Hindawi
Mathematical Problems in Engineering
Volume 2021, Article ID 5518950, 16 pages
https://doi.org/10.1155/2021/5518950

Research Article

An Improved Approach for Robust MPC Tuning Based on Machine Learning

Ning He,[1] **Mengrui Zhang**,[1] **and Ruoxia Li**[2]

[1]*School of Mechanical and Electrical Engineering, Xi'an University of Architecture and Technology, Xi'an 710055, China*
[2]*School of Information and Control Engineering, Xi'an University of Architecture and Technology, Xi'an 710055, China*

Correspondence should be addressed to Ruoxia Li; lrx0308@126.com

Received 10 January 2021; Revised 17 April 2021; Accepted 29 April 2021; Published 15 May 2021

Academic Editor: Juan P. Amezquita-Sanchez

A robust tuning method based on an artificial neural network for model predictive control (MPC) of industrial systems with parametric uncertainties is put forward in this work. Firstly, an efficient approach to characterize the mapping relationship between the controller parameters and the robust performance indices is established. As there are normally multiple conflicted robust performance indices to be considered in MPC tuning, the neural network is further used to fuse the indices to produce a simple label representing the acceptable level of the robust performance. Finally, an automated algorithm is proposed to tune the MPC parameters for the considered uncertain system to achieve the desired robust performance. In addition, the regulation of the pH value of the sewage treatment system is used to verify the effectiveness of the robust tuning algorithm which is described in this paper.

1. Introduction

Model predictive control (MPC) has been widely used in industrial communities due to its robustness, ability to tackle safety constraints, and inaccurate model [1, 2]. As we all know, PID control is normally applied at the system's base layer, while the MPC controllers are usually employed at the supervisory layer [3]. In MPC applications, the prediction horizon, control horizon, and weighting matrices in the cost function will significantly affect the closed-loop performance of the controlled system, and thus, the selection of the aforementioned parameters becomes one of the most important tasks for MPC design [4]. As control systems become more and more complex, factors such as input and output coupling, external interference, and time delay make it even more difficult to achieve an effective MPC tuning.

The MPC tuning methods in existing industrial applications are mainly based on engineering experience or numerical methods, which greatly increases the blindness of the controller design and, at the same time, consumes a lot of computation time [5]. Besides, since the model is only an approximation of the real process, it is inevitable to suffer a certain level of uncertainty; robust MPC tuning becomes a necessity [6]. In [7], the authors proposed a robust tuning method of MPC, which is on the basis of the min-max optimization. Such an approach can handle the model uncertainty problem explicitly and, meanwhile, could preclude MPC controllers from choosing large prediction and control horizons so that the online calculation time is reduced. In [8], the authors reduced the number of effective tuning parameters by modifying the controller structure and redesigned the MPC cost function properly. In [9], two robust tuning strategies are put forward for SISO uncertain paper-making processes which incorporated the total variation specification to user-friendly performance indices. In [10], the authors further proposed a rapid tuning strategy based on the closed-loop system structure for MPC parameters for MIMO paper-making system with first-order-plus-dead-time subsystems and uncertain model parameters. In [11], by adopting the sequential procedure, the authors developed a tuning method that took the reachable trajectories of each operating point of the controlled system as the reference to pursue an improved robust performance. Then, they applied the method to the electrical system to

verify its feasibility. In [12], the authors proposed a tuning method using the worst-case control scenario, which is characterized by the Morari resiliency index and the condition number, and a nonlinear multiobjective performance criterion. The resulting constrained nonlinear optimization problem is solved with PSO. In [13], based on the measured data in various operating conditions, a novel approach of the real-time compensation of the asymmetric behavior was investigated, which leads to an improved control performance.

With the rapid development of AI technology lately, researchers have made some attempts in the application of machine learning techniques in the process of MPC tuning. In [14], the authors put forward a framework using machine learning to approximate the tuning experience of human experts along with a gradient-free optimization algorithm to tune the MPC parameters. In [15], the authors proposed an online tuning method for the MPC parameters using particle swarm optimization (PSO) and online sequential extreme learning machine (OS-ELM). In [16], the authors used the artificial neural network for the tuning of FS-MPC for power electronic converters and verified the effectiveness of a practical FS-MPC regulated voltage source converter (VSC) for uninterruptible power supply (UPS) system. In [17], a dynamic system based on the projection neural network (PNN), which is well known for the parallel computational capability, is established to optimize the objective function of MPC; thereby, the computational efficiency is improved significantly which makes the proposed control design more practical.

Note that although there already exist MPC parameter tuning methods using neural network and PSO, none of them focus on the parameter tuning of an uncertain system based on robust time-domain performance indices. As model-plant mismatch is unavoidable in industrial applications, it is of great importance to develop a tuning method to handle this uncertainty. Besides, compared with the frequency-domain indices, their time-domain counterparts are more familiar to the site engineers, and thus, it is also necessary to incorporate the time-domain indices in the MPC tuning.

Given such a new problem, this paper uses the machine learning technique to develop an MPC tuning method to deal with both parametric uncertainty and robust time-domain performance indices. The contribution can be summarized as follows:

(i) A novel approach to characterize the relationship between the MPC parameters and the robust time-domain performance indices, e.g., worst-case overshoot, is established based on RBF neural network. According to the definition of parametric uncertainty and robust performance indices, the parameters of the neural network and the corresponding data acquisition method are also specifically designed.

(ii) As there normally exist conflicts between different time-domain robust indices, it is difficult to specify a suitable target for MPC tuning, and thus, BP neural network is employed to fuse the indices to produce a scalar label representing the acceptable level of the robust performance, such that the MPC tuning problem can be efficiently solved via the PSO algorithm.

The paper is organized as follows. Section 2 describes the structure of the model predictive controller and expresses the tuning problem. Section 3 provides a tuning algorithm for MPC parameters via machine learning. Then, in Section 4, a real system is utilized to verify the effectiveness of the raised algorithm. Finally, the concluding remarks are given in Section 5.

2. Preliminary and Problem Formulation

2.1. Nominal Model and Model Uncertainty. In this paper, we consider multiple-input multiple-output systems which can be expressed by the following discrete-time transfer function model:

$$G(s) = \begin{bmatrix} G_{11}(s) & G_{12}(s) & G_{13}(s) & \dots & G_{1n}(s) \\ G_{21}(s) & G_{22}(s) & G_{23}(s) & \dots & G_{2n}(s) \\ G_{31}(s) & G_{32}(s) & G_{33}(s) & \dots & G_{3n}(s) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ G_{m1}(s) & G_{m2}(s) & G_{m3}(s) & \dots & G_{mn}(s) \end{bmatrix}_{m\times n}, \quad (1)$$

in which $G_{ij}(s)$ demonstrates the transfer function between the i^{th} output and the j^{th} input.

Take the common FOPDT model structure for each subsystem following the industrial experience [9,10], which can be described as follows:

$$G_{pij}(s) = k_{pij}\frac{e^{-\theta_{pij}s}}{\tau_{pij}s+1}, \quad i = 1,2,3,\dots,m,\ j = 1,2,3,\dots,n, \quad (2)$$

where k_p, τ_p, and θ_p denote the process gain, time constant, and time delay. Since $G_{pij}(s)$ is hard to be known accurately, a nominal model $G_{0ij}(s)$ is defined to approach it, which can be expressed as

$$G_{0ij}(s) = k_{0ij}\frac{e^{-\theta_{0ij}s}}{\tau_{0ij}s+1}, \quad i = 1,2,3,\dots,m,\ j = 1,2,3,\dots,n. \quad (3)$$

The model parameters k_0, τ_0, and θ_0 are identified through the input or output data of the real process and are used to predict the state of the MPC controller. However, it is inevitable that $G_{0ij}(s)$ is different from $G_{pij}(s)$, and to consider such a model mismatch, the parametric uncertainty is used, which refers to the difference in the model parameters:

$$\begin{aligned} k_{pij} &\in \left[\underline{k}_{pij}, \overline{k}_{pij}\right], \\ \theta_{pij} &\in \left[\underline{\theta}_{pij}, \overline{\theta}_{pij}\right], \\ \tau_{pij} &\in \left[\underline{\tau}_{pij}, \overline{\tau}_{pij}\right], \end{aligned} \quad (4)$$

where $i = 1, 2, 3, \ldots, m$, $j = 1, 2, 3, \ldots, n$. Note that compared with other types of uncertainty specifications, parametric uncertainty is easier for site engineers to specify based on their knowledge of the controlled system and thus is employed in this work. Based on the considered parametric uncertainty, a set of possible models can be denoted as follows:

$$\prod : = \left\{ G_p(s) :\ k_{pij} \in \left[\underline{k}_{pij}, \overline{k}_{pij}\right], \theta_{pij} \in \left[\underline{\theta}_{pij}, \overline{\theta}_{pij}\right], \tau_{pij} \in \left[\underline{\tau}_{pij}, \overline{\tau}_{pij}\right], \quad | i = 1, 2, \ldots, m, j = 1, 2, \ldots, n \right\}. \tag{5}$$

Furthermore, the state-space model can be obtained as

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k), \\ y(k) &= Cx(k), \end{aligned} \tag{6}$$

where $u \in R$ is the input variable, $y \in R$ is the output variable, x is the state-space vector, and k is the sampling instant.

