Scientific Research Publishing

Journal of Computer and Communications, 2025, 13(4), 35-49
https://www.scirp.org/journal/jcc
ISSN Online: 2327-5227
ISSN Print: 2327-5219

Distributed Learning for Echo State Networks with Dynamic Event-Triggered Consensus

Xiyu Gong[1], Wu Ai[1,2*]

[1]School of Mathematics and Statistics, Guilin University of Technology, Guilin, China
[2]Guangxi Colleges and Universities Key Laboratory of Applied Statistics, Guilin, China
Email: *aiwu818@gmail.com

How to cite this paper: Gong, X.Y. and Ai, W. (2025) Distributed Learning for Echo State Networks with Dynamic Event-Triggered Consensus. *Journal of Computer and Communications*, **13**, 35-49.
https://doi.org/10.4236/jcc.2025.134003

Received: March 5, 2025
Accepted: April 14, 2025
Published: April 17, 2025

Abstract

This paper is devoted to studying the dynamic event-triggered consensus problem in distributed Echo State Networks (ESNs). The traditional ESN problem can be formulated as a set of distributed sub-problems with consensus constraints, which are solved using the Zero-Gradient-Sum (ZGS) distributed optimization strategy. Additionally, each agent employs a distributed dynamic event-triggered strategy based on its internal dynamic variables to achieve asymptotic consensus convergence. The purposeful dynamic triggering strategy is more communication-efficient compared to static triggering strategies and eliminates the need for continuous communication to update controllers or monitor triggering thresholds. Numerical simulation results are presented to validate the efficacy of the proposed algorithm.

Keywords

Echo State Network (ESN), Dynamic Event-Triggered Control, Optimization

1. Introduction

Echo state network is a novel type of recurrent neural network. Compared to other recurrent neural networks, ESN can handle time series classification and regression tasks by simply training the weight matrix connecting the hidden layer to the output layer using linear regression [1]. In practical applications, due to the large volume of data and privacy concerns, distributed ESNs have been widely applied [2]. In recent years, extensive research has been conducted on ESNs. For example, Zhu proposed a distributed learning method for DeepESN (DESN) based on improved singular value decomposition (SVD) and alternating direction method of multipliers (ADMM), which effectively reduces computation and communication costs for training DESNs in edge environments [3]. In [4], a novel distributed ESN

model integrated with an auto-encoder (AE-DESNm) eliminates the dimension disaster problem and uses an extreme learning machine (ELM) to extract features for estimating key variables. Integrating ESN with the decentralized average consensus (DAC) [5] approach facilitates achieving consensus in output values. Over time, considerable advancements have been made in methods for enhancing ESNs. For instance, a hybrid evolutionary algorithm that combines a competitive swarm optimizer with local search has been applied to mitigate the challenges associated with the random selection of input and reservoir weights [6]. Additionally, an optimized echo state network (O-ESN) proposed in [7] utilizes binary particle swarm optimization to minimize network complexity while improving generalization capabilities.

In a distributed framework, the assumption for achieving consensus convergence is often that each agent communicates continuously with its neighbors, which typically requires unlimited communication resources. However, a perfect communication network with infinite bandwidth capacity is usually not feasible in real-life scenarios [8]. To address the resource consumption issue in continuous communication among multi-agents, Lu proposed a general sampling framework where the sampled signals originate from a known union of subspaces, and the sampling operator is linear. Furthermore, the framework identified the minimum sampling requirements for several classes of signals [9]. If the continuous sampling signals fluctuate minimally, agents will engage in continuous unnecessary communication, resulting in a large amount of wasted samples [10]. The introduction of the event-based control method has garnered widespread interest, as it adjusts the sampling sequence based on events related to the system state, thus avoiding the communication resource wastage associated with traditional periodic sampling [11]. An event-triggered consensus protocol and triggering law proposed in [12] can achieve global consensus if and only if the underlying directed graph has a directed spanning tree. Moreover, it is found that the existence of a directed spanning tree is always a necessary and sufficient condition for achieving global consensus, regardless of whether input saturation is present. This conclusion remains valid even when more complex nonlinear dynamic behaviors are introduced. The fully distributed event-triggered algorithm proposed by Berneberg ensures asymptotic convergence to the average consensus state while allowing each agent to independently select a desired minimum inter-event time, thereby guaranteeing non-Zeno behavior [13]. In [14], static and dynamic event-triggered methods are introduced as secondary control techniques to restore voltage frequency and improve power sharing accuracy.

A new dynamic event-triggered control strategy, which includes internal dynamic variables, was proposed in [15]. Compared to static event-triggered strategies, it offers more advantages. As a result, dynamic event-triggered strategies have been used in recent years to solve the consensus problem in multi-agent systems (MAS). Later, dynamic event triggering was applied to islanded DC microgrids [16] and vehicle platoon systems [17], where the data from neighboring

DGs are used only at event-triggered times, thus reducing the burden on the communication network. In our previous work [18], we applied an event-triggered control policy to the communication component of the problem to avoid unnecessary transmissions, where the agents send messages only when the negotiation is critically needed. In [19], a distributed cooperative learning algorithm designed for stochastic configuration networks with fixed-time convergence, known as FixD-SCN, which is presented to tackle the 'Big Data' problem. To our best knowledge, little effort has been made in the field of dynamic event-triggered distributed learning over peer-to-peer networks, especially for RNNs.

The purpose of this paper is to propose a distributed learning algorithm for training the ESN, in which all agents converge asymptotically while minimizing the use of communication resources during the process. To accomplish this objective, we extend the multi-agent system with dynamic event-triggered algorithms to address the distributed learning problem. Initially, we convert the centralized ESN into a distributed form by breaking down the training process into smaller components, each subject to equality constraints on the output parameters. Next, a dynamic event-triggered algorithm is used to reduce communication during the process of optimizing the distributed ESN output weights. The proposed distributed algorithm is inspired by the use of dynamic event-triggered techniques to achieve distributed consensus convergence [20]. This work extends that of Scardapane *et al.* [21] by computing the final optimal consensus of ESN output parameters. The primary contributions presented in this paper are summarized below.

- A fully decentralized algorithm for training ESNs is introduced, where the training process is collaboratively executed without relying on a central coordinator.
- The convergence with dynamic event-triggered mechanism of a distributed learning algorithm is addressed to provide an asymptotically convergent.
- The convergent property of the proposed algorithm is mathematically substantiated when the Lyapunov theory is employed.

