SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

November 2024, Vol. 67, Iss. 11, 212205:1–212205:16
https://doi.org/10.1007/s11432-023-4146-6

Interval analysis for neural networks with application to fault detection

Zhenhua WANG[1,2†], Youdao MA[1†], Song ZHU[3], Thach Ngoc DINH[4] & Yi SHEN[1,2*]

[1]*Department of Control Science and Engineering, Harbin Institute of Technology, Harbin* 150001*, China;*
[2]*National Key Laboratory of Complex System Control and Intelligent Agent Cooperation, Harbin* 150001*, China;*
[3]*School of Mathematics, China University of Mining and Technology, Xuzhou* 221116*, China;*
[4]*The CEDRIC-Lab, Conservatoire National des Arts et Métiers (CNAM), Paris* 75141*, France*

Received 22 March 2023/Revised 29 June 2023/Accepted 19 September 2023/Published online 22 October 2024

Abstract This paper investigates an interval analysis method for neural networks and applies it to fault detection for systems with unknown but bounded measurement noise. First, a novel interval analysis method is presented, which can compute the bounds of the output of a feedforward neural network subject to a bounded input. By applying the proposed interval analysis method to a network trained with fault-free system data, adaptive thresholds for fault detection are computed. Finally, one can acquire fault detection results via a fault detection strategy. The proposed method can achieve tight bounds of the network output and employ simple operations, which leads to accurate fault detection results and a low computational burden. A numerical simulation and an experiment on an AC servo motor are given to illustrate the effectiveness and superiority of the proposed method.

Keywords interval analysis, feedforward neural network, fault detection, adaptive thresholds

1 Introduction

Acquiring and analyzing fault information in time is necessary since a slight fault in a modern control system may cause performance degradation and even serious damage. Therefore, fault diagnosis techniques have been widely studied in the past decades due to their ability to improve the safety and reliability of control systems [1–6]. As an important part of fault diagnosis, fault detection aims to detect the occurrence of system faults. In general, fault detection methods can be divided into model-based methods, data-driven methods, and knowledge-based methods wherein the first type can compute residuals and their related thresholds based on the system model [7–9]. By comparing the signal and thresholds, a fault alarm will be given when a fault occurs [10, 11]. In [12], a sliding mode observer-based method was proposed for helicopter systems with system uncertainties to detect sensor, actuator, and component faults. Various types of adaptive threshold computation methods based on set-membership theory have been studied in recent years. For example, interval observers [13–15] via monotonic system theory and set-based methods via the propagation of geometric set [16–19]. Despite their promising performance on fault detection, the model-based methods require accurate mathematical models of the studied dynamic systems as a priori. For those large-scale operations, it is nontrivial to create such an accurate mechanism model.

Data-driven methods apply multivariate statistics or neural network-based approaches, offering attractive choices for fault diagnosis [20–22]. These methods have become one of the most fruitful areas in both research and applications over the last two decades. Typically the Hotelling's T^2 statistic and the squared prediction error can give overall information of measured data for subsequent fault detection. Principal component analysis and partial least squares (PLS) are the most commonly adopted methods to characterize fault-free statistics [23, 24]. Ref. [25] introduced a recursive modified PLS method for quality-relevant fault detection and complexity reduction. Compared with traditional PLS methods,

* Corresponding author (email: shen@hit.edu.cn)
† Zhenhua WANG and Youdao MA have the same contribution to this work.

the method in [25] can deal with time-varying factors. In recent years, neural networks have enjoyed tremendous progress in the control community due to their promising performance in approximating nonlinear functions. Some of the appealing structures include feedforward networks, convolutional neural networks, and recurrent neural networks [26–28]. Fault detection has employed neural networks in various ways. Some are adopted as classifiers to categorize fault-free and faulty situations and then the fault detection problem is converted to a classification problem [20, 29, 30]. The performance of this technique relies heavily on a balanced dataset containing a large number of fault-free and faulty training samples. However, gathering sufficient fault samples is difficult in practice since control systems are fault-free in most situations and it will be treated quickly when there is a fault. An unbalanced dataset will lead to performance degradation of fault detection, which may cause false alarms and missing alarms. To address this problem, in tremendous literature, neural networks are trained based on fault-free samples and adopted as an approximate of the fault-free system model to generate estimates [31]. Fault alarms are raised once the measurement data are inconsistent with the data predicted by the neural network.

Fault detection relying on neural network estimators becomes extremely challenging because the existence of system uncertainties such as measurement noise may cause inconsistency between the predicted data and the measurement data. To address these uncertainties, set-membership methods can be used to compute the bounds of all admissible fault-free measurement data. Once the measurement data exceeds these bounds, a fault alarm will be triggered. In this process, computing the bounds of the neural network output is essential. Various works have applied set-membership methods to state space systems. For example, [13] constructed an interval observer based on the monotone system theory to estimate the interval bounds of the residual and achieved accurate fault detection for multiagent systems. Nevertheless, only limited set-membership methods have been investigated on the computation of bounds of the neural network output [32, 33]. In recent years, some mathematical tools and analysis methods have been developed for determining the bounds of the network output, such as mixed-integer linear programming [34], semidefinite programming [35–37], and abstract interpretation [38, 39]. The semidefinite programming bounds the output for neural networks by solving linear matrix inequalities which are derived based on quadratic constraints and the S-procedure. Here the quadratic constraints of activation functions are acquired based on their sector-bounded and slope restricted nonlinearities. Meanwhile, those of zonotopic- or ellipsoidal-bounded conditions are acquired according to the geometric performance of these bounded sets. In [35], semidefinite programming was employed to compute the ellipsoidal bounded output for feedforward neural networks with commonly used activation functions. However, constructing and solving linear matrix inequalities in semidefinite programming introduces additional conservatism and high computational burden, making it difficult to compute tight bounds of the network output. The abstract interpretation divides the neural network into multiple layers so that the complex arithmetic can be transformed into a series of simple operations. Then the bounds of the network output are computed by representing and propagating the geometric set through these layers in order. Based on the abstract interpretation, [38] computed the bounds of the output for complex neural networks via zonotopes. However, during the propagation in abstract interpretation, the extensive use of intersection and union of geometrical sets makes the bounds of the network output conservative and greatly increases the computational burden. It is also worth pointing out that seldom research focuses on fault detection according to the interval analysis of neural networks. To the best of our knowledge, only [36] presented a performance bound analysis method for neural networks and applied it to fault detection for systems with unknown dynamics and Gaussian uncertainties. It adopted an ellipsoidal confidence set to represent system uncertainties, and then it constructed quadratic constraints for the input layer, the hidden layers, and the output layer respectively based on the geometric performance or nonlinearities. By online solving linear matrix inequalities established by these quadratic constraints, fault detection thresholds were computed. However, the construct of linear matrix inequalities introduces large conservatism, and its online solution is computationally and operationally complex, which limits the application of the method in [36].

Motivated by the discussion above, we aim to propose an interval analysis method for neural networks and apply it to fault detection. The contributions of the proposed method are highlighted by the following items:

1. This paper proposes an interval analysis method for feedforward neural networks. Based on simple interval operations, the proposed method can compute a tight bound of the output of a feedforward neural network subject to bounded inputs with low computational complexity and high operation efficiency.

2. The proposed interval analysis method is applied to fault detection. The presented fault detection

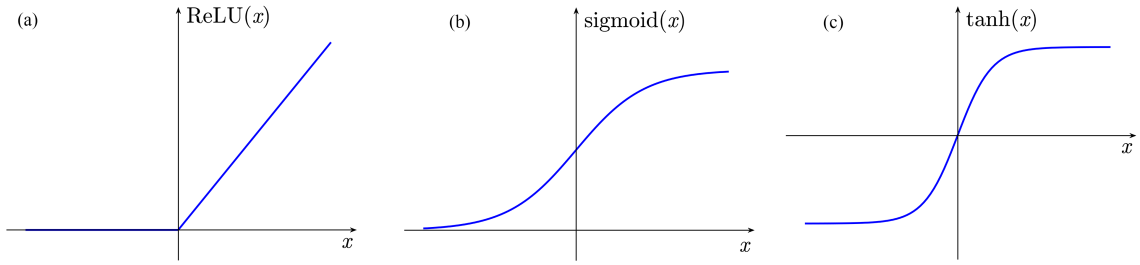

Figure 1 (Color online) Three commonly used activation functions: (a) ReLU; (b) sigmoid; (c) tanh.

method utilizes interval analysis layer by layer on a feedforward neural network trained with fault-free system data to obtain adaptive thresholds for fault detection. Compared with the method presented in [36], the proposed method can provide less conservative adaptive thresholds and hence yields a better fault detection performance.