2.2. MPC Formulation. In industrial applications, the MPC cost function with constraints usually can be designed in the following way:

$$\begin{aligned} J(y(k), u(k)) &= \sum_{i=1}^{P} \left(y(k+i) - y_{ss}\right)^T Q \left(y(k+i) - y_{ss}\right) + \sum_{j=1}^{M} u(k+j-1)^T R u(k+j-1) \\ \text{s.t} y(k+i) &= Cx(k+i) = C\left[A^i x(k) + \sum_{j=1}^{\min\{M,i\}} A^{i-j} B u(k+j-1)\right], \\ y_{\min}(k+i) &\le y(k+i) \le y_{\max}(k+i), \\ u_{\min}(k+j) &\le u(k+j) \le u_{\max}(k+j), \end{aligned} \tag{7}$$

where Q and R are controller parameters to be tuned, y_{ss} is the reference signal of y, and P and M are prediction and control horizons.

Therefore, the predicted output value of the model can be expressed via the following matrix expression:

$$\begin{aligned} Y(k) &= F_x x(0) + F_u U \\ \text{s.t.} Y(k) &= [y(k+1|k) y(k+2|k) \ldots y(k+P|k)]^T, \\ U &= [u(k) u(k+1) \ldots u(k+M-1)]^T, \\ F_x &= \left[CA \ \ CA^2 \ \ CA^3 \ \ \ldots \infty \ \ CA^P \right]^T, \end{aligned} \tag{8}$$

in which T indicates the matrix transpose, and $F_u = \begin{bmatrix} CB & 0 & 0 & \ldots & 0 \\ \sum_{j=1}^{2} CA^{j-1}B & CB & 0 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \sum_{j=1}^{M} CA^{j-1}B & \sum_{j=1}^{M-1} CA^{j-1}B & \ldots & \ldots & CB \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \sum_{j=1}^{P} CA^{j-1}B & \sum_{j=1}^{P-1} CA^{j-1}B & \ldots & \ldots & \sum_{j=1}^{P-M+1} CA^{j-1}B \end{bmatrix}$ and $x(0)$ is the current state vector of the system. Then, the MPC can be represented in the following quadratic programming problem based on which the control signals can be obtained:

$$\begin{aligned} \min J(Y(k), U) = \min & \begin{pmatrix} \left((Y(k) - Y_{ss})^T \operatorname{diag}\{Q, \ldots Q\}_{Pm \times Pm} (Y(k) - Y_{ss}) \right) \\ + \left(U^T \operatorname{diag}(R, \ldots R)_{Mn \times Mn} U \right) \end{pmatrix} \\ & Y_{ss} = [\, y_{ss} \ \ y_{ss} \ \ y_{ss} \ \ \cdots \ \ y_{ss} \,]^T_{Pm \times Pm}, \\ \text{s.t.} \quad & Y_{\min} \le Y \le Y_{\max}, \\ & U_{\min} \le U \le U_{\max}. \end{aligned} \tag{9}$$

2.3. Tuning Problem. In this work, we need to tune Q and R to keep the system robustly stable against the parametric uncertainty and each output tracks its target with a fast and stable response. But there are many contradictions in achieving the target. For example, a small overshoot often causes a large settling time, while a small settling time can be associated with a large overshoot. To make a balance in the process, we need to find the most appropriate set of parameters.

We choose overshoot and settling time as the main performance measures for the MPC tuning because they are simple and well suited for end users to evaluate the control effect. In the future application, other control performance indices can be directly added according to the specific needs following a similar procedure.

3. Robust MPC Tuning Based on Machine Learning

3.1. Controller Tuning Framework Based on Machine Learning. In this paper, an approach for adjusting the aforementioned MPC design parameters for system with parametric uncertainties is developed based on the machine learning technique, the overall framework of which is shown in Figure 1.

3.2. Robust Performance Calculation Based on Machine Learning. Many actual industrial control applications obtain the desired closed-loop performance through a tuning process. That is essentially an optimization problem, which, however, is normally solved by a human (see Figure 2). In this work, we are going to propose a method to achieve the desired controller parameters in a similar way as the human expert. As the MPC cost function captures a cost-benefit relationship between multiple, competing objectives, the performance of the system being considered often cannot be analyzed explicitly. Therefore, a common approach is to approximate it by exploiting the innate human ability to recognize different patterns. More specifically, the human expert acceptance level (denoted as ψ_h) of a given closed-loop performance (denoted as χ) is considered as the tuning objective. Then, the tuning problem can be expressed as

$$\begin{aligned} &\min_{\omega} \ \psi_h(\chi(\omega)) \\ &\text{s.t. } \omega \in \Omega, k_{pij} \in \left[\underline{k}_{pij}, \overline{k}_{pij}\right], \\ &\quad \theta_{pij} \in \left[\underline{\theta}_{pij}, \overline{\theta}_{pij}\right], \tau_{pij} \in \left[\underline{\tau}_{pij}, \overline{\tau}_{pij}\right] \\ &\quad (i = 1, 2, 3, \ldots, m, \ j = 1, 2, 3, \ldots, n), \end{aligned} \tag{10}$$

where $\omega = \begin{bmatrix} Q & R \end{bmatrix}^T$ presents the MPC tuning parameter and Ω indicates a set of admissible controller parameters. $\chi(\omega)$ is the robust performance of the uncertain system given a selected ω.

The explanation of ψ_h is given in Figure 3. For the two curves with the same reference output, the blue curve has the performance we hope to obtain in the process of controller parameter tuning rather than the red curve. So its ψ_h value is less than that of the red curve.

Note that as there is no explicit relationship between ω and ψ_h, machine learning technique is adopted in this work to characterize the robust performance and the corresponding acceptance level. Then, the tuning problem is approximated by

$$\begin{aligned} &\min_{\omega} \psi_{ml}(\Gamma(\chi(\omega))) \\ &\text{s.t. } \omega \in \Omega, k_{pij} \in \left[\underline{k}_{pij}, \overline{k}_{pij}\right], \\ &\quad \theta_{pij} \in \left[\underline{\theta}_{pij}, \overline{\theta}_{pij}\right], \tau_{pij} \in \left[\underline{\tau}_{pij}, \overline{\tau}_{pij}\right], \\ &\quad (i = 1, 2, 3, \ldots, m, \ j = 1, 2, 3, \ldots, n), \end{aligned} \tag{11}$$

where Γ is a feature extractor function which transforms outputs of the system (i.e., time-domain signals) into a vector consisting of relevant performance indices, and ψ_{ml} is an approximation to ψ_h. Then, the tuning framework can be expressed as the diagram shown in Figure 4.

Now, there are three major steps in the tuning of MPC parameters: (i) extract the robust performance of the system with parameter ω, (ii) obtain the human expert's acceptance level of the obtained performance, and (iii) optimize the MPC tuning parameter for the desired performance.

