frontiers in Energy Research

ORIGINAL RESEARCH
published: 07 February 2022
doi: 10.3389/fenrg.2022.796325

A Reduced-Order RNN Model for Solving Lyapunov Equation Based on Efficient Vectorization Method

Zhiying Chen[1], *Zhaobin Du*[1]*, *Feng Li*[2] *and Chengjun Xia*[1]

[1]School of Electric Power Engineering, South China University of Technology, Guangzhou, China, [2]The Grid Planning and Research Center of Guangdong Power Grid Corporation, Guangzhou, China

With the trend of electronization of the power system, a traditional serial numerical algorithm is more and more difficult to adapt to the demand of real-time analysis of the power system. As one of the important calculating tasks in power systems, the online solution of Lyapunov equations has attracted much attention. A recursive neural network (RNN) is more promising to become the online solver of the Lyapunov equation due to its hardware implementation capability and parallel distribution characteristics. In order to improve the performance of the traditional RNN, in this study, we have designed an efficient vectorization method and proposed a reduced-order RNN model to replace the original one. First, a new vectorization method is proposed based on the special structure of vectorized matrix, which is more efficient than the traditional Kronecker product method. Second, aiming at the expanding effect of vectorization on the problem scale, a reduced-order RNN model based on symmetry to reduce the solution scale of RNN is proposed. With regard to the accuracy and robustness, it is proved theoretically that the proposed model can maintain the same solution as that of the original model and also proved that the proposed model is suitable for the Zhang neural network (ZNN) model and the gradient neural network (GNN) model under linear or non-linear activation functions. Finally, the effectiveness and superiority of the proposed method are verified by simulation examples, three of which are standard examples of power systems.

Keywords: Lyapunov equation, vectorization, reduced-order RNN, symmetry, ZNN, GNN

OPEN ACCESS

Edited by:
Yan Xu,
Nanyang Technological University, Singapore

Reviewed by:
Yushuai Li,
University of Oslo, Norway
Dazhong Ma,
Northeastern University, China

***Correspondence:**
Zhaobin Du
epduzb@scut.edu.cn

Specialty section:
This article was submitted to Smart Grids, a section of the journal Frontiers in Energy Research

Received: *16 October 2021*
Accepted: *03 January 2022*
Published: *07 February 2022*

Citation:
Chen Z, Du Z, Li F and Xia C (2022) A Reduced-Order RNN Model for Solving Lyapunov Equation Based on Efficient Vectorization Method. Front. Energy Res. 10:796325. doi: 10.3389/fenrg.2022.796325

INTRODUCTION

With the trend of the electronic power system, the scale of system computing is increasing day by day, while the demand of real-time analysis and calculation in the process of system operation remains unchanged. Traditional serial algorithms cannot solve this contradiction well, so various parallel algorithms and distributed methods appear successively. In power system state estimation, Chen et al. (2017) have used the SuperLU_MT solver to estimate the state of the actual power grid, making full use of the parallel characteristics of multicore and multi-thread solver. Liu Z. et al. (2020) have fully explored the parallelism in the calculation of continuous power flow and applied the continuous Newton method power flow model to realize the parallel solution algorithm of continuous power flow based on GPU in large scale and multiple working conditions. Moreover, a novel distributed dynamic event-triggered Newton–Raphson algorithm is proposed to solve the double-mode energy management problem in a fully distributed fashion (Li et al., 2020). Similarly, Li Y. et al. (2019) proposed an event-triggered distributed algorithm with some desirable

features, namely, distributed execution, asynchronous communication, and independent calculation, which can solve the issues of day-ahead and real-time cooperative energy management for multienergy systems. Given that software algorithms are essentially run by hardware, implementing functions directly from hardware is also an option for real-time computing. For example, Hafiz et al. (2020) proposed a real-time stochastic optimization of energy storage management using deep learning–based forecasts for residential PV applications, where the key of the real-time computation is the hardware controller. It is worth pointing out that compared with the aforementioned methods, the neural dynamics method has greater potential in the field of real-time calculation of power systems (Le et al., 2019), and its time constant can reach tens of milliseconds (Chicca et al., 2014) because of its parallel distribution characteristics and the convenience of hardware implementation.

The Lyapunov equation is widely used in some scientific and engineering fields to analyze the stability of dynamic systems (He et al., 2017; He and Zhang, 2017; Liu J. et al., 2020). In addition, the Lyapunov equation plays an important role in the controller design and robustness analysis of non-linear systems (Zhou et al., 2009; Raković and Lazar, 2014). In the field of power systems, the balanced truncation method, controller design, and stability analysis are also inseparable from the solution of the Lyapunov equation (Zhao et al., 2014; Zhu et al., 2016; Shanmugam and Joo, 2021). Therefore, many solving algorithms have been proposed to solve the Lyapunov equation. For example, Bartels and Stewart proposed the Bartels–Stewart method (Bartels and Stewart, 1972), which is a numerically stable solution. Lin and Simoncini (Lin and Simoncini, 2013) proposed the minimum residual method for solving the Lyapunov equation. Stykel (2008) used the low-rank iterative method to solve the Lyapunov equation and verified the effectiveness of the method through numerical examples. However, the efficiency of these serial processing algorithms is not high in large-scale applications and related real-time processing (Xiao and Liao, 2016).

Recently, due to its parallelism and convenience of hardware implementation, recurrent neural networks have been proposed and designed to solve the Lyapunov equation (Zhang et al., 2008; Yi et al., 2011; Yi et al., 2013; Xiao et al., 2019). The RNN mainly includes the Zhang neural network (ZNN) and gradient neural network (GNN) (Zhang et al., 2008). Most of the research studies on RNN focus on the improvement of model convergence. For example, Yi et al. (2013) point out that when solving a stationary or a non-stationary Lyapunov equation, the convergence of the ZNN is better than that of GNN. Yi et al. (2011) used a power-sigmoid activation function (PSAF) to build an improved GNN model to accelerate the iterative convergence of Lyapunov equation. In (Xiao and Liao, 2016), the sign-bi-power activation function (SBPAF) is used to accelerate the convergence of the ZNN model for solving the Lyapunov equation and the proposed ZNN model has finite-time convergence, which is obviously better than the previous ZNN and GNN models. In recent years, some studies have considered the noise-tolerant ZNN model. In Xiao et al. (2019), two robust non-linear ZNN (RNZNN) are established to find the solution of the Lyapunov equation under various noise conditions. Different from previous ZNN models activated by the typical activation functions (such as the linear activation function, the bipolar sigmoid activation function, and the power activation function), these two RNZNN models have predefined time convergence in the presence of various noises.

However, both GNN and ZNN need to transform the solution matrix from the matrix form to the vector form through the Kronecker product, which is called vectorization of the RNN model (Yi et al., 2011). The use of the Kronecker product will make the scale of the problem to be solved larger. As the size of the problem increases, the scaling effect of the Kronecker product becomes more obvious. The enlargement effect of the Kronecker product on the model size will not only lead to insufficient memory when the RNN is simulated on software but also make the hardware implementation of the RNN model need more devices and wiring, which increases the volume of hardware, the complexity of hardware production, and the failure rate of hardware. However, no study has discussed the order reduction of the RNN model.

It should be pointed out that the vectorized RNN model needs to be solved using a hardware circuit. However, as the relevant research of the RNN for solving the Lyapunov equation is still in the stage of theoretical exploration and improvement, there are no reports about hardware products of the RNN solver of the Lyapunov equation. Relevant studies (Zhang et al., 2008; Yi et al., 2011; Yi et al., 2013; Xiao and Liao, 2016; Xiao et al., 2019) simulate the execution process of the RNN hardware circuit through the form of software simulation, and this study also adopts this form. It is undeniable that the results of software simulation are consistent with those of hardware implementation. Therefore, the theoretical derivation and simulation results of the RNN in this article and in the literature (Zhang et al., 2008; Yi et al., 2011; Yi et al., 2013; Xiao and Liao, 2016; Xiao et al., 2019) can be extended to the scenarios of hardware implementation.

The RNN is used to solve the Lyapunov equation, and the ultimate goal is to develop an effective online calculation model to solve the Lyapunov equation, so it is of great significance to improve the calculation speed of the RNN. Current studies focus on improving the computational speed of the RNN by improving the convergence of RNN. However, how to efficiently realize vectorization of the RNN model is also a breakthrough to improve the computational efficiency of the RNN method. At present, the Kronecker product is generally used to transform the solution matrix into the vector form (Horn and Johnson, 1991). The Kronecker product actually performs multiple matrix multiplication operations, and the time complexity of multiplying two n×n matrices is O (n^3), so the time complexity of the Kronecker product increases rapidly as the scale increases. This means that the traditional matrix vectorization method based on the Kronecker product still has room for optimization.

In summary, this article proposes an efficient method for vectorizing the RNN model based on the special structure of the vectorized matrix, which is more efficient than the traditional

expansion method by the Kronecker product. Aiming at the expanding effect of vectorization on the problem scale, a reduced-order RNN model based on symmetry was proposed for solving the time-invariant Lyapunov equation, and the validity and applicability of the reduced-order RNN model were proved theoretically. The main contributions of this article are as follows.

1) An efficient method for vectorization of RNN model is proposed. Compared with the traditional vectorization method, this method has higher efficiency and less time consumption.
2) The reduced-order RNN model for solving the Lyapunov equation based on symmetry is proposed, which greatly reduces the solution scale. It is proved theoretically that the proposed model can maintain the same solution as that of the original model. Meanwhile, it is proved theoretically that the proposed model is suitable for the ZNN model and GNN model under linear or non-linear activation functions.
3) Several simulation examples are given to verify the effectiveness and superiority of the proposed efficient method for vectorization of the RNN and the reduced-order RNN model. It is also verified that the neural dynamics method is suitable for solving the Lyapunov equation of power systems through three standard examples of power systems.

