Received December 3, 2019, accepted December 26, 2019, date of publication January 8, 2020, date of current version January 16, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2964835

Modeling Uncertain Dynamic Plants With Interval Neural Networks by Bounded-Error Data

SHOUPING GUAN, ZIHE ZHANG, AND ZHOUYING CUI
College of Information Science and Engineering, Northeastern University, Shenyang 110819, China

Corresponding author: Zihe Zhang (317802385@qq.com)

This work was supported by the National Natural Science Foundation of China under Grant 61573087.

ABSTRACT This paper presents a novel approach to building an interval dynamic model for an industrial plant with uncertainty by an interval neural network (INN). A new type of randomized learner model, named interval random vector functional-link network (IRVFLN), is proposed to take advantages of the inherent RVFLN in rapid modeling. The IRVFLN model is equipped with interval hidden input weights (and biases), which are randomly assigned from certain distribution/range and remain fixed, and the interval output weights can be evaluated by solving a couple of least squares problems. The comparative numerical experiments have verified the good potential of the proposed IRVFLN with the interval learner models produced by the error back-propagation algorithm. In the following modeling application, some measures for building IRVFLN with unknown but bounded (UBB) errors requirements are discussed in depth, in order to modeling an uncertain dynamic plant with IRVFLN by bounded-error data in either known or unknown error bounds. Finally, as a case study, the IRVFLN is applied to modeling a chemical interval dynamic plant with recycling, where the simulation results and generalization ability analysis demonstrate that the proposed method is suitable and effective.

INDEX TERMS Interval neural network (INN), random vector functional-link network (RVFLN), unknown but bounded (UBB) errors, uncertain dynamic system modeling.

I. INTRODUCTION

In some real-world industrial domains, such as chemical plants, robotic manipulators, nuclear reactors, electrical machines and large power networks, uncertainties unavoidably exist due to poor plant knowledge, nonlinearities, unknown internal or external noises, environmental influence, time-varying parameters, changing operating conditions, etc. Therefore, to build a very accurate model that can exactly describe the physical plant is of great challenge [1]–[3]. Technically, conventional models (like neural networks with point-valued parameters and inputs) cannot favourably deal with uncertainty issues mentioned above as the ordinary values (rather than ranges/intervals) may not reflect the indeterminacy of the practical industrial system. Instead, an interval model, which includes parameters varying in prescribed ranges that are perturbed by uncertainties, has a good potential to model the complex dynamics of a plant with their inherent flexibilities contributed by the interval components (the parameters, inputs, and outputs) [4]–[7]. Compared with the other uncertain models, such as the fuzzy set model [8], which requires the suitable membership functions and the expert rules, the stochastic model [9], which needs the accurate probability distribution of uncertainty, and etc., the interval model possesses the properties of the simplest description of uncertainty and the least requirements of a priori knowledge.

The system description based on the theory of unknown but bounded (UBB) errors is an effective way to establish an interval system model [10], [11]. At present, modelling an interval nonlinear system using UBB error theory is based on the assumption that the parameter estimation can be regarded as a set inversion problem, and, with the help of the interval analysis-based SIVIA (set inverter via interval analysis) algorithm for the obtainment of an approximate but reliable set of identification parameters [12], [13]. This kind of technique optimizes the parameters of the system under the condition that the models structure is known in advance. However, with increasing complexity in the process being modelled, it can be very difficult to determine a priori models structure

The associate editor coordinating the review of this manuscript and approving it for publication was Ming Xu.

through the process mechanism analysis. Furthermore, when the SIVIA algorithm is used to estimate the parameters of an interval system model, it is assumed that the parameters are compatible with the errors, which implies that the SIVIA algorithm may return an empty set if an incompatibility is found [14].

To address the problems that arise when modelling an uncertain dynamic plant based on UBB theory, we focus on a new interval modelling method by employing interval neural networks (INNs) in system design, which can effectively avoid problems such as the system models structural demands and the error compatibility requirement. Although several researchers have studied the development of prediction intervals (PIs) for NN forecasts [15], [16], the existing research on PIs is to use the point-valued neural network to realize the interval-valued prediction, which essentially belongs to statistical regression. Besides some limitations in understanding and difficulties in the training process of NN-based PI [16], it is very hard, or impossible to quantify the uncertainties in the system inputs and parameters, and it is difficult to provide dynamic long-range interval prediction, i.e., the previous outputs are fed back from the network itself to reflect the current point (interval) values to the future ones, while these are the necessary requirements of an uncertain dynamic process modeling. Therefore, as the name implies, NN-based PI models are suitable for forecasting, not modeling. As for the INN models, these abovementioned requirements are the fundamental characteristics, so to some extent, we can say that the INNs are the promotion of NN-based PIs. In fact, the research on the design of interval system controller based on INN models has achieved preliminary results [17]–[19].

Similar to [20], we further give the following definition: a neural network is called an INN if one of its input, output, or weight sets is interval-valued, and it follows the rules of interval arithmetic. This concept was introduced since the early 90's [21]–[24], and exhibited similar principles in terms of various perspectives, such as granular neural networks [25], [26]. Some researchers convert interval value into point value by some methods, and then use conventional point-valued neural network for processing, such as interval probabilistic neural network (IPNN) [27], interval radial basis function network (IRBFN)IRBFN [28], not within the scope of our discussion as they do not meet our above definition of network operation on interval arithmetic.

For problem-solving, error back-propagation (BP) algorithm is usually applied for training the interval feedforward neural network (termed IBPNN for simplification) [20]–[24], but still suffers from drawbacks such as the local minima solution, slow convergence and huge uncertainty in learner architecture and parameter settings. In this work, we extend the random vector functional-link network (RVFLN) [29]–[31] to interval version, named interval RVFLN (IRVFLN), by combing the trivial architecture of RVFLN (without direct link from inputs to the outputs) with some interval analysis principles [32]. Similar as performed in RVFLN, the hidden weights (and biases) of IRVFLN are randomly assigned with interval values and remain fixed. The output weights however are expressed as intervals with both lower and upper bound components, which can be analytically evaluated by solving two separate least squares problems.

Since the bounded errors in the UBB method can be described by intervals, it is easy to model an interval plant by using the IRVFLN under the condition of known error bounds. While in the condition of unknown error bounds, which is a modelling problem that cannot be settled in the UBB method, considering that a real (or point) value is a special case of an interval and resorting to the weighted strategy, the penalty factor is employed to improve the IRVFLN learning algorithm. In this way, the IRVFLN can be adapted to modelling an interval system with either known or unknown error bounds. Our experimental results on four numerical examples and an industrial case study for interval modelling of an uncertain recycling plant have demonstrated a good potential of IRVFLN in uncertainty modelling tasks.

In summary, there are three aspects to the contributions, as described below. First, a novel strategy is proposed to model a dynamic uncertain system using the INN to overcome the problems existing in the UBB errors method. Second, a new type of randomized learner model, named interval random vector functional-link network (IRVFLN), is proposed to take advantages of the inherent RVFLN in rapid modeling. Finally, an improved IRVFLN with penalty factors in learning algorithm is applied to model a dynamic industrial plant by bounded-error data in either known or unknown error bounds.

The remainder of this paper is organized as follows. The basics of the interval arithmetic, RVFLN and UBB method are reviewed in Section II. Section III details the architecture of IRVFLN and the proposed learning algorithm. Experimental study on four artificial examples with the comparison with IBPNN is presented in Section IV, with the robustness analysis for random selection range of IRVFLN. Considering the situation of unknown error bounds, an improved IRVFLN learning algorithm is proposed by adding penalty factors, and then it is used to model an uncertain dynamic chemical recycling plant in Section V, the generalization ability of the proposed network in the recurrent formation with the joint inputs (points and intervals) is also discussed in this section. Section VI concludes this work with further remarks.

