arrival of a signal at receivers located at different positions [40]. By utilizing this algorithm, we can establish an error equation among the sound source and microphones. In the context of sound source localization, the signal corresponds to the sound wave, the receiver represents the microphone, and the signal source corresponds to the sound source.

Taking 2D moving sound source as an example, we need to define the coordinates of n microphones and the sound source:

$$N=\begin{bmatrix} x_1 & x_2 & \cdots & x_n \\ y_1 & y_2 & \cdots & y_n \end{bmatrix}\in\mathbb{R}^{2\times n}, l(t)=\begin{bmatrix} x(t) \\ y(t) \end{bmatrix}\in\mathbb{R}^2, \tag{46}$$

where $n \geqslant 4$. Microphone positions are chosen at random and do not change, while the source moves along the trajectory over time. Based on the relationship between the source and the microphones, we obtain

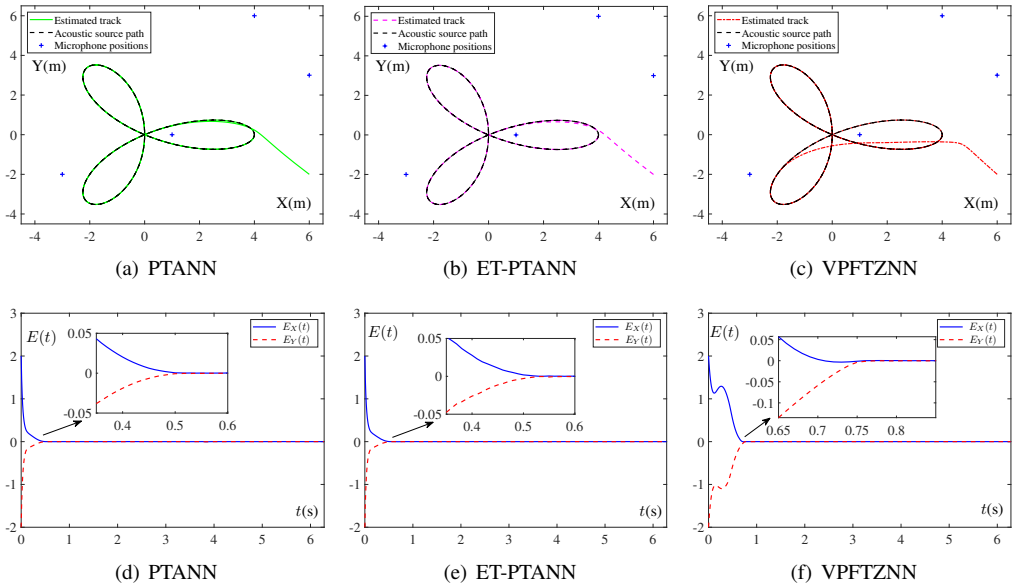
$$d_i(t)=vT_i(t)=\sqrt{(x_i-x(t))^2+(y_i-y(t))^2}, \tag{47}$$
$$\Delta T_i(t)=T_i(t)-T_1(t), \tag{48}$$
$$v\Delta T_i(t)=vT_i(t)-vT_1(t)=d_i(t)-d_1(t). \tag{49}$$

Here, $i\in\{1,\cdots,n\}$; $d_i(t)$ represents the distance between the sound source and the i-th microphone; $v=340$ m/s is the speed of sound; $T_i(t)$ represents the time that the sound arrives at the i-th microphone; $\Delta T_i(t)$ represents the time difference of arrival between the i-th microphone and the 1-th microphone.

Through a series of derivations [41], we obtain

$$\begin{bmatrix} z_{31}(t) & z_{32}(t) \\ z_{41}(t) & z_{42}(t) \\ \vdots & \vdots \\ z_{n1}(t) & z_{n2}(t) \end{bmatrix}\begin{bmatrix} x(t) \\ y(t) \end{bmatrix}=\begin{bmatrix} -s_3(t) \\ -s_4(t) \\ \vdots \\ -s_n(t) \end{bmatrix}, \tag{50}$$

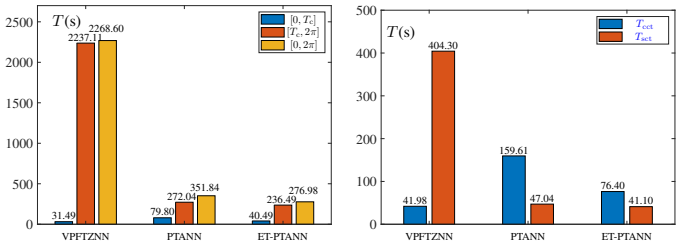
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI10.1109/TNNLS.2024.3354936,



(a) PTANN (b) ET-PTANN (c) VPFTZNN

(d) PTANN (e) ET-PTANN (f) VPFTZNN

Fig. 8. Sound source localization trajectories and coordinate errors for PTANN, ET-PTANN and VPFTZNN models.