In order to show the contributions of this study more clearly, the logical graph using the RNN model for solving the Lyapunov equation is shown in **Figure 1**, and the main novelties and differences of this article from Refs Yi et al. (2011); Yi et al. (2013); Xiao and Liao (2016); Xiao et al. (2019) are shown in **Table 1**.

In **Table1**, items and numbers correspond to the three steps of **Figure 1**. The relevant references include Yi et al. (2011); Yi et al. (2013); Xiao and Liao (2016); Xiao et al. (2019).

In conclusion, Refs (Yi et al., 2011; Yi et al., 2013; Xiao and Liao, 2016; Xiao et al., 2019) focus on constructing a stronger RNN model to improve the convergence and noise-tolerant ability, including using different activation functions and neural networks. However, this study focuses on the vectorization method and the reduced-order RNN model.

PROBLEM FORMULATION AND RELATED WORK

Problem Formulation

Consider the following well-known Lyapunov equation (Yunong Zhang and Danchi Jiang, 1995)

$$A^{\mathrm{T}}X(t) + X(t)A = -C, \tag{1}$$

where $A \in \mathbb{R}^{n\times n}$ is a constant stable real matrix and $C \in \mathbb{R}^{n\times n}$ is a constant symmetric positive-definite matrix. The objective is to find the unknown matrix $X(t) \in \mathbb{R}^{n\times n}$ to make the Lyapunov matrix **Eq. 1** hold true. Let $X^* \in \mathbb{R}^{n\times n}$ denote the theoretical solution of **Eq. 1**.

In addition, two of the most relevant works (i.e., GNN and ZNN models) are presented to solve the Lyapunov **Eq. 1** in the following.

GNN

According to the principle of GNN (Yi et al., 2011) and combined with the characteristics of Lyapunov equation, a corresponding GNN model can be designed to solve the Lyapunov equation. The design steps are as follows:

First, construct an energy function based on norm as follows:

$$\Delta = \frac{\|A^{\mathrm{T}}X(t) + X(t)A + C\|_{\mathrm{F}}^2}{2} \tag{2}$$

where $\|.\|_{\mathrm{F}}$ means F-norm. The minimum value of the energy function is the solution of the Lyapunov equation.

Second, based on the principle of the negative gradient descent of the GNN, the following formula can be constructed:

$$-\frac{\partial\Delta}{\partial X} = -A\left(A^{\mathrm{T}}X(t) + X(t)A + C\right) - \left(A^{\mathrm{T}}X(t) + X(t)A + C\right)A^{\mathrm{T}} \tag{3}$$

By introducing the adjustable positive parameter γ, the following GNN model can be obtained:

$$\begin{aligned}\dot{X}(\mathrm{t}) = &-\gamma A\left(A^{\mathrm{T}}X(t) + X(t)A + C\right)\\ &-\gamma\left(A^{\mathrm{T}}X(t) + X(t)A + C\right)A^{\mathrm{T}}\end{aligned} \tag{4}$$

where $\gamma > 0$, $X(t) \in \mathbb{R}^{n\times n}$, and $X(0) \in \mathbb{R}^{n\times n}$ is the initial value of $X(t)$.

Finally, the conventional linear GNN (**Eq. 4**) can be improved into the following non-linear expression by employing a non-linear activation function array $\mathcal{F}(\cdot)$:

$$\begin{aligned}\dot{X}(\mathrm{t}) = &-\gamma\left(A\mathcal{F}\left(A^{\mathrm{T}}X(t) + X(t)A + C\right)\right.\\ &\left.+\mathcal{F}\left(A^{\mathrm{T}}X(t) + X(t)A + C\right)A^{\mathrm{T}}\right)\end{aligned} \tag{5}$$

where $\mathcal{F}(\cdot)$: $\mathbb{R}^{n\times n} \to \mathbb{R}^{n\times n}$ denotes a matrix-valued activation function array of the GNN models. In this study, the bipolar sigmoid activation function (BPAF) is selected as the representative of the non-linear activation function of the GNN model for simulation because of its strong convergence (Yi et al., 2011). The expression of BPAF is as follows:

$$\mathcal{F}(x) = \frac{1 - \exp(-\delta x)}{1 + \exp(-\delta x)} \tag{6}$$

where δ is a constant and $\delta > 1$.

ZNN

First, following Zhang et al.'s design method (Zhang et al., 2002), we can define the following matrix-valued error function to monitor the solution process of Lyapunov **Eq. 1**:

$$E(t) = A^{\mathrm{T}}X(t) + X(t)A + C \tag{7}$$

Then in view of the definition of $E(t)$ and the design formula $\mathrm{d}E(t)/dt = -\gamma\varphi(E(t))$, the dynamic equation of the ZNN model for solving the online Lyapunov **Eq. 1** is derived as follows:

$$A^{\mathrm{T}}\dot{X}(\mathrm{t}) + \dot{X}(\mathrm{t})A = -\gamma\varphi\left(A^{\mathrm{T}}X(t) + X(t)A + C\right) \tag{8}$$

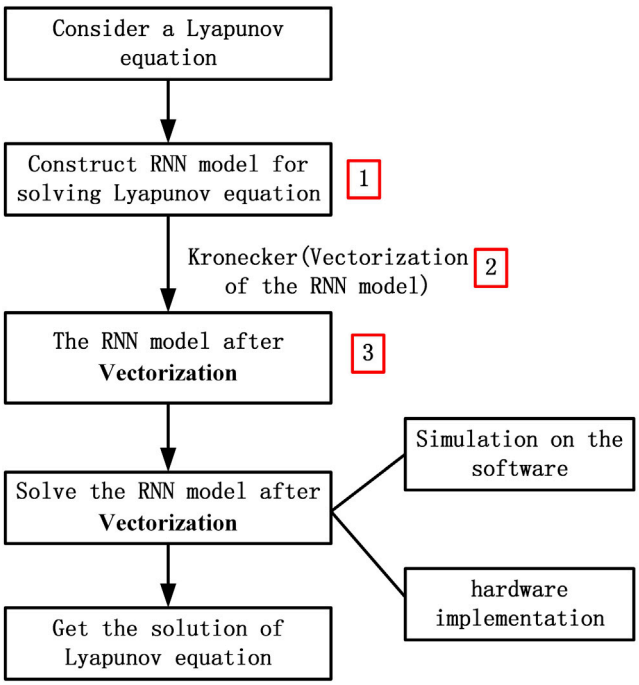

FIGURE 1 | Logical graph of using a RNN model for solving the Lyapunov equation.

TABLE 1 | Main novelties and differences of this article from the relevant references.

Number	Item	Refs. [9,15-17]	This article
1	Constructing RNN model	✓	✗
2	Vectorization	✗	✓
3	Reduced-order RNN model	✗	✓

where $\varphi(\cdot)$: $\mathbb{R}^{n\times n} \to \mathbb{R}^{n\times n}$ denotes a matrix-valued activation function array of the ZNN models. The definition of γ in the ZNN model is the same as that in the GNN model.

In this study, the RNZNN-1 model is selected as the representative of the non-linear activation function of the ZNN model for simulation because of its strong convergence (Xiao et al., 2019). The expression of the non-linear activation function in the RNZNN-1 model is as follows:

$$\varphi(x) = \left(a_1|x|^{\eta} + a_2|x|^{\omega}\right)\operatorname{sign}(x) + a_3 x + a_4 \operatorname{sign}(x) \tag{9}$$

where design parameters $0 < \eta < 1$, $\omega > 1$, $a_1 > 0$, $a_2 > 0$, $a_3 \geq 0$, $a_4 \geq 0$, and $\operatorname{sign}(x)$ denotes the signum function.

AN EFFICIENT METHOD FOR VECTORIZATION OF RNN MODEL

General Method of Vectorizing RNN Model

The RNN model needs to be transformed to the vector form so that it can be used for software simulation (Li X. et al., 2019) and hardware implementation.

Vectorization of the GNN Model

Yi et al. (2011) pointed out that the vectorization of GNN model is as follows:

$$\begin{aligned}\operatorname{vec}\dot{X}(t) = &-\gamma\big((A\otimes I)\mathcal{F}\big((A^{\mathrm{T}}\otimes I)\operatorname{vec}X(t) + (I\otimes A^{\mathrm{T}})\\ &\operatorname{vec}X(t) + \operatorname{vec}C\big) + (I\otimes A)\mathcal{F}\big((A^{\mathrm{T}}\otimes I)\operatorname{vec}X(t)\\ &+ (I\otimes A^{\mathrm{T}})\operatorname{vec}X(t) + \operatorname{vec}C\big)\big)\\ = &-\gamma\big((A\oplus A)\mathcal{F}\big((A^{\mathrm{T}}\oplus A^{\mathrm{T}})\operatorname{vec}X(t) + \operatorname{vec}C\big)\big)\end{aligned} \tag{10}$$

where

$$A \oplus A = A\otimes I + I\otimes A \tag{11}$$

$$A^{\mathrm{T}} \oplus A^{\mathrm{T}} = A^{\mathrm{T}}\otimes I + I\otimes A^{\mathrm{T}} \tag{12}$$

where ⊗ means the Kronecker product. Given $X = [x_{ij}] \in \mathbb{R}^{n\times n}$, we can vectorize X as a column vector, $\operatorname{vec}(X) \in \mathbb{R}^{n^2\times 1}$, which is defined as $\operatorname{vec}(X) = [x_{11},\ldots,x_{1n},x_{21},\ldots,x_{n1},\ldots,x_{nn}]^{\mathrm{T}}$.