II. PRELIMINARIES

A. INTERVAL OPERATIONS

Here, we briefly review some preliminaries for interval arithmetic, which is a generalization of ordinary arithmetic to closed intervals [32]. A continuous subset $A = [\underline{a}, \bar{a}]$ of a real number set is called an interval, and the lower and upper bounds of the interval are represented by $\underline{a}$ and $\bar{a}$, respectively. In this paper, interval numbers are denoted by upper case letters such as A and B, interval vectors are described by bold upper case letters such as $\boldsymbol{A}$ and $\boldsymbol{B}$, real numbers are

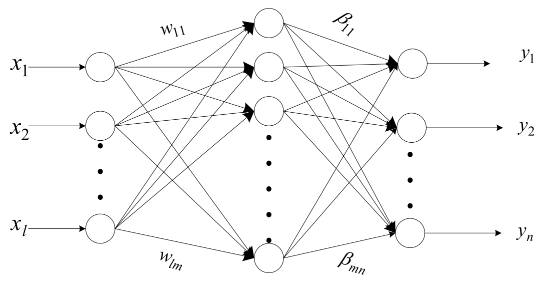

FIGURE 1. **Structure of RVFLN.**

represented by lower case letters such as a and b, and vectors are represented by bold lower case letters such as $\boldsymbol{a}$ and $\boldsymbol{b}$. Some interval arithmetic operations are defined as follows:

Addition:

$$A + B = [\underline{a} + \underline{b}, \bar{a} + \bar{b}]. \tag{1}$$

Subtraction:

$$A - B = [\underline{a} - \bar{b}, \bar{a} - \underline{b}]. \tag{2}$$

Multiplication:

$$A \times B = [\min(\underline{ab}, \bar{a}\underline{b}, \underline{a}\bar{b}, \bar{a}\bar{b}), \max(\underline{ab}, \bar{a}\underline{b}, \underline{a}\bar{b}, \bar{a}\bar{b})]. \tag{3}$$

Division (excluding division by an interval B containing zero):

$$A/B = [\min(\underline{a}/\underline{b}, \bar{a}/\underline{b}, \underline{a}/\bar{b}, \bar{a}/\bar{b}), \max(\underline{a}/\underline{a}, \bar{a}/\underline{b}, \underline{a}/\bar{b}, \bar{a}/\bar{b})]. \tag{4}$$

The sigmoid function, which is used as the activation function of hidden nodes, is extended to the case of an interval input as

$$f(Net) = \{f(x) | x \in Net\}. \tag{5}$$

where Net is an interval input and

$$f(x) = 1/(1 + exp(\text{-}\, x)). \tag{6}$$

Since the sigmoid function (6) is a strictly increasing function, the interval output in (5) can be calculated as

$$f(Net) = f([\underline{net}, \overline{net}]) = [f(\underline{net}), f(\overline{net})]. \tag{7}$$

B. RANDOM VECTOR FUNCTIONAL-LINK NETWORK

A RVFLN can be viewed as a classical single-layer feedforward network with the hidden weights (and biases) randomly assigned and output weights evaluated by least squares method [29]. A typical architecture of RVFLN is shown in Fig.1.

The relationship between the inputs and outputs of the RVFLN with the structure l-m-n can be defined as follows

$$y_k(x) = \sum_{j=1}^{m} \beta_{jk} \cdot f(w_j^T \cdot x + \theta_j), \quad k = 1, 2, \cdots, n. \tag{8}$$

where β_{jk} is the output weight linking the jth node of the hidden layer and the kth node of the output layer; $x = (x_1, x_2, \cdots, x_l)^T$ is the input vector; $w_j = (w_{1j}, w_{2j}, \ldots, w_{lj})^T$ is the hidden weight vector that links the input layer and the jth node of the hidden layer; θ_j is the hidden-layer bias of the jth node; l, m and n are the numbers of input, hidden and output layer nodes, respectively; and $f(\cdot)$is a basis function of the hidden layer nodes.

In RVFLN, the parameters of the hidden layer (w_j and θ_j) are assigned randomly from $[-\lambda, \lambda], \lambda > 0$, and this is done independently from the training samples; the linear parameters β_{jk} of the output layer can be tuned using quadratic optimization techniques, and then the problems of local optimization and slow convergence can be effectively solved [29].

C. PROBLEM FORMULATION UNDER UBB CONDITION

The UBB problem is described below

$$y = f(\boldsymbol{x}, \boldsymbol{q}) + e. \tag{9}$$

where the input and output samples are real values, that is (x_i, y_i), and the error belongs to an interval, that is $e_i \in E_i = [\underline{e}_i, \bar{e}_i]$, where $\underline{e}_i$ and $\bar{e}_i$ are error bounds. The task of parameter estimation is to determine the parameter set $\boldsymbol{Q}$ to make the error $\boldsymbol{E}$ be satisfied, where $\boldsymbol{q} \subset \boldsymbol{Q}$,$\boldsymbol{E} = \{E_i\}$.

When the error bounds $\underline{e}_i$,$\bar{e}_i$ are known, according to the output measured data and error, we can obtain the desired output interval, which is $Y_i = [y_i - \bar{e}_i, y_i - \underline{e}_i]$. Expand model f to the interval form F, and redefine the model (9) as follows

$$Y = F(\boldsymbol{x}, \boldsymbol{Q}). \tag{10}$$

Then, the parameter estimation can be transformed into a regression problem. We can find the optimal solution of the interval parameter set $\boldsymbol{Q}$ according to the real input and interval output data pairs of (x_i, Y_i).

If we only consider the mapping relationship and neglect the model's internal structural features, we can easily establish the mathematical model of an uncertain system according to the input and output data by using INN. The model relationship is as follows

$$Y = F(\boldsymbol{x}, \boldsymbol{Q}) = INN(\boldsymbol{x}, \boldsymbol{W}, \boldsymbol{\Theta}). \tag{11}$$

where W and Θ are the network parameters. This means that the interval network structure is adopted instead of the theoretical model structure, the interval parameters of the system are replaced by the interval parameters of the network, and the learning process of the network parameters is used instead of the interval parameters identification. This new approach can effectively solve the problems that existed in the uncertain system parameter estimation under the UBB condition and provide a new way to model uncertain systems.

When the error bounds are unknown, which means that $\underline{e}_i$ and $\bar{e}_i$ are not available, then the system output yi is just a random real value between a certain range, and we cannot obtain the output interval $Y_i = [y_i - \bar{e}_i, y_i - \underline{e}_i]$. In this case, the parameter estimation of (10) cannot be determined by using the UBB method. At present, there is no effective

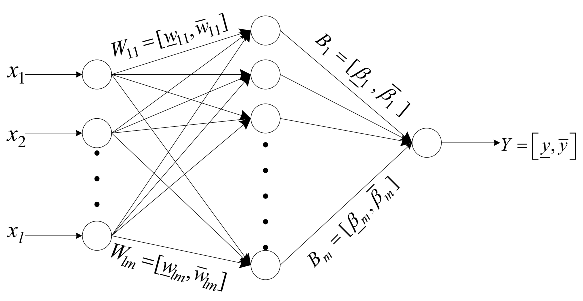
x_1
x_2
x_l
$W_{11}=[\underline{w}_{11},\bar{w}_{11}]$
$W_{lm}=[\underline{w}_{lm},\bar{w}_{lm}]$
$B_1=[\underline{\beta}_1,\bar{\beta}_1]$
$B_m=[\underline{\beta}_m,\bar{\beta}_m]$
$Y=[\underline{y},\bar{y}]$

FIGURE 2. Structure of the IRVFLN.

approach to solve this problem in the UBB modelling field, which motivates us to employ INNs for problem-solving.

III. INTERVAL RANDOM VECTOR FUNCTIONAL-LINK NETWORK

A. ARCHITECTURE OF IRVFLN

The IRVFLN has three layers, including the input layer, hidden layer and output layer. The weights in the network have interval values, and the activation function of the hidden unit is the sigmoid function. Here, for simplicity, we assume that the IRVFLN has one output, and then the point input/interval output architecture of IRVFLN with the structure l-m-1 is described in Fig.2

Where $W_{ij} = [\underline{w}_{ij}, \bar{w}_{ij}]$ is an interval that links the ith node of the input layer with the jth node of the hidden layer, $i = 1, 2 \ldots, l, j = 1, 2, \ldots, m$, and l and m are the numbers of input and hidden layer nodes, respectively. $\Theta_j = [\underline{\theta}_j, \bar{\theta}_j]$ is the interval bias of the jth hidden node, $B_j = [\underline{\beta}_j, \bar{\beta}_j]$ is an interval that links the jth hidden unit with the output unit, $x_p = (x_{p1}, x_{p2}, \ldots, x_{pl})^T$ is the point-valued input vector of the pth sample, and $Y_p = [\underline{y}_p, \bar{y}_p]$ is an interval output of the pth sample. The activation function is denoted by $f(\cdot)$.

Different from the point-valued RVFLN, the parameters of the hidden layer ($\boldsymbol{W_j} = (W_{1j}, W_{2j}, \ldots, W_{lj})$ and Θ_j) in the IRVFLN are intervals and assigned randomly from the uniform distribution interval $[-\lambda, \lambda], \lambda > 0$, which means that they are the subintervals of $[-\lambda, \lambda]$. The training task is to determine the interval parameters B_j of the output layer by using the quadratic optimization technique.