(a) computation time (b) average computation time

Fig. 9. Computation time of different phases of neural network models in sound source localization.

where

$$\begin{aligned} z_{i1}(t) =& \frac{2(x_i - x_1)}{v\Delta T_i(t)} - \frac{2(x_2 - x_1)}{v\Delta T_2(t)}, \\ z_{i2}(t) =& \frac{2(y_i - y_1)}{v\Delta T_i(t)} - \frac{2(y_2 - y_1)}{v\Delta T_2(t)}, \\ s_i(t) =& v(\Delta T_i(t) - \Delta T_2(t)) + \frac{x_1^2 + y_1^2 - x_i^2 - y_i^2}{v\Delta T_i(t)} \\ & - \frac{x_1^2 + y_1^2 - x_2^2 - y_2^2}{v\Delta T_2(t)}, \forall i \in \{3, \cdots, n\}. \end{aligned} \tag{51}$$

For convenience, (50) is rewritten as

$$l(t) = Z^{\dagger}(t)s(t), \tag{52}$$

where $Z(t) = [z_{31}(t), z_{32}(t); \cdots; z_{n1}(t), z_{n2}(t)]$, $l(t) = [x(t), y(t)]^{\mathrm{T}}$, and $s(t) = [-s_3(t), -s_4(t), \cdots, -s_n(t)]^{\mathrm{T}}$.

As shown in (50), $Z(t)$ and $s(t)$ are known, and we can obtain the position of the sound source by calculating $Z^{\dagger}(t)$. Since $\mathrm{rank}(Z(t)) = 2$ is full rank, we can have the following model:

$$\begin{cases} M(t)\dot{X}(t) = -\lambda(t)\Phi\left(M(t)X(t) - Z^{\mathrm{T}}(t)\right) \\ \qquad\qquad - \dot{M}(t)X(t) + \dot{Z}^{\mathrm{T}}(t), \\ l(t) = X(t)s(t), \end{cases} \tag{53}$$

where $M(t) = Z^{\mathrm{T}}(t)Z(t)$. By substituting specific activation function and parameters, we obtain the neural network model. For comparison purposes, we have established the PTANN, ET-PTANN, and VPFTZNN models. The number of microphones is 4, the initial state $l(0) = [6, -2]^{\mathrm{T}}$, $k_1 = 2$, $k_2 = 2$, $k_3 = 0.5$, $\lambda = 1$, $p = 5/13$, $h = 5$, $\varpi = 13$, $T = 0.5$. The figures below displays the results.

Fig. 8 displays the sound source localization trajectories and coordinate errors of the PTANN, ET-PTANN, and VPFTZNN models. Comparing the trajectories in Figs. 8(a), 8(b), and 8(c), we observe that the PTANN and ET-PTANN models align with the actual moving sound source trajectory earlier. Examining the coordinate errors in Fig. 8(d), 8(e), and 8(f), we find that the PTANN and ET-PTANN models demonstrate faster convergence rates.

Fig. 9 illustrates the computation time of the neural network models at different phases of sound source localization. In the time interval of $[0, 2\pi]$ s, the neural network models exhibit the following order of computation time from high to low: VPFTZNN, PTANN, and ET-PTANN. During the convergence phase, the VPFTZNN model has the lowest computation time, at 31.49 s, but its computation time during the stable phase increases to 2237.11 s. In contrast, the PTANN and ET-PTANN models have computation time of 272.04 s and 236.49 s, respectively, during the stable phase, resulting in lower overall computation time. The introduction of the event-triggered mechanism significantly reduces the computation cost of the ET-PTANN model during the convergence phase.

In conclusion, the application of the PTANN and ET-PTANN models to mobile sound source localization demonstrates their practicality and efficiency in real-world scenarios.

VII. Conclusion

In this paper, the PTANN and ET-PTANN models have been proposed to efficiently compute time-varying tensor MP inverse. The PTANN model has a novel adaptive parameter

and activation function, while the ET-PTANN model combines the evolution formula with event trigger. Theoretical analysis has confirmed the predefined-time convergence of both models and highlighted their improved computational efficiency. Simulation example and practical application have demonstrated the superior convergence rate and efficiency of the PTANN and ET-PTANN models. Future research will focus on further enhancing the computational efficiency of ANN models and exploring their applications in real-world scenarios.