Since the order of matrix addition and matrix transpose is interchangeable (Cheng and Chen, 2017),

$$(Y+Z)^{\mathrm{T}} = Y^{\mathrm{T}} + Z^{\mathrm{T}} \tag{13}$$

Applying this property to **Eq. 11**, we can get

$$(A\otimes I + I\otimes A)^{\mathrm{T}} = (A\otimes I)^{\mathrm{T}} + (I\otimes A)^{\mathrm{T}} \tag{14}$$

According to Chen and Zhou (2012), the relationship between the matrix transpose and Kronecker product is as follows:

$$(Y\otimes Z)^{\mathrm{T}} = Y^{\mathrm{T}}\otimes Z^{\mathrm{T}} \tag{15}$$

Applying this property to **Eq. 14**, we can get

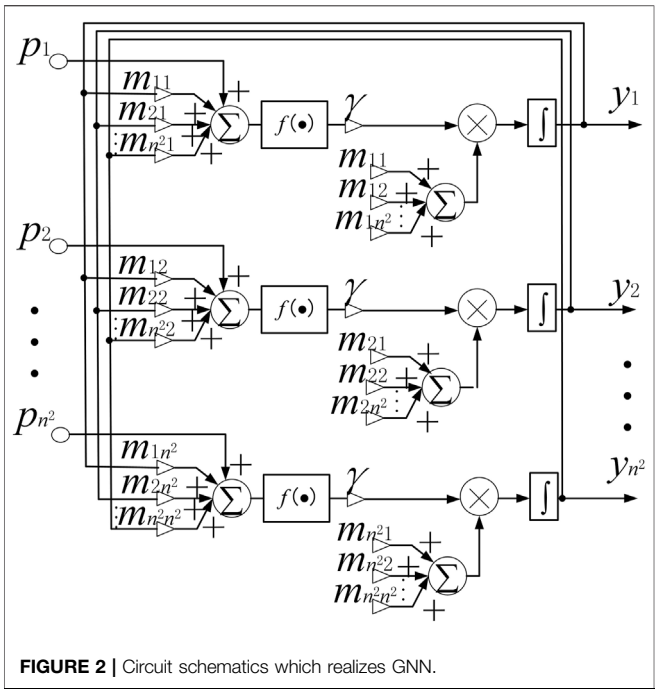

FIGURE 2 | Circuit schematics which realizes GNN.

$$(A \otimes I)^{\mathrm{T}} + (I \otimes A)^{\mathrm{T}} = A^{\mathrm{T}} \otimes I^{\mathrm{T}} + I^{\mathrm{T}} \otimes A^{\mathrm{T}} \tag{16}$$

Considering $I = I^{\mathrm{T}}$ and combining **Eqs 11**, **12**, **14** and **16**, we can get

$$(A \oplus A)^{\mathrm{T}} = A^{\mathrm{T}} \oplus A^{\mathrm{T}} \tag{17}$$

Vectorization of the ZNN Model

The vectorization process of the ZNN model is similar to that of the GNN. Carry out Kronecker product on **Eq. 8**, and we can get:

$$\begin{aligned}(A^{\mathrm{T}} \oplus A^{\mathrm{T}})\mathrm{vec}\dot{X}(t) &= -\gamma\varphi\left((A^{\mathrm{T}} \otimes I)\mathrm{vec}X(t) + (I \otimes A^{\mathrm{T}})\mathrm{vec}X(t) + \mathrm{vec}C\right)\\ &= -\gamma\varphi\left((A^{\mathrm{T}} \oplus A^{\mathrm{T}})\mathrm{vec}X(t) + \mathrm{vec}C\right)\end{aligned} \tag{18}$$

Vectorization of the RNN Model

By comparing **Eqs 10**, **17** and **18**, it can be seen that the key of vectorization of the RNN model is to solve $A^{\mathrm{T}} \oplus A^{\mathrm{T}}$.

According to **Eq. 12**, the calculation of $A^{\mathrm{T}} \oplus A^{\mathrm{T}}$ can be divided into three steps:

1) Calculate $A^{\mathrm{T}} \otimes I$

$$A^{\mathrm{T}} \otimes I = \begin{bmatrix} a_{11} & \cdots & 0 & \cdots & \cdots & \cdots & a_{n1} & \cdots & 0 \\ \vdots & \ddots & \vdots & \cdots & \cdots & \cdots & \vdots & \ddots & \vdots \\ 0 & \cdots & a_{11} & \cdots & \cdots & \cdots & 0 & \cdots & a_{n1} \\ \vdots & \vdots & \vdots & \ddots & & & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & & \ddots & & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & & & \ddots & \vdots & \vdots & \vdots \\ a_{1n} & \cdots & 0 & \cdots & \cdots & \cdots & a_{nn} & \cdots & 0 \\ \vdots & \ddots & \vdots & \cdots & \cdots & \cdots & \vdots & \ddots & \vdots \\ 0 & \cdots & a_{1n} & \cdots & \cdots & \cdots & 0 & \cdots & a_{nn} \end{bmatrix} \tag{19}$$

where $\begin{bmatrix} a_{ij} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & a_{ij} \end{bmatrix}$ is a diagonal matrix with n rows and n columns. $A^{\mathrm{T}} \otimes I$ is a matrix with n^2 rows and n^2 columns.

2) Calculate $I \otimes A^{\mathrm{T}}$

$$\begin{aligned} I \otimes A^{\mathrm{T}} &= \begin{bmatrix} a_{11} & \cdots & a_{n1} & \cdots & \cdots & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \cdots & \cdots & \cdots & \vdots & \ddots & \vdots \\ a_{1n} & \cdots & a_{nn} & \cdots & \cdots & \cdots & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & & & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & & \ddots & & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & & & \ddots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & \cdots & \cdots & \cdots & a_{11} & \cdots & a_{n1} \\ \vdots & \ddots & \vdots & \cdots & \cdots & \cdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & \cdots & \cdots & a_{1n} & \cdots & a_{nn} \end{bmatrix} \\ &= \begin{bmatrix} A^{\mathrm{T}} & 0 & \cdots & 0 \\ 0 & A^{\mathrm{T}} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & A^{\mathrm{T}} \end{bmatrix} \end{aligned} \tag{20}$$

where $I \otimes A^{\mathrm{T}}$ is a matrix with n^2 rows and n^2 columns.

3) Add $A^{\mathrm{T}} \otimes I$ to $I \otimes A^{\mathrm{T}}$

$$A^{\mathrm{T}} \otimes I + I \otimes A^{\mathrm{T}} = \begin{bmatrix} 2a_{11} & \cdots & a_{n1} & \cdots & a_{n1} & \cdots & 0 \\ \vdots & \ddots & \vdots & \cdots & \vdots & \ddots & \vdots \\ a_{1n} & \cdots & a_{nn} + a_{11} & \cdots & 0 & \cdots & a_{n1} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ a_{1n} & \cdots & 0 & \cdots & a_{11} + a_{nn} & \cdots & a_{n1} \\ \vdots & \ddots & \vdots & \cdots & \vdots & \ddots & \vdots \\ 0 & \cdots & a_{1n} & \cdots & a_{1n} & \cdots & 2a_{nn} \end{bmatrix} \tag{21}$$

TABLE 2 | Comparison of the original-order RNN and the reduced-order RNN under Example 1.

	Linearity		Non-linearity	
	Original-order RNN	Reduced-order RNN	Original-order RNN	Reduced-order RNN
Scale	9	6	9	6
Proportion	66.7%			
ZNN F-norm	2.8400e-5	2.4700e-5	3.1127e-4	1.9507e-4
GNN F-norm	5.4000e-5	1.5208e-4	3.4400e-5	4.0500e-5

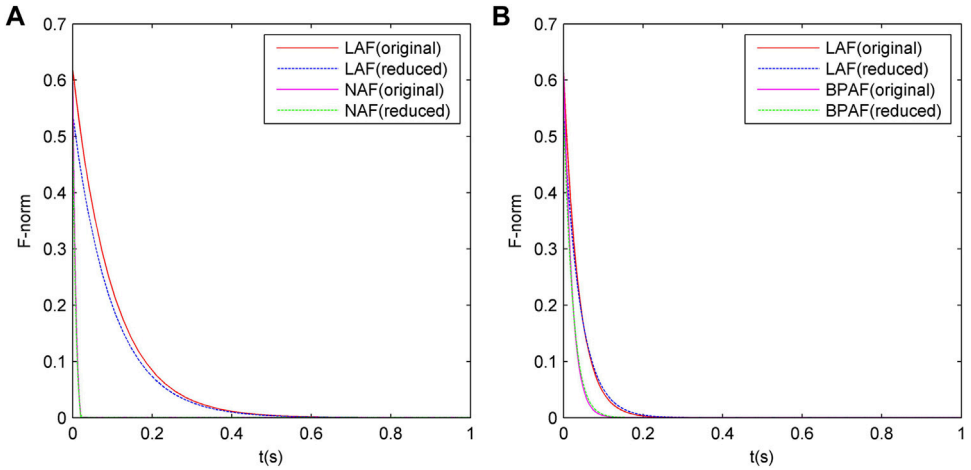

FIGURE 3 | Comparison of the convergence between the original-order and the reduced-order RNN models under Example 1. **(A)** is ZNN, **(B)** is GNN.

TABLE 3 | Comparison of the original-order RNN and the reduced-order RNN under Example 2.