The output of the jth hidden unit can be written as

$$U_{pj} = [\underline{u}_{pj}, \bar{u}_{pj}] = f(\sum_{i=1}^{l} W_{ij}x_{pi} - \Theta_j). \tag{12}$$

where

$$\underline{u}_{pj} = f\left(\sum_{\substack{i=1\\x_{pi}\geq 0}}^{l} \underline{w}_{ij}\cdot x_{pi} + \sum_{\substack{i=1\\x_{pi}<0}}^{l} \bar{w}_{ij}\cdot x_{pi} - \bar{\theta}_j\right). \tag{13}$$

$$\bar{u}_{pj} = f\left(\sum_{\substack{i=1\\x_{pi}\geq 0}}^{l} \bar{w}_{ij}\cdot x_{pi} + \sum_{\substack{i=1\\x_{pi}<0}}^{l} \underline{w}_{ij}\cdot x_{pi} - \underline{\theta}_j\right). \tag{14}$$

The output can be denoted as follows

$$Y_p = [\underline{y}_p, \bar{y}_p] = \sum_{j=1}^{m} B_j U_{pj}. \tag{15}$$

where

$$\underline{y}_p = \sum_{\substack{j=1\\ \underline{\beta}_j\geq 0}}^{m} \underline{\beta}_j \cdot \underline{u}_{pj} + \sum_{\substack{j=1\\ \underline{\beta}_j< 0}}^{m} \underline{\beta}_j \cdot \bar{u}_{pj}. \tag{16}$$

$$\bar{y}_p = \sum_{\substack{j=1\\ \bar{\beta}_j\geq 0}}^{m} \bar{\beta}_j \cdot \bar{u}_{pj} + \sum_{\substack{j=1\\ \bar{\beta}_j< 0}}^{m} \bar{\beta}_j \cdot \underline{u}_{pj}. \tag{17}$$

For simplicity, define uu_{pj} and ul_{pj} as

$$ul_{pj} = \begin{cases} \underline{u}_{pj} & , \underline{\beta}_j \geqslant 0 \\ \bar{u}_{pj}, & \underline{\beta}_j < 0 \end{cases}, \quad uu_{pj} = \begin{cases} \bar{u}_{pj}, & \bar{\beta}_j \geqslant 0 \\ \underline{u}_{pj}, & \bar{\beta}_j < 0 \end{cases} \tag{18}$$

Substituting (18) with (16) and (17), then we have the following

$$\underline{y}_p = \sum_{j=1}^{m} \underline{\beta}_j \cdot ul_{pj}, \bar{y}_p = \sum_{j=1}^{m} \bar{\beta}_j \cdot uu_{pj}. \tag{19}$$

B. LEARNING ALGORITHM

Given the structure of IRVFLN described in Fig. 2, we know that the network inputs are point values, while the output is an interval. Thus, the training dataset can be described as $\{(x_1, D_1), (x_2, D_2), \cdots, (x_P, D_P)\}$, where x_p denotes the point-valued input and $D_p = [\underline{d}_p, \bar{d}_p]$ denotes the interval output,p= $1, 2, \cdots$,P, which means the pth sample of total P.

Our objective is to let the network output interval sufficiently approximate the target interval. From that purpose, we can formulate the learning cost function as follows

$$E_T = \sum_{p=1}^{P} E_p = \frac{1}{2}\sum_{p=1}^{P}\left(\left(\underline{y}_p - \underline{d}_p\right)^2 + \left(\bar{y}_p - \bar{d}_p\right)^2\right). \tag{20}$$

Technically, the parameters $B_j = [\underline{\beta}_j, \bar{\beta}_j]$ need to be tuned to ensure that the IRVFLN can obtain an optimal performance when the gradient of the error function in (20) vanishes with respect to B_j, as we detail in the following.

First, the derivative of the cost function with respect to $\underline{\beta}_j$ and $\bar{\beta}_j$, respectively, to be zero:

$$\frac{\partial E_T}{\partial \underline{\beta}_j} = \sum_{p=1}^{P} \frac{\partial E_p}{\partial \underline{\beta}_j} = \sum_{p=1}^{P} (\underline{y}_p - \underline{d}_p) \cdot \frac{\partial \underline{y}_p}{\partial \underline{\beta}_j} = 0. \tag{21}$$

$$\frac{\partial E_T}{\partial \bar{\beta}_j} = \sum_{p=1}^{P} \frac{\partial E_p}{\partial \bar{\beta}_j} = \sum_{p=1}^{P} (\bar{y}_p - \bar{d}_p) \cdot \frac{\partial \bar{y}_p}{\partial \bar{\beta}_j} = 0. \tag{22}$$

Since the derivation process of $\frac{\partial E_T}{\partial \underline{\beta}_j}$ is similar to $\frac{\partial E_T}{\partial \bar{\beta}_j}$, we only give the derivation process of $\frac{\partial E_T}{\partial \underline{\beta}_j}$. Based on (21), we have

$$\sum_{p=1}^{P}(\sum_{h=1}^{m}\underline{\beta}_h \cdot ul_{ph} - \underline{d}_p) \cdot \frac{\partial \underline{y}_p}{\partial \underline{\beta}_j} = 0. \tag{23}$$

which leads to

$$\sum_{p=1}^{P}\left(\sum_{h=1}^{m}\underline{\beta}_h \cdot ul_{ph}\right) \cdot \frac{\partial \underline{y}_p}{\partial \underline{\beta}_j} = \sum_{p=1}^{P}\underline{d}_p \cdot \frac{\partial \underline{y}_p}{\partial \underline{\beta}_j}. \tag{24}$$

From (19), we get $\frac{\partial \underline{y}_p}{\partial \underline{\beta}_j} = ul_{pj}$. Then, (24) can be written as follows

$$\sum_{p=1}^{P}\left(ul_{pj} \cdot \sum_{h=1}^{m}\underline{\beta}_h \cdot ul_{ph}\right) = \sum_{p=1}^{P}\left(ul_{pj} \cdot \underline{d}_p\right). \tag{25}$$

Now we can rewrite (25) into the matrix form, that is

$$UL^T \cdot UL \cdot \underline{\boldsymbol{\beta}} = UL^T \cdot \underline{\boldsymbol{d}}. \tag{26}$$

where $\underline{\boldsymbol{\beta}} = \left[\underline{\beta}_1, \underline{\beta}_2, \cdots, \underline{\beta}_m\right]^T$,$\underline{\boldsymbol{d}} = \left[\underline{d}_1, \underline{d}_2, \cdots, \underline{d}_p\right]^T$,and

$$UL = \begin{bmatrix} ul_{11} & \cdots & ul_{1m} \\ \vdots & \ddots & \vdots \\ ul_{P1} & \cdots & ul_{Pm} \end{bmatrix}$$

Similarly, as for the upper component, we have

$$UU^T \cdot UU \cdot \bar{\boldsymbol{\beta}} = UU^T \cdot \bar{\boldsymbol{d}}. \tag{27}$$

where $\bar{\boldsymbol{\beta}} = [\bar{\beta}_1, \bar{\beta}_2, \cdots, \bar{\beta}_m]^T$,$\bar{\boldsymbol{d}} = [\bar{d}_1, \bar{d}_2, \cdots, \bar{d}_P]^T$,and

$$UU = \begin{bmatrix} uu_{11} & \cdots & uu_{1m} \\ \vdots & \ddots & \vdots \\ uu_{P1} & \cdots & uu_{Pm} \end{bmatrix}$$

Combine (26) with (27), we obtain the evaluation of the output weight $\underline{\boldsymbol{\beta}}$, the lower component, and $\bar{\boldsymbol{\beta}}$, the upper component, as follows

$$\underline{\boldsymbol{\beta}} = \left(UL^T \cdot UL\right)^{-1} \cdot \left(UL^T \cdot \underline{\boldsymbol{d}}\right). \tag{28}$$

$$\bar{\boldsymbol{\beta}} = \left(UU^T \cdot UU\right)^{-1} \cdot \left(UU^T \cdot \bar{\boldsymbol{d}}\right). \tag{29}$$

Combine the above results, the following **Algorithm 1** is obtained to describe our proposed learning scheme for building IRVFLN model.

Remark 1: From (18), we can see that if we want to obtain UU and UL, the value of the attribute $B_j = \left[\underline{\beta}_j, \bar{\beta}_j\right]$ (i.e. positive or negative) must be known. Therefore, we randomly assign a value of B_j at first.