It is worth pointing out that, although [36] has proposed a bound analysis method for neural networks and applied it to fault detection, it assumed system uncertainties satisfy Gaussian distribution and used an ellipsoidal confidence set to characterize the noise. Also, it constructed linear matrix inequalities based on quadratic constraints and solved them to compute fault detection thresholds, which introduces large conservatism and leads to high computational burden and time expense. In contrast, the proposed method adopts more general unknown but interval-bounded assumptions of uncertainties. Meanwhile, it uses simple interval arithmetic to propagate interval bounds layer by layer to obtain the bounds of the outputs. Therefore, the proposed method has less conservatism and higher computational efficiency.

The remainder of this paper is organized as follows: Section 2 introduces the main results of the interval analysis method for neural networks. In Section 3, a neural network-based fault detection method is illustrated. Simulation and experiment results are given in Section 4 to indicate the validity and superiority of the proposed methods. Finally, Section 5 draws a conclusion.

The notations used in this paper are standard. $\mathbb{R}$, $\mathbb{R}^n$ and $\mathbb{R}^{m\times n}$ denote all the real numbers, the n-dimensional and $m\times n$-dimensional Euclidean space, respectively. The symbols $\leqslant$ and $\geqslant$ should be understood elementwise. For a vector $x\in\mathbb{R}^n$, $x(i)$, $1\leqslant i\leqslant n$ represents its ith element. For a matrix A, $A^+=\max(A,0)$ and $A^-=A^+-A$. Given three sets $\mathcal{P}$, $\mathcal{Q}$ and $\mathcal{R}$, a function $\mathcal{F}:\mathcal{P}\times\mathcal{Q}\to\mathcal{R}$ takes $p\in\mathcal{P}$, $q\in\mathcal{Q}$ and returns a $r\in\mathcal{R}$.

2 Interval analysis for neural networks

As shown in Figure 1, the commonly used activation functions, such as restricted linear unit (ReLU), sigmoid, and tanh, are monotonically increasing. In this section, we propose an interval analysis method for feedforward neural networks with all its activation functions monotonic increasing. By using the proposed method, the interval bounds of the output of a feedforward neural network can be acquired if its input is interval bounded.

The following lemma is essential for interval analysis:

Lemma 1([40]). For a vector $x\in\mathbb{R}^n$ satisfying $\underline{x}\leqslant x\leqslant\bar{x}$ and a constant matrix $A\in\mathbb{R}^{m\times n}$, the following inequality holds

$$A^+\underline{x}-A^-\bar{x}\leqslant Ax\leqslant A^+\bar{x}-A^-\underline{x}. \tag{1}$$

A feedforward neural network $\mathcal{N}:\mathbb{R}^{n_1}\to\mathbb{R}^{n_l}$, as shown in Figure 2, is also known as a multilayer perceptron. Among them, the l-layer feedforward neural network is structured as

$$z_1=\zeta, \tag{2a}$$
$$z_\eta=\phi_\eta(W_\eta z_{\eta-1}+b_\eta),\quad \eta=2,\cdots,l-1, \tag{2b}$$
$$\hat{z}=W_l z_{l-1}+b_{\eta-1}, \tag{2c}$$

in which $\hat{z}\in\mathbb{R}^{n_l}$ is the output data of $\mathcal{N}$ corresponding to the input data $\zeta\in\mathbb{R}^{n_1}$, namely the network output and the network input, respectively. $W_\eta\in\mathbb{R}^{n_\eta\times n_{\eta-1}}$ represents the weight matrix in the ηth layer and $b_\eta\in\mathbb{R}^{n_\eta}$ is the bias vector. $z_\eta\in\mathbb{R}^{n_\eta}$ represents the intermediate data in the ηth layer. The

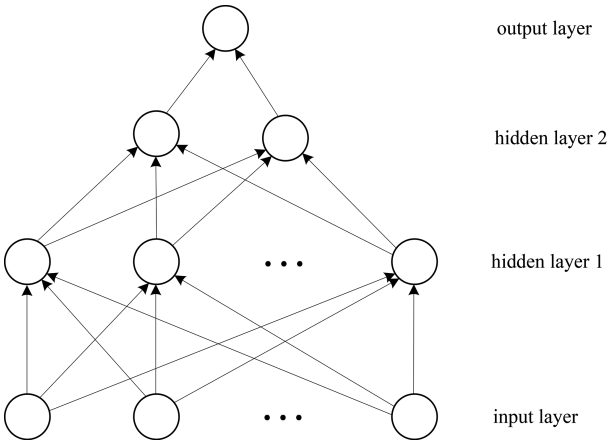

Figure 2 The structure of a 4-layer feedforward neural network.

nonlinear activation function of the ηth layer $\phi_\eta : \mathbb{R}^{n_\eta} \to \mathbb{R}^{n_\eta}$ is applied elementwise, which is formed as follows:

$$\phi_\eta(\cdot) = \begin{bmatrix} \psi_\eta(\cdot) \\ \vdots \\ \psi_\eta(\cdot)) \end{bmatrix}, \tag{3}$$

where $\psi_\eta : \mathbb{R} \to \mathbb{R}$ is the activation function of individual neuron. It should be noted that the proposed method is effective for all the monotonic increasing activation functions. Especially, for the commonly used ReLU, if its input is a vector $x \in \mathbb{R}^n$, it can be formulated as

$$\mathrm{ReLU}(x) = \begin{bmatrix} \max(0, x(1)) \\ \vdots \\ \max(0, x(n)) \end{bmatrix}. \tag{4}$$

From (2), the feedforward neural network $\mathcal{N}$ is organized as a sequence of layers. The output of one layer is the input of the next layer. The first layer computed by (2a) is the input layer, based on which $\mathcal{N}$ gets its network input. The last layer computed by (2c) is the output layer, based on which $\mathcal{N}$ gives its network output. The layers computed by (2b) between the input layer and the output layer are the hidden layers and the ηth hidden layer is the $\eta+1$th layer of $\mathcal{N}$. Each hidden layer performs a linear affine transformation (5a) followed by a nonlinear activation function (5b), as follows:

$$\xi_\eta = W_\eta z_{\eta-1} + b_\eta, \tag{5a}$$

$$z_\eta = \phi(\xi_\eta), \tag{5b}$$

where $\eta = 2, \cdots, l-1$ and ξ_η represents the intermediate data of the $\eta-1$th hidden layer before activation.

In this section, we aim to obtain the interval bounds of the network output $\hat{z}$ for the feedforward neural network (2) such that

$$\underline{z} \leqslant \hat{z} \leqslant \bar{z}, \tag{6}$$

when the network input ζ is interval bounded as follows:

$$\underline{\zeta} \leqslant \zeta \leqslant \bar{\zeta}. \tag{7}$$

Applying the relationship between the output and input of the neural network directly to acquire the bounds of its output is difficult due to its complex nonlinear characteristics. Notice that all the divided layers have linear and simple nonlinear performances, it is easy to analyze the relationship between the output and the input of an individual layer and propagate the bounds through the divided layers one by one to compute the bounds of the network output. Therefore, the following theorem based on layer-by-layer interval analysis is proposed.

Theorem 1. For an l-layer feedforward neural network $\mathcal{N}$ structured as (2) with all activation functions monotonic increasing, if its network input ζ satisfies the interval bounded condition (7), then its network output is interval bounded and satisfies (6). Moreover, the upper and lower interval bounds $\bar{z}$ and $\underline{z}$ are given by

$$\begin{cases} \bar{z} = W_l^+ \bar{z}_{l-1} - W_l^- \underline{z}_{l-1} + b_l, \\ \underline{z} = W_l^+ \underline{z}_{l-1} - W_l^- \bar{z}_{l-1} + b_l, \end{cases} \tag{8}$$

where $\bar{z}_{l-1}$ and $\underline{z}_{l-1}$ are computed iteratively by

$$\begin{cases} \bar{\xi}_\eta = W_\eta^+ \bar{z}_{\eta-1} + W_\eta^- \underline{z}_{\eta-1} + b_\eta, \\ \underline{\xi}_\eta = W_\eta^+ \underline{z}_{\eta-1} + W_\eta^- \bar{z}_{\eta-1} + b_\eta, \\ \bar{z}_\eta(i) = \psi(\bar{\xi}_\eta(i)), \\ \underline{z}_\eta(i) = \psi(\underline{\xi}_\eta(i)), \end{cases} \tag{9}$$

for $\eta = 2, \cdots, l-1$ with

$$\begin{cases} \bar{z}_1 = \bar{\zeta}, \\ \underline{z}_1 = \underline{\zeta}. \end{cases} \tag{10}$$

Proof. We analyze the input layer, the hidden layers, and the output layer respectively to verify this theorem.