3.2.1. Robust Time-Domain Performance Calculation. Given the perturbed systems in Π, the time-domain robust performance indices are employed to characterize the robust performance as they are more intuitive to end users in the industry. More specifically, the worst-case overshoot and settling time are considered, the definition of which is defined as follows.

Definition 1 (worst-case overshoot). The worst-case overshoot of a set of responses with the same final value is the maximum value of all the responses minus the final value divided by the final value.

Definition 2 (worst-case settling time). The worst-case settling time of a set of responses with the same final value is the maximum time required for all the responses to arrive and stay within a predetermined final percentage range. The calculation of worst-case performance is shown in Algorithm 1.

The illustration of the abovementioned indices is shown in Figure 5.

Note that there may exist a certain relationship between the system model parameters and the robust time-domain performance indices, but such a relationship is implicit and cannot be expressed by a definite formula. Thus, we use an artificial neural network to establish the mapping relationship between the MPC controller parameters and the robust time-domain indices (i.e., Γ in equation (11)) for the system with parametric uncertainty.

Due to the large dimensionality of the inputs and outputs, local approximation network radial basis function (RBF) is selected to ensure that the network has a fast learning convergence speed, and according to the sample size, a standard RBF network or generalized RBF network can be selected. Here, the standard RBF network is taken as an example without a morbid problem.

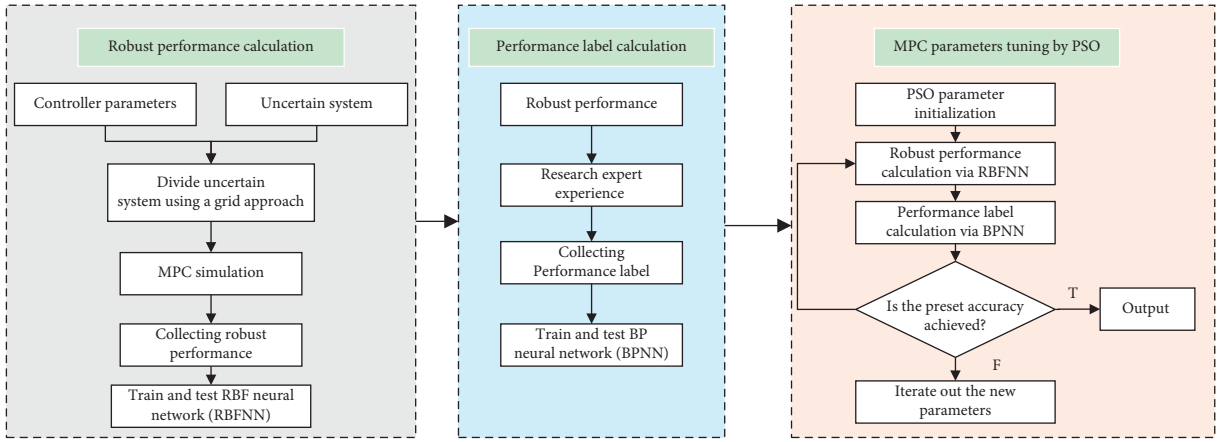


Figure 1: The overall framework of the proposed tuning method.

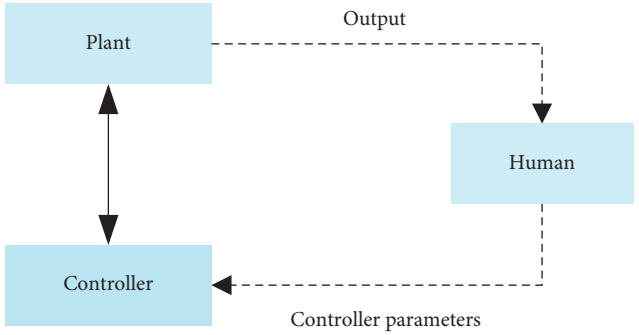


Figure 2: Controller tuning framework with a human in the loop.

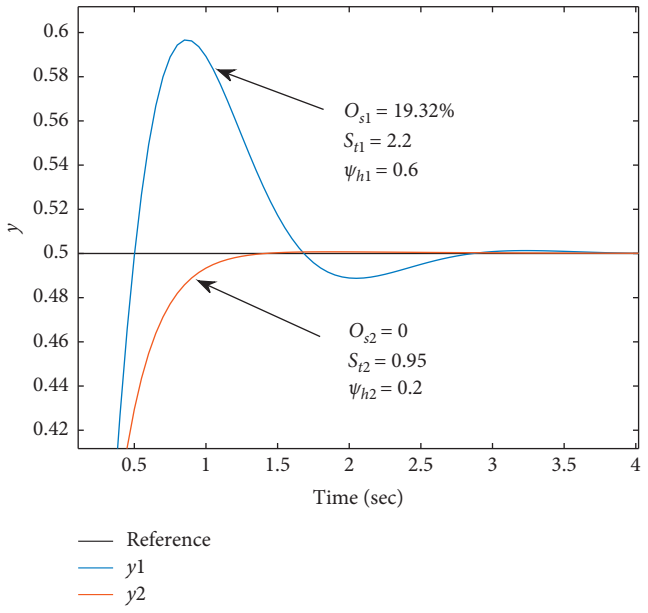


Figure 3: The illustration of the performance label.

In this paper, each sample of the RBF neural network contains a set of controller parameters q and r and the corresponding robust time-domain performance indices of uncertain system. In practice, if the overshoot or settling time is too big, such control is considered meaningless. In order to ensure the representativeness of the samples, the grid method is used to sample the performance parameters within the acceptable range, and then, the corresponding controller parameters are derived as the training database of the RBF network. If each index takes Δ numbers within its allowable range, there are $2m\Delta$ training set samples in total. Finally, Δ^* groups of controller parameters and their corresponding robust performance indices are randomly generated as test sets. If the training set samples cannot reach the required network accuracy, the segmentation of the above interval can be further refined.

Note that every group of worst-case performance indices needs 2^d (d is the number of uncertain parameters in the system model) curves to generate a reasonable result. Considering the requirement of the BPNN in this work, the $2m\Delta$ groups of datasets are obtained when training RBF network is directly utilized, and therefore, it is necessary to have at least $2m\Delta \times 2^d$ curves to generate a reasonable result. The reason to consider 2^d curves is that in order to characterize the worst-case performance, the polyhedron system representation [3] in robust control theory is employed, which indicates that the worst performance of the uncertain system mostly appears at the vertex system of the polyhedron system, and therefore, the largest and smallest possible values of each model parameter of the uncertain system need to be considered, resulting in 2^d curves for each group of robust indices. Note that, compared with the existing method to evaluate the worst-case time-domain performance (e.g., brutal search method), the aforementioned method is much simpler, since the required number of curves is significantly reduced, which helps to achieve the network in a more efficient way.

The input layer of the RBF network has two inputs, which represent the two parameters q and r of the controller, respectively. The number of neurons in the hidden layer is $2m\Delta$, which is the same as the number of samples in the training set; the number of neurons in the output layer is $2m$, which represents the worst-case dynamic time-domain performances of the model uncertain system. Gaussian radial basis function is used in the network.

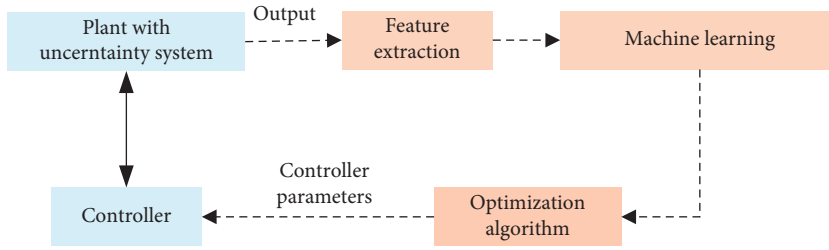


Figure 4: Controller tuning framework with machine learning in the loop.