	Linearity		Non-linearity	
	Original-order RNN	**Reduced-order RNN**	**Original-order RNN**	**Reduced-order RNN**
Scale	25	16	25	16
Proportion	64%			
ZNN F-norm	2.6800e-5	2.1400e-5	3.1300e-5	5.1600e-6
GNN F-norm	8.1400e-5	1.7200e-5	1.6487e-4	1.8600e-5

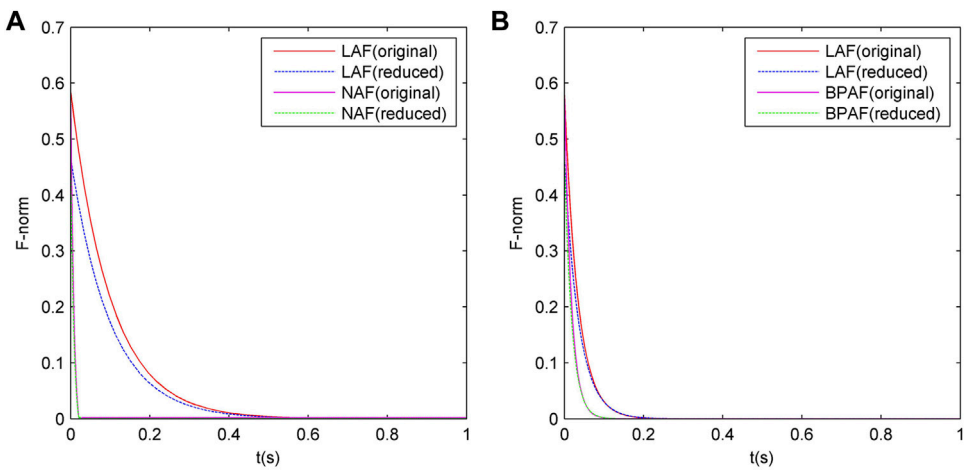

FIGURE 4 | Comparison of the convergence between the original-order and the reduced-order RNN models under Example 2. **(A)** is ZNN, **(B)** is GNN.

TABLE 4 | Comparison of the original-order RNN and the reduced-order RNN under Example 3.

	Linearity		Non-linearity	
	Original-order RNN	**Reduced-order RNN**	**Original-order RNN**	**Reduced-order RNN**
Scale	100	55	100	55
Proportion	55%			
ZNN F-norm	5.4400e-5	4.0800e-5	5.1745e-4	6.9400e-5
GNN F-norm	9.8582e-4	9.9281e-4	3.8700e-5	8.3900e-5

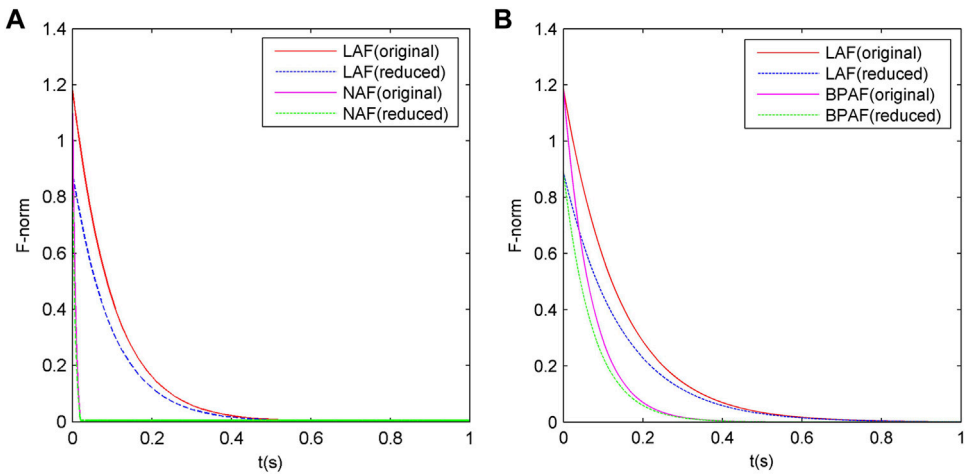

FIGURE 5 | Comparison of the convergence between the original-order and the reduced-order RNN models under Example 3. **(A)** is ZNN, **(B)** is GNN.

TABLE 5 | Comparison of the original-order ZNN and the reduced-order ZNN under Examples 4-6.

	Linearity (15)		Linearity (35)		Linearity (97)	
	Original-order ZNN	**Reduced-order ZNN**	**Original-order ZNN**	**Reduced-order ZNN**	**Original-order ZNN**	**Reduced-order ZNN**
Scale	225	120	1225	630	9409	4753
Proportion	53.3%		51.4%		50.5	
ZNN F-norm	3.3541e-2	3.2454e-2	3.4016e-2	1.5757e-2	1.6214e-3	3.1051e-3

An Efficient Method for Vectorization of RNN Model

According to the previous analysis, no matter how the matrix A is changed, the matrix structure of $A^{\mathrm{T}} \oplus A^{\mathrm{T}}$ is fixed. Based on the special structure of $A^{\mathrm{T}} \oplus A^{\mathrm{T}}$, an efficient method for vectorization of the RNN model is proposed in this article. The steps are as follows.

1) Create a matrix with n^2 rows and n^2 columns named K and fill K with the elements of A according to **Eq. 19**.
2) Fill K with the elements of A according to **Eq. 20**.
3) Add the corresponding element of A to the diagonal element of K according to **Eq. 21**.

The vectorization method of the RNN model proposed in this article is still based on the Kronecker product, but the time complexity is greatly reduced. Because the vectorization method proposed in this article replaces matrix multiplication with assignment and addition.

THE REDUCED-ORDER RNN MODEL FOR SOLVING LYAPUNOV EQUATIONS BASED ON SYMMETRY

Since the solution of **Eq. 1**, X^*, is always symmetric, as long as the upper trigonometric elements of X^* are solved, the

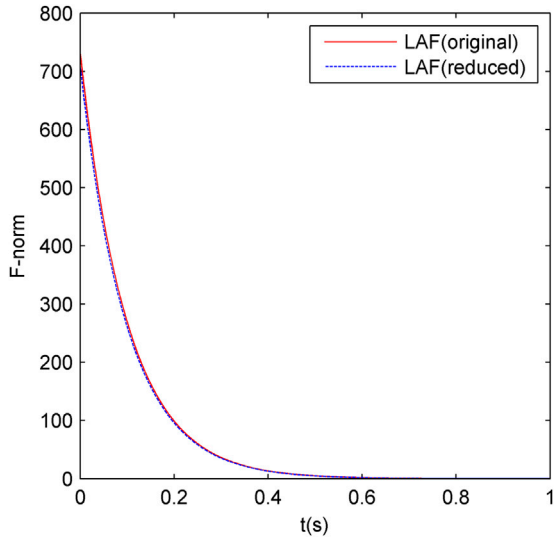

FIGURE 6 | Comparison of the convergence between the original-order and the reduced-order ZNN models under Example 4.

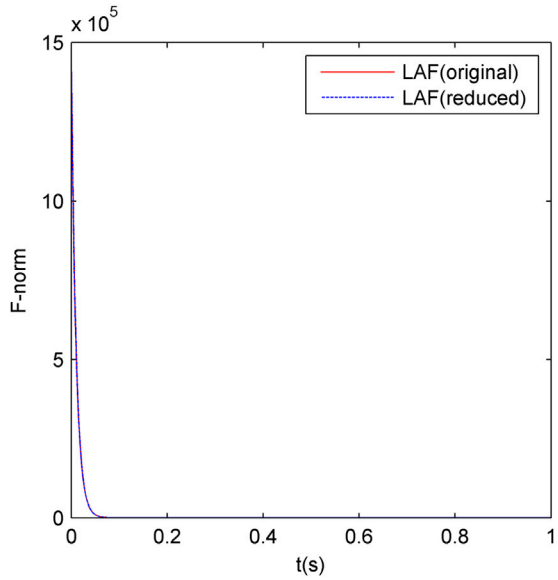

FIGURE 7 | Comparison of the convergence between the original-order and the reduced-order ZNN models under Example 5.

lower trigonometric elements of X^* can be obtained correspondently, which can greatly reduce the computational amount of solving the Lyapunov equations. Based on this idea, a reduced-order RNN model for solving the Lyapunov equation based on symmetry is proposed in this article.

The Reduced-Order ZNN Model With Linear Activation Function

Vectorization

Let's consider a ZNN model with linear activation function after vectorization. The formula is as follows:

$$\begin{bmatrix} k_{11} & k_{12} & \dots & k_{1,n+1} & \dots & k_{1,n^2} \\ k_{21} & k_{22} & \dots & k_{2,n+1} & \dots & k_{2,n^2} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ k_{n+1,1} & k_{n+1,2} & \dots & k_{n+1,n+1} & \dots & k_{n+1,n^2} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ k_{n^2,1} & k_{n^2,2} & \dots & k_{n^2,n+1} & \dots & k_{n^2,n^2} \end{bmatrix} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \vdots \\ \dot{x}_{n+1} \\ \vdots \\ \dot{x}_{n^2} \end{bmatrix} = -\gamma \left(\begin{bmatrix} k_{11} & k_{12} & \dots & k_{1,n+1} & \dots & k_{1,n^2} \\ k_{21} & k_{22} & \dots & k_{2,n+1} & \dots & k_{2,n^2} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ k_{n+1,1} & k_{n+1,2} & \dots & k_{n+1,n+1} & \dots & k_{n+1,n^2} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ k_{n^2,1} & k_{n^2,2} & \dots & k_{n^2,n+1} & \dots & k_{n^2,n^2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n+1} \\ \vdots \\ x_{n^2} \end{bmatrix} + \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_{n+1} \\ \vdots \\ c_{n^2} \end{bmatrix} \right) \tag{22}$$

where $K = A^{\mathrm{T}} \oplus A^{\mathrm{T}}$.

For the convenience of later discussion, $S \in \mathbb{R}^{n \times n}$ is constructed. Assign the following values to S as follows:

$$S = \begin{bmatrix} 1 & 2 & \cdots & n \\ n+1 & n+2 & \cdots & 2n \\ \vdots & \vdots & \ddots & \vdots \\ n(n-1)+1 & n(n-1)+2 & \cdots & n^2 \end{bmatrix} \tag{23}$$

Each element of S is the index number of the element of A at the same position.