First, we analyze the bounded condition of the input layer. Substituting (7) into (2a) yields that

$$\underline{\zeta} = \underline{z}_1 \leqslant z_1 \leqslant \bar{z}_1 = \bar{\zeta}, \tag{11}$$

which implies (10).

Then, we derive the propagation in the hidden layers. We denote the upper and lower bounds of the output of the $\eta-1$th layer as $\bar{z}_{\eta-1}$ and $\underline{z}_{\eta-1}$, respectively. The bounds of the input of the ηth layer are also $\bar{z}_{\eta-1}$ and $\underline{z}_{\eta-1}$. Therefore, for the linear affine transformation (5a) in the ηth layer, we can obtain the following inequalities according to Lemma 2:

$$\underline{\xi}_\eta \leqslant \xi_\eta \leqslant \bar{\xi}_\eta, \tag{12}$$

where

$$\begin{cases} \bar{\xi}_\eta = W_\eta^+ \bar{z}_{\eta-1} - W_\eta^- \underline{z}_{\eta-1} + b_\eta, \\ \underline{\xi}_\eta = W_\eta^+ \underline{z}_{\eta-1} - W_\eta^- \bar{z}_{\eta-1} + b_\eta. \end{cases} \tag{13}$$

Moreover, according to (3) and (5b), the following equation holds in the activation function operation:

$$z_\eta(i) = \psi(\xi_\eta(i)). \tag{14}$$

Since ψ is monotonic increasing, based on the effect of a monotonic increasing function, as shown in Figure 3, (14) leads to

$$\underline{z}_\eta(i) \leqslant z_\eta(i) \leqslant \bar{z}_\eta(i), \quad i = 1, \cdots, n_t, \tag{15}$$

where

$$\begin{cases} \bar{z}_\eta(i) = \psi(\bar{\xi}_\eta(i)), \\ \underline{z}_\eta(i) = \psi(\underline{\xi}_\eta(i)). \end{cases} \tag{16}$$

Note that the hidden layers of the feedforward neural network $\mathcal{N}$ are organized as a sequence of layers with the similar structure (5), and the output of a hidden layer is the input of the next hidden layer. Therefore, the bounds of z_{l-1} can be deduced iteratively based on (12) and (15) for $\eta = 2, \cdots, l-1$, i.e., (9), with the initial condition (10).

Finally, we analyze the output layer to compute the tight bounds of the network output. Based on (9), the input of the output layer is bounded by $\bar{z}_{l-1}$ and $\underline{z}_{l-1}$. Applying Lemma 2 to (2c), the bounds of the network output can be obtained by (8). The proof is complete.

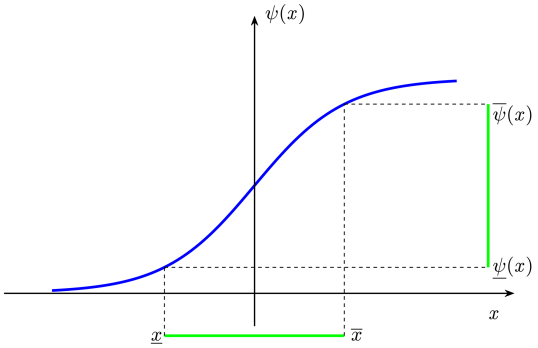

Figure 3 (Color online) The effect of a monotonic increasing activation function on an interval input.

Remark 1. The condition that all the activation functions in $\mathcal{N}$ are monotonic increasing is essential for the proposed interval analysis method. It is easily satisfied since almost all the commonly adopted activation functions are monotonic increasing, such as ReLU, sigmoid, tanh, and leaky ReLU. This also indicates that the proposed method can deal with interval analysis problems for a wide range of neural networks.

Based on the proposed interval analysis method for neural networks, one can provide the bounds of the network output $\hat{z}$ when the network input ζ is interval bounded. After acquiring interval analysis results, we apply them to the computation of adaptive thresholds for fault detection.

3 Neural network-based fault detection

An important stage of fault detection is to compute upper and lower adaptive thresholds to identify whether a fault has occurred. The section aims to provide an adaptive thresholds computation method based on a feedforward neural network. According to the acquired thresholds, we present a fault detection strategy to achieve fault detection results.

Consider the following discrete-time system:

$$\begin{cases} x_{k+1} = \mathcal{F}(x_k, u_k, f_{a,k}), \\ y_k = \mathcal{H}(x_k) + v_k + f_{s,k}, \end{cases} \tag{17}$$

where $x_k \in \mathbb{R}^{n_x}$, $u_k \in \mathbb{R}^{n_u}$, $y_k \in \mathbb{R}^{n_y}$, $v_k \in \mathbb{R}^{n_v}$, $f_{a,k} \in \mathbb{R}^{n_x}$ and $f_{s,k} \in \mathbb{R}^{n_y}$ are the state vector, the system input, the system output, the measurement noise, the additive actuator fault and the additive sensor fault, respectively. $\mathcal{F} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$ and $\mathcal{H} : \mathbb{R}^{n_x} \to \mathbb{R}^{n_y}$ are unknown continuous functions.

Assumption 1. The measurement noise v_k is unknown but bounded as follows:

$$\underline{v} \leqslant v_k \leqslant \bar{v}, \tag{18}$$

where $\bar{v} \in \mathbb{R}^{n_v}$ and $\underline{v} \in \mathbb{R}^{n_v}$ are two known vectors.

The system (17) in the fault-free situation is formulated as

$$\begin{cases} x_{k+1} = \mathcal{F}(x_k, u_k), \\ y_k = \mathcal{H}(x_k) + v_k, \end{cases} \tag{19}$$

where $\mathcal{F}(x_k, u_k)$ is computed by substituting $f_{a,k} = 0$ into $\mathcal{F}(x_k, u_k, f_{a,k})$.

Since the mapping $\mathcal{F}$ and $\mathcal{H}$ are unknown, the data-driven approach is adopted to analyze this dynamic system. We construct a feedforward neural network $\mathcal{N} : \mathbb{R}^{s_u n_u + s_y n_y} \to \mathbb{R}^{n_y}$ according to the time-series system input and output of (19). Its optimal weight matrices and bias vectors are computed by supervised training. During training, the vector $\tilde{u}_{k-1}$ which consists of the past s_u system inputs $u_{k-1}, \cdots, u_{k-s_u}$ and s_y system outputs $y_{k-1}, \cdots, y_{k-s_y}$ is adopted as the training network input, i.e.,

$$\tilde{u}_{k-1} = \begin{bmatrix} u_{k-1} & \cdots & u_{k-s_u} & y_{k-1} & \cdots & y_{k-s_y} \end{bmatrix}^{\mathrm{T}}. \tag{20}$$

Meanwhile, the current system output y_k is adopted as the corresponding labeled output.

Define the ideal output in the noise-free situation as

$$y_k^* = \mathcal{H}(x_k), \tag{21}$$

which follows that

$$y_k^* = y_k - v_k. \tag{22}$$

Note that the fault-free data of a control system are always easily acquired. We assume that $\mathcal{N}$ is trained based on adequate and sufficient data so that it can give the following equation:

$$y_k^* = \mathcal{N}(\tilde{u}_{k-1}^*) + \epsilon_k, \tag{23}$$

where

$$\tilde{u}_{k-1}^* = \begin{bmatrix} u_{k-1} & \cdots & u_{k-s_u} & y_{k-1}^* & \cdots & y_{k-s_y}^* \end{bmatrix}^{\mathrm{T}}, \tag{24}$$

and ϵ_k is bounded by

$$\underline{\epsilon} \leqslant \epsilon_k \leqslant \bar{\epsilon}, \tag{25}$$

in which $\underline{\epsilon}$, $\bar{\epsilon} \in \mathbb{R}^{n_y}$ are two known constants.