(1) Input: the uncertainty intervals $k_{pij} \in [\underline{k}_{pij}, \overline{k}_{pij}]$, $\theta_{pij} \in [\underline{\theta}_{pij}, \overline{\theta}_{pij}]$, $\tau_{pij} \in [\underline{\tau}_{pij}, \overline{\tau}_{pij}]$, nominal system $G_{0ij}(s)$ $(i = 1, 2, 3 \ldots m, j = 1, 2, 3 \ldots n)$, controller parameter set $Q_{\text{data}}(\delta)$ and $R_{\text{date}}(\delta)$ $(\delta = 1, 2, 3 \ldots 2m\Delta)$, output reference Y_{ref};
(2) Output: ρ;
(3) Divide the uncertain interval of k_{pij}, θ_{pij}, τ_{pij} into η equal parts;
(4) for $\delta = 1{:}\ 1{:}\ 2m\Delta$ do
(5) for $l = 1{:}\ 1{:}\ 2^d$ do
(6) Transform the transfer function ($k_{pij}(l)$, $\theta_{pij}(l)$, $\tau_{pij}(l)$) into state space ($A(l)$, $B(l)$, $C(l)$);
(7) $U(k) \leftarrow f_{\text{MPC}}(Y_{\text{ref}}, Q(\delta), R(\delta), G_0(s))$;
(8) $x(k+1) \leftarrow A(l)x(k) + B(l)u(k)$;
(9) $y(k) \leftarrow C(l)x(k)$;
(10) $O_s(l) \leftarrow ((\max(y(k)) - y_{\text{ref}})/y_{\text{ref}})$;
(11) $S_t(l) \leftarrow \min k(|y - y_{\text{ref}}| \le 2\% y_{\text{ref}})$;
(12) end for
(13) $\rho(\delta) \leftarrow [\max(0_s), \max(S_t)]$;
(14) end for

Algorithm 1: Calculation of the worst-case performance indices.

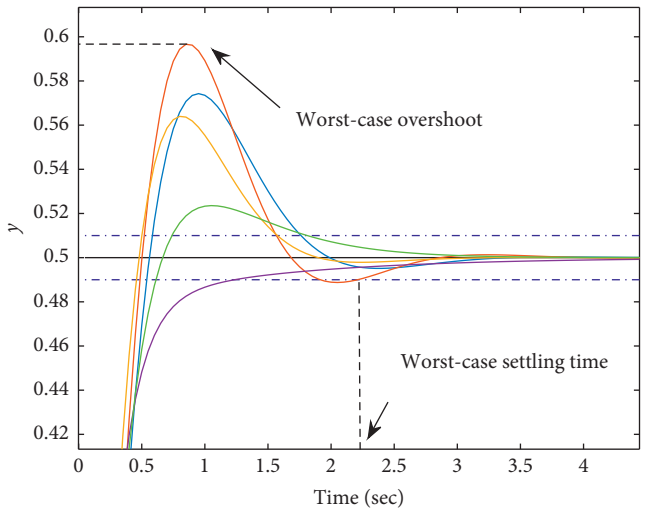


Figure 5: The definition of the considered worst-case performance indices.

The training of RBF neural network needs 4 steps: (1) preprocessing; (2) data standardization; (3) determination of the center vector and standardized constant of Gaussian function; and (4) obtaining the weight matrix of output layer by the recursive least square method. More specifically, the output function is given by

$$\rho^* = \sum_{\delta=1}^{\Delta} \varepsilon_\delta f_{\text{RBF}}(p_\delta - c_i), \tag{12}$$

in which ρ^* is output variables, ρ_δ is output variables $(\delta = 1, 2, 3 \ldots 2m\delta)$, ε_δ is the weight from hidden layer node to output layer, and f_{RBF} is the basis function of RBF network. The basis function of the RBF network in this paper is the Gaussian function. The specific expression is as follows:

$$f_{\text{RBF}}(p_\delta - c_i) = e^{-(1/2\sigma^2)\|p_\delta - c_i\|}, \tag{13}$$

in which σ^2 is the variance of the Gaussian function and c_i is the center of the Gaussian function.

3.2.2. Performance Label Calculation. As there may exist a conflict between different robust performance indices, we employ a performance label ψ_h to characterize the acceptance of a given pair of robust indices based on the experience of human experts. Note that although different expert's ψ_h is likely to be personalized to some extent, the codomain of each human cost function to achieve ψ_h is a set of elements representing the quantities' assessment of the relevant data. In this work, these data consist of the worst-case time-domain robust performance indices of the system, while the codomain elements are the nonnegative real numbers, referred to as performance labels.

For the purpose of demonstration, the performance label belongs to the interval [0, 1], capturing the "acceptable" level of the closed-loop performance. More specifically, label 0 denotes the best possible performance while a label greater than 0 denotes worsen performance, and 1 means the least "acceptable" performance. Naturally, any label greater than 1 captures the "unacceptable" closed-loop performance. The label assignments corresponding to these worst-case robust performance indices are obtained by collecting expert experience.

Given the dataset above, approximating the human cost function is a typical supervised learning problem. Note that there may exist a difference between the label values from each expert because he or she may have different preferences and concerns about characterizing the performance. Therefore, the regression method is used to approximate the human cost function. In this work, BP neural network is used to establish the mapping relationship between the system performance indices and the label.

The training data of the BP network are obtained by investigation. First of all, we generate a number of output curves to illustrate the considered robust time-domain indices (e.g., Figure 5), and these outputs are given performance labels ψ by experts.

Then, the required BP neural network can be established. The number of nodes in the input layer is $2m$, which is decided by the output dimension of the system. The output layer has just one node, indicating the value of the label. We chose the number of nodes in the hidden layer by an empirical formula $\sqrt{2m+1}+\partial$, where the value range of ∂ is usually $[1, 2, 3 \ldots 10]$.

There are $2m\Delta$ inputs and output training sample vectors are represented by x_δ and x_δ, respectively $(\delta = 1, 2, 3 \ldots \delta)$. The input vector is $x_\delta = (x_{\delta 1}, \ldots, x_{\delta \varrho})^T, \varrho = 1, \ldots, \mathrm{I}_{\mathrm{BP}}$, the output vector of the network is $o_\delta = (o_{\delta 1}, \ldots, o_{\delta \varrho})^T$, $\varrho = 1, \ldots, \mathrm{J}_{\mathrm{BP}}$, and the target output vector is g_δ. Note $w_{\varrho\epsilon}$ is the weight of the ϱ component of the input vector mapped to the ϵ component of the output vector, which is randomly allocated in the first calculation. BP network modifies weight $w_{\varrho\epsilon}$ by the gradient steepest descent method through the feedback of output result, so that the sum of square error between output value and the target value is minimum, that is, (14). Repeat until the error is less than the set threshold.

$$\min \frac{1}{2} \sum_{\epsilon=1}^{J_{\mathrm{BP}}} \left(o_{\delta\epsilon} - g_{\delta\epsilon}\right)^2, \tag{14}$$

$$\Delta w_{\varrho\epsilon} = \sum_{\varrho=1}^{I_{\mathrm{BP}}} \eta_{\mathrm{BP}} \left(o_{\delta\epsilon} - g_{\delta\epsilon}\right) x_{\delta\varrho}, \tag{15}$$

in which η_{BP} is the learning rate of the network. The activation function of the BP network is the sigmoid function, i.e.,

$$\mathrm{sigmoid}(z) = \frac{1}{1+e^{-z}}. \tag{16}$$

3.3. Tuning Process. As mentioned above, using the two trained neural networks, the performance label ψ^* can be quickly obtained in the process of MPC parameter optimization. The specific algorithm for robust performance labeling is shown in Algorithm 2.

Based on Algorithm 2, an effective robust MPC tuning method is proposed in this section, which can be summarized in Algorithm 3.

Interpretations: the position μ of each particle contains all the information of Q and R matrices. The greater the particle swarm size N is, the larger the search range will be, the easier the global optimal solution would be obtained, but the longer the corresponding running time will be required. λ^* is the maximum iteration number of particles, and ξ is the precision of the optimization target. c_1, c_2, and W are the local speed, global speed, and flight acceleration of particles, respectively. The faster the particles fly, the faster the optimization speed is, but it is also easier to miss the optimal location. The final pg^* is the optimal control parameters of the MPC given the considered model uncertainty; $\psi^*(pg^*)$ is the corresponding performance label.