Notation: Denote the set of real (natural) numbers as $\mathbb{R}$ ($\mathbb{N}$). Let $\mathbf{1}_m$ ($\mathbf{0}_m$) represent the m-dimensional column vector where each entry is 1 (0). The matrix $\boldsymbol{I}_L \in \mathbb{R}^{L\times L}$ is the identity matrix. Letting $\|\cdot\|$ and $\|\cdot\|_1$ represent the Euclidean norm (2-norm) and 1-norm, respectively. $\rho(\cdot)$ denotes the spectral radius of a matrix. $\top$ is used to denote the transpose of a vector or matrix and $\otimes$ to represent the Kronecker product. For a vector $\boldsymbol{x} = [x_1, x_2, \cdots, x_D]^\top \in \mathbb{R}^D$, $|\boldsymbol{x}| = [|x_1|, \cdots, |x_D|]^\top$, denote the υ-norm of $\boldsymbol{x}$ by $\|\boldsymbol{x}\|_\upsilon = \left(\sum_{i=1}^{D} |x_i|^\upsilon\right)^{\frac{1}{\upsilon}}$, denote the sign of $\boldsymbol{x}$ by $\operatorname{sign}(\boldsymbol{x}) = [\operatorname{sign}(x_1), \cdots, \operatorname{sign}(x_D)]^\top$, where $\operatorname{sign}(\cdot)$ represents the sign function; denote $\boldsymbol{x}^{[\alpha]} = \left[x_1^{[\alpha]}, \cdots, x_D^{[\alpha]}\right]^\top$ with $x_i^{[\alpha]} = |x_i|^\alpha \operatorname{sign}(x_i)$, where $|\cdot|$ denotes the absolute value. Then we have $x_i \operatorname{sign}(x_i) = |x_i|$, $x_i x_i^{[\alpha]} = |x_i|^{\alpha+1}$ and

$|x_i|x_i^{[\alpha]} = x_i^{[\alpha+1]}$. Let $\preceq$ denote the matrix inequality sign, *i.e.*, $\boldsymbol{A} \preceq \boldsymbol{B}$, indicating that $\boldsymbol{B} - \boldsymbol{A}$ is a positive semidefinite matrix.

2. Problem Formulation

2.1. Graph Theory

We design a communication network of J agents distributed over a geographic area. Mathematically, we represent the communication network by a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{A})$ of order J with $\mathcal{V} = \{1, \cdots, J\}$ depicting the set of vertices, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ representing the edge set, and $\boldsymbol{A} = [a_{ij}] \in \mathbb{R}^{J \times J}$ illustrates the weighted adjacency matrix, where $a_{ij} > 0$ if $(j,i) \in \mathcal{E}$ and $a_{ij} = 0$, otherwise. The graph has no loops or multiple edges. If there is a connection between node i and node j, $(j,i) \in \mathcal{E}$. The set of neighbors of node i is represented by $\mathcal{N}_i = \{j : (j,i) \in \mathcal{E}\}$. Define the Laplacian matrix $\mathcal{L} = [l_{ij}] \in \mathbb{R}^{J \times J}$, where $l_{ij} = -a_{ij}$ for $i \neq j$ and $l_{ii} = \sum_{j=1}^{J} a_{ij}$. The matrix $\mathcal{L}$ is symmetric and positive semi-definite, with all eigenvalues being non-negative.

2.2. Problem Formulation

Lemma 1 (graphs) *For an undirected connected graph $\mathcal{G}$ with J vertices, the eigenvalues of $\mathcal{L}$ contains only one zero entry. $0 = \lambda_1(\mathcal{L})$ possessing a right eigenvector, $\mathbf{1}$, i.e., $\mathcal{L}\mathbf{1} = \mathbf{0}$, and other have positive real parts. Suppose $0 = \lambda_1(\mathcal{L}) < \lambda_2(\mathcal{L}) \leq \cdots \leq \lambda_J(\mathcal{L}) = \lambda_{\max}(\mathcal{L})$ in an ascending order. The minimum nonzero eigenvalue $\lambda_2(\mathcal{L}) = \min_{\boldsymbol{x} \neq 0, \mathbf{1}^\top \boldsymbol{x} = 0} \dfrac{\boldsymbol{x}^\top \mathcal{L} \boldsymbol{x}}{\boldsymbol{x}^\top \boldsymbol{x}}$, where $\boldsymbol{x} = [x_1, \cdots, x_J]^\top$. We have*

$$\boldsymbol{x}^\top \mathcal{L} \boldsymbol{x} = \frac{1}{2} \sum_{i=1}^{J} \sum_{j=1}^{J} a_{ij} (x_j - x_i)^2. \tag{1}$$

Furthermore, if $\mathbf{1}^\top \boldsymbol{x} = 0$, then $\boldsymbol{x}^\top \mathcal{L} \boldsymbol{x} \geq \lambda_2(\mathcal{L}) \boldsymbol{x}^\top \boldsymbol{x}$.

Lemma 2 (nonsingular) *Let $\kappa_1, \kappa_2, \cdots, \kappa_J \geq 0$. Then*

$$\left(\sum_{i=1}^{J} \kappa_i \right)^p \leq \sum_{i=1}^{J} \kappa_i^p \leq J^{1-p} \left(\sum_{i=1}^{J} \kappa_i \right)^p \tag{2}$$

if $0 < p \leq 1$, and

$$J^{1-p} \left(\sum_{i=1}^{J} \kappa_i \right)^p \leq \sum_{i=1}^{J} \kappa_i^p \leq \left(\sum_{i=1}^{J} \kappa_i \right)^p \tag{3}$$

if $1 < p < \infty$.

2.3. Distributed ESN over Distributed Datasets in a Multi-Agent Networks

ESN consists of three parts: an input layer, a hidden layer (reservoir), and an output layer. It is a variant of feedforward neural networks, with the training process involving feeding back output results to the hidden layer to adjust

connection weights. The input vector $\boldsymbol{x}_n \in \mathbb{R}^{D_{\text{in}}}$, with a dimensionality of D_{in}, is fed into a reservoir of dimensionality D_{res}. The internal state of the reservoir, $\boldsymbol{g}_n \in \mathbb{R}^{D_{\text{res}}}$, is computed as follows:

$$\boldsymbol{g}_n = f_{\text{res}}\left(\boldsymbol{\beta}_{\text{in}}^{\text{res}}\boldsymbol{x}_n + \boldsymbol{\beta}_{\text{res}}^{\text{res}}\boldsymbol{g}_{n-1} + \boldsymbol{\beta}_{\text{out}}^{\text{res}}\boldsymbol{y}_{n-1}\right), \tag{4}$$

where $\boldsymbol{\beta}_{\text{in}}^{\text{res}} \in \mathbb{R}^{D_{\text{res}}\times D_{\text{in}}}, \boldsymbol{\beta}_{\text{res}}^{\text{res}} \in \mathbb{R}^{D_{\text{res}}\times D_{\text{res}}}$ and $\boldsymbol{\beta}_{\text{out}}^{\text{res}} \in \mathbb{R}^{D_{\text{res}}\times D_{\text{out}}}$ are matrices generated randomly. The nonlinear function $f_{\text{res}}(\cdot)$ is applied, where $\boldsymbol{y}_{n-1} \in \mathbb{R}^{D_{\text{out}}}$ denotes the output from the previous time step. The output at the current step is then calculated as

$$\boldsymbol{y}_n = f_{\text{out}}\left(\boldsymbol{\beta}_{\text{in}}^{\text{out}}\boldsymbol{x}_n + \boldsymbol{\beta}_{\text{res}}^{\text{out}}\boldsymbol{g}_n\right). \tag{5}$$