The purpose of this paper is to compute tight bounds of the output of $\mathcal{N}$ based on interval analysis and then to achieve fault detection for system (17). Therefore, how to refine the training process to obtain the network $\mathcal{N}$ is not considered in this paper.

Although the relationship between y_k^* and $\tilde{u}_{k-1}^*$ is acquired, we have no access to accurate noise-free ideal output in practice. However, it is easy to gather the actual measurement output. According to (22) and Assumption 1, the input of the feedforward neural network is interval bounded.

From the above discussion, we present the following theorem to compute the interval bounds of y_k in the fault-free situation, which can be adopted as adaptive thresholds for fault detection.

Theorem 2. For a system (19), given an $l-$layer feedforward neural network $\mathcal{N}$ trained by the fault-free data, then the adaptive thresholds $\bar{y}_k$ and $\underline{y}_k$ for fault detection are provided as follows:

$$\begin{cases} \bar{y}_k = \bar{z}_{k,l} + \bar{v} + \bar{\epsilon}, \\ \underline{y}_k = \underline{z}_{k,l} + \underline{v} + \underline{\epsilon}, \end{cases} \tag{26}$$

where the bounds $\bar{z}_{k,l}$ and $\underline{z}_{k,l}$ of the network output such that $\underline{z}_{k,l} \leqslant \mathcal{N}(\tilde{u}_{k-1}^*) \leqslant \bar{z}_{k,l}$ are obtained by

$$\begin{cases} \bar{z}_k = W_l^+ \bar{z}_{l-1} - W_l^- \underline{z}_{l-1} + b_l, \\ \underline{z}_k = W_l^+ \underline{z}_{l-1} - W_l^- \bar{z}_{l-1} + b_l, \end{cases} \tag{27}$$

and

$$\begin{cases} \bar{\xi}_\eta = W_\eta^+ \bar{z}_{\eta-1} + W_\eta^- \underline{z}_{\eta-1} + b_\eta, \\ \underline{\xi}_\eta = W_\eta^+ \underline{z}_{\eta-1} + W_\eta^- \bar{z}_{\eta-1} + b_\eta, \\ \bar{z}_\eta(i) = \psi(\bar{\xi}_\eta(i)), \\ \underline{z}_\eta(i) = \psi(\underline{\xi}_\eta(i)), \end{cases} \tag{28}$$

for $\eta = 2, \cdots, l-1$. And the bounds $\bar{\tilde{u}}_{k-1}$ and $\underline{\tilde{u}}_{k-1}$ of the network input such that $\underline{\tilde{u}}_{k-1} \leqslant \tilde{u}_{k-1}^* \leqslant \bar{\tilde{u}}_{k-1}$ are computed as follows:

$$\bar{\tilde{u}}_{k-1} = \begin{bmatrix} u_{k-1} \\ \vdots \\ u_{k-s_u} \\ y_{k-1} - \underline{v} \\ \vdots \\ y_{k-s_y} - \underline{v} \end{bmatrix}, \quad \underline{\tilde{u}}_{k-1} = \begin{bmatrix} u_{k-1} \\ \vdots \\ u_{k-s_u} \\ y_{k-1} - \bar{v} \\ \vdots \\ y_{k-s_y} - \bar{v} \end{bmatrix}. \tag{29}$$

Algorithm 1 The neural network-based fault detection method.

Input: $\bar{v}$, $\underline{v}$, s_u, s_y, large numbers of fault-free system input data and system output data;
Output: The fault detection result;
1: Construct the training data set: $\tilde{u}_{k-1}$ by (20) as network input and y_k as the labelled output;
2: Train the neural network $\mathcal{N}$ with the training data set to obtain (23);
3: Determine $\bar{\epsilon}$ and $\underline{\epsilon}$ by testing on fault-free training data;
4: **for** $k \geqslant \max(s_u, s_y)$ **do**
5: Compute $\bar{\tilde{u}}_{k-1}$ and $\underline{\tilde{u}}_{k-1}$ via (29);
6: Set $\bar{z}_1 = \bar{\tilde{u}}_{k-1}$, $\underline{z}_1 = \underline{\tilde{u}}_{k-1}$, and compute $\bar{z}_{k,l}$, $\underline{z}_{k,l}$ via (27) and (28);
7: Compute the adaptive thresholds $\bar{y}_k$ and $\underline{y}_k$ via (26);
8: **end for**
9: Achieve fault detection results via the fault detection strategy (38).

Proof. We first derive the bounds of the network input. Since the system inputs are all known fixed values, the upper and lower bound of u_j, $j = k - s_u, \cdots, k-1$ are both u_j. i.e.,

$$u_j \leqslant u_j \leqslant u_j. \tag{30}$$

Meanwhile, from the definition of the ideal output, the following equality holds for all $m = k-s_y, \cdots, k-1$:

$$y_m^* = y_m - v_m. \tag{31}$$

It follows from Assumption 1 and (31) that

$$y_m - \bar{v} \leqslant y_m^* \leqslant y_m - \underline{v}. \tag{32}$$

Substituting (30) and (32) into (24) yields

$$\underline{\tilde{u}}_{k-1} \leqslant \tilde{u}_{k-1}^* \leqslant \bar{\tilde{u}}_{k-1}, \tag{33}$$

where $\underline{\tilde{u}}_{k-1}$ and $\bar{\tilde{u}}_{k-1}$ are given by (29).

Then, following the result of Theorem 1, the interval bounds of the network output under (33) can be obtained by (27)–(29).

Finally, we compute the bounds $\bar{y}_k$ and $\underline{y}_k$ of the system output y_k in the fault-free situation, which can be used as adaptive thresholds. From (23), the bounds of y_k^* are provided by

$$\underline{y}_k^* \leqslant y_k^* \leqslant \bar{y}_k^*, \tag{34}$$

where

$$\begin{cases} \bar{y}_k^* = \bar{z}_{k,l} + \bar{\epsilon}, \\ \underline{y}_k^* = \underline{z}_{k,l} + \underline{\epsilon}. \end{cases} \tag{35}$$

Substituting (22) into (34) yields that

$$\underline{z}_{k,l} + \underline{\epsilon} \leqslant y_k - v_k \leqslant \bar{z}_{k,l} + \bar{\epsilon}, \tag{36}$$

which follows that

$$\underline{y}_k \leqslant y_k \leqslant \bar{y}_k, \tag{37}$$

where $\bar{y}_k$ and $\underline{y}_k$ are obtained by (26). The proof is complete.

It should be noted that the adaptive thresholds provided by Theorem 2 represent the bounds of the measurement output y_k in the fault-free situation. Therefore, if there is no fault occurs, the measurement output will always fall within the adaptive thresholds. Based on this, the following fault detection strategy is obtained:

$$\begin{cases} \underline{y}_k \leqslant y_k \leqslant \bar{y}_k \ \Rightarrow \text{Fault-free}, \\ \text{Otherwise} \quad \Rightarrow \text{Faulty}. \end{cases} \tag{38}$$

For better illustration, we give Algorithm 1 which summarizes the main steps of the proposed neural network-based fault detection method.

Remark 2. It is worth mentioning that the proposed method can be extended to systems with multiplicative faults. As pointed out in [7], the multiplicative fault of actuators can be described as

$$u_{f,k} = (I_{n_u} - \Lambda)u_k, \tag{39}$$

where $u_{f,k} \in \mathbb{R}^{n_u}$ is the actual output of actuators and $\Lambda \in \mathbb{R}^{n_u \times n_u}$ describes the effect of fault. Theoretically, a multiplicative fault can be converted into an additive fault, as follows:

$$u_{f,k} = u_k - \Lambda u_k = u_k + f_k, \tag{40}$$

where $f_k \in \mathbb{R}^{n_u}$ is an additive fault. Similarly, the multiplicative fault of sensors can be converted into additive faults. Therefore, our fault detection method can be extended to systems with multiplicative faults by converting multiplicative faults into additive faults.

Remark 3. In this paper, we develop an interval analysis method for feedforward neural networks and apply it to adaptive threshold computation for fault detection. Note that the feedforward neural networks are an open-loop system without closed-loop dynamics, usually stability analysis is not considered for the feedforward neural networks.

Remark 4. In the literature, the interval observer-based fault detection methods require a known and accurate physical model, and use interval observers for fault detection. Different from the interval observer-based method, the proposed method only requires a neural network trained with system data, and uses simple interval arithmetic to compute fault detection thresholds.