We can expand X^* to $\mathrm{vec}X^* \in \mathbb{R}^{n^2 \times 1}$. Assuming that x_2^* and x_{n+1}^* are, respectively, the elements of the 1st row and the $n+1$th row of $\mathrm{vec}X^*$. Due to the symmetry, x_2^* will be equal to x_{n+1}^*.

Reduce the Column Number of *K*

If the Kronecker product is directly carried out on **Eq. 1**, then

$$\begin{bmatrix} k_{11} & k_{12} & \dots & k_{1,n+1} & \dots & k_{1,n^2} \\ k_{21} & k_{22} & \dots & k_{2,n+1} & \dots & k_{2,n^2} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ k_{n+1,1} & k_{n+1,2} & \dots & k_{n+1,n+1} & \dots & k_{n+1,n^2} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ k_{n^2,1} & k_{n^2,2} & \dots & k_{n^2,n+1} & \dots & k_{n^2,n^2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n+1} \\ \vdots \\ x_{n^2} \end{bmatrix} = -\begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_{n+1} \\ \vdots \\ c_{n^2} \end{bmatrix} \tag{24}$$

We can use $K\mathrm{vec}X = -\mathrm{vec}C$ to express **Eq. 24**. Lan (2017) points out that if A is stable and C is symmetrically positive definite, then the Lyapunov **Eq. 1** has a unique symmetric positive definite solution. Therefore, the K matrix of **Eq. 24** must be invertible.

Multiply both sides of **Eq. 22** by the inverse matrix of K, then we get

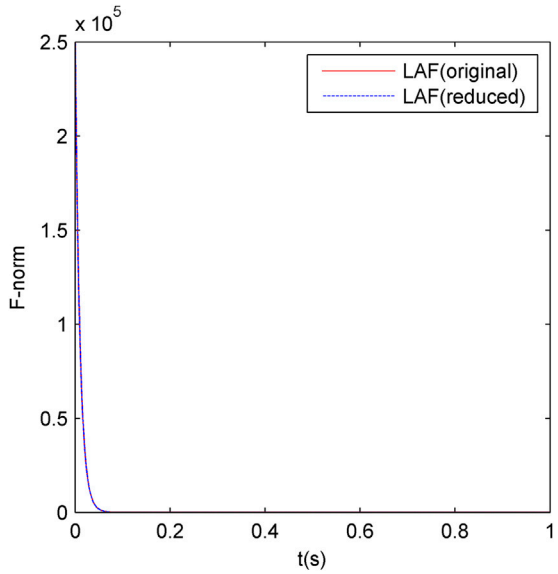

FIGURE 8 | Comparison of the convergence between the original-order and the reduced-order ZNN models under Example 6.

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \vdots \\ \dot{x_{n+1}} \\ \vdots \\ \dot{x_{n^2}} \end{bmatrix} = -\gamma \left(\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n+1} \\ \vdots \\ x_{n^2} \end{bmatrix} + K^{-1} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_{n+1} \\ \vdots \\ c_{n^2} \end{bmatrix} \right) \tag{25}$$

From **Eq. 25**, we can see that $K^{-1}\text{vec}C$ is the solution of the Lyapunov **Eq. 1**, which means $K^{-1}\text{vec}C = \text{vec}X^*$. Since X^* is a symmetric matrix, the differential equations of $x_2(t)$ and $x_{n+1}(t)$ are the same. If $x_2(0)$ and $x_{n+1}(0)$ are equal, then the time domain trajectories of $x_2(t)$ and $x_{n+1}(t)$ are the same, namely, $\dot{x_2}(t) = \dot{x_{n+1}}(t)$ and $x_2(t) = x_{n+1}(t)$. Therefore, for **Eq. 22**, the column $n+1$ of K can be added to the second column. Similarly, the same operation of column addition can be performed on the other columns in the symmetric positions. So the column number of K reduces to $0.5(n+1)n$. **Eq. 22** becomes

$$\begin{bmatrix} k_{11} & k_{12} & \dots & k_{1,0.5(n+1)n} \\ k_{21} & k_{22} & \dots & k_{2,0.5(n+1)n} \\ \vdots & \vdots & \ddots & \vdots \\ k_{n+1,1} & k_{n+1,2} & \dots & k_{n+1,0.5(n+1)n} \\ \vdots & \vdots & \vdots & \vdots \\ k_{n^2,1} & k_{n^2,2} & \dots & k_{n^2,0.5(n+1)n} \end{bmatrix} \begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \vdots \\ \dot{x}_{0.5(n+1)n} \end{bmatrix} =$$
$$-\gamma \left(\begin{bmatrix} k_{11} & k_{12} & \dots & k_{1,0.5(n+1)n} \\ k_{21} & k_{22} & \dots & k_{2,0.5(n+1)n} \\ \vdots & \vdots & \ddots & \vdots \\ k_{n+1,1} & k_{n+1,2} & \dots & k_{n+1,0.5(n+1)n} \\ \vdots & \vdots & \vdots & \vdots \\ k_{n^2,1} & k_{n^2,2} & \dots & k_{n^2,0.5(n+1)n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{0.5(n+1)n} \end{bmatrix} + \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_{n+1} \\ \vdots \\ c_{n^2} \end{bmatrix} \right), \tag{26}$$

Reduce the Row Number of *K*

When the steady state is considered, the differential term of **Eq. 26** is 0, and we can get:

$$\begin{bmatrix} k_{11} & k_{12} & \dots & k_{1,0.5(n+1)n} \\ k_{21} & k_{22} & \dots & k_{2,0.5(n+1)n} \\ \vdots & \vdots & \ddots & \vdots \\ k_{n+1,1} & k_{n+1,2} & \dots & k_{n+1,0.5(n+1)n} \\ \vdots & \vdots & \vdots & \vdots \\ k_{n^2,1} & k_{n^2,2} & \dots & k_{n^2,0.5(n+1)n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{0.5(n+1)n} \end{bmatrix} = -\begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_{n+1} \\ \vdots \\ c_{n^2} \end{bmatrix} \tag{27}$$

As mentioned before, if A is stable and C is symmetrically positive definite, then Lyapunov **Eq. 1** has a unique symmetric positive definite solution. Therefore, the number of equations should be the same as the number of variables (Cheng and Chen, 2017), namely, the rank of the coefficient matrix of **Eq. 27** is equal to $0.5(n+1)n$, which means the row vector set of the coefficient matrix of **Eq. 27** is linearly correlated.

We can construct the augmented matrix of **Eq. 27** and name it as G. If we define the first row of G as the vector α_1, the second

TABLE 6 | Comparison of the time cost of two vectorization methods for RNN models.

Scale	Method A (ms)	Method B (ms)	Proportion (%)
3	0.032	0.117	19.7
5	0.049	0.132	37.1
10	0.117	0.225	52.0
15	0.221	9.677	2.3
20	0.811	2.245	36.1
30	1.818	9.349	19.4
35	2.070	28.916	7.2
40	3.206	28.254	11.3
50	5.106	71.949	7.1
97	34.983	921.778	3.8

row as the vector α_2, and so on, the row n^2 is defined as the vector α_{n^2}.

According to the knowledge of linear algebra, the vector set $\alpha_1, \alpha_2, \ldots, \alpha_{n^2}$ is linearly dependent only if at least one of the vectors in the set can be represented linearly by the other vectors.

Let us define the vectors which can be represented linearly by the other vectors as the redundant vectors. Suppose that

$$\alpha_{n^2} = h_1\alpha_1 + h_2\alpha_2 + \ldots + h_{n^2-1}\alpha_{n^2-1} \tag{28}$$

where $h_1, h_2, \ldots, h_{n^2-1}$ are real numbers and at least one of them is not equal to 0. Then α_{n^2} is a redundant vector. As long as the redundant vectors are found out and the equations of their corresponding rows are deleted, a new augmented matrix with full row rank can be obtained.

According to the aforementioned analysis, as long as A is stable and C is symmetrically positive definite, then the redundant vectors must exist, which means there are always some vectors that satisfy **Eq. 28**. However, A and C are independent of each other. Considering there are always some vectors satisfying **Eq. 28** in the case of any stable A and any symmetrically positive definite C, there is only one possibility that for some redundant vector, there is another vector that is equal to it, and the row indexes of both of them are symmetric in the matrix S. Only in this way, based on the symmetric characteristics of matrix C, can the redundant vectors always satisfy **Eq. 28** when A and C are independent of each other.

For a redundant vector, its own row index and the row index of another vector equal to it can form a pair of indexes. We can use these index pairs to find the redundant vectors and delete the corresponding rows. In general, in an index pair, the equation corresponding to the index whose value is larger is selected for deletion.

After the row deletion, the row number of K also reduces to $0.5(n+1)n$. **Eq. 26** becomes

$$\begin{bmatrix} k_{11} & k_{12} & \cdots & k_{1,0.5(n+1)n} \\ k_{21} & k_{22} & \cdots & k_{2,0.5(n+1)n} \\ \vdots & \vdots & \ddots & \vdots \\ k_{0.5(n+1)n,1} & k_{0.5(n+1)n,2} & \cdots & k_{0.5(n+1)n,0.5(n+1)n} \end{bmatrix} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \vdots \\ \dot{x}_{0.5(n+1)n} \end{bmatrix}$$
$$= -\gamma\left(\begin{bmatrix} k_{11} & k_{12} & \cdots & k_{1,0.5(n+1)n} \\ k_{21} & k_{22} & \cdots & k_{2,0.5(n+1)n} \\ \vdots & \vdots & \ddots & \vdots \\ k_{0.5(n+1)n,1} & k_{0.5(n+1)n,2} & \cdots & k_{0.5(n+1)n,0.5(n+1)n} \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{0.5(n+1)n} \end{bmatrix} + \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_{0.5(n+1)n} \end{bmatrix}\right) \tag{29}$$

We can use $K_r\mathrm{vec}\dot{X}_r = -\gamma(K_r\mathrm{vec}X_r + \mathrm{vec}C_r)$ to express **Eq. 29**.