Depending on the specific dataset, the parameters $\boldsymbol{\beta}_{\text{in}}^{\text{out}} \in \mathbb{R}^{D_{\text{out}}\times D_{\text{in}}}$ and $\boldsymbol{\beta}_{\text{res}}^{\text{out}} \in \mathbb{R}^{D_{\text{out}}\times D_{\text{res}}}$ are adapted, so as an invertible nonlinear function $f_{\text{out}}(\cdot)$. To train the readout, assume that there is a sequence of N input-output pairs $(\boldsymbol{x}_1, \boldsymbol{y}_1), \cdots, (\boldsymbol{x}_N, \boldsymbol{y}_N)$. The inputs are mapped to the reservoir, generating a series of internal states $\boldsymbol{g}_1, \cdots, \boldsymbol{g}_N$. At this stage, due to the unavailability of the ESN output for feedback, the designated output is substituted in (4). We obtain the hidden matrix $\boldsymbol{H}$ and output matrix $\boldsymbol{S}$ as

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{x}_1^{\top}, \boldsymbol{g}_1^{\top} \\ \vdots \\ \boldsymbol{x}_N^{\top}, \boldsymbol{g}_N^{\top} \end{bmatrix}, \quad \boldsymbol{S} = \begin{bmatrix} f_{\text{out}}^{-1}\left(\boldsymbol{y}_1^{\top}\right) \\ \vdots \\ f_{\text{out}}^{-1}\left(\boldsymbol{y}_N^{\top}\right) \end{bmatrix}. \tag{6}$$

The optimal weight vector for the output is obtained by solving the subsequent regularized least-squares problem:

$$\boldsymbol{\beta}^* = \arg\min_{\boldsymbol{\beta}} \left\{ \frac{1}{2}\left\|\boldsymbol{H}\boldsymbol{\beta} - \boldsymbol{S}\right\|^2 + \frac{C}{2}\left\|\boldsymbol{\beta}\right\|^2 \right\}, \tag{7}$$

where $\boldsymbol{\beta} = \left[\boldsymbol{\beta}_{\text{in}}^{\text{out}}, \boldsymbol{\beta}_{\text{res}}^{\text{out}}\right]^{\top} \in \mathbb{R}^{(D_{\text{in}}+D_{\text{res}})\times D_{\text{out}}}$ and $C > 0$ is a regularization parameter. The solution to the problem in (7) can be written as

$$\boldsymbol{\beta}^* = \left(\boldsymbol{H}^{\top}\boldsymbol{H} + C\boldsymbol{I}_L\right)^{-1}\boldsymbol{H}^{\top}\boldsymbol{S}, \tag{8}$$

where $L = D_{\text{in}} + D_{\text{res}}$.

To transform the centralized ESN into a distributed ESN, at the outset, we break down the centralized ESN task into several subtasks, each corresponding to a specific agent. Obtaining the optimal solution for the regularized problem leads to the determination of the output weights $\boldsymbol{\beta}^*$.

$$\begin{aligned} \min_{\{\boldsymbol{\beta}_i\},\{\boldsymbol{\xi}_{n,i}\}} \quad & \frac{1}{2}\sum_{i=1}^{J}\sum_{n=1}^{N_i}\left\|\boldsymbol{\xi}_{n,i}\right\|^2 + \frac{C}{2J}\sum_{i=1}^{J}\left\|\boldsymbol{\beta}_i\right\|^2 \\ \text{s.t.} \quad & \boldsymbol{\beta}_i^{\top}\boldsymbol{h}_{n,i} = \boldsymbol{s}_{n,i}^{\top} - \boldsymbol{\xi}_{n,i}^{\top}, \ \forall i \in \mathcal{V}, n = 1, \cdots, N_i, \\ & \boldsymbol{\beta}_i = \boldsymbol{\beta}_j, \ \forall i \in \mathcal{V}, j \in \mathcal{N}_i. \end{aligned} \tag{9}$$

The training error vector for the output neurons, corresponding to the training example $\boldsymbol{x}_{n,i}$ at the i-th learning agent, is denoted by $\boldsymbol{\xi}_{n,i} \in \mathbb{R}^{D_{\text{out}}}$. To convert a centralized ESN into a fully distributed ESN, the overall output weights $\boldsymbol{\beta}$ are

first duplicated and assigned to each agent. Equality constraints are then enforced along the edges of the communication network. By incorporating the constraints into the objective function, problem (9) can be reformulated as

$$\begin{aligned} \min_{\{\boldsymbol{\beta}_i\}} \quad & \sum_{i=1}^{J}\left(\frac{1}{2}\left\|\boldsymbol{H}_i\boldsymbol{\beta}_i-\boldsymbol{S}_i\right\|^2+\frac{C}{2J}\left\|\boldsymbol{\beta}_i\right\|^2\right) \\ \text{s.t.} \quad & \boldsymbol{\beta}_i=\boldsymbol{\beta}_j,\ \forall i\in\mathcal{V}, j\in\mathcal{N}_i, \end{aligned} \tag{10}$$

where the hidden matrix $\boldsymbol{H}_i$ and the training target matrix $\boldsymbol{S}_i$ are autonomously acquired from the respective dataset $\mathcal{D}_i$ observed by agent i itself. For simplicity in presentation, we begin by simplifying problem (10) as

$$\boldsymbol{\beta}^*=\arg\min_{\boldsymbol{\beta}}\left\{\sum_{i=1}^{J}u_i(\boldsymbol{\beta})\triangleq u(\boldsymbol{\beta})\right\}, \tag{11}$$

where

$$u_i(\boldsymbol{\beta})=\frac{1}{2}\left\|\boldsymbol{H}_i\boldsymbol{\beta}-\boldsymbol{S}_i\right\|^2+\frac{C}{2J}\left\|\boldsymbol{\beta}\right\|^2 \tag{12}$$

is the individual function performed by agent i locally. Agent i is capable of independently optimizing its local objective and establishing local consensus constraints through communication with neighboring agent j.

2.4. Distributed ESN Solved by the ZGS Algorithm

The ZGS algorithm aims to achieve optimal consensus on a manifold where the sum of the gradients of local functions is consistently zero. Let $\boldsymbol{\beta}_i(t)$ denote the state of agent i estimating the unknown optimum $\boldsymbol{\beta}^*$ at iterative step k, with $\boldsymbol{\beta}_i(0)$ indicating the initial value. The objective is to drive $\boldsymbol{\beta}_i(t)$ of each agent asymptotically to $\boldsymbol{\beta}^*$, *i.e.*, assuming convexity in the objective function, the global objective can be optimally achieved uniquely. According to the form of the objective function, it is known that it is a convex function. The global objective can be uniquely achieved optimally.