REFERENCES

[1] J. Long, Y. Peng, T. Zhou, L. Zhao, and J. Li, "Fast and stable hyperspectral multispectral image fusion technique using Moore–Penrose inverse solver," *Appl. Sci.*, vol. 11, no. 16, p. 7365, 2021.

[2] N. Yu, R. Yang, and M. Huang, "Deep common spatial pattern based motor imagery classification with improved objective function," *Int. J. Netw. Dyn. Intell.*, pp. 73–84, 2022.

[3] B. Rahi, M. Li, and M. Qi, "A review of techniques on gait-based person re-identification," *Int. J. Netw. Dyn. Intell.*, pp. 66–92, 2023.

[4] H. Zhuang, Z. Lin, and K. A. Toh, "Blockwise recursive Moore–Penrose inverse for network learning," *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 52, no. 5, pp. 3237–3250, 2021.

[5] S. Zhang, Y. Dong, Y. Ouyang, Z. Yin, and K. Peng, "Adaptive neural control for robotic manipulators with output constraints and uncertainties," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 11, pp. 5554–5564, 2018.

[6] M. Liang and B. Zheng, "Further results on Moore–Penrose inverses of tensors with application to tensor nearness problems," *Comput. Math. Appl.*, vol. 77, no. 5, pp. 1282–1293, 2019.

[7] H. Ma, N. Li, P. S. Stanimirović, and V. N. Katsikis, "Perturbation theory for Moore–Penrose inverse of tensor via Einstein product," *Comput. Appl. Math.*, vol. 38, pp. 1–24, 2019.

[8] V. Y. Pan, F. Soleymani, and L. Zhao, "An efficient computation of generalized inverse of a matrix," *Appl. Math. Comput.*, vol. 316, pp. 89–101, 2018.

[9] P. S. Stanimirović, F. Roy, D. K. Gupta, and S. Srivastava, "Computing the Moore-Penrose inverse using its error bounds," *Appl. Math. Comput.*, vol. 371, p. 124957, 2020.

[10] D. D. Zontini, M. L. Mirkoski, and J. A. Santos, "Error bounds in the computation of outer inverses with generalized Schultz iterative methods and its use in computing of Moore-Penrose inverse," *Appl. Math. Comput.*, vol. 440, p. 127664, 2023.

[11] Y. Zhang, D. Guo, and Z. Li, "Common nature of learning between back-propagation and Hopfield-type neural networks for generalized matrix inversion with simplified models," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 4, pp. 579–592, 2013.

[12] P. S. Stanimirović, M. D. Petković, and D. Gerontitis, "Gradient neural network with nonlinear activation for computing inner inverses and the Drazin inverse," *Neural Process Lett.*, vol. 48, no. 1, pp. 109–133, 2018.

[13] X. Lv, L. Xiao, Z. Tan, Z. Yang, and J. Yuan, "Improved gradient neural networks for solving Moore–Penrose inverse of full-rank matrix," *Neural Process Lett.*, vol. 50, pp. 1993–2005, 2019.

[14] Y. Zhang and S. S. Ge, "Design and analysis of a general recurrent neural network model for time-varying matrix inversion," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 16, no. 6, pp. 1477–1490, 2005.

[15] Z. Tan, W. Li, L. Xiao, and Y. Hu, "New varying-parameter ZNN models with finite-time convergence and noise suppression for time-varying matrix Moore–Penrose inversion," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 8, pp. 2980–2992, 2019.

[16] F. Yu, L. Liu, L. Xiao, K. Li, and S. Cai, "A robust and fixed-time zeroing neural dynamics for computing time-variant nonlinear equation using a novel nonlinear activation function," *Neurocomputing*, vol. 350, pp. 108–116, 2019.

[17] J. Dai, L. Luo, L. Xiao, L. Jia, and X. Li, "An intelligent fuzzy robustness ZNN model with fixed-time convergence for time-variant Stein matrix equation," *Int. J. Intell. Syst.*, vol. 37, no. 12, pp. 11 670–11 691, 2022.

[18] J. Jin, J. Zhu, J. Gong, and W. Chen, "Novel activation functions-based ZNN models for fixed-time solving dynamirc Sylvester equation," *Neural Comput. & Applic.*, vol. 34, no. 17, pp. 14 297–14 315, 2022.