Remark 2: The matrix $\left(UL^T \cdot UL\right)$ and $\left(UU^T \cdot UU\right)$ in (28) and (29) may not always be reversible. To overcome this issue, one can use the generalized pseudo inverse for problem-solving.

Algorithm 1 IRVFLN Algorithm

```
Require:Training dataset $X = \{X_p\}$(p= $1, 2, \cdots$ ,P), the sigmoid active function $f(\cdot)$, number of nodes: $l$ input, $m$ hidden, $n$ output.
Ensure: Trained IRVFLN model
1: Initialize the IRVFLN structure.
2: Initialize the weights $W_{ij}$ and biases $\Theta_j$: randomly set $W_{ij}$ and $\Theta_j$.
3: Calculate the outputs of the hidden layer for all samples in $X$, i.e., calculate $\underline{u}_{pj}(X_p)$,$\bar{u}_{pj}(X_p)$. For convenience, define two temporary variables $\underline{\phi}_{pj}(X_p)$ and $\bar{\phi}_{pj}(X_p)$ in the following procedure.
4: for $p$=1 to $P$ do
5:   for $j$= 1 to $m$ do
6:     for $i$ =1 to $l$ do
7:       if $x_{pi} \geq 0$ then
8:         $\left(\underline{\phi}_{pj}(X_p), \bar{\phi}_{pj}(X_p)\right) \leftarrow \left(\sum_{i=1}^{l}\underline{w}_{ij} \cdot x_{pi} - \bar{\theta}_j, \sum_{i=1}^{l}\bar{w}_{ij} \cdot x_{pi} - \underline{\theta}_j\right)$
9:       else
10:        $\left(\underline{\phi}_{pj}(X_p), \bar{\phi}_{pj}(X_p)\right) \leftarrow \left(\sum_{i=1}^{l}\bar{w}_{ij} \cdot x_{pi} - \bar{\theta}_j, \sum_{i=1}^{l}\underline{w}_{ij} \cdot x_{pi} - \underline{\theta}_j\right)$
11:      end if
12:    end for
13:    $\left(\underline{u}_{pj}(X_p), \bar{u}_{pj}(X_p)\right) \leftarrow \left(f(\underline{\phi}_j(X_p)), f(\bar{\phi}_j(X_p))\right)$
14:  end for
15: end for
16: Randomly assign values to $\underline{\boldsymbol{\beta}}$ and $\bar{\boldsymbol{\beta}}$, and then calculate $ul_{pj}$ and $uu_{pj}$ according to (18).
17: Construct the matrix $UL$ and $UU$ according to (26) and (27), respectively.
18: Calculate $\underline{\boldsymbol{\beta}}$ and $\bar{\boldsymbol{\beta}}$ according to (28) and (29), respectively.
19: Return IRVFLN model.
```

IV. PERFORMANCE EVALUATION

This section investigates the performance of IRVFLN and then compares it with the popular model IBPNN, and then analyzes the effectiveness of the random selection range to the network regression capability.

A. ANALYSIS WITH NUMERICAL DATASETS

Four sets of datasets obtained from the test functions are considered in order to analyze the performance of IRVFLN on regression tasks.

Func1: $Y = \sin x + [0.2, 0.6], x \in U[0, 1]$,

Func2: $Y = 10\sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5 \cdot [0.3, 0.6], x_i \in U[0, 1]$, Func3: $Y = \sqrt{(x_2x_3 - 1/(x_3x_4))/x_1} + [0.1, 0.5]$, $x_1 \in U[0, 100]$, $x_2 \in U[40\pi, 560\pi]$, $x_3 \in U[0, 1]$, $x_4 \in U[1, 11]$

Func4: $Y = 0.79 + 1.27 \cdot [-1, 1] \cdot x_1x_2 + 1.56x_1x_4 + 3.42x_2x_5 + 2.06x_3x_4x_5$, $x_i \in U[0, 1]$, $(i = 1, 2, 3, 4, 5)$.

These functions are Sin, Friedman #1, Friedman #3, and Multi. The domain variables of all the functions are randomly generated using a uniform probability distribution $U[a, b]$, where a and b are the lower and upper bounds of the corresponding variables, respectively. The number of the training data $[x_i, Y_i]$ generated is 1000 for all datasets.

To suit the performance analysis of the interval values, we define the root mean square error (RMSE) of the output lower bound, upper bound and midpoint as follows:

$$RMSE_L = \sqrt{\frac{1}{P'}\sum_{p=1}^{P'}(\underline{y}_p - \underline{d}_p)^2}. \tag{30}$$

$$RMSE_U = \sqrt{\frac{1}{P'}\sum_{p=1}^{P'}(\bar{y}_p - \bar{d}_p)^2}. \tag{31}$$

$$RMSE_M = \sqrt{\frac{1}{P'}\sum_{p=1}^{P'}\left[MID(Y_p) - MID(D_p)\right]^2}. \tag{32}$$

where $RMSE_L$, $RMSE_U$ and $RMSE_M$ are the RMSE of the outputs lower bound, upper bound and midpoint, respectively, and P' is the number of testing samples. In (32), MID means the midpoint of an interval, i.e. $MID(Y_p) = (\underline{y}_p + \bar{y}_p)/2$. It should be stated that the $RMSE_M$ will be mainly used in practice because the midpoint of the output interval of an IRVFLN indicates the prediction value of the uncertain system.

To illustrate the performance superiority of the proposed IRVFLN, we compare the IRVFLN with the conventional IBPNN. Due to the different learning processes of the IRVFLN and IBPNN, the selection of the network structure is also different. To compare the IRVFLN and IBPNN reasonably and effectively, the optimal results for both networks are selected for comparison. For the IRVFLN, the trial-and-error method described in the next subsection IV-B can be used to determine the network structure and the distribution range for each test function, where we set 1-80-1 and $\lambda = 2$ for Func1, 5-40-1 and $\lambda = 1$ for Func2, 4-80-1 and $\lambda = 1$ for Func3, and 5-70-1 and $\lambda = 2$ for Func4.

Similarly, the trial-and-error method can also be used to determine the network structure of the IBPNN for each regression task, which means to set a suitable hidden node number to obtain an optimal performance for the IBPNN. Comprehensively considering the network's ability to approximate and converge, the network structure of IBPNN for each test function is given as 1-5-1, 5-8-1, 4-8-1, and 5-7-1. Unlike the IRVFLN, the IBPNN adopts the iterative training process. Therefore, we need to assign the termination condition in the calculation process. There are two conventional options for the termination, namely, by giving the iteration number or limiting the iteration accuracy. Here, to ensure the optimization of the cost function, we use the latter method. That is, if the absolute error between two iterative times is less than 0.0001, stop the training process.

TABLE 1. Comparison of training performance between IRVFLN and IBPNN.

Dataset	IRVFLN		IBPNN	
	Training time (s)	Average error	Training time (s)	Average error
Func1	0.2821	$1.0102*10^{-13}$	2.9089	$1.0907*10^{-4}$
Func2	1.0541	0.0114	97.5083	1.3299
Func3	1.0249	0.2464	99.4382	3.6365
Func4	1.0443	$1.3821*10^{-4}$	114.7802	0.0692

TABLE 2. Comparison of the RMSEs between IRVFLN and IBPNN.

Dataset	IRVFLN			IBPNN		
	$RMSE_L$	$RMSE_U$	$RMSE_M$	$RMSE_L$	$RMSE_U$	$RMSE_M$
Func1	$8.25*10^{-7}$	$9.79*10^{-7}$	$8.94*10^{-7}$	0.0085	0.01	0.0092
Func2	0.0692	0.0698	0.0609	1.9198	1.9142	1.9169
Func3	0.3703	0.4978	0.3847	2.465	1.8654	2.1073
Func4	0.0135	0.0124	0.0108	0.1802	0.1974	0.1275

Table 1 shows the performance comparison between the IRVFLN and IBPNN in the training process based on the above four datasets. By considering the training time and the average error (E_T/P), we can see that the IRVFLN can obtain a faster training speed and a smaller average error.

After training of the two type interval networks, the prediction performance will be evaluated on the testing dataset. Fig. 3 shows the test results, where the dotted blue line indicates the upper and lower bounds of the samples, while the solid red line indicates the upper and lower bounds of the network outputs. The left diagram is the prediction results of the IRVFLN and the right diagram is that of the IBPNN. We can see that the IRVFLN has better prediction accuracy. If the test function is simple, the two types of interval networks have similar forecasting precision, whereas if the test function is relatively complicated, the IRVFLN is significantly better than the IBPNN. Table 2 shows the RMSEs of the IRVFLN and IBPNN from the Fig. 3 results, which indicates that the IRVFLN performs better than the IBPNN.