4 Numerical and experiment results

In this section, a numerical example and an experiment on an AC servo motor are provided to verify the effectiveness and superiority of the proposed method.

4.1 Numerical simulation example

To demonstrate the viability and validity of the proposed fault detection method, a numerical example slightly modified from [36] is utilized, which is formulated as follows:

$$\begin{cases} x_{k+1} = 0.8 \begin{bmatrix} \cos(0.6\pi) & -\sin(0.6\pi) \\ \sin(0.6\pi) & \cos(0.6\pi) \end{bmatrix}, \\ y_k = x_k + v_k + f_{s,k}, \end{cases}$$

where v_k is bounded by

$$v_k^T v_k \leqslant 0.01. \tag{41}$$

Since all the possible noise is bounded in this ellipsoid, it can be regarded as a 100%-confidence ellipsoid for the method presented in [36]. Meanwhile, (41) indicates that v_k satisfies the following unknown but bounded condition:

$$\begin{bmatrix} -0.1 \\ -0.1 \end{bmatrix} \leqslant v_k \leqslant \begin{bmatrix} 0.1 \\ 0.1 \end{bmatrix}. \tag{42}$$

Selecting $s_u = 0$ and $s_y = 2$, we can construct a four-layer feedforward neural network $\mathcal{N}$ (4 neurons in the input layer, 10 neurons in the first hidden layer, 2 neurons in the second layer and 1 neuron in the output layer) with the monotonic increasing activation function ReLU. By training $\mathcal{N}$ with a huge

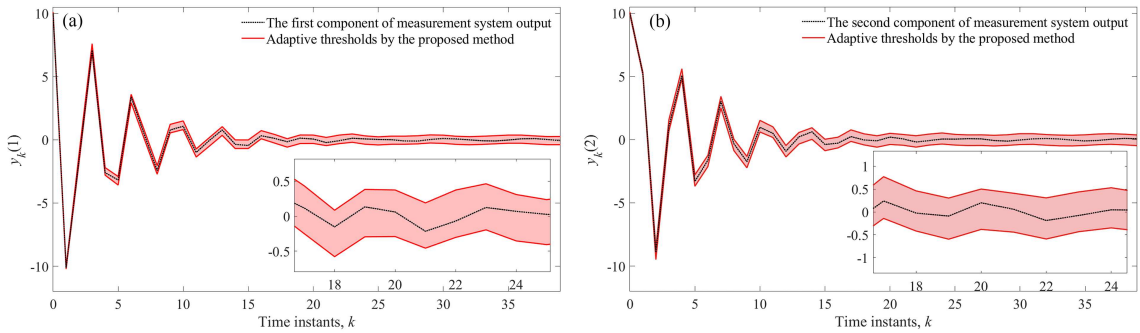

Figure 4 (Color online) System output and adaptive thresholds by the proposed method in the fault-free situation.

number of system inputs and measurement outputs, we obtain its optimal parameters as follows:

$$W_2 = \begin{bmatrix} -0.103 & -0.081 & -0.074 & 0.187 \\ -0.038 & 0.021 & 0.020 & 0.048 \\ -0.124 & -0.139 & -0.137 & 0.272 \\ 0.087 & -0.058 & -0.113 & -0.089 \\ -0.215 & 0.197 & 0.405 & 0.208 \\ 0.069 & -0.058 & -0.085 & -0.042 \\ 0.000 & 0.000 & -0.000 & 0.000 \\ 0.225 & -0.221 & -0.384 & -0.192 \\ -0.193 & -0.196 & -0.073 & 0.200 \\ -0.000 & -0.000 & -0.000 & -0.000 \end{bmatrix}, \quad b_2 = \begin{bmatrix} 2.331 \\ 1.324 \\ 3.986 \\ 0.159 \\ -0.040 \\ -0.225 \\ -1.913 \\ -0.050 \\ 3.570 \\ -1.110 \end{bmatrix}, \quad W_3 = \begin{bmatrix} 0.614 & 0.726 \\ 0.445 & 0.403 \\ 0.501 & 0.630 \\ -0.036 & 0.136 \\ 0.367 & -0.300 \\ 0.088 & -0.024 \\ -0.000 & 0.001 \\ -0.373 & 0.287 \\ 0.640 & 0.838 \\ 0.000 & 0.000 \end{bmatrix}^{\mathrm{T}},$$

$$b_3 = \begin{bmatrix} 0.686 \\ 0.647 \end{bmatrix}, \quad W_4 = \begin{bmatrix} -0.464 & 1.023 \\ -1.449 & 0.835 \end{bmatrix}, \quad b_4 = \begin{bmatrix} -5.354 \\ 3.089 \end{bmatrix}.$$

First of all, we test the proposed method in a fault-free situation. The fault-free system output and the proposed adaptive thresholds are plotted in Figure 4. As is shown, the adaptive thresholds provided by Theorem 2 always enclose the system output, which follows that there is no fault occurs according to the given fault detection strategy.

A sensor fault is adopted to show that the proposed method is effective in detecting faults, which is formulated as

$$f_{s,k} = \begin{cases} 0 & k < 20, \\ \begin{bmatrix} 0.4^T & 0.4^T \end{bmatrix}^T & k \geqslant 20. \end{cases}$$

The simulation results are depicted in Figure 5, which gives the measurement output and the adaptive thresholds provided by Theorem 2. Before the sensor fault occurs, the system output always falls within the adaptive thresholds, whereas y_k goes outside the adaptive thresholds quickly after that. This indicates that a fault has occurred.

Moreover, a comparison between the adaptive thresholds obtained from the proposed method and the traditional semidefinite programming method presented in [36] is given to demonstrate the superiority of the proposed method. This comparison study is implemented in the environment of MATLAB R2021a under AMD Ryzen 7 5800H with Radeon Graphics @3.20GHz. The YALMIP toolbox is used to solve the linear matrix inequalities involved in the method presented in [36]. The thresholds of the method presented in [36] are also plotted in Figure 5. For an easier and more obvious comparison, we transform the ellipsoidal bounds provided by the traditional method into the interval bounds. Notice that the proposed method provides less conservative thresholds, which quantifies a promising fault detection performance. The thresholds provided by the method presented in [36] can hardly detect the occurrence of the sensor fault since y_k is always contained within them.

For a more obvious demonstration, Figure 6 depicts the fault alarm signal σ_k of these two methods. $\sigma_k = 0$ indicates that $\underline{y}_k \leqslant y_k \leqslant \bar{y}_k$ holds, which means there is no fault occurs, whereas $\sigma_k = 1$ means

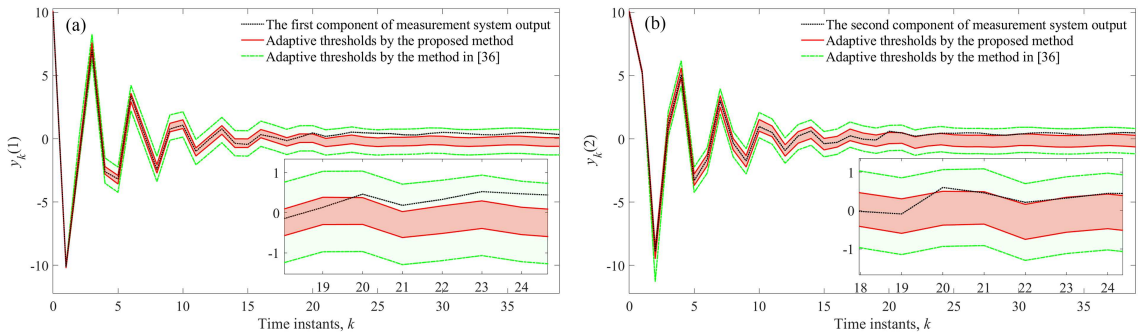

Figure 5 (Color online) System output components and adaptive thresholds by the proposed method and the method presented in [36] when a sensor fault occurs.