[19] Z. Qi, Y. Ning, L. Xiao, Y. He, J. Luo, and B. Luo, "Predefined-time zeroing neural networks with independent prior parameter for solving time-varying plural Lyapunov tensor equation," *IEEE Trans. Neural Netw. Learn. Syst.*, 2023, in press with DOI: 10.1109/TNNLS.2022.3233050.

[20] Y. Chai, H. Li, D. Qiao, S. Qin, and J. Feng, "A neural network for moore–penrose inverse of time-varying complex-valued matrices," *Int. J. Comput. Intell. Syst.*, vol. 13, no. 1, pp. 663–671, 2020.

[21] G. Sowmya and P. Thangavel, "Convergence of a finite-time zhang neural network for moore–penrose matrix inversion," in *Proceedings of International Conference on Intelligent Computing, Information and Control Systems: ICICCS 2020*. Springer, 2021, pp. 797–808.

[22] L. Xiao, X. Li, W. Huang, and L. Jia, "Finite-time solution of time-varying tensor inversion by a novel dynamic-parameter zeroing neural-network," *IEEE Trans. Ind. Inform.*, vol. 18, no. 7, pp. 4447–4455, 2021.

[23] L. Xiao, X. Li, L. Jia, and S. Liu, "Improved finite-time solutions to time-varying Sylvester tensor equation via zeroing neural networks," *Appl. Math. Comput.*, vol. 416, p. 126760, 2022.

[24] X. Wang, Y. Sun, and D. Ding, "Adaptive dynamic programming for networked control systems under communication constraints: a survey of trends and techniques," *Int. J. Netw. Dyn. Intell.*, pp. 85–98, 2022.

[25] P. Tallapragada and N. Chopra, "On event triggered tracking for nonlinear systems," *IEEE Trans. Autom. Control*, vol. 58, no. 9, pp. 2343–2348, 2013.

[26] Y. Li, Y. Li, and S. Tong, "Event-based finite-time control for nonlinear multi-agent systems with asymptotic tracking," *IEEE Trans. Autom. Control*, vol. 68, no. 6, pp. 3790–3797, 2022.

[27] X. Ge, Q. L. Han, and Z. Wang, "A threshold-parameter-dependent approach to designing distributed event-triggered H_∞ consensus filters over sensor networks," *IEEE Trans. Cybern.*, vol. 49, no. 4, pp. 1148–1159, 2018.

[28] B. Demirel, E. Ghadimi, D. E. Quevedo, and M. Johansson, "Optimal control of linear systems with limited control actions: Threshold-based event-triggered control," *IEEE Trans. Control Netw. Syst.*, vol. 5, no. 3, pp. 1275–1286, 2017.

[29] L. Xing, C. Wen, Z. Liu, H. Su, and J. Cai, "Event-triggered adaptive control for a class of uncertain nonlinear systems," *IEEE Trans. Autom. Control*, vol. 62, no. 4, pp. 2071–2076, 2016.

[30] E. Jiménez Rodríguez, A. J. Muñoz Vázquez, J. D. Sánchez Torres, M. Defoort, and A. G. Loukianov, "A Lyapunov-like characterization of predefined-time stability," *IEEE Trans. Autom. Control*, vol. 65, no. 11, pp. 4922–4927, 2020.

[31] C. Hu, H. He, and H. Jiang, "Fixed/preassigned-time synchronization of complex networks via improving fixed-time stability," *IEEE Trans. Cybern.*, vol. 51, no. 6, pp. 2882–2892, 2020.

[32] L. Sun, B. Zheng, C. Bu, and Y. Wei, "Moore–Penrose inverse of tensors via Einstein product," *Linear Multilinear Algebra*, vol. 64, no. 4, pp. 686–698, 2016.

[33] Q. W. Wang and X. Xu, "Iterative algorithms for solving some tensor equations," *Linear Multilinear Algebra*, vol. 67, no. 7, pp. 1325–1349, 2019.

[34] N. A. F. Senan, "A brief introduction to using ode45 in MATLAB," *University of California at Berkeley, USA*, 2007.

[35] D. Houcque and R. R. McCormick, "Applications of MATLAB: Ordinary differential equations (ODE)," *Robert R. McCormick School of Engineering and Applied Science-Northwestern University, Evanston*, 2008.

[36] A. B. Kisabo, C. Osheku, A. M. Lanre, and A. Aliyu Funmilayo, "Ordinary differential equations: MATLAB/Simulink® solutions," *Int J Eng Sci*, vol. 3, no. 8, pp. 1–7, 2012.