4. Industrial Example

This section applies the developed new tuning algorithm to the actual application of the pH adjustment process of sewage treatment system shown by Figure 6 to illustrate the efficiency of the tuning algorithms. The pH neutralization process system of sewage is composed of inlet wastewater flow, buffer fluid flow, and acid neutralizer flow in the neutralization tank to obtain the height of wastewater flow at the outlet and the liquid level of the storage tank. Among them, the flow rate of acid neutralizer flow and the flow rate of sewage flow at the inlet are taken as control variables, and the pH value of sewage flow at the output port and the height of the liquid level in the storage tank are taken as the output quantities. The model predictive controller is used, and the proposed method is employed to tune the relevant parameters of the controller to achieve the purpose of adjusting the pH value of sewage.

The process can be characterized by a two-input two-output system:

$$G_p(s) = \begin{bmatrix} k_{11}\dfrac{e^{-\theta_1 s}}{\tau_1 s+1} & k_{12}\dfrac{e^{-\theta_1 s}}{\tau_1 s+1} \\ \dfrac{k_2}{\tau_2 s+1} & \dfrac{k_2}{\tau_2 s+1} \end{bmatrix}. \tag{17}$$

Considering potential model-plant mismatch, the real model parameters are considered to be within the following ranges:

$$\begin{aligned} k_{11} &\in [-0.65, -0.3], \\ k_{12} &\in [0.3, 0.65], \\ k_2 &\in [0.89, 1.1], \\ \tau_1 &\in [85, 95], \\ \tau_2 &\in [180, 210], \\ \theta_1 &= 30. \end{aligned} \tag{18}$$

(1) Input: controller parameters Q and R;
(2) Output: worst-case performance label ψ^* of the uncertainty system;
(3) Obtain the worst-case system robust performance matrix ρ^* with the control of MPC via RBFNN;
(4) Obtain the performance label ψ^* of the matrix ρ^* via BPNN.

Algorithm 2: Performance label calculation based on machine learning.

(1) Input: the uncertainty intervals $k_{pij} \in [\underline{k}_{pij}, \overline{k}_{pij}]$, $\theta_{pij} \in [\underline{\theta}_{pij}, \overline{\theta}_{pij}]$, $\tau_{pij} \in [\underline{\tau}_{pij}, \overline{\tau}_{pij}]$, and nominal system $G_{0ij}(s)$ $(i = 1, 2, 3 \ldots m, j = 1, 2, 3 \ldots n)$;
(2) Output: the optimal tuning results pg^*;
(3) Initializing PSO optimizer parameters: $N \leftarrow 2m\Delta$, $c_1 \leftarrow 2$, $c_2 \leftarrow 2$, $w \leftarrow 0.2$, narvs$\leftarrow m^2 + n^2$, $t \leftarrow 0$, $\lambda^* \leftarrow 50$, $\xi^* = 0.1$;
(4) Initializing the position μ and flying speed v of the particles;
(5) $t \leftarrow 0$, $\psi^*(pg^*) \leftarrow \infty$;
(6) while $t \le \lambda^*$ or $\psi^*(pg^*) \le \xi^*$ do
(7) $t \leftarrow t + 1$;
(8) for $e = 1{:}\ 1{:}\ N$ do
(9) $\psi^*(e) \leftarrow$ Algorithm 1 $(\psi^*(\mu))$;
(10) if $\psi^*(e) \le \psi^*(pi(e))$ then
(11) $pi(e) \leftarrow \mu(e)$;
(12) end if
(13) $pg^* = \min(pt)$;
(14) if $\psi^*(pg^*) \le \psi^*(pg)$ then
(15) $pg \leftarrow pg^*$;
(16) end if
(17) $v = w \times v + c_1 \text{rand} \times (pi(e) - \mu) + c_2 \text{rand} \times (pg - \mu)$;
(18) $\mu = \mu + v$;
(19) end for
(20) end while

Algorithm 3: Robust tuning of MPC based on machine learning.

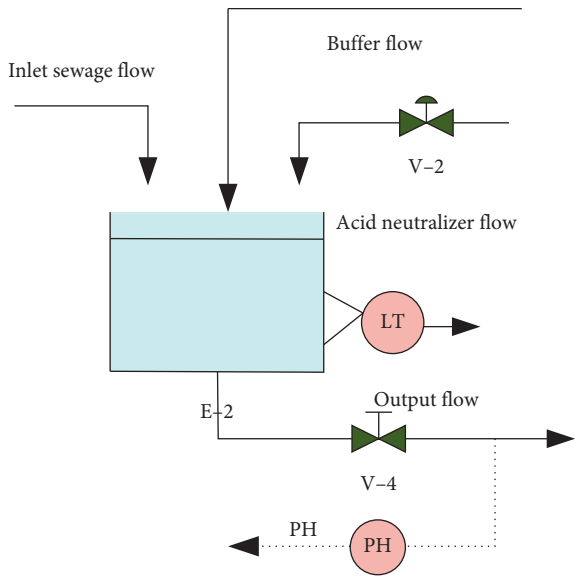


Figure 6: Schematic diagram of the sewage treatment system.

The nominal system considered is shown as follows:

$$\begin{aligned} k_{110} &= -0.4, \\ k_{120} &= 0.4, \\ k_{20} &= 1, \\ \tau_{10} &= 90, \\ \tau_{20} &= 200, \\ \theta_{10} &= 30. \end{aligned} \tag{19}$$

Note that the model is identified via an advanced industrial control software package for the use of MPC.

The prediction and control horizons are set to $P = M = 5$, and the initial operating conditions are as follows:

$$\begin{aligned} y(0) &= [0, 0]^T, \\ u(0) &= [0, 0]^T, \end{aligned} \tag{20}$$

and the references in (8) are $y_{ss} = \begin{bmatrix} y_{ss\text{ph}} \\ y_{ss\text{level}} \end{bmatrix} = \begin{bmatrix} 6 \\ 15 \end{bmatrix}$. The tuning parameters for the considered MPC in (8) are Q and R, where

$$\begin{aligned} Q &= \begin{bmatrix} q_{11} & q_{12} \\ q_{21} & q_{22} \end{bmatrix}, \\ R &= \begin{bmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{bmatrix}. \end{aligned} \tag{21}$$

For illustration purpose, the weighting matrix is simplified as follows:

$$\begin{aligned} Q &= q \times E, \\ R &= r \times E. \end{aligned} \tag{22}$$

In our experiment, according to the requirements of the actual system for these indices, it is divided into $\Delta = 10$ performance intervals in the range of change, and a sample is selected in each interval, a total of $10 \times 4 = 40$ samples as the training set. Then randomly select $\Delta^* = 15$ groups from the remaining samples as the test set. And parameters of supervised machine learning are set as follows:

$$\begin{aligned} \Delta &= 20, \\ d &= 5, \\ \partial &= 5. \end{aligned} \tag{23}$$

The root mean square error (RMSE) which can reflect the prediction stability of networks is introduced to evaluate the prediction performance of the network.

$$\text{RMSE} = \sqrt{\frac{1}{\Delta^*} \sum_{\delta=1}^{\Delta^*} \left(H_r - H_p\right)^2}, \tag{24}$$

in which H_r is the actual output and H_p is the reference output of the network. In our experiment, the training accuracy of the RBF network and BP network is 0.001, and the error curves from the testing are shown in Figures 7–8. The RMSE of the RBF network and BP network is 0.8073 and 0.0951, respectively. Note that although the obtained error value of the RBF network is a bit high, such accuracy is acceptable for the robust tuning problem at hand because, given the high level of model uncertainty considered in this work, an average error of 0.1448 for worst-case overshoot and worst-case settling time would not affect the overall tuning performance from a robust control point of view. Furthermore, as the number of data samples of the industrial control system is normally limited, it is reasonable that the construction of the network is stopped once the accuracy meets the design requirement. As for the BP network, since the relationship between the robust indices and performance label is relatively simple, the accuracy becomes higher as shown in Figure 7.

Then, the effectiveness of the proposed robust time-domain performance calculation method is tested and the results are shown in Figure 9, in which the red curves indicate the worst-case performance obtained by the RBF network from Algorithm 1, and the blue curves indicate the outputs generated with system parameters randomly selected in the region shown in equation (13). More specifically, the robust time-domain performance from the brutal force search is $\begin{bmatrix} 0.11 & 0.0602 & 33 & 28 \end{bmatrix}^T$, while that from the proposed network is $\begin{bmatrix} 0.1477 & 0.0756 & 33.9222 & 29.1417 \end{bmatrix}^T$.