Reduced-Order GNN Model With Linear Activation Function

Consider a GNN model with linear activation function after vectorization. The formula is as follows:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \vdots \\ \dot{x}_{n+1} \\ \vdots \\ \dot{x}_{n^2} \end{bmatrix} = -\gamma \begin{bmatrix} k_{11} & k_{12} & \cdots & k_{1,n+1} & \cdots & k_{1,n^2} \\ k_{21} & k_{22} & \cdots & k_{2,n+1} & \cdots & k_{2,n^2} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ k_{n+1,1} & k_{n+1,2} & \cdots & k_{n+1,n+1} & \cdots & k_{n+1,n^2} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ k_{n^2,1} & k_{n^2,2} & \cdots & k_{n^2,n+1} & \cdots & k_{n^2,n^2} \end{bmatrix}^{\mathrm{T}}$$
$$\left(\begin{bmatrix} k_{11} & k_{12} & \cdots & k_{1,n+1} & \cdots & k_{1,n^2} \\ k_{21} & k_{22} & \cdots & k_{2,n+1} & \cdots & k_{2,n^2} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ k_{n+1,1} & k_{n+1,2} & \cdots & k_{n+1,n+1} & \cdots & k_{n+1,n^2} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ k_{n^2,1} & k_{n^2,2} & \cdots & k_{n^2,n+1} & \cdots & k_{n^2,n^2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n+1} \\ \vdots \\ x_{n^2} \end{bmatrix} + \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_{n+1} \\ \vdots \\ c_{n^2} \end{bmatrix}\right) \tag{30}$$

When the steady state is considered, the differential term of **Eq. 30** is 0, and **Eq. 30** is changed into **Eq. 24**. From the aforementioned derivation, it can be known that both $K\mathrm{vec}X = -\mathrm{vec}C$ and $K_r\mathrm{vec}X_r = -\mathrm{vec}C_r$ can obtain the solution of Lyapunov **Eq. 1**. On this basis, an attempt is made to construct a reduced-order GNN model with linear activation function after vectorization, as follows:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \vdots \\ \dot{x}_{0.5(n+1)n} \end{bmatrix} = -\gamma \begin{bmatrix} k_{11} & k_{12} & \cdots & k_{1,0.5(n+1)n} \\ k_{21} & k_{22} & \cdots & k_{2,0.5(n+1)n} \\ \vdots & \vdots & \ddots & \vdots \\ k_{0.5(n+1)n,1} & k_{0.5(n+1)n,2} & \cdots & k_{0.5(n+1)n,0.5(n+1)n} \end{bmatrix}^{T}$$
$$\left(\begin{bmatrix} k_{11} & k_{12} & \cdots & k_{1,0.5(n+1)n} \\ k_{21} & k_{22} & \cdots & k_{2,0.5(n+1)n} \\ \vdots & \vdots & \ddots & \vdots \\ k_{0.5(n+1)n,1} & k_{0.5(n+1)n,2} & \cdots & k_{0.5(n+1)n,0.5(n+1)n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{0.5(n+1)n} \end{bmatrix} + \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_{0.5(n+1)n} \end{bmatrix}\right). \tag{31}$$

When the steady state is considered, **Eq. 31** is changed into $K_r\mathrm{vec}X_r = -\mathrm{vec}C_r$, which means the solution of Lyapunov **Eq. 1** can be finally obtained by solving **Eq. 31**.

Reduced-Order RNN Model With Non-Linear Activation Functions

Before every non-linear activation function is introduced into the linear RNN, it will be theoretically proved that their introduction can guarantee the correct convergence of RNN. However, the reduced-order RNN in this article does not change the structure of RNN, but only changes the size of RNN. Because both the problem scale and the specific values of the matrix are generally expressed in symbolic form in the theoretical derivation (Xiao and Liao, 2016; Xiao et al., 2019), and the reduced-order RNN model in this article is still applicable to the relevant theoretical proof of introducing the non-linear activation functions into the linear RNN. In other words, the reduced-order method in this article can be applied to the RNN model with non-linear activation functions.

The Generation of the Reduced-Order RNN Model

The steps for generating the reduced-order RNN model are as follows:

1) S is constructed, and the construction logic is as described above.
2) Considering that the indexes in the symmetric positions of S can form $\frac{n(n-1)}{2}$ index pairs, we construct a matrix L and fill L with $\frac{n(n-1)}{2}$ index pairs. It is important to note that all of the elements in the first column of L must be the upper trigonometric elements (excluding diagonal elements) of S. For the convenience of presentation, suppose M is the first column of L, and N is the second column of matrix L.
3) For K and vec$C$ of RNN model, the rows corresponding to the element values of N are deleted.
4) For K and vec$C$ obtained in step 3, add the symmetric columns according to the element values of M and N, and the columns obtained by the addition will replace the columns corresponding to the element values of M, while the columns corresponding to the element values of N will be deleted. For vec$\dot{X}(t)$ and vec$X(t)$, the rows corresponding to the element values of N will be deleted.

It should be pointed out that in mathematical proof, if the order of row reduction and column reduction is exchanged, the correctness of the reduced-order RNN model cannot be proved or another proof method is needed to complete the proof. However, in the case that the mathematical proof has been completed, the order of row reduction and column reduction does not affect the final result, since we know in advance which rows and columns are to be deleted. In the aforementioned steps of generating the reduced-order RNN model, the reason why we carry out step 3 first is that it can reduce the computational amount of the column addition to achieving higher computational efficiency.

The Significance of the Reduced-Order RNN Model

In order to better explain the value and significance of the reduced-order RNN model proposed in this article, the differences before and after the order reduction are shown from the perspectives of software simulation and hardware implementation, respectively. For the convenience of discussion, the GNN is taken as an example to illustrate.

Simulation on the Software

We use the ode45 function of MATLAB to solve the GNN model after vectorization. By comparing **Eq. 30** and **Eq. 31**, it can be seen that the memory requirement of the reduced-order GNN model is much smaller than that of the original GNN model. Therefore, the reduced-order GNN model greatly alleviates the problem of insufficient memory that may occur in the software simulation of the GNN.

Hardware Implementation

When we use the traditional GNN model, the structure of the circuit diagram is shown as **Figure 2** (Yi et al., 2011).

Where $M = [m_{ij}] \in R^{n^2\times n^2} = A \oplus A$, $P = [p_1, p_2, \ldots, p_{n^2}]^{\mathrm{T}} = \mathrm{vec}C \in R^{n^2\times 1}$, $Y = [y_1, y_2, \ldots, y_{n^2}]^{\mathrm{T}} = \mathrm{vec}X \in R^{n^2\times 1}$.

When we use the reduced-order GNN model, the structure of the circuit diagram is shown as **Figure 2**, except that the n^2 in the diagram becomes $0.5(n+1)n$.

So the reduced-order GNN model greatly reduces the number of devices and wiring required for the hardware realization of GNN model, which is conducive to reducing the volume of hardware, the complexity of hardware production, and the failure rate of hardware.

ILLUSTRATIVE VERIFICATION

The simulation examples in this article are all completed on the MATLAB 2013b platform. In this article, the ode45 function of MATLAB is used to simulate the iterative process of RNN (Zhang et al., 2008). The corresponding computing performance is tested on a personal computer with Intel Core i7-4790 CPU @3.2GHz and 8 GB RAM.

Since there are great differences between software and hardware in the principle of realizing the integral function, there will be a big gap between the time cost in simulating the RNN process using software and the time cost in implementing the RNN model using hardware. Considering the research on the RNN model used to solve the Lyapunov equation is still in the stage of theoretical exploration, and has not reached the stage of hardware production for the time being, this article does not discuss the influence of the proposed reduced-order RNN model on the time consuming of RNN.

The Reduced-Order RNN Model for Solving Lyapunov Equation Based on Symmetry

Example 1

Let us consider the Lyapunov **Eq. 1** with the following coefficient matrices:

$$A = \begin{bmatrix} -11 & 2 & 3 \\ 4 & -7 & 6 \\ 1 & 8 & -12 \end{bmatrix} \text{ and } C = I^{3\times 3}$$

where A is similar to Example II in Xiao et al. (2019). However, A and C of Example II in Xiao et al. (2019) do not fit the definition of **Eq. 1**, so we change A a little bit and set C to be the identity matrix.

In this example, we set $\gamma = 10$, $\eta = 0.25$, $\omega = 4$, $a_1 = a_2 = a_3 = a_4 = 1$, and $\delta = 4$ (Yi et al., 2011; Xiao et al., 2019).

In order to demonstrate the advantages of the reduced-order RNN model, this article compares the performance of the reduced-order RNN and the original-order RNN, as shown in **Table 2**. In **Table 2**, linearity and non-linearity, respectively, mean the linear activation function and the non-linear activation function. Scale means the row number of vec$X$ or vecX_r. Proportion means the row number of vecX_r divided by the row number of vec$X$. The ZNN F-norm and GNN F-norm, respectively, mean $\| A^{\mathrm{T}}X + XA + C \|_F$ at the end of the simulation of ZNN and GNN.