B. ROBUSTNESS ANALYSIS FOR RANDOM SELECTION RANGE

In the process of constructing an IRVFLN, there are two relatively difficult problems. One is how to set the uniform distribution range λ, which determines how to randomly assign the initial values of the weight $\boldsymbol{W_j}$ and the bias Θ_j, and the other is how to set the node number m in the hidden layer. Here, we use a trial-and-error method to choose the distribution range $[-\lambda, \lambda]$ and the hidden node number m. That is, we randomly initialize the hidden weights and biases in different distribution ranges and select the different hidden nodes. Then, we train the IRVFLN using the testing function or a real process dataset. Finally, we analyze the RMSE

TABLE 3. Setting analysis of the distribution range and hidden nodes.

m	$\lambda = 0.5$		$\lambda = 1$		$\lambda = 2$		$\lambda = 3$	
	$RMSE_L$	$RMSE_U$	$RMSE_L$	$RMSE_U$	$RMSE_L$	$RMSE_U$	$RMSE_L$	$RMSE_U$
60	$2.62*10^{-5}$	$1.92*10^{-5}$	$6.97*10^{-6}$	$2.16*10^{-5}$	$1.81*10^{-6}$	$1.09*10^{-5}$	$2.44*10^{-5}$	$2.04*10^{-5}$
70	$1.64*10^{-5}$	$2.23*10^{-5}$	$1.20*10^{-5}$	$1.43*10^{-5}$	$1.30*10^{-6}$	$1.49*10^{-6}$	$2.51*10^{-5}$	$1.79*10^{-5}$
80	$1.09*10^{-5}$	$7.91*10^{-6}$	$7.76*10^{-6}$	$1.06*10^{-5}$	$7.79*10^{-7}$	$9.20*10^{-7}$	$7.22*10^{-6}$	$9.99*10^{-6}$
90	$2.04*10^{-5}$	$1.64*10^{-5}$	$6.93*10^{-6}$	$1.15*10^{-5}$	$1.03*10^{-5}$	$9.91*10^{-7}$	$7.35*10^{-6}$	$8.23*10^{-6}$
100	$1.01*10^{-5}$	$1.59*10^{-5}$	$5.26*10^{-6}$	$8.07*10^{-6}$	$1.13*10^{-6}$	$5.96*10^{-7}$	$5.09*10^{-6}$	$1.20*10^{-5}$

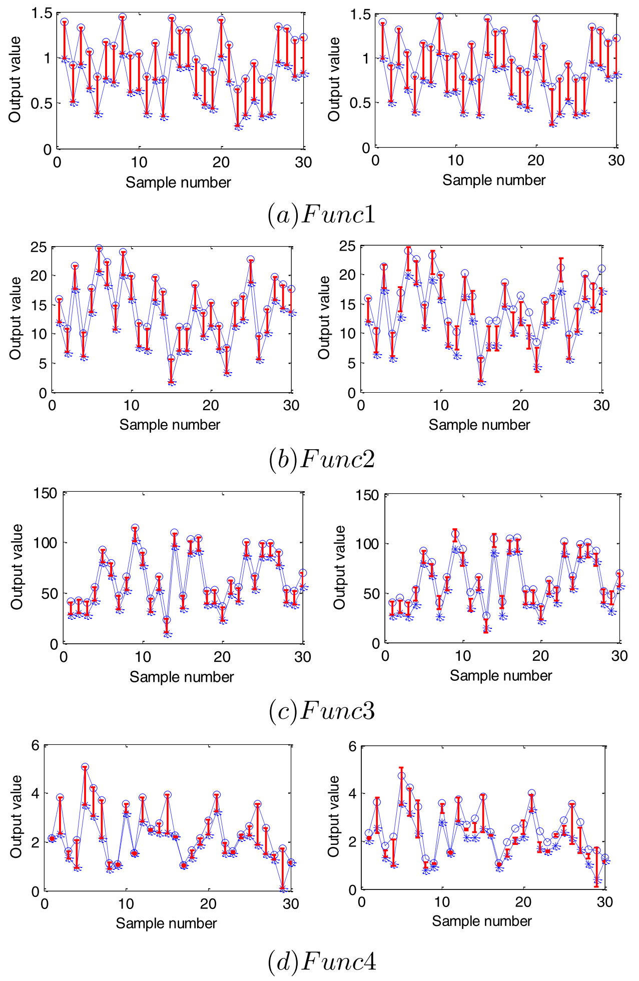

FIGURE 3. Comparison of the test results between IRVFLN and IBPNN.

of the validation dataset to determine the final parameters of λ and m.

The prior mentioned function Fun1, i.e. $Y = \sin x + [0.2, 0.6], x \in U[0, 1]$, is selected to show the trial-and-error method to choose the distribution range λ and the hidden node number m, and the simulation results are demonstrated in Table 3. The detailed steps are that, for a given hidden node number m, the uniform distribution selected for the weight $\boldsymbol{W_j}$ and the bias Θ_j is $[-\lambda, +\lambda]$, where different values are assigned to λ to examine the generalization performance. All choices for λ are assessed using a 10-fold cross-validation procedure. For each λ, we run the algorithm 10 times for each experiment. For each run, the algorithm randomly initiates the hidden weights and biases in the range $[-\lambda, +\lambda]$, calculates the other parameters based on the training datasets, and then verifies the IRVFLNs performance over one validation dataset. The performance measurements from all folds are collected and averaged over the ten iterations. Due to the randomness of the weights, biases and data sampling, each λ value is simulated ten times for more reliable results. Then, the results from all simulations are averaged, which means that the $RMSE_L$ and $RMSE_U$ in Table 3 are the averaged values on the testing dataset.

The simulation results shown in Table 3 indicate that the choice of random selection range has a significant impact on the network performance, and the uniform distribution $[-2, +2]$ ($\lambda = 2$) and the hidden nodes $m = 80$ seem to be relatively satisfactory.

V. APPLICATION STUDIES OF MODELING AN UNCERTAIN INDUSTRIAL PLANT

In this section, the application to modelling an uncertain system with the proposed network under UBB condition is discussed, and some substantial improvements are proposed.

A. APPLICATION ANALYSIS UNDER UBB CONDITION

The application of the IRVFLN under UBB condition contains two cases: one is that the error bounds are known, and the other is the error bounds are unknown.

Case 1: the error bounds are known

The architecture and the learning algorithm of the IRVFLN in sections III-A and III-B can be used to model an interval plant when the error bounds are known. In $Y = f(\boldsymbol{x}, \boldsymbol{q}) + [\underline{e}, \bar{e}]$, we know the lower bound $\underline{e}$ and the upper bound $\bar{e}$. Based on that, the point-valued input and interval output sample pairs can be obtained. Thus, we can train the parameters of the IRVFLN to obtain an INN prediction model.

Case 2: the error bounds are unknown

In the industrial processes, we usually obtain the input-output datasets with point values in the case of the error bounds being unknown. For the uncertain system, the outputs of the system are different, even with the same input, due to some random parameters that exist. Thus, the output actually can be seen as an interval, although the interval bounds are unknown. For instance, for the interval function $Y = \sin(x) + E$, if the bounds $\underline{e}$ and $\bar{e}$ are unknown, then it can be rewritten as $y = \sin(x) + rand[\underline{e}, \bar{e}]$, where $rand(\cdot)$ means obtaining a random value from an interval. The output can be qualitatively shown in Fig. 4 through several iterations,

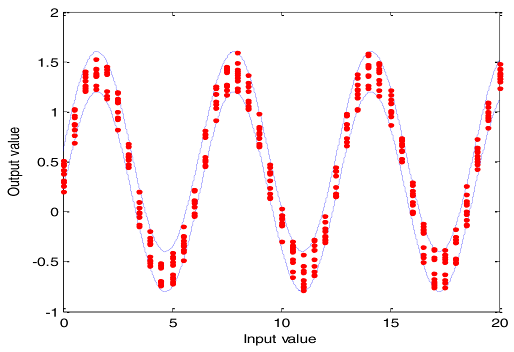

FIGURE 4. Interval function output of unknown error bounds.

where the dotted blue lines indicate the unknown upper and lower bounds of the samples, while the solid red dots indicate the point values.

In fact, we should generate the sample output as much as possible so that we can ensure that the randomly generated target output is trained to fall within the predicted range of the proposed network as much as possible.