Assumption 1 *For each $i\in\mathcal{V}$, the function u_i demonstrates strong convexity and maintains twice continuous differentiability.*

Under Assumption 1, there is a unique minimizer $\boldsymbol{\beta}_i^*$, *i.e.*,

$$\boldsymbol{\beta}_i^*=\left(\boldsymbol{H}_i^{\top}\boldsymbol{H}_i+\frac{C}{J}\boldsymbol{I}_L\right)^{-1}\boldsymbol{H}_i^{\top}\boldsymbol{S}_i \tag{13}$$

such that the gradient $\nabla u_i\left(\boldsymbol{\beta}_i^*\right)=0$. The ZGS algorithm is formulated as an iterative procedure under the assumption of a zero gradient for u_i. The following implicit distributed algorithm is considered to achieve the goal.

$$\begin{aligned} &\nabla u_i\left(\boldsymbol{\beta}_i(t+1)\right)-\nabla u_i\left(\boldsymbol{\beta}_i(t)\right) \\ &=\gamma_1\sum_{j\in\mathcal{N}_i}a_{ij}\left(\boldsymbol{\beta}_j(t)-\boldsymbol{\beta}_i(t)\right)^{[\alpha_1]}+\gamma_2\sum_{j\in\mathcal{N}_i}a_{ij}\left(\boldsymbol{\beta}_j(t)-\boldsymbol{\beta}_i(t)\right)^{[\alpha_2]} \end{aligned} \tag{14}$$

with $\boldsymbol{\beta}_i(0)=\boldsymbol{\beta}_i^*$, where $\gamma_1,\gamma_2>0, 0<\alpha_1<1, \alpha_2>1$ and a_{ij} represents entries in the weighted adjacency matrix $\boldsymbol{A}$ of graph $\mathcal{G}$. Under the assumption that the graph $\mathcal{G}$ is undirected and connected, we have

$$\sum_{i=1}^{J}\left(\nabla u_i\left(\boldsymbol{\beta}_i(t+1)\right)-\nabla u_i\left(\boldsymbol{\beta}_i(t)\right)\right)=0. \tag{15}$$

Assuming $\boldsymbol{\beta}_i(0)=\boldsymbol{\beta}_i^*$, we have

$$\sum_{i=1}^{J}\nabla u_i\left(\boldsymbol{\beta}_i(0)\right)=\sum_{i=1}^{J}\nabla u_i\left(\boldsymbol{\beta}_i^*\right)=0. \tag{16}$$

Consequently, the consensus value $\boldsymbol{\beta}^o$ must be the optimizer $\boldsymbol{\beta}^*$. In such sense, a shared $\boldsymbol{\beta}^*$ is determined for each agent, ensuring $\nabla u\left(\boldsymbol{\beta}^*\right)=\sum_{i=1}^{J}\nabla u_i\left(\boldsymbol{\beta}^*\right)=0$. So that (10) holds. Furthermore, the gradient of the function $u_i(\boldsymbol{\beta})$ with respect to $\boldsymbol{\beta}$ can be easily computed as $\nabla u_i(\boldsymbol{\beta})=\left(\boldsymbol{H}_i^{\top}\boldsymbol{H}_i+\frac{C}{J}\boldsymbol{I}_L\right)\boldsymbol{\beta}-\boldsymbol{H}_i^{\top}\boldsymbol{S}_i$. The Hessian of $u_i(\boldsymbol{\beta})$ is

$$\nabla^2 u_i(\boldsymbol{\beta})=\boldsymbol{H}_i^{\top}\boldsymbol{H}_i+\frac{C}{J}\boldsymbol{I}_L\triangleq\tilde{\boldsymbol{\Omega}}_i. \tag{17}$$

The matrix $\tilde{\boldsymbol{\Omega}}_i$ is static, nonsingular, symmetric and positive definite. Letting $\theta_i=\lambda_{\min}\left(\tilde{\boldsymbol{\Omega}}_i\right)>0$ and $\Theta_i=\lambda_{\max}\left(\tilde{\boldsymbol{\Omega}}_i\right)>0$ denote the minimum and maximum eigenvalues of $\tilde{\boldsymbol{\Omega}}_i$, respectively, then

$$\theta_i I_L\preceq\tilde{\boldsymbol{\Omega}}_i\preceq\Theta_i I_L. \tag{18}$$

From the gradient of $u_i(\boldsymbol{\beta})$, we have $\nabla u_i\left(\boldsymbol{\beta}_i(t+1)\right)-\nabla u_i\left(\boldsymbol{\beta}_i(t)\right)$ $=\tilde{\boldsymbol{\Omega}}_i\left(\boldsymbol{\beta}_i(t+1)-\boldsymbol{\beta}_i(t)\right)$. Then, $\boldsymbol{\beta}_i(t+1)-\boldsymbol{\beta}_i(t)=\tilde{\boldsymbol{\Omega}}_i^{-1}\left(\nabla u_i\left(\boldsymbol{\beta}_i(t+1)\right)-\nabla u_i\left(\boldsymbol{\beta}_i(t)\right)\right)$. Substituting this expression into the left-hand side of (14), an algorithm that iteratively solves the problem in a distributed manner is designed. For each agent $i\in\mathcal{V}$, we design an algorithm for distributed ESN (ZGS-ESN). The ZGS-ESN iterates as follows:

$$\begin{aligned}\boldsymbol{\beta}_i(t+1)=\boldsymbol{\beta}_i(t)&+\gamma_1\tilde{\boldsymbol{\Omega}}_i^{-1}\sum_{j\in\mathcal{N}_i}a_{ij}\left(\boldsymbol{\beta}_j(t)-\boldsymbol{\beta}_i(t)\right)^{[\alpha_1]}\\&+\gamma_2\tilde{\boldsymbol{\Omega}}_i^{-1}\sum_{j\in\mathcal{N}_i}a_{ij}\left(\boldsymbol{\beta}_j(t)-\boldsymbol{\beta}_i(t)\right)^{[\alpha_2]}\end{aligned} \tag{19}$$

where $\gamma_1,\gamma_2>0, 0<\alpha_1<1, \alpha_2>1$, with initial values

$$\boldsymbol{\beta}_i(0)=\tilde{\boldsymbol{\Omega}}_i^{-1}\boldsymbol{H}_i^{\top}\boldsymbol{S}_i, \tag{20}$$

where the matrix $\tilde{\boldsymbol{\Omega}}_i$ is defined in (17). However, the continuous updating strategy may result in elevated communication consumption and frequent updates. The dynamic event-triggered strategy is an effective approach to overcome these disadvantages, in which the information interactions only happen at the trigger instants.