(1) Input: the uncertainty intervals $k_{pij} \in [\underline{k}_{pij}, \overline{k}_{pij}]$, $\theta_{pij} \in [\underline{\theta}_{pij}, \overline{\theta}_{pij}]$, $\tau_{pij} \in [\underline{\tau}_{pij}, \overline{\tau}_{pij}]$, nominal system $G_{0ij}(s)$ $(i = 1, 2, 3 \ldots m, j = 1, 2, 3 \ldots n)$, controller parameter set $Q_{\text{data}}(\delta)$ and $R_{\text{date}}(\delta)$ $(\delta = 1, 2, 3 \ldots 2m\Delta)$, output reference Y_{ref};

(2) Output: ρ;

(3) Divide the uncertain interval of k_{pij}, θ_{pij}, τ_{pij} into η equal parts;

(4) for $\delta = 1{:}\ 1{:}\ 2m\Delta$ do

(5) for $l = 1{:}\ 1{:}\ 2^d$ do

(6) Transform the transfer function $(k_{pij}(l), \theta_{pij}(l), \tau_{pij}(l))$ into state space $(A(l), B(l), C(l))$;

(7) $U(k) \leftarrow f_{\text{MPC}}(Y_{\text{ref}}, Q(\delta), R(\delta), G_0(s))$;

(8) $x(k+1) \leftarrow A(l)x(k) + B(l)u(k)$;

(9) $y(k) \leftarrow C(l)x(k)$;

(10) $O_s(l) \leftarrow ((\max(y(k)) - y_{\text{ref}})/y_{\text{ref}})$;

(11) $S_t(l) \leftarrow \min k(|y - y_{\text{ref}}| \le 2\% y_{\text{ref}})$;

(12) end for

(13) $\rho(\delta) \leftarrow [\max(0_s), \max(S_t)]$;

(14) end for

Now, we test the proposed tuning method. Figure 10 shows the change of the performance label while Algorithm 2 is searching for the optimal MPC parameters. With the PSO iteration process going on, the label value decreases until it converges. The solution is 0.3426 and the corresponding tuning results are $q = 8.57$ and $r = 4.54$, and the optimization time is 23 s; Figure 11 shows the optimal control parameters obtained by the brutal search. The global optimal solution is $q = 8.57$ and $r = 4.55$, the corresponding

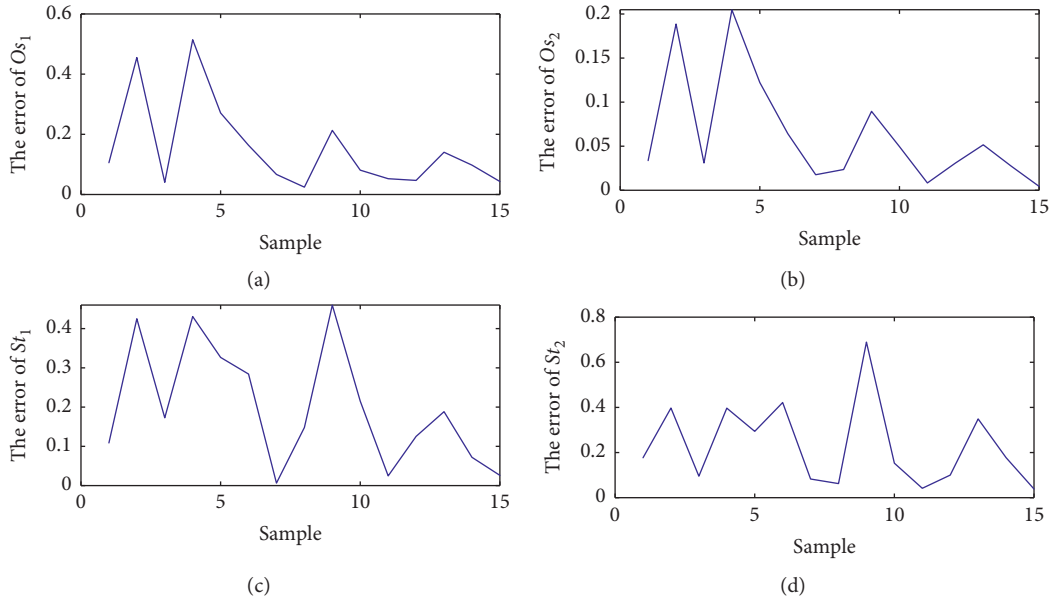


Figure 7: The error curves of the RBF network.

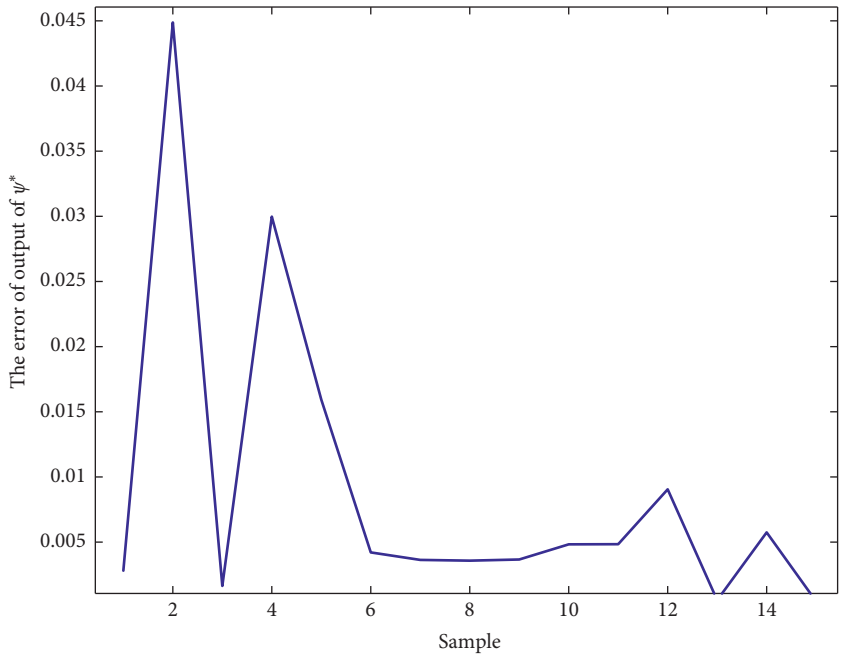


Figure 8: The error curves of the BP network.

label level value is 0.3450, and the optimization time is 1.18h. Compared with the brute force search method, the tuning method based on machine learning requires only 0.5% of the running time which can still obtain a similar tuning result, which verifies the effectiveness of the method described in the invention.

In order to further verify the effectiveness of the method proposed in this paper, we selected three groups of controller parameters according to tuning guides utilized in industry and compared them with those obtained by Algorithm 3, and the tuning parameters of each case are shown in Table 1. Assume that the actual parameters of the system are $k_{11} = -0.55$, $k_{12} = 0.55$, $k_2 = 1.05$, $\tau_1 = 90$, $\tau_2 = 194$, and $\theta_1 = 30$, which are different from the nominal system in equation (14).

The corresponding performance indices and corresponding ψ^* of each group of controller parameters are shown in Table 2. It can be seen that the controller parameters corresponding to group (d), that is, the parameters obtained through the algorithm described in this paper, have the best control effect for the uncertain system.