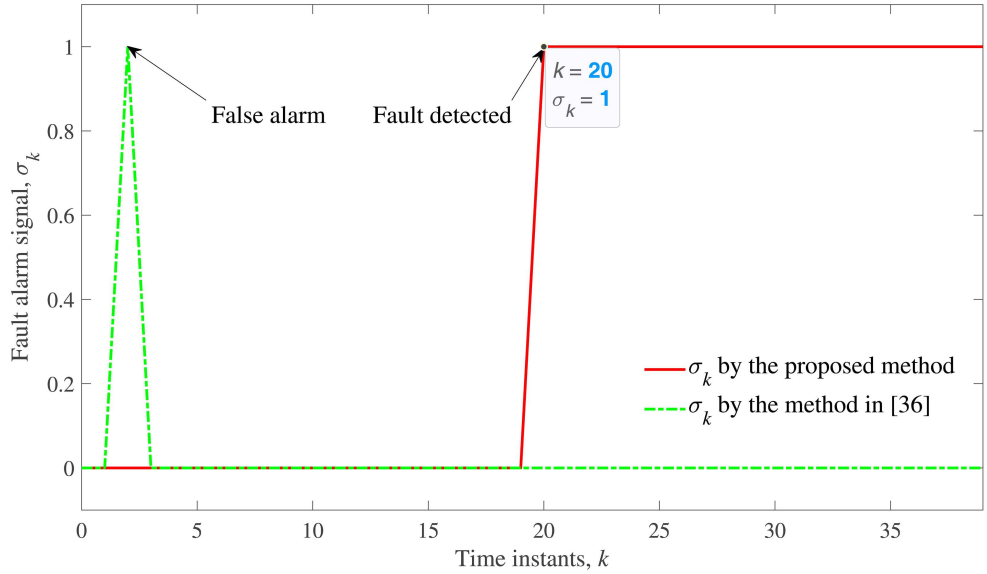

Figure 6 (Color online) Fault alarm signals by the proposed method and the method presented in [36].

Table 1 Online computational time (s) comparison results of the proposed method and the method presented in [36]

	1	2	3	4	5	6
The proposed method	0.0048	0.0037	0.0037	0.0037	0.0037	0.0037
The method in [36]	751.8091	745.3214	751.9798	762.5996	755.5893	758.0649

there exists a fault. From Figure 6, it is obvious that σ_k of the proposed method changes from 0 to 1 when $k = 20$. Meanwhile, that of the method presented in [36] reveals that $\sigma_k = 0$ from $k = 0$ to $k = 40$ except $k = 3$. This means the traditional method cannot effectively detect the sensor fault for the simulated system and may even yield a false alarm. These results show that the proposed method can detect the fault more accurately compared with the traditional method.

The online computational time of these two approaches is equally compared in our simulation. We run the program of these two methods 6 times and record the computational time for 40 time instants in Table 1. As is shown in Table 1, the merit of the proposed method includes that its online computational time is much shorter than the method presented in [36]. Roughly speaking, the main reason is that the proposed method avoids solving linear matrix inequalities. High calculation efficiency makes the proposed method more practical in real-world systems.

4.2 AC servo motor experiment

We construct an AC servo motor experimental platform in Figure 7. The experimental platform consists of a host computer, an ARM microprocessor, a CANbus module, an AC servo motor, and its servo system. The framework of the constructed experimental platform is shown in Figure 8. In the platform, the host

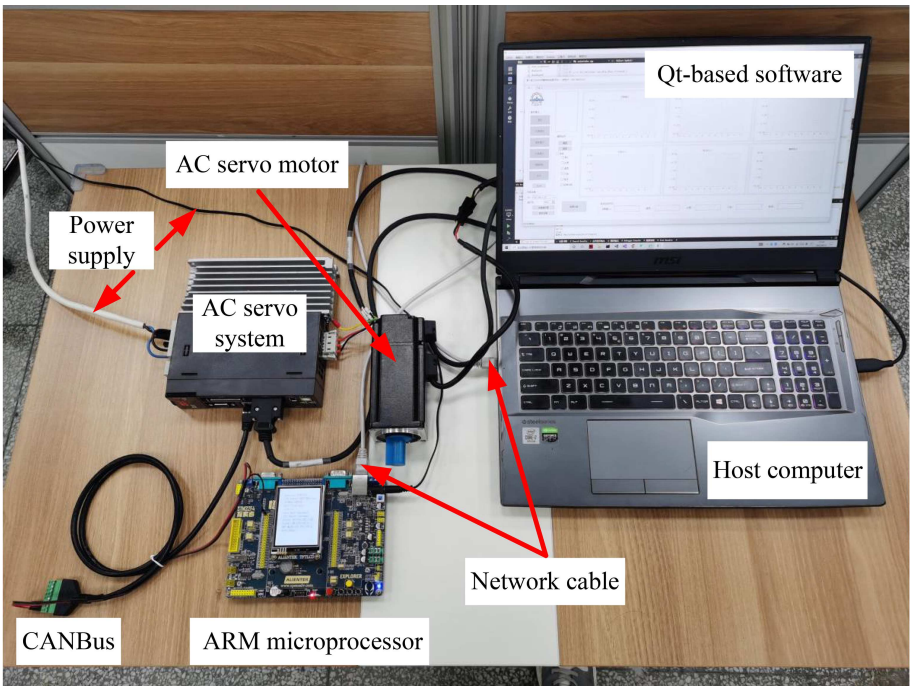

Figure 7 (Color online) AC servo motor experimental platform.

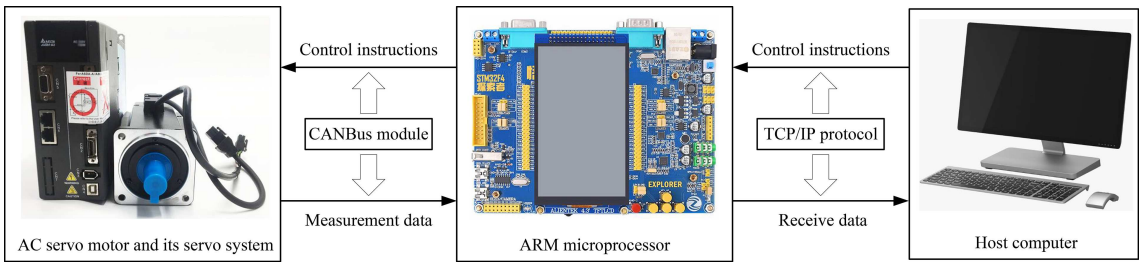

Figure 8 (Color online) The framework of the AC servo motor experimental platform.

computer equipped with the AMD Ryzen 7 5800H with Radeon Graphics @3.20GHz processor can receive data from the ARM microprocessor through the TCP/IP protocol and send the control instructions to the ARM microprocessor. The ARM microprocessor is a development board with an STM32F407ZGT6 as its core and the UCOSII as its operating system. A Delta series servo motor system consisting of an ECMA-C10604RS permanent magnet synchronous motor and an ASDA-A2-0421-M servo drive is utilized. This servo motor system adopts a widely used three-loop closed-loop control method and offers various operating modes. Here we use the torque mode based on current loop control, as this mode can effectively simulate the occurrence of actuator or sensor faults in the servo motor. The CANbus module is used to transmit the real-time measurement data such as torque, position and velocity of the AC servo motor between the ARM microprocessor and the AC servo system. In addition, the AC servo system will respond to control instructions and drive the servo motor to implement corresponding actions.

The accurate mechanism modeling of an AC servo motor can not be realized. To handle this issue, a data-driven method is used to build the mathematical model of the AC servo motor. According to a large number of control torque data u_k ($\mathrm{N}\cdot\mathrm{m}$) and output velocity data y_k (m/s) data measured from a fault-free AC servo system, we construct a three-layer feedforward neural network (2 neurons in the input layer, 6 neurons in the hidden layer and 1 neuron in the output layer) with the monotonic increasing activation function ReLU, i.e.,

$$\begin{aligned} z_1 &= \left[u_{k-1}^T \ y_{k-1}^T\right]^{\mathrm{T}}, \\ z_2 &= \mathrm{ReLU}(W_2 z^1 + b_2), \\ y_k &= \hat{z}_k = W_3 z_2 + b_3, \end{aligned}$$