3. Distributed ESN with Dynamic Event-Triggered Consensus

Under the dynamic event-triggered strategy, we consider the following distributed algorithm:

$$\begin{aligned}\boldsymbol{\beta}_i(t+1)=\boldsymbol{\beta}_i(t)+\tilde{\boldsymbol{\Omega}}_i^{-1}\sum_{j\in N_i}a_{ij}\Big(\gamma_1\left(\boldsymbol{\beta}_j\left(t_k^i\right)-\boldsymbol{\beta}_i\left(t_k^i\right)\right)^{[\alpha_1]}\\+\gamma_2\left(\boldsymbol{\beta}_j\left(t_k^i\right)-\boldsymbol{\beta}_i\left(t_k^i\right)\right)^{[\alpha_2]}\Big),\ t\in\left[t_k^i,t_{k+1}^i\right)\end{aligned}\tag{21}$$

with initial values as in (20), where t_k^i is the latest triggering time for agent i. Drawing inspiration from [24], we begin by introducing the following combined measurement variable:

$$p_i(t)=\sum_{j=1}^{N}a_{ij}\left(\boldsymbol{\beta}_j(t)-\boldsymbol{\beta}_i(t)\right).\tag{22}$$

Based on the definition of the combined measurement, we express the measurement error as follows :

$$e_i(t)=p_i\left(t_k^i\right)-p_i(t).\tag{23}$$

Agent i requires the combined measurement $p_i(t)$ only at specific time intervals, known as triggering times (or event times), which are denoted by $t_0^i,t_1^i,t_2^i,\cdots$. At each triggering time t_k^i, agent i must calculate $p_i\left(t_k^i\right)$ through communication with its neighbors. Given the current triggering time t_k^i, the subsequent triggering time t_{k+1}^i is determined by the following triggering mechanism:

$$\begin{cases}t_{k+1}^i=\inf\left\{t>t_k^i\mid h_i^d\left(e_i(t),\bigcup_{j\in N_i}x_j(t),\eta_i(t)\right)\geq 0\right\},\\ \dot{\eta}_i(t)=g_i\left(e_i(t),p_i(t),\eta_i(t)\right),\end{cases}\tag{24}$$

where $\eta_i(t)$ represents the internal dynamic state and is the measurement error $e_i(t)$ that needs to be defined.

The triggering function $h_i^d(\cdot)$ is affected not only by the state of the controlled system but also by the internal dynamic state, denoted as $\eta_i(t)$. This type of triggering mechanism, referred to as a dynamic event-triggering mechanism, was first introduced for single controlled system in [15]. When the internal dynamic state $\eta_i(t)$ is disregarded , the dynamic event-triggering approach simplifies to a static event-triggering mechanism, expressed as $h_i^s\left(e_i(t),\bigcup_{j\in N_i}\boldsymbol{\beta}_j(t)\right)\geq 0$. In this study, the dynamic triggering conditions are defined by the following equations.

$$\begin{cases}t_{k+1}^i=\inf\left\{t>t_k^i\mid\left\|e_i(t)\right\|^2-\delta_i\left\|p_i(t)\right\|^2-\pi_i\eta_i(t)\geq 0\right\},\\ \dot{\eta}_i(t)=-\tau_i\eta_i(t)+\theta_i\left(\delta_i\left\|p_i(t)\right\|^2-\left\|e_i(t)\right\|^2\right),\ \eta_i(0)>0,\end{cases}\tag{25}$$

where, $\theta_i=68$, $\tau_i=0.03$, $\pi_i=0.002$ and $\delta_i=0.01$. In a multi-agent system, both static and dynamic event-triggering mechanisms need to be decentralized, meaning that each triggering mechanism is restricted to accessing information from its neighboring agents.

4. Main Results

In this section, a numerical example is provided to verify the effectiveness of the

proposed algorithm.

4.1. Description of the Datasets

The ZGS-FxTdt-ESN algorithm was validated on synthetic dataset designed for nonlinear system identification and the prediction of chaotic time series. For a large-scale simulation analysis, we utilize datasets that are approximately one to two orders of magnitude larger than those used in earlier studies. To be specific, for each dataset, we create 50 sequences, each comprising 2,000 elements. These sequences start from varying initial conditions, resulting in a total of 100,000 samples for each experiment.

The Mackey-Glass chaotic time-series (referred to as MKG) dataset is used for prediction tasks. Characterized in continuous time, this series follows the subsequent differential equation:

$$\dot{x}_n = -0.1x_n + \frac{0.2x_{n-\tau}}{1+x_{n-30}^{10}}. \tag{26}$$

For $\tau > 16.8$, the time-series given by (26) undergoes integration using a fourth-order Runge-Kutta method with a time step of 0.1. Subsequently, it is sampled every 10 time-instants. The task then becomes a 10-step-ahead prediction problem

$$y_n = x_{n+10}. \tag{27}$$

The functional form of Geometric Brownian Motion (GBM) is: $S_t = S_{t-1} \cdot \mathrm{e}^{\left(\mu - \frac{1}{2}\sigma^2\right)dt + \sigma dW}$. In this model: $\mu = 0.0003$ represents the drift rate, governing the long-term trend of the asset price; $\sigma = 0.005$ denotes the volatility, determining the magnitude of price fluctuations; $S(t)$ is the asset price at time t; $dW(t)$ is the increment of Brownian motion over an infinitesimal time interval dt. Based on this model, we will generate 50 time series, each containing 2000 consecutive time steps of simulated asset price data.

4.2. The Setting of the Algorithms

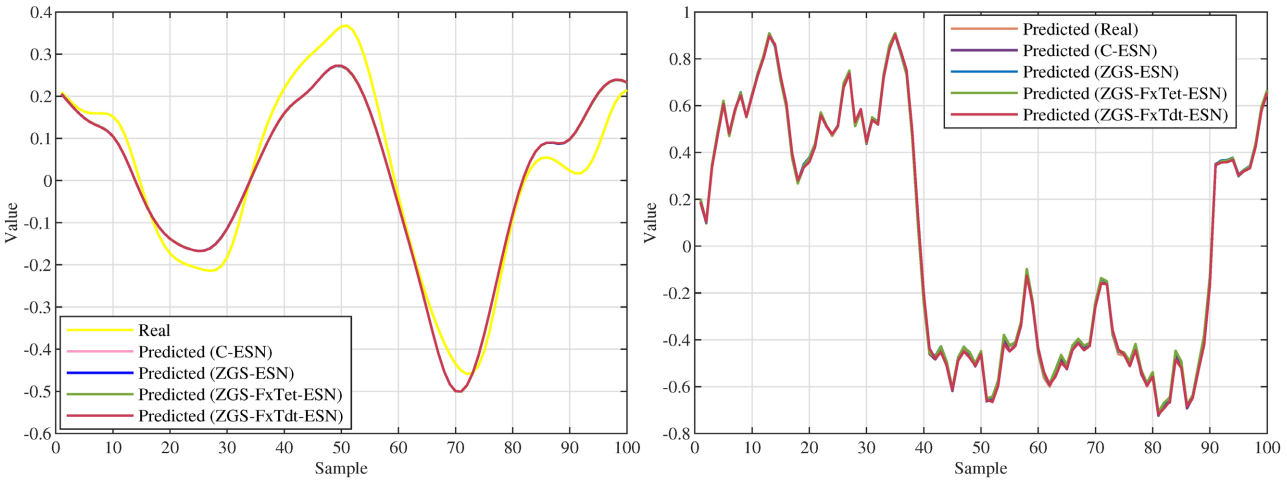
We create an agent network employing a model with a randomly generated topology with a 55% connectivity rate. Our experimentation involves varying the number of agents, starting from 4 and incrementing by 4 up to 20. To assess testing errors, we employ a 3-fold cross-validation on the initial set of 50 sequences. Each sequence is 2000 units in length. In every fold of the cross-validation process, the training sequences are distributed across the agents, after which we assess the performance of the four algorithms listed below:

- C-ESN: This algorithm is a single-agent ESN. The training data is centralized, and the ESN is trained by directly addressing the problem.
- L-ESN: In this algorithm, each agent independently trains a localized ESN using its own data, with no communication taking place. The average testing error is computed across all agents.
- ZGS-ESN: This algorithm set $\gamma_2 = 0$ and $\alpha_1 = 1$. The maximum number of iterations is 400 with an error level $\epsilon = 10^{-6}$. The parameter $\gamma_1 = 0.01$. ZGS-

FxTet-ESN: This algorithm is an extension of the ZGS-FxT-ESN with $\omega = 0.01$. We set the maximum number of iterations at 400, and the error threshold at $\epsilon = 10^{-6}$.

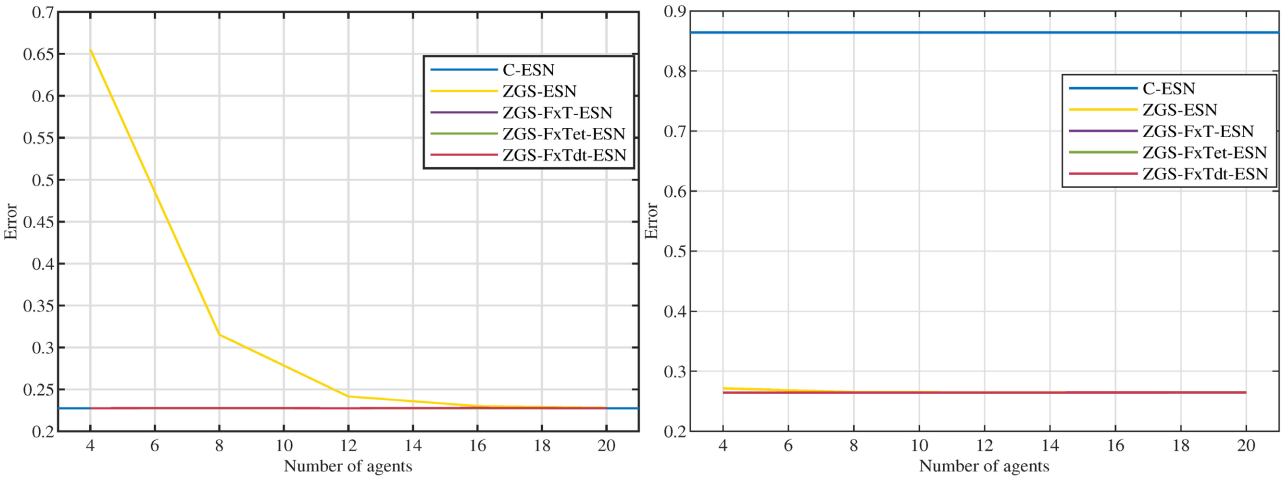
- ZGS-FxTdt-ESN: We set the maximum number of iterations at 400, $\omega = 0.01$ and the error threshold at $\epsilon = 10^{-6}$.

Each of the algorithms employs an identical ESN architecture, which is detailed in the following subsection. The threefold cross-validation process is iterated 10 times with variations in ESN initialization and data partitioning . Errors from each iteration and fold are accumulated. The trained ESN is applied to the test sequences to compute the error , and the resulting predicted outputs $\tilde{y}_1, \cdots, \tilde{y}_{N_{\text{test}}}$ are compiled. In this context, N_{test} denotes the count of testing samples excluding the initial washout elements from the test sequences. The normalized root mean-square error (N-RMSE) is defined as follows:



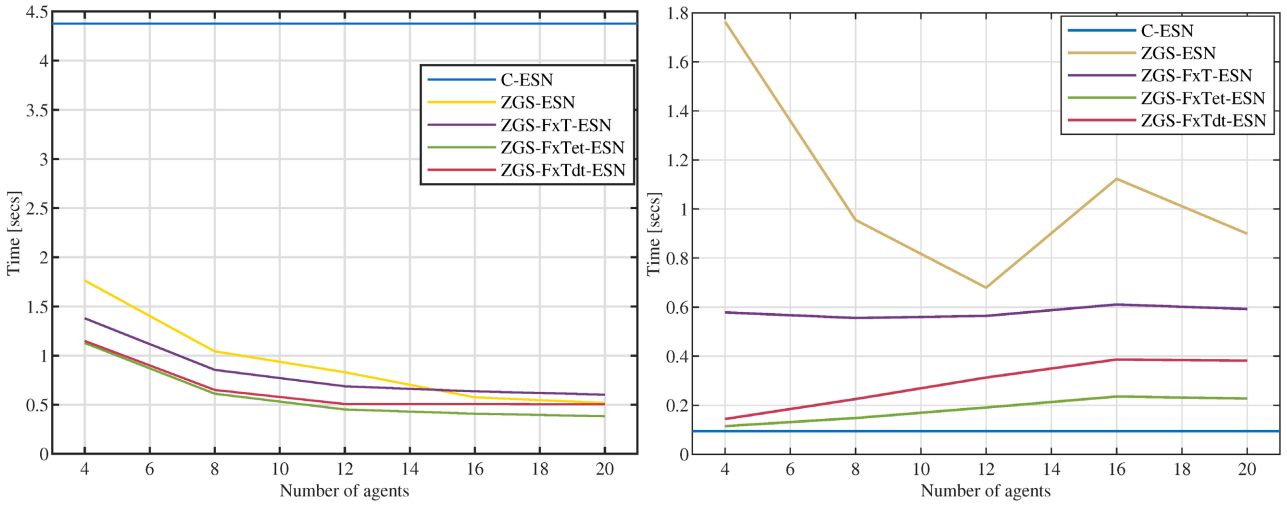
(a) [MKG] (b) [GBM]

Figure 1. The output of the algorithms for the network of 20 agents on MKG and GBM dataset. The performance of the distributed algorithms (ZGS-ESN, ZGS-FxTet-ESN and ZGS-FxTdt-ESN) is averaged across the agents.



(a) [MKG] (b) [GBM]

Figure 2. Evolution of testing error (NRMSE) for networks in the range of 4 to 20 agents.



(a) [MKG] (b) [GBM]

Figure 3. Evolution of training time for networks in the range of 4 to 20 agents.