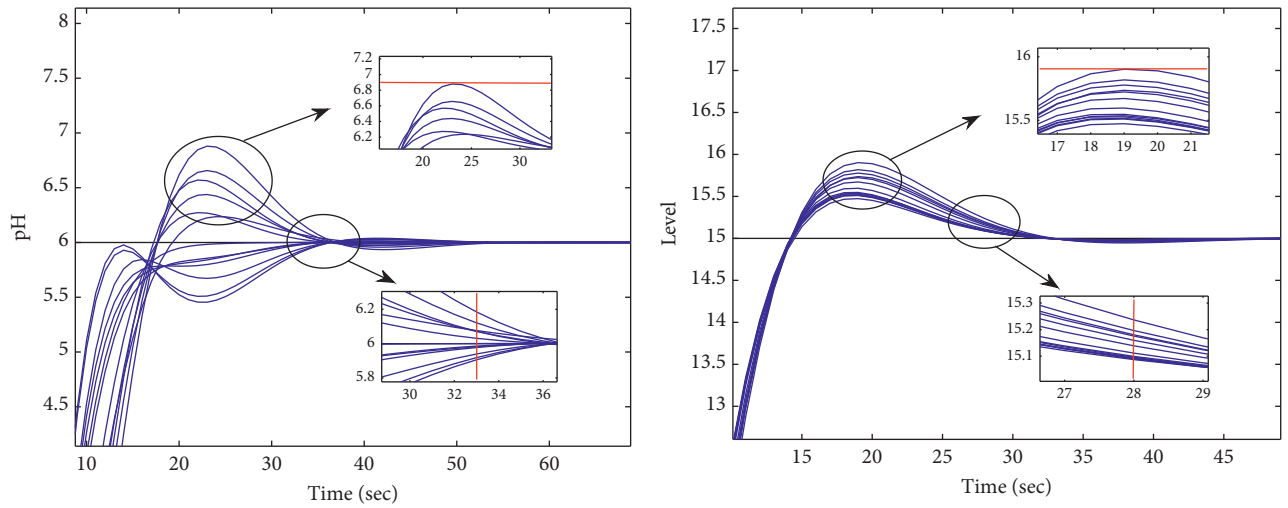


Figure 9: The effectiveness verification of the RBF-based robust performance calculation method.

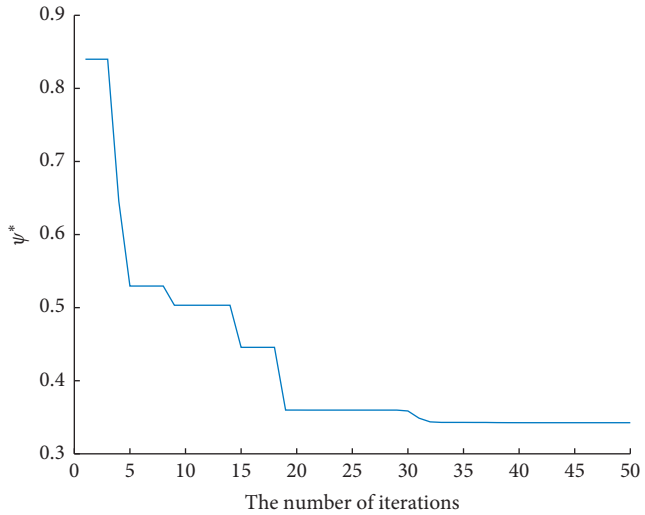


Figure 10: The optimization results of Algorithm 3.

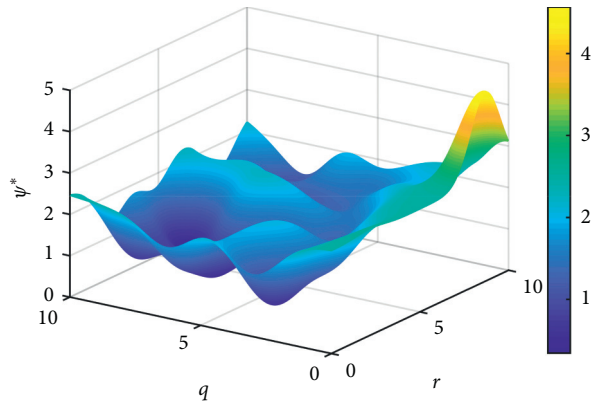


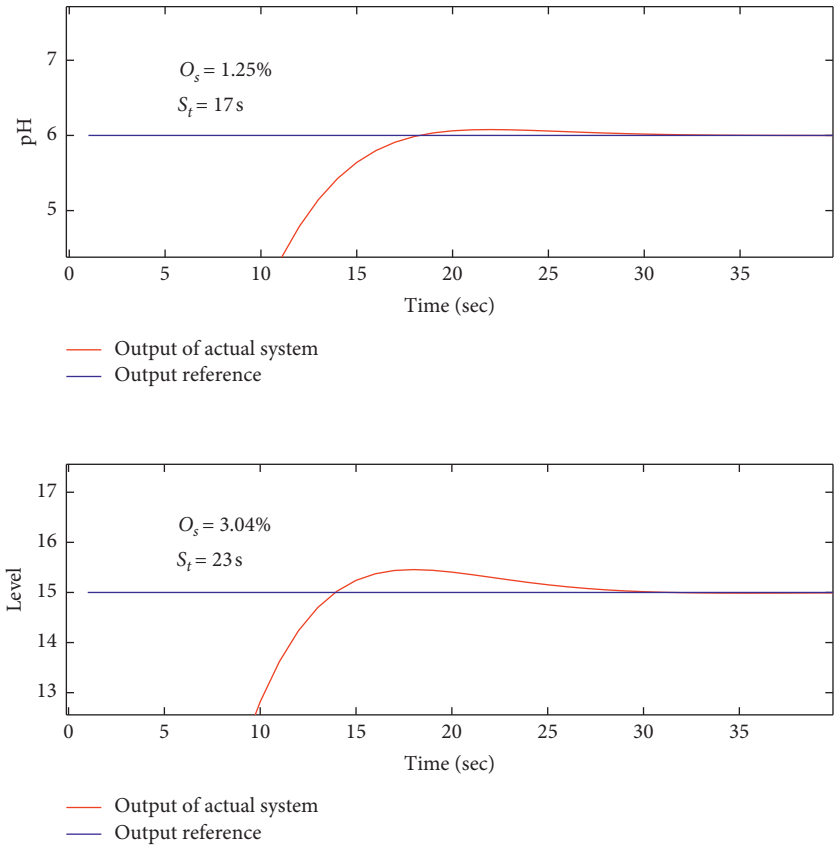
Figure 11: The optimization results of the brutal search method.

Table 1: Parameters of MPC.

	Q	R
(a)	7.22	8.36
(b)	4.56	9.27
(c)	5.79	3.99
(d)	8.57	4.54

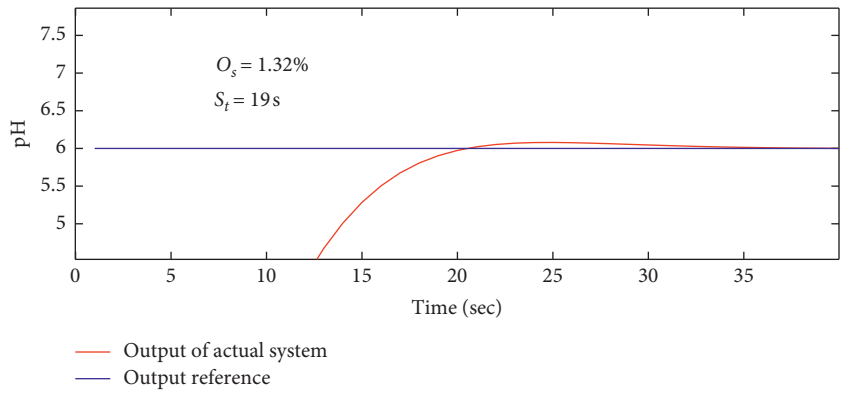
Table 2: Performance indices of uncertain system.