In order to study the effect of order reduction method proposed in this article on the convergence of the RNN model, the F-norm curves of the original-order RNN model and the reduced-order RNN model are drawn, as shown in **Figure 3**. In **Figure 3A**, LAF means the ZNN model with linear activation functions. NAF means the ZNN model with the non-linear activation function of **Eq. 9**. In **Figure 3B**, LAF means the GNN model with linear activation functions. BPAF means the GNN model with the non-linear activation function of **Eq. 6**. In both **Figure 3A** and **Figure 3B**, F-norm refers to $\| A^{\mathrm{T}}X(t) + X(t)A + C \|_F$.

Example 2

To enlarge the scale of the example, we consider Lyapunov **Eq. 1** with the following coefficient matrices:

$$A = \begin{bmatrix} -17 & 3 & 4 & 5 & 6 \\ 3 & -17 & 3 & 4 & 5 \\ 4 & 3 & -17 & 3 & 4 \\ 5 & 4 & 3 & -17 & 3 \\ 6 & 5 & 4 & 3 & -17 \end{bmatrix} \text{ and } C = I^{5\times 5}$$

where A is similar to Example III in Xiao et al. (2019). However, A and C of Example III in Xiao et al. (2019) do not fit the definition of **Eq. 1**, so we change A a little bit and set C to be the identity matrix.

In this example, RNN's model parameters are the same as Example 1. Similar to Example 1, we can get **Table 3** and **Figure 4**. The definitions of all nouns in **Table 3** are the same as those in **Table 2**, and the definitions of all nouns in **Figure 4** are the same as those in **Figure 3**.

Example 3

$$A = \begin{bmatrix} -50 & 5 & 2 & 5 & 9 & 5 & 8 & 3 & 6 & 10 \\ 8 & -51 & 10 & 10 & 7 & 1 & 4 & 4 & 3 & 8 \\ 4 & 3 & -51 & 2 & 4 & 10 & 3 & 9 & 8 & 5 \\ 6 & 10 & 8 & -49 & 6 & 10 & 5 & 1 & 2 & 5 \\ 2 & 2 & 9 & 2 & -47 & 5 & 1 & 1 & 7 & 5 \\ 7 & 9 & 9 & 2 & 1 & -47 & 2 & 2 & 2 & 4 \\ 3 & 6 & 1 & 9 & 3 & 4 & -42 & 7 & 4 & 6 \\ 7 & 10 & 4 & 6 & 2 & 10 & 10 & -44 & 7 & 6 \\ 7 & 1 & 3 & 6 & 2 & 4 & 6 & 7 & -44 & 9 \\ 8 & 5 & 9 & 2 & 3 & 2 & 1 & 5 & 1 & -44 \end{bmatrix} \text{ and } C = I^{10\times 10}$$

A 10*10 matrix is randomly generated and then α-shift is applied to the matrix to make it stable (Yang et al., 1993), which is the generation process of A of Example 3 in this article. C is set to be the identity matrix.

In this example, RNN's model parameters are the same as Example 1. Similar to Example 1, we can get **Table 4** and **Figure 5**. The definitions of all nouns in **Table 4** are the same as those in **Table 2**, and the definitions of all nouns in **Figure 5** are the same as those in **Figure 3**.

Based on the information in the aforementioned three tables, we can draw the following conclusions:

a) The reduced-order RNN model has a very obvious effect, with the scale reduced by about 33–45%. Moreover, the effect of the reduced-order RNN model becomes more obvious with the increase in the size of the example. According to $\lim_{n\to\infty} \frac{0.5n(n+1)}{n^2} = 0.5$, it can be seen that when the size of the example is larger, the percentage of the scale decrease is closer to 50%. The reduced-order RNN model not only greatly alleviates the problem of insufficient memory in the software simulation of RNN but also greatly reduces the number of devices and wiring required for the hardware realization of RNN model, which is conducive to reducing the volume of hardware, the complexity of hardware production, and the failure rate of hardware.
b) Under different case scales, whether it is ZNN or GNN, whether it is linear activation function or non-linear activation function, the steady-state errors of the reduced-order RNN model are very close to 0, which means the reduced-order RNN model can always converge to the correct solution of the Lyapunov equation. This indicates that the reduced-order RNN model is applicable to ZNN and GNN, as well as the scenarios of linear activation function and non-linear activation function, which is consistent with the theoretical derivation results above.
c) Under different case scales, the difference in the steady-state accuracy between the reduced-order RNN and the original-order RNN is very small, indicating that the reduced-order RNN basically does not affect the steady-state accuracy of RNN.

Based on the information in the aforementioned three figures, we can draw the following conclusions:

a) Under different case scales, the reduced-order RNN models with linear or non-linear activation functions either have a little effect on the iterative convergence characteristics or enhance the convergence at the beginning of the iteration process and have a little effect on the convergence at the end of it.
b) Under the non-linear activation functions, the convergence of the ZNN model is always stronger than that of the GNN model when other conditions are fixed.
c) Under the linear activation function, the convergence of the ZNN model is weaker than that of the GNN model when the size of the examples is small (e.g., Example 1 and Example 2). The convergence of the linear ZNN model is stronger than that of the linear GNN model when the size of the examples is large (e.g., Example 3).
d) For both ZNN and GNN, the convergence of the RNN model with non-linear activation function is always stronger than that of the linear RNN model.
e) With the increase in the size of the examples, the convergence of ZNN is basically unchanged, while the convergence of GNN will become significantly worse.

Example 4

In order to verify the applicability of neural dynamics method to the power system, the corresponding Lyapunov equation describing system controllability is generated for the IEEE three-machine nine-node system according to the principle of the balanced truncation method in (Zhao et al., 2014). The input signal is the rotor speed deviation and the output signal is the auxiliary stabilizing signal (Zhu et al., 2016). It should be noted that the IEEE standard systems used in this article come from the examples of the PST toolkit (Lan, 2017), and the linearization process of the system is realized by the svm_mgen.m of PST toolkit. We set $\gamma = 10$. $A \in \mathbb{R}^{15\times 15}$ and $C \in \mathbb{R}^{15\times 15}$ of the Lyapunov equation are detailed in **Supplementary Material**.

In this example, the linear ZNN was selected for testing. Similar to Example 1, we can get **Table 5** and **Figure 6**. The definitions of all nouns in **Table 5** are the same as those in **Table 2** and the definitions of all nouns in **Figure 6** are the same as those in **Figure 3**.

Example 5

Similar to Example 4, we generate the corresponding Lyapunov equation describing system controllability of the IEEE 16-machine

system. $A \in \mathbb{R}^{35\times35}$ and $C \in \mathbb{R}^{35\times35}$ of the Lyapunov equation are detailed in **Supplementary Material**. We set $\gamma = 100$. The simulation results are shown in **Table 5** and **Figure 7**. The definitions of all nouns in **Figure 7** are the same as those in **Figure 3**.

Example 6

Similar to Example 4, we generate the corresponding Lyapunov equation describing system controllability of the IEEE 48-machine system. $A \in \mathbb{R}^{97\times97}$ and $C \in \mathbb{R}^{97\times97}$ of the Lyapunov equation are detailed in **Supplementary Material**. We set $\gamma = 100$. The simulation results are shown in **Table 5** and **Figure 8**. The definitions of all nouns in **Figure 8** are the same as those in **Figure 3**.

It can be seen from **Table 5** and **Figures 6–8** that the neural dynamics method used to solve Lyapunov equations is also suitable for solving Lyapunov equations in power systems, and the reduced-order RNN mosdels proposed in this article is effective in the example of power systems. Moreover, with the increase in the power system scale, the convergence and steady-state accuracy of ZNN model are almost unchanged, indicating the applicability of the RNN model to power systems of different scales.

It is worth mentioning that the integration between the electric power and natural gas systems has been steadily enhanced in recent decades. The incorporation of natural gas systems brings, in addition to a cleaner energy source, greater reliability and flexibility to the power system (Liu et al., 2021). Since the dynamic model of the electricity–gas coupled system can be expressed by differential-algebraic equations (Zhang, 2005; Yang, 2020), which means the dynamic model of the electricity–gas coupled system is the same as that of the power system, the aforementioned applicability analysis of the methods proposed in this article for large power systems are also applicable to large electricity–gas coupled systems.

An Efficient Method for Vectorization of RNN Model

Table 6 compares the time cost of the RNN model vectorization method proposed in this article and the traditional RNN model vectorization method. For the sake of convenience, the former is called method A and the latter is called method B. In order to better demonstrate the effect of the vectorization method of RNN model proposed in this article, four examples are added, as shown in **Table 6**. Four newly added examples are generated in the same way as Example 3 and are detailed in **Supplementary Material**, where ms means millisecond; scale means the order of A; and proportion refers to the time taken by method A divided by the time taken by method B.

It can be seen from **Table 6** that method A is significantly better than method B in terms of time cost, with the decrease in time cost between 48 and 98%. With the increase in the size of the examples, the proportion of time cost improvement generally increases. It should be pointed out that when the system sizes are 15, 35, and 97, the corresponding examples are the IEEE standard systems mentioned before, which indicates that the vectorization method proposed in this article is also effective in the example of power systems.

CONCLUSION

1) We propose an efficient method for vectorizing RNN models, which can achieve higher computational efficiency than the traditional method of vectorizing RNN based on the Kronecker product.
2) In order to reduce the solving scale of the RNN model, a reduced-order RNN model for solving the Lyapunov equation was proposed based on symmetry. At the same time, it is proved theoretically that the proposed model can maintain the same solution as that of the original model, and it is also proved that the proposed model is suitable for both the ZNN model and GNN model under linear or non-linear activation functions.
3) Several simulation examples are given to verify the effectiveness and superiority of the proposed method, while three standard examples of power systems are given to verify that the neural dynamics method is suitable for solving the Lyapunov equation of power systems.