For the unknown error bounds, we propose two methods. One method is to generate a large amount of data. For different outputs of the same input, we take the maximum and minimum values to form an output interval. Then, we convert it into case 1 and use the proposed IRVFLN in section III to solve it.

Another direct method is to treat the point output as a special interval, which has the same upper and lower bounds. Then, the different outputs of the same input are all put into the IRVFLN as independent samples for training. We also select the function of Func1 in subsection IV-A, i.e. $y = \sin(x) + rand[0.2, 0.6], x \in [0, 1]$, to verify the above method. We run the function ten times with the same inputs, and then we randomly obtain 10 outputs under the same inputs and produce a total of 5000 pairs of input and output samples to be the training dataset. Afterward, we run it another time to obtain 50 pairs samples to be the testing dataset. Set the IRVFLN structure as 1-80-1 with the uniform distribution $[-2, +2]$, and then train the IRVFLN according to the algorithm in section III. Afterward, we use the testing dataset to verify the network performance, and the result is shown in Fig. 5, where the solid red line represents the target value and the dotted blue line represents the output of the network.

We can see that although the overall trend is forecasted, the target point value is not ideally contained in the output range, and the number of the target value that has fallen inside the output interval is only 24, which is 48% of the 50 total target values.

To solve this problem, we improve the learning algorithm in subsection III-B to address the form of input-output point values. Here, we establish a model whose input and output are point values. The dataset form is $\{(x_1, d_1), (x_2, d_2), \cdots, (x_P, d_P)\}$, where x_p is the pth sample input, and d_p is the pth sample output.

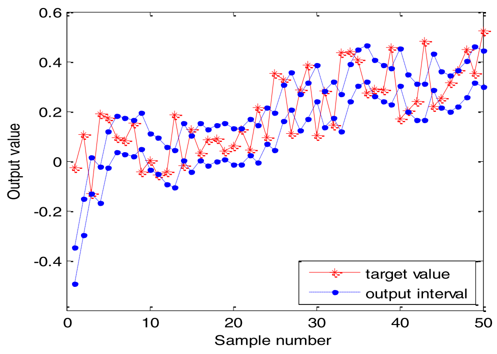

FIGURE 5. Testing result of unknown error bounds.

The training target is to make the sample output d_p as much as possible included in the output interval of the network $Y_p = [\underline{y}_p, \bar{y}_p]$, i.e. $\underline{y}_p \leq d_p \leq \bar{y}_p$. Without loss of generality, let $d_p = \underline{d}_p = \bar{d}_p$, then the target is rewritten as

$$\underline{y}_p \leq \underline{d}_p \leq \bar{d}_p \leq \bar{y}_p. \tag{33}$$

Aiming at this issue, we redefine the cost function with the penalty factor as follows.

$$E_T = \sum_{p=1}^{P} E_p = \sum_{p=1}^{P} \left(\underline{v}_p \underline{E}_p + \bar{v}_p \bar{E}_p\right). \tag{34}$$

where

$$\underline{E}_p = \frac{1}{2}\left(\underline{y}_p - \underline{d}_p\right)^2, \bar{E}_p = \frac{1}{2}\left(\bar{y}_p - \bar{d}_p\right)^2. \tag{35}$$

$\underline{v}_p$ and $\bar{v}_p$ are the penalty factors in $\underline{E}_p$ and $\bar{E}_p$, defined as follows:

$$\underline{v}_p = \begin{cases} \delta, & \underline{d}_p \geq \underline{y}_p \\ 1, & \underline{d}_p < \underline{y}_p \end{cases}, \quad \bar{v}_p = \begin{cases} \delta, & \bar{d}_p \leq \bar{y}_p \\ 1, & \bar{d}_p > \underline{y}_p \end{cases} \tag{36}$$

where σ is a very small positive constant less than 1.

From the cost function, it can be seen that when the IRVFLN output is satisfied with (33), the penalty factor is very small and the weights are changed slightly. However, when the output is not met (33), the penalty factor is 1, the weights are changed significantly, and the learning effect of the network is more obvious. The learning algorithm can be derived from the cost function (34) in the same manner as in subsection III-B, finally, we have

$$\underline{\boldsymbol{\beta}} = \left(UL^T \cdot VL \cdot UL\right)^{-1} \cdot \left(UL^T \cdot VL \cdot \underline{\boldsymbol{d}}\right). \tag{37}$$

$$\bar{\boldsymbol{\beta}} = \left(UU^T \cdot VU \cdot UU\right)^{-1} \cdot \left(UU^T \cdot VU \cdot \bar{\boldsymbol{d}}\right). \tag{38}$$

where

$$VL = diag\left(\underline{v}_1, \underline{v}_2, \cdots, \underline{v}_P\right)$$
$$VU = diag\left(\bar{v}_1, \bar{v}_2, \cdots, \bar{v}_P\right)$$

Then, we train the network using the improved algorithm with the penalty factor. The network setting, training and testing datasets are the same as the IRVFLN aforementioned when no penalty factor is used. The testing result is shown

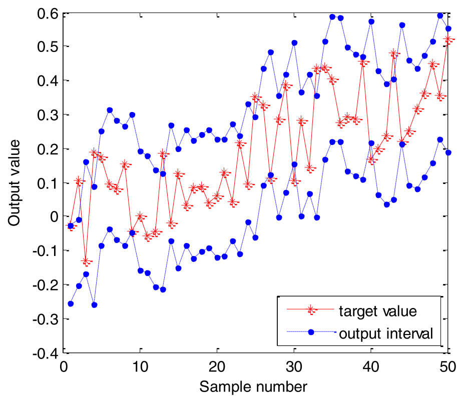

FIGURE 6. Testing result with the penalty factor of unknown error bounds.

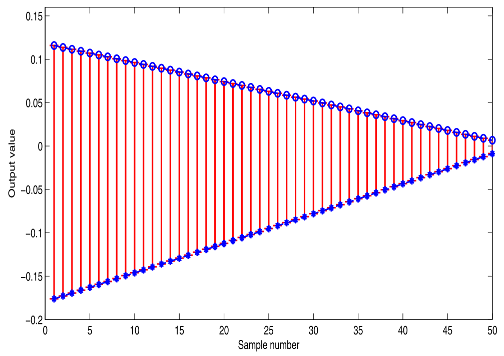

FIGURE 7. Network evaluation of known error bounds.

in Fig. 6, where the solid red line represents the target value and the dotted blue line represents the output of the network.

As is shown in Fig. 6, the improved algorithm makes more target points fall in the output intervals. The number of target points in the output intervals is 42, which is 84% of the 50 total target values, and the number that is no more than 20% of the network output range is 8. The results show that the improved algorithm with the penalty factor can effectively forecast the output interval, and the percentage of the point target contained within the output interval is ideal.

B. INTERVAL MODELLING OF AN UNCERTAIN RECYCLING PLANT

Plants with a recycling operation are often encountered in practical process industries. Taiwo [33] pointed out that it is more reasonable to build an interval model to describe a plant with recycling incorporating a recycling compensator. However, as far as we know, there are almost no papers exploring the challenging problem associated with the modelling of such an interval plant.

We select the transfer function of the recycling operation as the benchmark interval plant model, which can be described as follows [33].

$$G(s) = \frac{y(s)}{u(s)} = \frac{4s+0.5}{2s^2+8.25s+1-0.5k_R\exp(-\tau s)+0.975\exp(-8s)}. \tag{39}$$

where k_R is the steady gain of the recycling operation, $k_R \in [1.17, 2.73]$, τ is system delay and $\tau \in [4.8, 11.2]$.

Case 1: modelling of known error bounds

When the error bounds are known, we need to generate the point input and interval output samples to be the training and testing datasets. Then, the model (39) should be rewritten as

$$G(s) = \frac{Y(s)}{u(s)} = \frac{4s+0.5}{2s^2+8.25s+1-0.5K_R\exp(-Ts)+0.975\exp(-8s)}. \tag{40}$$

where $K_R = [1.17, 2.73]$ and $T = [4.8, 11.2]$.

Model (40) is a point-valued input and interval-valued output transfer function, which can be used to generate the point-interval sample pairs according to the interval operations. The system input is selected as $u(k) = 0.05*\sin(2\pi k/1000)(k = 1, 2, \cdots, 2000)$, and then 2000 total samples are obtained, of which 1950 pairs are used as the training dataset and the remaining 50 pairs are used as the testing dataset.