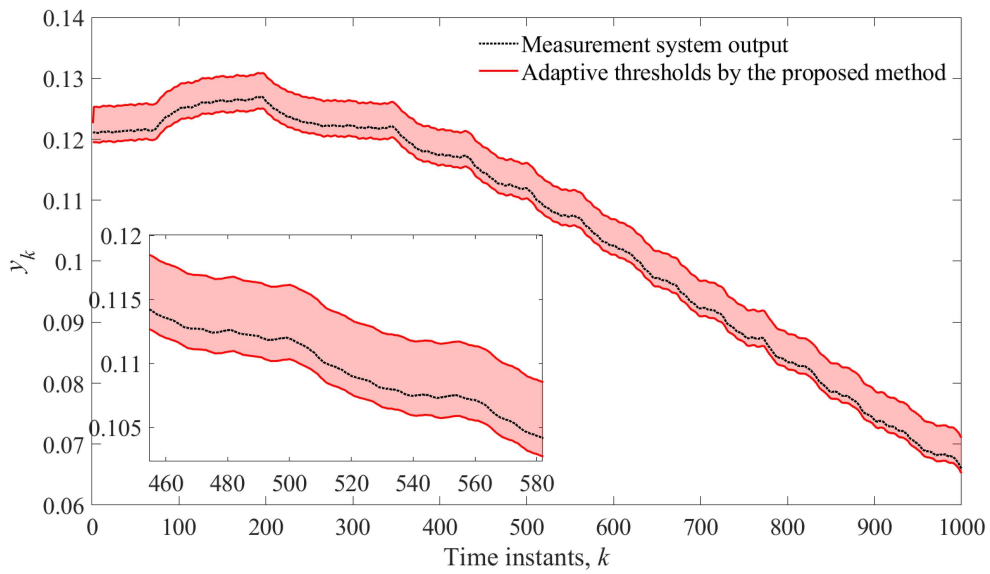

Figure 9 (Color online) System output and adaptive thresholds by the proposed method in the fault-free situation.

in which optimal parameters $W_2 \in \mathbb{R}^{6\times 2}$, $W_3 \in \mathbb{R}^{1\times 6}$, $b_2 \in \mathbb{R}^6$, $b_3 \in \mathbb{R}$ obtained based on the supervised training are as follows:

$$W_2 = \begin{bmatrix} 96.1302 & 16.1437 \\ 71.7327 & 19.2563 \\ 28.8103 & 21.8346 \\ 0.0731 & 0.5118 \\ 58.2756 & 20.1121 \\ -2.1957 & -29.6930 \end{bmatrix}, \quad b_2 = \begin{bmatrix} -14.4419 \\ -9.6889 \\ -3.0448 \\ -0.5366 \\ -0.7732 \\ -1.6730 \end{bmatrix},$$

$$W_3 = \begin{bmatrix} -0.0166 & -0.0007 & 0.0009 & 2.3148 & -0.0164 & -0.1210 \end{bmatrix}, \quad b_3 = 1.0133.$$

In this experiment, the control torque is set as

$$u_k = 0.05 + 0.01\sin(0.5kT_s - 3.75),$$

where $T_s = 0.005$s is the sampling time. Besides, the measurement noise v_k (m/s) of the AC servo motor satisfies the following unknown but bounded condition:

$$-0.0015 \leqslant v_k \leqslant 0.0015.$$

We first verify that the proposed method is effective in a fault-free situation. The fault-free system output and the proposed adaptive thresholds are plotted in Figure 9. As is shown, the system output keeps within the adaptive thresholds provided by Theorem 2 during this experiment. Based on the proposed fault detection strategy, it is concluded that there is no fault occurs.

Then we demonstrate that the proposed method is an effective tool for detecting the occurrence of faults. The first fault that we consider is an actuator fault which is formulated as

$$f_{a,k} = \begin{cases} 0 & k < 500, \\ -0.01\ \mathrm{N\cdot m} & k \geqslant 500. \end{cases}$$

The experiment results are depicted in Figures 10 and 11. Figure 10 gives the measurement output and the adaptive thresholds provided by Theorem 2 and Figure 11 depicts the fault alarm signal σ_k of the proposed method. As is shown, the system output always falls within the adaptive thresholds when no actuator fault occurs. However, after the occurrence of the actuator fault, y_k goes outside the adaptive

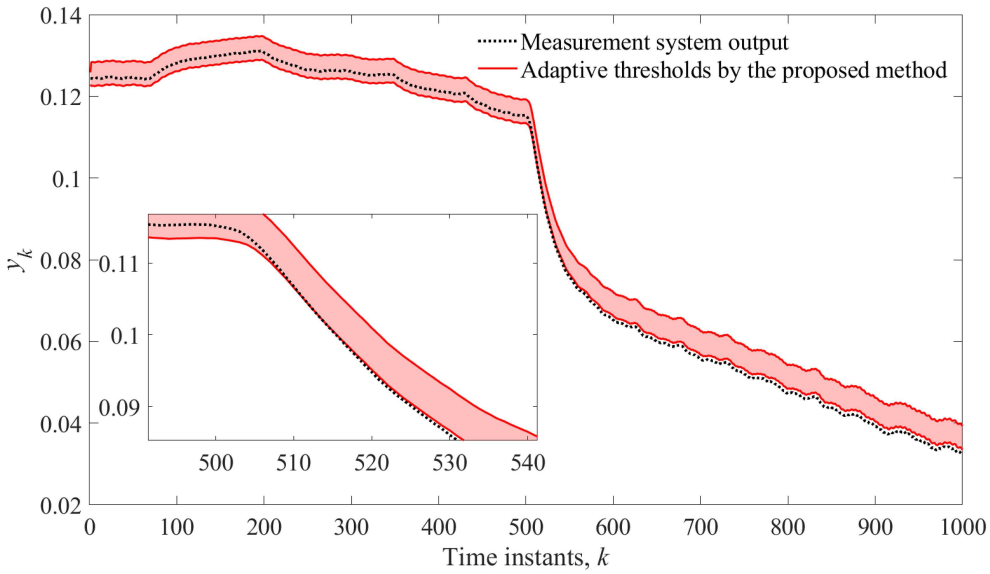

Figure 10 (Color online) System output and adaptive thresholds by the proposed method when an actuator fault occurs.

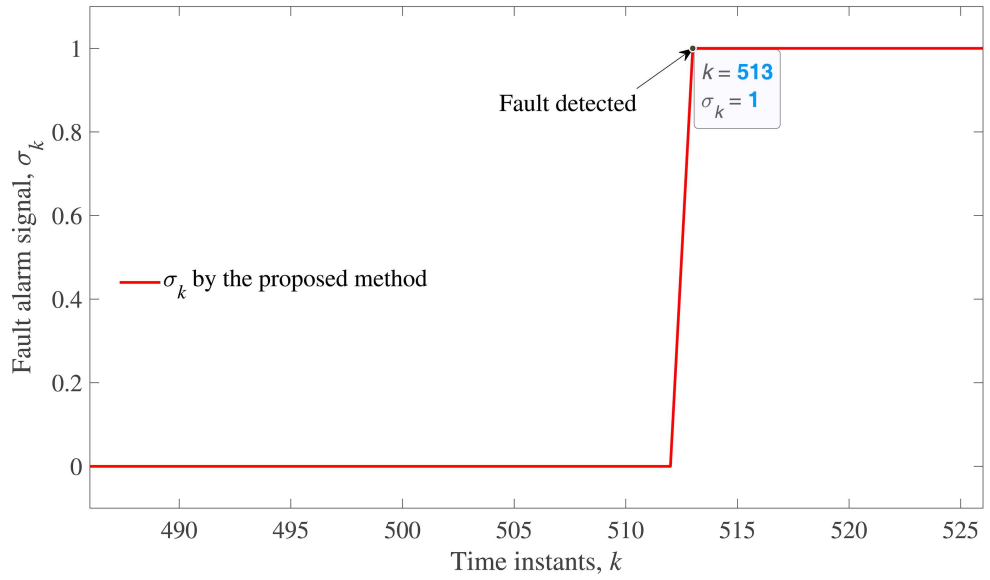

Figure 11 (Color online) Fault alarm signals by the proposed method.

Table 2 Online computational time of the proposed method

	1	2	3	4	5	6
Time (s)	0.001972	0.001184	0.001099	0.001080	0.001273	0.001196

thresholds quickly at $k = 513$. Based on the proposed fault detection strategy, we can conclude that the system is faulty. The experiment results show that accurate fault detection results can be achieved by using the proposed method.

On the other hand, we also record the online computational time of the proposed approach. We run the program of this method 6 times and record the computational time for the period from $k = 485$ to $k = 525$ in Table 2. Also evident is the fact that in this experiment, the sampling time is 0.005 s. From the computational time recorded in Table 2, the average computational time for one time instant is far less than the sampling time, which reveals its effectiveness for online fault detection.