[37] D. Gerontitis, R. Behera, P. Tzekis, and P. Stanimirović, "A family of varying-parameter finite-time zeroing neural networks for solving time-varying Sylvester equation and its application," *J. Comput. Appl. Math.*, vol. 403, p. 113826, 2022.

[38] L. Xiao and Y. He, "A noise-suppression ZNN model with new variable parameter for dynamic Sylvester equation," *IEEE Trans. Ind. Inform.*, vol. 17, no. 11, pp. 7513–7522, 2021.

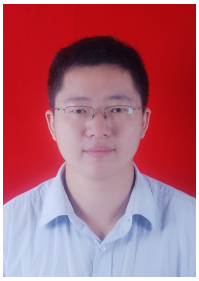
[39] S. Li, S. Chen, and B. Liu, "Accelerating a recurrent neural network to finite-time convergence for solving time-varying Sylvester equation by using a sign-bi-power activation function," *Neural Process Lett.*, vol. 37, pp. 189–205, 2013.

[40] T. Wang, H. Xiong, H. Ding, and L. Zheng, "TDOA-based joint synchronization and localization algorithm for asynchronous wireless sensor networks," *IEEE Trans. Commun.*, vol. 68, no. 5, pp. 3107–3124, 2020.

[41] L. Jin, J. Yan, X. Du, X. Xiao, and D. Fu, "RNN for solving time-variant generalized Sylvester equation with applications to robots and acoustic source localization," *IEEE Trans. Ind. Inform.*, vol. 16, no. 10, pp. 6359–6369, 2020.

Zhaohui Qi is currently a professor at College of Information Science and Engineering, Hunan Normal University, Changsha, P.R.China. Zhaohui Qi received his Ph.D. degree in computer application technology from Tianjin University, Tianjin, China, in 2006. His research interests include Pattern recognition, Intelligent information processing, and Bioinformatics.

Yingqiang Ning received the B.S. degree in electronic commerce from Hunan University of Finance and Economics, Changsha, China, in 2020. He is currently pursuing the master's degree in computer technology with the College of Information Science and Engineering, Hunan Normal University, Changsha, China. His research interests include neural networks and robotics.



Lin Xiao received the B.S. degree in electronic information science and technology from Hengyang Normal University, Hengyang, China, in 2009, and the Ph.D. degree in communication and information systems from Sun Yat-sen University, Guangzhou, China, in 2014.

He is currently a Professor with the College of Information Science and Engineering, Hunan Normal University, Changsha, China. He has authored over 100 articles in international conferences and journals. His main research interests include neural networks, robotics, and intelligent information processing.

Zidong Wang (Fellow, IEEE) received the B.Sc. degree in mathematics in 1986 from Suzhou University, Suzhou, China, and the M.Sc. degree in applied mathematics and the Ph.D. degree in electrical engineering from the Nanjing University of Science and Technology, Nanjing, China, in 1990 and 1994, respectively.

He is currently Professor of Dynamical Systems and Computing in the Department of Computer Science, Brunel University London, U.K. From 1990 to 2002, he held teaching and research appointments in universities in China, Germany and the UK. Prof. Wang's research interests include dynamical systems, signal processing, bioinformatics, control theory and applications. He has published a number of papers in international journals. He is a holder of the Alexander von Humboldt Research Fellowship of Germany, the JSPS Research Fellowship of Japan, William Mong Visiting Research Fellowship of Hong Kong.

Prof. Wang serves (or has served) as the Editor-in-Chief for *International Journal of Systems Science*, the Editor-in-Chief for *Neurocomputing*, the Editor-in-Chief for *Systems Science & Control Engineering*, and an Associate Editor for 12 international journals, including IEEE TRANSACTIONS ON AUTOMATIC CONTROL, IEEE TRANSACTIONS ON CONTROL SYSTEMS TECHNOLOGY, IEEE TRANSACTIONS ON NEURAL NETWORKS, IEEE TRANSACTIONS ON SIGNAL PROCESSING, and IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS–PART C. He is a Member of the Academia European, a Member of the European Academy of Sciences and Arts, an Academician of the International Academy for Systems and Cybernetic Sciences, a Fellow of the IEEE, a Fellow of the Royal Statistical Society, and a member of program committee for many international conferences.

Yongjun He received the B.E. degree in electronic information engineering from Jishou University, Jishou, China, in 2019 and the M.E. degree in electronics and communication engineering from Jishou University, Jishou, China, in 2022.

He is currently studying toward the Ph.D degree in operational research and cybernetics from College of Mathematics and Statistics, Hunan Normal University, Changsha, China. His research interests include neural networks and