Considering that the midpoint of the IRVFLN output interval is used as the prediction value, we then select the network inputs as $u(k)$, $u(k-1)$, $u(k-2)$, $u(k-3)$, $u(k-4)$, MIDY(k-1), MIDY(k-2), MIDY(k-3), and MIDY(k-4). That means there are nine neurons in the input layer, in which there are three previous inputs (point values) and four previous outputs (point values), the output neuron is 1 as Y(k) (interval value), and the hidden layer neurons are set as 80. Then, the structure of the IRVFLN is 9-80-1 with the recurrent formation of point inputs and interval output.

After training the IRVFLN using the conventional learning algorithm in subsection III-B, the network prediction performance is verified on the testing dataset. The testing result is shown in Fig. 7, where the solid red line represents the network output intervals and the dotted blue line indicates the target intervals. We can see that the boundary of the network output is close to the target, which demonstrates that the proposed IRVFLN can effectively forecast the output interval of the uncertain system. The RMSE is also calculated according to (30)-(32), they are $RMSE_L = 0.0560$, $RMSE_U = 0.0511$ and $RMSE_M = 0.3035$.

It should be stated that the learning algorithm with the penalty factor in subsection V-A can also be applied in this situation, where the sample pairs have the point-interval formation. The same training and testing datasets as above are used, which lead to almost the same simulation results as those in Fig. 7. That indicates that the basic IRVFLN model can satisfy the requirements of the interval plant modelling under known error bounds.

Case 2: modelling of unknown error bounds When the error bound is unknown, we need to generate the sample pairs in the form of point-point form input-output data. Then, model (39)

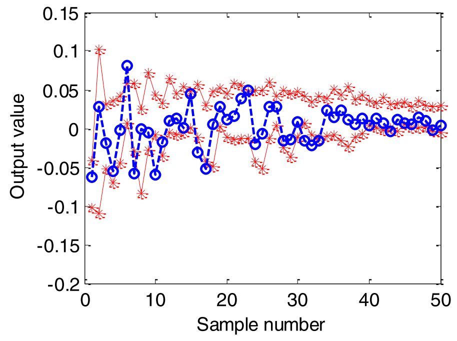

FIGURE 8. Network evaluation of unknown error bounds.

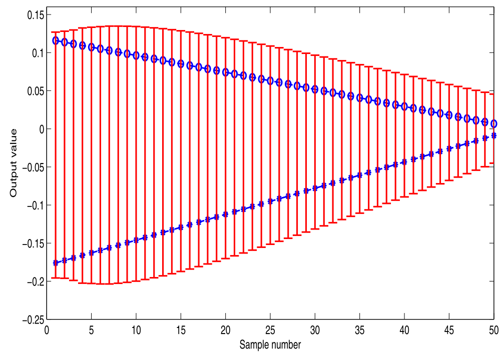

FIGURE 9. Network evaluation of interval output feedback.

should be rewritten as

$$\begin{aligned} G(s) &= y(s)/u(s) \\ &= (4s+0.5)/\Big\{2s^2+8.25s+1-0.5 \\ &\quad \cdot rand(K_R)\cdot \exp[-rand(T)\cdot s]+0.975\exp(-8s)\Big\}. \end{aligned} \tag{41}$$

Model (41) is a real-valued input and real-valued output transfer function, which can be used to generate the point-point sample pairs. The system input is also selected as $u(k) = 0.05 * \sin(2\pi k/1000)$, but the sampling number is selected as $k = 1, 2, \cdots, 10000$, which is five times of that in the situation of known error bounds. Then, 10000 total samples are obtained, of which 9950 pairs are used as the training dataset, and the remaining 50 pairs are used as the testing dataset. The network structure, the input variables and the output variable are the same as that in the situation of known error bounds in Case 1.

Use the improved algorithm in subsection V-A to train the IRVFLN and then to verify it on the testing dataset. The test result is shown in Fig. 8, in which the solid red line represents the network output intervals and the dotted blue line represents the target real values. The absolute number of the target values in the output intervals is 44, in the network output range not exceeding 5% is 3, and in the output range not exceeding 10% is 3. Because the target samples are the real values, the $RMSE_L$ and $RMSE_U$ are meaningless, and the $RMSE_M$ is 0.0144.

From the test results, we can see that the target values are mostly included in the output intervals, which indicates that the proposed IRVFLN can predict the output of an uncertain system in the condition of unknown error bounds.

C. DISCUSSION

The IRVFLN model mentioned above is the recurrent formation with the point inputs and interval output. Considering the properties of the interval network, the interval inputs and interval output model of IRVFLN can also be set up, even for the joint inputs of points and intervals, because the point value can be treated as a special interval with zero width. In practical dynamic modeling, the inputs of control variables are usually point values, but the feedback variables of the previous outputs may be the intervals in the prediction application because the prediction values are intervals, which can reflect the predicting uncertainties to the future forecasting in the long-range prediction. That is not difficult for INNs but is hard to implement for the NN-based PIs mentioned in [16].

The IRVFLN model in section III should be minor modified to possess the capacity in processing the interval inputs. Then from (12), the output of the jth node of the hidden layer can be obtained as

$$U_{pj} = \left[\underline{u}_{pj}, \bar{u}_{pj}\right] = f\left(\sum_{i=1}^{l}(W_{ij}\cdot X_{pi}) - \Theta_j\right). \tag{42}$$

where

$$\underline{u}_{pj} = f\left(\sum_{i=1}^{l}\left[\underline{w}_{ij}, \bar{w}_{ij}\right]\cdot\left[\underline{x}_{pi}, \bar{x}_{pi}\right] - \bar{\theta}_j\right). \tag{43}$$

$$\bar{u}_{pj} = f\left(\sum_{i=1}^{l}\left[\underline{w}_{ij}, \bar{w}_{ij}\right]\cdot\left[\underline{x}_{pi}, \bar{x}_{pi}\right] - \underline{\theta}_j\right). \tag{44}$$

The interval multiplication in (43) and (44) is calculated according to interval arithmetic rule (3).

Model (40) (i.e. Case 1 in the previous subsection) is used to generate sample pairs, producing the same training dataset and testing dataset. Considering the previous output intervals are fed back to the input, then the network inputs are selected as point variables $u(k)$, $u(k-1)$, $u(k-2)$, $u(k-3)$, $u(k-4)$, and interval variables *Y(k-1)*, *Y(k-2)*, *Y(k-3)*, *Y(k-4)*, which construct a joint input vector. The network structure is also set to 9-80-1, and the learning algorithm with penalty factors in subsection V-A is selected to train the IRVFLV model with the interval feedback. The testing result is shown in Fig. 9, in which the red line represents the network output intervals and the blue line indicates the target intervals.

From Fig. 9, we can see that the target intervals are all contained by the network output bounds. Compared with Fig.7, the prediction intervals are wider than that ones, that means the interval feedback imposes a greater impact on the prediction output then the point feedback. That is a reasonable illustration because intervals contain more information then points, and the current uncertainty will affect the following uncertainties prediction. It can conclude that with the prediction steps increase, the network output interval will become more wider, therefore the prediction range should be limited in the interval output feedback.

VI. CONCLUSION

This work proposes a new type of randomized learner model, called interval random vector functional-link network, for dealing with uncertainty system modelling problems. Distinguished from the architecture of RVFL network, the hidden input weights (and biases) in the presented IRVFLN model are expressed by intervals with lower and upper bounds that are randomly assigned. With the help of some properties in interval arithmetic theory, the output weights of IRVFLN model can be represented by intervals, of which the lower and upper components can be evaluated by solving least squares problems. We compare the proposed IRVFLN model with interval learner models produced by the back-propagation algorithm on four artificial examples, the results have verified the advantages of IRVFLN model.

As an important application area, the INNs can be used to model an uncertain system based on the bounded error data, which is a conventional field dominated by the UBB error theory method. Different from the developed NN-based PIs algorithms, the INNs not only can predict the output intervals, but also can describe the uncertainties in the input/output and the network parameters with intervals, possessing the potential to be the long-range prediction with the interval outputs feedback, such that they can be used in the controller design [17]–[19].

The detailed approach is proposed to model an uncertain industrial process system under the condition of known or unknown bounds, the simulations demonstrate that the IRVFLN is suitable and effective, which opens a new direction for uncertain system modeling.

One limitation of our work should be mentioned with more clarification, that is, IRVFLN still suffer from some issues that inherit from RVFL networks, such as reported in [34]. In contrast, the advanced learner model, stochastic configuration networks (SCNs), proposed by Wang *et al.* [35], can exhibit more good potential in industrial system modelling with preferable learning ability and sound generalization performance, and will essentially improve the proposed IRVFLN and produce more advanced interval randomized models.