5 Conclusion

In this paper, a novel interval analysis method is presented for feedforward neural networks. Given a feedforward neural network with interval bounded inputs, the proposed interval analysis method can compute the interval bounds of its output by propagating the bounds through its layers. Then, using the network trained with the fault-free system data, a neural network-based fault detection method is presented to compute adaptive thresholds for fault detection. Fault detection results are acquired by integrating the adaptive thresholds and a fault detection strategy. A numerical simulation and an experiment on an AC servo motor are given to demonstrate that the proposed method can achieve tight adaptive thresholds and low time costs. Reducing the conservatism of the proposed method and extending the proposed method to other types of neural networks such as convolutional neural networks and recurrent neural networks will be our future work.

Acknowledgements This work was supported by National Natural Science Foundation of China (Grant No. 62373125) and the Grant from Si-Yuan Collaborative Innovation Alliance of Artificial Intelligence Science and Technology (Grant No. HTKJ2023SY-502003).

References

1 Zhou D, Qin L, He X, et al. Distributed sensor fault diagnosis for a formation system with unknown constant time delays. Sci China Inf Sci, 2018, 61: 112205
2 Wang Z, Shi P, Lim C-C. H_-/H_∞ fault detection observer in finite frequency domain for linear parameter-varying descriptor systems. Automatica, 2017, 86: 38–45
3 Zhu J W, Gu C Y, Ding S X, et al. A new observer-based cooperative fault-tolerant tracking control method with application to networked multiaxis motion control system. IEEE Trans Ind Electron, 2021, 68: 7422–7432
4 Liu Z, Han Z, Zhao Z, et al. Modeling and adaptive control for a spatial flexible spacecraft with unknown actuator failures. Sci China Inf Sci, 2021, 64: 152208
5 Mao Z, Yan X G, Jiang B, et al. Adaptive fault-tolerant sliding-mode control for high-speed trains with actuator faults and uncertainties. IEEE Trans Intell Transp Syst, 2020, 21: 2449–2460
6 Hu C, Luo J, Kong X, et al. Novel fault subspace extraction methods for the reconstruction-based fault diagnosis. J Process Control, 2021, 105: 129–140
7 Wang Z, Shen Y. Model-Based Fault Diagnosis: Methods for State-Space Systems. Singapore: Springer Nature, 2022
8 Zhou D, Zhao Y, Wang Z, et al. Review on diagnosis techniques for intermittent faults in dynamic systems. IEEE Trans Ind Electron, 2020, 67: 2337–2347
9 Zhu X, Liu Y, Fang J, et al. Fault detection for a class of linear systems with integral measurements. Sci China Inf Sci, 2021, 64: 132207
10 Zhu F, Tang Y, Wang Z. Interval-observer-based fault detection and isolation design for t-s fuzzy system based on zonotope analysis. IEEE Trans Fuzzy Syst, 2021, 30: 945–955
11 Guo S, Tang M, Huang D, et al. State estimation and finite-frequency fault detection for interconnected switched cyber-physical systems. Sci China Inf Sci, 2023, 66: 192204
12 Raghappriya M, Kanthalakshmi S. Sliding mode observer-based fault detection for helicopter system. J Control Decision, 2023, 10: 465–475
13 Zhang Z H, Yang G H. Distributed fault detection and isolation for multiagent systems: an interval observer approach. IEEE Trans Syst Man Cybern Syst, 2020, 50: 2220–2230
14 Garbouj Y, Dinh T N, Raissi T, et al. Optimal interval observer for switched takagi-sugeno systems: an application to interval fault estimation. IEEE Trans Fuzzy Syst, 2022, 29: 2296–2309
15 Wang Z, Lim C C, Shen Y. Interval observer design for uncertain discrete-time linear systems. Syst Control Lett, 2018, 116: 41–46
16 Wang J, Shi Y, Zhou M, et al. Active fault detection based on set-membership approach for uncertain discrete-time systems. Intl J Robust NOnlinear, 2020, 30: 5322–5340
17 Wang Z, Zhang Y, Shen M, et al. Ellipsoidal set-membership filtering for discrete-time linear time-varying systems. IEEE Trans Autom Control, 2023, 68: 5764–5774
18 Fei Z, Yang L, Sun X M, et al. Zonotopic set-membership state estimation for switched systems with restricted switching. IEEE Trans Automat Contr, 2022, 67: 6127–6134
19 Wang Z, Dinh T N, Zhang Q, et al. Fast interval estimation for discrete-time linear systems: an L1 optimization method. Automatica, 2022, 137: 110029
20 Yu W, Zhao C. Broad convolutional neural network based industrial process fault diagnosis with incremental learning capability. IEEE Trans Ind Electron, 2020, 67: 5081–5091
21 Chen H, Jiang B, Ding S X, et al. Data-driven fault diagnosis for traction systems in high-speed trains: a survey, challenges, and perspectives. IEEE Trans Intell Transp Syst, 2022, 23: 1700–1716
22 Li H, Wu Y, Chen M, et al. Adaptive multigradient recursive reinforcement learning event-triggered tracking control for multiagent systems. IEEE Trans Neural Netw Learn Syst, 2023, 34: 144–156
23 Tao Y, Shi H, Song B, et al. A novel dynamic weight principal component analysis method and hierarchical monitoring strategy for process fault detection and diagnosis. IEEE Trans Ind Electron, 2020, 67: 7994–8004
24 Si Y, Wang Y, Zhou D. Key-performance-indicator-related process monitoring based on improved kernel partial least squares. IEEE Trans Ind Electron, 2021, 68: 2626–2636
25 Feng X, Kong X, Du B, et al. Adaptive lii-rmpls based data-driven process monitoring scheme for quality-relevant fault detection. J Control Decision, 2022, 9: 477–488
26 Zhao J, Guo W, Zhang Z, et al. A coupled convolutional neural network for small and densely clustered ship detection in SAR images. Sci China Inf Sci, 2019, 62: 042301
27 Pang N, Zhao X, Wang W, et al. Few-shot text classification by leveraging bi-directional attention and cross-class knowledge. Sci China Inf Sci, 2021, 64: 130103

28 Pei H, Si X S, Hu C, et al. Bayesian deep-learning-based prognostic model for equipment without label data related to lifetime. IEEE Trans Syst Man Cybern Syst, 2023, 53: 504–517
29 Chen Z, Gryllias K, Li W. Intelligent fault diagnosis for rotary machinery using transferable convolutional neural network. IEEE Trans Ind Inf, 2020, 16: 339–349
30 Shao H, Xia M, Han G, et al. Intelligent fault diagnosis of rotor-bearing system under varying working conditions with modified transfer convolutional neural network and thermal images. IEEE Trans Ind Inf, 2021, 17: 3488–3496
31 Wen L, Li X, Gao L, et al. A new convolutional neural network-based data-driven fault diagnosis method. IEEE Trans Ind Electron, 2018, 65: 5990–5998
32 Zhu S, Gao Y, Hou Y, et al. Reachable set estimation for memristive complex-valued neural networks with disturbances. IEEE Trans Neural Netw Learn Syst, 2023, 34: 11029–11034
33 Venzke A, Chatzivasileiadis S. Verification of neural network behaviour: formal guarantees for power system applications. IEEE Trans Smart Grid, 2021, 12: 383–397
34 Kantchelian A, Tygar J D, Joseph A. Evasion and hardening of tree ensemble classifiers. In: Proceedings of International Conference on Machine Learning, 2016. 2387–2396
35 Fazlyab M, Morari M, Pappas G J. Probabilistic verification and reachability analysis of neural networks via semidefinite programming. In: Proceedings of 2019 IEEE 58th Conference on Decision and Control (CDC), 2019. 2726–2731
36 Hashemi N, Fazlyab M, Ruths J. Performance bounds for neural network estimators: applications in fault detection. In: Proceedings of 2021 American Control Conference (ACC), 2021. 3260–3266
37 Fazlyab M, Morari M, Pappas G J. Safety verification and robustness analysis of neural networks via quadratic constraints and semidefinite programming. IEEE Trans Automat Contr, 2022, 67: 1–15
38 Gehr T, Mirman M, Drachsler-Cohen D, et al. AI2: safety and robustness certification of neural networks with abstract interpretation. In: Proceedings of 2018 IEEE Symposium on Security and Privacy (SP), 2018. 3–18
39 Mirman M, Gehr T, Vechev M. Differentiable abstract interpretation for provably robust neural networks. In: Proceedings of the 35th International Conference on Machine Learning, 2018. 3578–3586
40 Efimov D, Raïssi T, Chebotarev S, et al. Interval state observer for nonlinear time varying systems. Automatica, 2013, 49: 200–205