Because the neural dynamics method has parallel distribution characteristics and hardware implementation convenience, its convergence and computation time are not sensitive to the system scale. Considering the current development level and trend of the very large-scale integration (VLSI) chip and the ultra large-scale integration (ULSI) chip, the wide application of the neural dynamics method in large-scale systems is expected.

In addition, the research on the RNN model used to solve the Lyapunov equation is mainly in the stage of theoretical improvement and exploration, and there are few reports about hardware products. The hardware product design will be the main content of the next stage.

DATA AVAILABILITY STATEMENT

The original contributions presented in the study are included in the article/**Supplementary Material**, further inquiries can be directed to the corresponding author.

AUTHOR CONTRIBUTIONS

All authors listed have made a substantial, direct, and intellectual contribution to the work and approved it for publication.

FUNDING

This work was supported in part by the Key-Area Research and Development Program of Guangdong Province (2019B111109001), the National Natural Science Foundation of China (51577071), and the Southern Power Grid Corporation's Science and Technology Project (Project No. 037700KK52190015 (GDKJXM20198313)).

SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/fenrg.2022.796325/full#supplementary-material

REFERENCES

Bartels, R. H., and Stewart, G. W. (1972). Solution of the Matrix Equation AX + XB = C [F4]. *Commun. ACM* 15 (9), 820–826. doi:10.1145/361573.361582

Chen, Z., and Zhou, J. (2012). *Introduction to Matrix Theory*. Beijing: Beihang University Press.

Chen, Q., Gong, C., Zhao, J., Wang, Y., and Zou, D. (2017). Application of Parallel Sparse System Direct Solver Library Super LU_MT in State Estimation. *Automation Electric Power Syst.* 41 (3), 83–88. doi:10.7500/AEPS20160607008

Cheng, K., and Chen, X. (2017). *Linear Algebra*. Chongqing: Chongqing University Press.

Chicca, E., Stefanini, F., Bartolozzi, C., and Indiveri, G. (2014). Neuromorphic Electronic Circuits for Building Autonomous Cognitive Systems. *Proc. IEEE* 102 (9), 1367–1388. doi:10.1109/JPROC.2014.2313954

Hafiz, F., Awal, M. A., Queiroz, A. R. d., and Husain, I. (2020). Real-Time Stochastic Optimization of Energy Storage Management Using Deep Learning-Based Forecasts for Residential PV Applications. *IEEE Trans. Ind. Applicat.* 56 (3), 2216–2226. doi:10.1109/TIA.2020.2968534

He, W., and Zhang, S. (2017). Control Design for Nonlinear Flexible Wings of a Robotic Aircraft. *IEEE Trans. Contr. Syst. Technol.* 25 (1), 351–357. doi:10.1109/TCST.2016.2536708

He, W., Ouyang, Y., and Hong, J. (2017). Vibration Control of a Flexible Robotic Manipulator in the Presence of Input Deadzone. *IEEE Trans. Ind. Inf.* 13 (1), 48–59. doi:10.1109/TII.2016.2608739

Horn, R. A., and Johnson, C. R. (1991). *Topics in Matrix Analysis*. Cambridge: Cambridge University Press.

Lan, X. (2017). *Research on Model Order Reduction Method and Predictive Control Algorithm of Grid Voltage Control System* (Beijing: North China Electric Power University). [dissertation/master's thesis].

Le, X., Chen, S., Li, F., Yan, Z., and Xi, J. (2019). Distributed Neurodynamic Optimization for Energy Internet Management. *IEEE Trans. Syst. Man. Cybern, Syst.* 49 (8), 1624–1633. doi:10.1109/TSMC.2019.2898551

Li, X., Yu, J., Li, S., Shao, Z., and Ni, L. (2019a). A Non-linear and Noise-Tolerant ZNN Model and its Application to Static and Time-Varying Matrix Square Root Finding. *Neural Process. Lett.* 50 (2), 1687–1703. doi:10.1007/s11063-018-9953-y

Li, Y., Zhang, H., Liang, X., and Huang, B. (2019b). Event-Triggered-Based Distributed Cooperative Energy Management for Multienergy Systems. *IEEE Trans. Ind. Inf.* 15 (4), 2008–2022. doi:10.1109/TII.2018.2862436

Li, Y., Gao, D. W., Gao, W., Zhang, H., and Zhou, J. (2020). Double-Mode Energy Management for Multi-Energy System via Distributed Dynamic Event-Triggered Newton-Raphson Algorithm. *IEEE Trans. Smart Grid* 11 (6), 5339–5356. doi:10.1109/TSG.2020.3005179

Lin, Y., and Simoncini, V. (2013). Minimal Residual Methods for Large Scale Lyapunov Equations. *Appl. Numer. Maths.* 72, 52–71. doi:10.1016/j.apnum.2013.04.004

Liu, J., Zhang, J., and Li, Q. (2020a). Upper and Lower Eigenvalue Summation Bounds of the Lyapunov Matrix Differential Equation and the Application in a Class Time-Varying Nonlinear System. *Int. J. Control.* 93 (5), 1115–1126. doi:10.1080/00207179.2018.1494389

Liu, Z., Chen, Y., Song, Y., Wang, M., and Gao, S. (2020b). Batched Computation of Continuation Power Flow for Large Scale Grids Based on GPU Parallel Processing. *Power Syst. Techn.* 44 (3), 1041–1046. doi:10.13335/j.1000-3673.pst.2019.2050

Liu, H., Shen, X., Guo, Q., and Sun, H. (2021). A Data-Driven Approach towards Fast Economic Dispatch in Electricity-Gas Coupled Systems Based on Artificial Neural Network. *Appl. Energ.* 286, 116480. doi:10.1016/j.apenergy.2021.116480

Raković, S. V., and Lazar, M. (2014). The Minkowski-Lyapunov Equation for Linear Dynamics: Theoretical Foundations. *Automatica* 50 (8), 2015–2024. doi:10.1016/j.automatica.2014.05.023

Shanmugam, L., and Joo, Y. H. (2021). Stability and Stabilization for T-S Fuzzy Large-Scale Interconnected Power System with Wind Farm via Sampled-Data Control. *IEEE Trans. Syst. Man. Cybern, Syst.* 51 (4), 2134–2144. doi:10.1109/TSMC.2020.2965577

Stykel, T. (2008). Low-rank Iterative Methods for Projected Generalized Lyapunov Equations. *Electron. Trans. Numer. Anal.* 30 (1), 187–202. doi:10.1080/14689360802423530

Xiao, L., and Liao, B. (2016). A Convergence-Accelerated Zhang Neural Network and its Solution Application to Lyapunov Equation. *Neurocomputing* 193, 213–218. doi:10.1016/j.neucom.2016.02.021

Xiao, L., Zhang, Y., Hu, Z., and Dai, J. (2019). Performance Benefits of Robust Nonlinear Zeroing Neural Network for Finding Accurate Solution of Lyapunov Equation in Presence of Various Noises. *IEEE Trans. Ind. Inf.* 15 (9), 5161–5171. doi:10.1109/TII.2019.2900659

Yang, J., Chen, C. S., Abreu-garcia, J. A. D., and Xu, Y. (1993). Model Reduction of Unstable Systems. *Int. J. Syst. Sci.* 24 (12), 2407–2414. doi:10.1080/00207729308949638

Yang, H. (2020). *Dynamic Modeling and Stability Studies of Integrated Energy System of Electric, Gas and Thermal on Multiple Time Scales* (Hunan: Changsha University of Science & Technology). [dissertation/master's thesis].

Yi, C., Chen, Y., and Lu, Z. (2011). Improved Gradient-Based Neural Networks for Online Solution of Lyapunov Matrix Equation. *Inf. Process. Lett.* 111 (16), 780–786. doi:10.1016/j.ipl.2011.05.010

Yi, C., Chen, Y., and Lan, X. (2013). Comparison on Neural Solvers for the Lyapunov Matrix Equation with Stationary & Nonstationary Coefficients. *Appl. Math. Model.* 37 (4), 2495–2502. doi:10.1016/j.apm.2012.06.022

Yunong Zhang, Z., and Danchi Jiang, M. T. J. (1995). *A Recurrent Neural Network for Solving Sylvester Equation with Time-Varying Coefficients*. New York: Academic Press.

Zhang, Y., Jiang, D., and Wang, J. (2002). A Recurrent Neural Network for Solving Sylvester Equation with Time-Varying Coeffificients. *IEEE Trans. Neural Netw.* 13 (5), 1053–1063. doi:10.1109/TNN.2002.1031938

Zhang, Y., Chen, K., Li, X., Yi, C., and Zhu, H. (2008). "Simulink Modeling and Comparison of Zhang Neural Networks and Gradient Neural Networks for Time-Varying Lyapunov Equation Solving," in Proceedings of IEEE International Conference on Natural Computation. Jinan. IEEE, 521–525. doi:10.1109/ICNC.2008.47

Zhang, Y. (2005). *Study on the Methods for Analyzing Combined Gas and Electricity Networks* (Beijing: China Electric Power Research Institute). [dissertation/master's thesis].

Zhao, H., Lan, X., Xue, N., and Wang, B. (2014). Excitation Prediction Control of Multi-machine Power Systems Using Balanced Reduced Model. *IET Generation, Transm. Distribution* 8 (6), 1075–1081. doi:10.1049/iet-gtd.2013.0609

Zhou, B., Duan, G.-R., and Li, Z.-Y. (2009). Gradient Based Iterative Algorithm for Solving Coupled Matrix Equations. *Syst. Control. Lett.* 58 (5), 327–333. doi:10.1016/j.sysconle.2008.12.004

Zhu, Z., Geng, G., and Jiang, Q. (2016). Power System Dynamic Model Reduction Based on Extended Krylov Subspace Method. *IEEE Trans. Power Syst.* 31 (6), 4483–4494. doi:10.1109/TPWRS.2015.2509481

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.