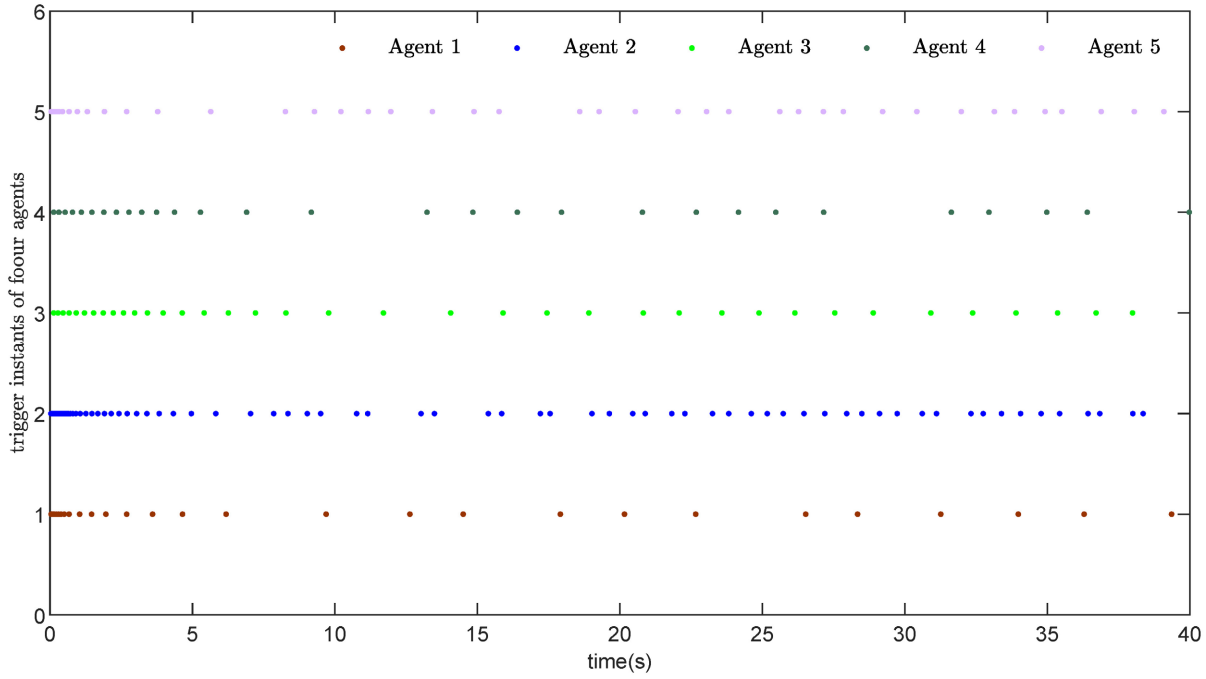


Figure 4. Event-triggered instants of the agents.

$$\mathrm{NRMSE}=\sqrt{\frac{\sum_{n=1}^{N_{\text{test}}}\left(\tilde{y}_n-y_n\right)^2}{N_{\text{test}}\hat{\sigma}_t}}. \tag{28}$$

In this context, $\hat{\sigma}_t$ signifies an empirical estimate of the variance in the actual output samples $y_1,\cdots,y_{N_{\text{test}}}$, where N_{test} is the count of testing samples, and $\tilde{y}_1,\cdots,\tilde{y}_{N_{\text{test}}}$, denote the predicted outputs.

4.3. ESN Architecture

A regularization factor of $C=10$ and a reservoir size of $D_{\text{res}}=300$ were

selected, and these choices proved to be effective in all situations. In the reservoir, we employ $\tanh(\cdot)$ nonlinearities, and for the output function, a scaled identity $f(s)=\alpha_t s$ is utilized. In all instances, the parameter α_t is set by searching within the set $\{0.1,0.3,\cdots,0.9\}$. The matrix $\boldsymbol{\beta}_{\text{in}}^{\text{res}}$ is populated with entries drawn from a uniform distribution within the range $[-\alpha_i,\alpha_i]$. The parameter α_i is explored within the same range as α_t. The matrix $\boldsymbol{\beta}_{\text{out}}^{\text{res}}$, which links the output to the reservoir, is initialized as a complete matrix with entries drawn from a uniform distribution within the range $[-\alpha_f,\alpha_f]$. The parameter α_f is searched within the same range as α_t. $\alpha_f=0$ is allowed for the case in which no output feedback is required.

The reservoir matrix $\boldsymbol{\beta}_{\text{res}}^{\text{res}}$ is initialized using a uniform distribution in the range of $[-1,1]$. Subsequently, around 55% of its connections are set to zero to foster sparsity. Lastly, the matrix is adjusted in scale to achieve a target spectral radius ρ, with ρ being explored within the same range as α_t. Furthermore, uniform noise is introduced during the state update of the reservoir, where the noise is sampled from a uniform distribution within the interval $[0,10^{-3}]$. Additionally, the initial $D=100$ elements are excluded from each sequence. Table 1 displays the ultimate configurations obtained through the grid-search process.

Table 1. Description of the parameters.

Dataset	ρ	α_i	α_t	α_f
MKG	0.9	0.3	0.5	0

4.4. Numerical Results

The test outputs of the four algorithms are presented in Figure 1, which demonstrate the prediction capability of the ESN. ZGS-FxTdt-ESN consistently follows the performance of the centralized solution across all scenarios. In Figure 1, the output value of ZGS-FxTdt-ESN basically coincides with the actual output value, indicating that ZGS-FxTdt-ESN maintains a relatively accurate output value. With the increasing number of agents, five algorithms show their test errors in Figure 2. We can see the test error of ZGS-FxTdt-ESN closely matches that of ZGS-FxTet-ESN. This shows that ZGS-FxTdt-ESN has the same test error as ZGS-FxTet-ESN, but ZGS-FxTdt-ESN has more advantages. Not only can its accuracy be comparable to that of ZGS-FxTet-ESN, but it also has a faster convergence speed. Figure 1 and Figure 2 indicate that the output values of C-ESN, ZGS-ESN and ZGS-FxTdt-ESN coincide and their test errors are all identical to that of ZGS-FxTet-ESN , which all suggests that their accuracies are similar. It has a faster convergence speed and only updates the state at the moment when the trigger condition is reached, which can save communication costs more. As shown in Figure 3, ZGS-FxTet-ESN achieves a significantly shorter training time than ZGS-FxTdt-ESN, primarily due to the latter's need for continuous internal-state

monitoring. In Figure 4, the event-triggered instants of agents are depicted, and it can be inferred that Zeno behavior is absent.

5. Conclusions

This paper presents a novel distributed learning algorithm designed for the training of a specialized recurrent neural network for the first time. The algorithm trained an identical ESN model among multiple agents. These agents were linked through a communication topology that operated without a central coordinator, relying solely on information exchange with neighboring nodes. The ZGS method was applied to solve the corresponding distributed optimization problem. The algorithm also incorporates the dynamic event-triggered method in order to reduce communication costs.