REFERENCES

[1] C.-T. Chen and M.-D. Wang, "Robust controller design for interval process systems," *Comput. Chem. Eng.*, vol. 21, no. 7, pp. 739–750, Mar. 1997.

[2] B. M. Patre and P. Deore, "Robust stability and performance for interval process plants," *ISA Trans.*, vol. 46, no. 3, pp. 343–349, Jun. 2007.

[3] Y. Zhang, X. Jin, Y. Feng, and G. Rong, "Data-driven robust optimization under correlated uncertainty: A case study of production scheduling in ethylene plant," *Comput. Chem. Eng.*, vol. 109, pp. 48–67, Jan. 2018.

[4] M. Dahleh, A. Tesi, and A. Vicino, "An overview of extremal properties for robust control of interval plants," *Automatica*, vol. 29, no. 3, pp. 707–721, May 1993.

[5] M. Corradini and G. Orlando, "A switching controller for the output feedback stabilization of uncertain interval plants via sliding modes," *IEEE Trans. Autom. Control*, vol. 47, no. 12, pp. 2101–2107, Dec. 2002.

[6] H. Chapellat and S. P. Bhattacharyya, "A generalization of Kharitonov's theorem; Robust stability of interval plants," *IEEE Trans. Autom. Control*, vol. 34, no. 3, pp. 306–311, Mar. 1989.

[7] B. Fang, "Design of PID controllers for interval plants with time delay," *J. Process Control*, vol. 24, no. 10, pp. 1570–1578, Oct. 2014.

[8] D. Zhai and J. M. Mendel, "Uncertainty measures for general type-2 fuzzy sets," *Inf. Sci.*, vol. 181, no. 3, pp. 503–518, Feb. 2011.

[9] J. Helton, "Uncertainty and sensitivity analysis in the presence of stochastic and subjective uncertainty," *J. Stat. Comput. Simul.*, vol. 57, nos. 1–4, pp. 3–76, Apr. 1997.

[10] F. Schweppe, "Recursive state estimation: Unknown but bounded errors and system inputs," *IEEE Trans. Autom. Control*, vol. AC-13, no. 1, pp. 22–28, Feb. 1968.

[11] E.-W. Bai, H. Cho, and R. Tempo, "Convergence properties of the membership set," *Automatica*, vol. 34, no. 10, pp. 1245–1249, Oct. 1998.

[12] L. Jaulin and E. Walter, "Guaranteed nonlinear parameter estimation from bounded-error data via interval analysis," *Math. Comput. Simul.*, vol. 35, no. 2, pp. 123–137, Apr. 1993.

[13] L. Jaulin and E. Walter, "Set inversion via interval analysis for nonlinear bounded-error estimation," *Automatica*, vol. 29, no. 4, pp. 1053–1064, Jul. 1993.

[14] M. Kieffer and E. Walter, "Interval analysis for guaranteed non-linear parameter and state estimation," *Math. Comput. Model. Dyn. Syst.*, vol. 11, no. 2, pp. 171–181, Jun. 2005.

[15] A. Khosravi, S. Nahavandi, D. Creighton, and A. F. Atiya, "Comprehensive review of neural network-based prediction intervals and new advances," *IEEE Trans. Neural Netw.*, vol. 22, no. 9, pp. 1341–1356, Sep. 2011.

[16] H. M. D. Kabir, A. Khosravi, M. A. Hosen, and S. Nahavandi, "Neural network-based uncertainty quantification: A survey of methodologies and applications," *IEEE Access*, vol. 6, pp. 36218–36234, 2018.

[17] M. Liu, "Robust H_∞ control for uncertain delayed nonlinear systems based on standard neural network models," *Neurocomputing*, vol. 71, nos. 16–18, pp. 3469–3492, Oct. 2008.

[18] Z. Wang, L. Huang, and Y. Wang, "Robust decentralized adaptive control for a class of uncertain neural networks with time-varying delays," *Appl. Math. Comput.*, vol. 215, no. 12, pp. 4154–4163, Feb. 2010.

[19] C.-J. Cheng, "Robust control of a class of neural networks with bounded uncertainties and time-varying delays," *Comput. Math. Appl.*, vol. 56, no. 5, pp. 1245–1254, Sep. 2008.

[20] M. Beheshti, A. Berrached, A. De Korvin, C. Hu, and O. Sirisaengtaksin, "On interval weighted three-layer neural networks," in *Proc. 31st Annu. Simulation Symp.*, Nov. 2002, pp. 188–194.

[21] H. Ishibuchi and H. Tanaka, "An extension of the BP-algorithm to interval input vectors-learning from numerical data and expert's knowledge," in *Proc. IEEE Int. Joint. Conf. Neural Netw.*, Nov. 1991, pp. 1588–1593.

[22] H. Ishibuchi, H. Okada, and H. Tanaka, "A neural network with interval weights and its learning algorithm," in *Proc. IJCNN*, 1992, pp. 507–511.

[23] H. Ishibuchi, H. Tanaka, and H. Okada, "An architecture of neural networks with interval weights and its application to fuzzy regression analysis," *Fuzzy Sets Syst.*, vol. 57, no. 1, pp. 27–39, Jul. 1993.

[24] K. Kwon, H. Ishibuchi, and H. Tanaka, "Nonlinear mapping of interval vectors by neural networks," in *Proc. Int. Conf. Neural Netw. (IJCNN)*, Nagoya, Japan, vol. 1, Aug. 2005, pp. 758–761.

[25] M. Song and W. Pedrycz, "Granular neural networks: Concepts and development schemes," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 4, pp. 542–553, Apr. 2013.

[26] M. G. Cimino, B. Lazzerini, F. Marcelloni, and W. Pedrycz, "Genetic interval neural networks for granular data regression," *Inf. Sci.*, vol. 257, pp. 313–330, Feb. 2014.

[27] P. A. Kowalski and P. Kulczycki, "Interval probabilistic neural network," *Neural Comput. Appl.*, vol. 28, no. 4, pp. 817–834, Apr. 2017.

[28] S.-F. Su, C.-C. Chuang, C. W. Tao, J.-T. Jeng, and C.-C. Hsiao, "Radial basis function networks with linear interval regression weights for symbolic interval data," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 42, no. 1, pp. 69–80, Feb. 2012.

[29] S. Scardapane and D. Wang, "Randomness in neural networks: An overview," *WIREs Data Mining Knowl Discovery*, vol. 7, no. 2, p. e1200, Mar. 2017, doi: 10.1002/widm.1200.

[30] Y.-H. Pao, G.-H. Park, and D. J. Sobajic, "Learning and generalization characteristics of the random vector functional-link net," *Neurocomputing*, vol. 6, no. 2, pp. 163–180, Apr. 1994.

[31] B. Igelnik and Y.-H. Pao, "Stochastic choice of basis functions in adaptive function approximation and the functional-link net," *IEEE Trans. Neural Netw.*, vol. 6, no. 6, pp. 1320–1329, Nov. 1995.

[32] R. E. Moore, *Interval Analysis*. Upper Saddle River, NJ, USA: Prentice-Hall, 1966.

[33] O. Taiwo, "The design of robust control systems for plants with recycle," *Int. J. Control*, vol. 43, no. 2, pp. 671–678, Feb. 1986.

[34] D. Wang and M. Li, "Stochastic configuration networks: Fundamentals and algorithms," *IEEE Trans. Cybern.*, vol. 47, no. 10, pp. 3466–3479, Oct. 2017.

[35] D. Wang and M. Li, "Robust stochastic configuration networks with kernel density estimation for uncertain data regression," *Inf. Sci.*, vols. 412–413, pp. 210–222, Oct. 2017.

SHOUPING GUAN received the B.S. and M.S. degrees in automatic control from the Harbin Shipbuilding Engineering Institute, Harbin, China, in 1989 and 1992, respectively, and the Ph.D. degree in control engineering from Northeastern University, Shenyang, China, in 1995. He is currently a Professor with the College of Information Science and Engineering, Northeastern University, Shenyang. His research interests include the practical application of computer control system theory, embedded systems, intelligent control, modeling and optimization of industrial process, and precision measurement and control.

ZIHE ZHANG received the B.S. degree in automation from Northeastern University, Shenyang, China, in 2018, where she is currently pursuing the master's degree in control theory and control engineering. Her research interests include interval neural networks and its application.

ZHOUYING CUI received the B.S. degree in automatic control from Yanshan University, Qinhuangdao, China, in 2017. She is currently pursuing the master's degree in control engineering from Northeastern University, Shenyang, China. Her research interests include interval neural networks and its application.