The effectiveness of the introduced algorithm was investigated by simulations. In the domain of distributed multi-agent systems, the convergence feature is a primary index for evaluating the performance of a distributed learning algorithm. Directions for future research could include considering dynamic handling of event triggering conditions to make it more applicable to real-life scenarios.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China (No. 62166013), the Natural Science Foundation of Guangxi (No. 2022GXNSFAA035499) and the Foundation of Guilin University of Technology (No. GLUTQD2007029).

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Zhang, H., Wang, Z. and Liu, D. (2014) A Comprehensive Review of Stability Analysis of Continuous-Time Recurrent Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, **25**, 1229-1262. https://doi.org/10.1109/tnnls.2014.2317880

[2] Verykios, V.S., Bertino, E., Fovino, I.N., Provenza, L.P., Saygin, Y. and Theodoridis, Y. (2004) State-of-the-Art in Privacy Preserving Data Mining. *ACM SIGMOD Record*, **33**, 50-57. https://doi.org/10.1145/974121.974131

[3] Zhu, M., Zhou, J., Cai, H., He, X. and Xiao, F. (2023) A Distributed Learning Method for Deep Echo State Network Based on Improved SVD and ADMM. 2023 *IEEE* 20*th International Conference on Mobile Ad Hoc and Smart Systems* (*MASS*), Toronto, 25-27 September 2023, 270-278. https://doi.org/10.1109/mass58611.2023.00040

[4] He, Y., Chen, L., Xu, Y., Zhu, Q. and Lu, S. (2023) A New Distributed Echo State Network Integrated with an Auto-Encoder for Dynamic Soft Sensing. *IEEE Transactions on Instrumentation and Measurement*, **72**, 1-8. https://doi.org/10.1109/tim.2022.3228278

[5] Alaviani, S.S. and Elia, N. (2019) Distributed Average Consensus over Random Net-

works. 2019 *American Control Conference* (*ACC*), Philadelphia, 10-12 July 2019, 1854-1859. https://doi.org/10.23919/acc.2019.8815134

[6] Wang, H. and Yan, X. (2015) Optimizing the Echo State Network with a Binary Particle Swarm Optimization Algorithm. *Knowledge-Based Systems*, **86**, 182-193. https://doi.org/10.1016/j.knosys.2015.06.003

[7] Long, J., Zhang, S. and Li, C. (2020) Evolving Deep Echo State Networks for Intelligent Fault Diagnosis. *IEEE Transactions on Industrial Informatics*, **16**, 4928-4937. https://doi.org/10.1109/tii.2019.2938884

[8] Zhang, B., Han, Q. and Zhang, X. (2017) Recent Advances in Vibration Control of Offshore Platforms. *Nonlinear Dynamics*, **89**, 755-771. https://doi.org/10.1007/s11071-017-3503-4

[9] Lu, Y.M. and Do, M.N. (2008) A Theory for Sampling Signals from a Union of Subspaces. *IEEE Transactions on Signal Processing*, **56**, 2334-2345. https://doi.org/10.1109/tsp.2007.914346

[10] Zhang, X., Han, Q. and Yu, X. (2016) Survey on Recent Advances in Networked Control Systems. *IEEE Transactions on Industrial Informatics*, **12**, 1740-1752. https://doi.org/10.1109/tii.2015.2506545

[11] Heemels, W.P.M.H., Sandee, J.H. and Van Den Bosch, P.P.J. (2008) Analysis of Event-Driven Controllers for Linear Systems. *International Journal of Control*, **81**, 571-590. https://doi.org/10.1080/00207170701506919

[12] Yi, X., Yang, T., Wu, J. and Johansson, K.H. (2019) Distributed Event-Triggered Control for Global Consensus of Multi-Agent Systems with Input Saturation. *Automatica*, **100**, 1-9. https://doi.org/10.1016/j.automatica.2018.10.032

[13] Berneburg, J. and Nowzari, C. (2019) Distributed Dynamic Event-Triggered Coordination with a Designable Minimum Inter-Event Time. 2019 *American Control Conference* (*ACC*), Philadelphia, 10-12 July 2019, 1424-1429. https://doi.org/10.23919/acc.2019.8815227

[14] Chen, J., Yue, D., Dou, C., Weng, S., Xie, X., Li, Y., *et al.* (2022) Static and Dynamic Event-Triggered Mechanisms for Distributed Secondary Control of Inverters in Low-Voltage Islanded Microgrids. *IEEE Transactions on Cybernetics*, **52**, 6925-6938. https://doi.org/10.1109/tcyb.2020.3034727

[15] Girard, A. (2015) Dynamic Triggering Mechanisms for Event-Triggered Control. *IEEE Transactions on Automatic Control*, **60**, 1992-1997. https://doi.org/10.1109/tac.2014.2366855

[16] Lu, J., Zhang, X., Zhang, B., Hou, X. and Wang, P. (2022) Distributed Dynamic Event-Triggered Control for Voltage Restoration and Current Sharing in DC Microgrids. *IEEE Transactions on Sustainable Energy*, **13**, 619-628. https://doi.org/10.1109/tste.2021.3123372

[17] Chen, J., Zhang, H. and Yin, G. (2023) Distributed Dynamic Event-Triggered Secure Model Predictive Control of Vehicle Platoon against Dos Attacks. *IEEE Transactions on Vehicular Technology*, **72**, 2863-2877. https://doi.org/10.1109/tvt.2022.3215966

[18] Ai, W. and Wang, D. (2020) Distributed Stochastic Configuration Networks with Cooperative Learning Paradigm. *Information Sciences*, **540**, 1-16. https://doi.org/10.1016/j.ins.2020.05.112

[19] Li, S., Ai, W. and Ge, X. (2020) Fixed-Time Distributed Cooperative Learning for Stochastic Configuration Networks. 2020 *Chinese Automation Congress* (*CAC*), Shanghai, 6-8 November 2020, 3380-3383. https://doi.org/10.1109/cac51589.2020.9326656

[20] Hu, W., Yang, C., Huang, T. and Gui, W. (2020) A Distributed Dynamic Event-Triggered Control Approach to Consensus of Linear Multiagent Systems with Directed Networks. *IEEE Transactions on Cybernetics*, **50**, 869-874. https://doi.org/10.1109/tcyb.2018.2868778

[21] Scardapane, S., Wang, D. and Panella, M. (2016) A Decentralized Training Algorithm for Echo State Networks in Distributed Big Data Applications. *Neural Networks*, **78**, 65-74. https://doi.org/10.1016/j.neunet.2015.07.006

[22] Bapat, R.B. (2010) Graphs and Matrices, Volume 27. Springer.

[23] Zuo, Z. (2015) Nonsingular Fixed-Time Consensus Tracking for Second-Order Multi-Agent Networks. *Automatica*, **54**, 305-309. https://doi.org/10.1016/j.automatica.2015.01.021

[24] Fan, Y., Feng, G., Wang, Y. and Song, C. (2013) Distributed Event-Triggered Control of Multi-Agent Systems with Combinational Measurements. *Automatica*, **49**, 671-675. https://doi.org/10.1016/j.automatica.2012.11.010