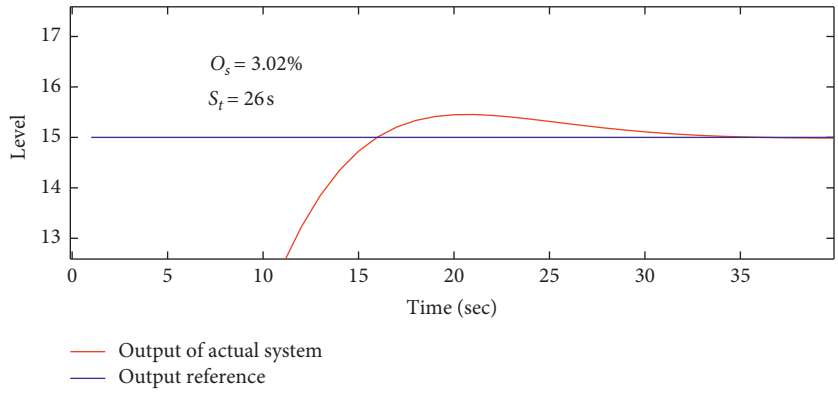
	O_s (pH) (%)	O_s (level) ((%)	S_t (pH) (s)	S_t (level) (s)	ψ^*
(a)	1.25	3.04	17	23	0.4849
(b)	1.32	3.02	19	26	0.4354
(c)	1.22	3.03	16	20	0.3594
(d)	1.17	3.02	15	19	0.3574



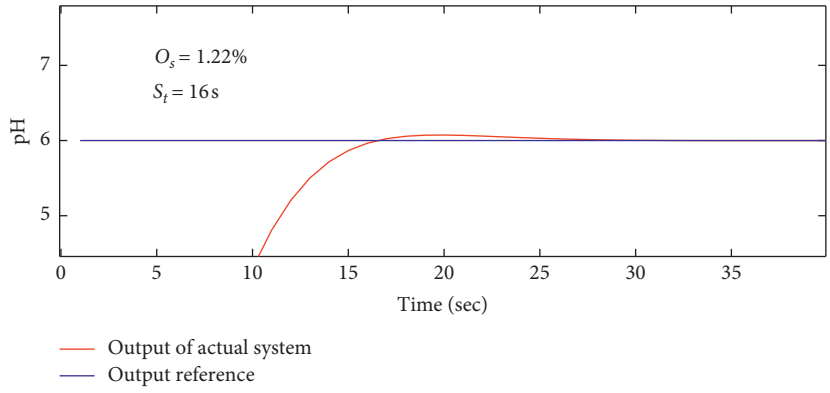
(a)

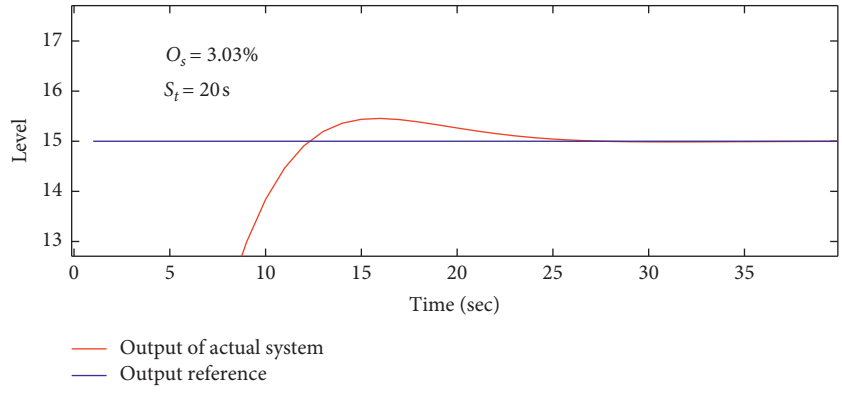
Figure 12: Continued.





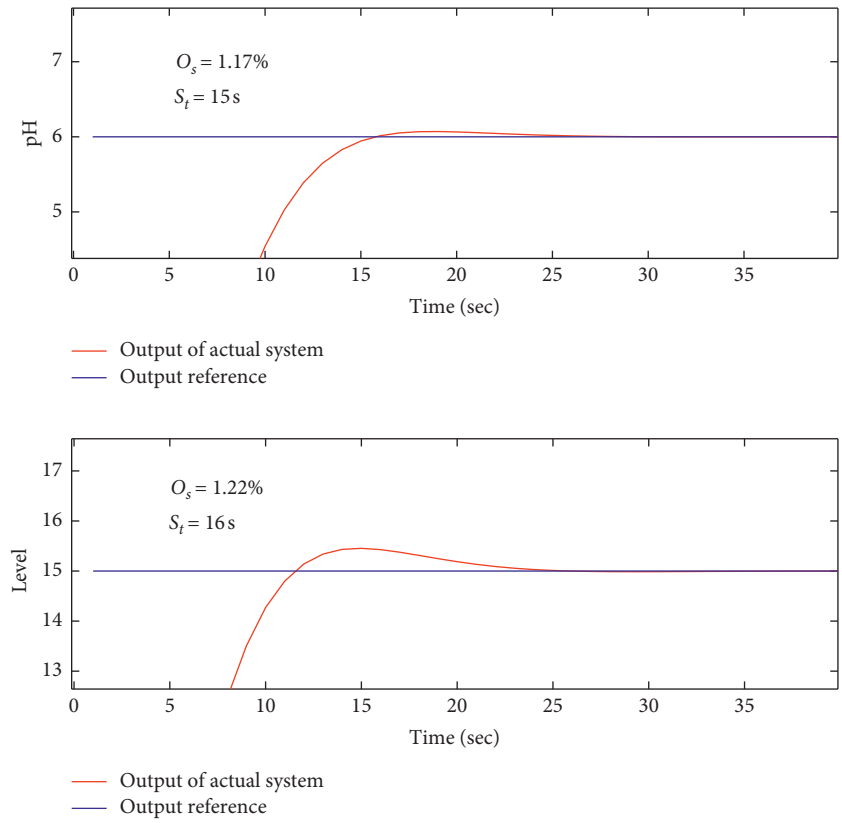
(b)





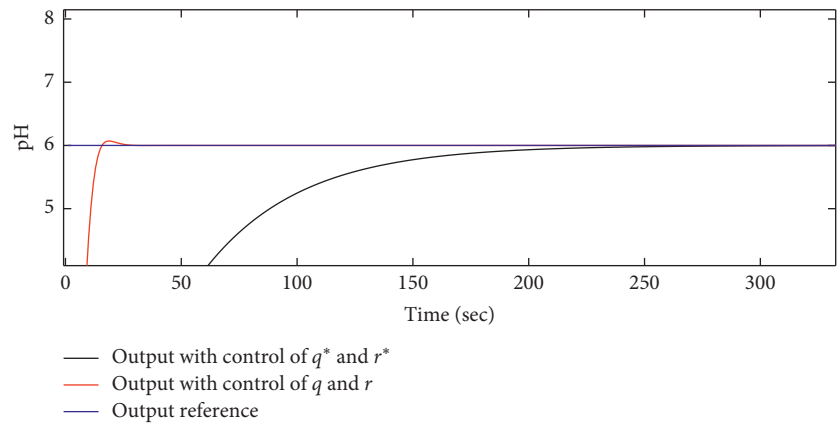
(c)

Figure 12: Continued.



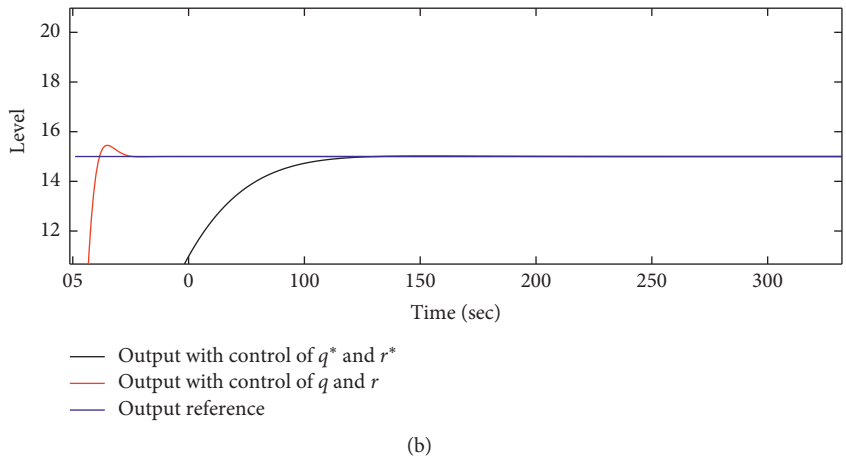
(d)

Figure 12: Uncertain system outputs under different controller parameters. (a) $\psi^* = 0.4849$. (b) $\psi^* = 0.4354$. (c) $\psi^* = 0.3594$. (d) $\psi^* = 0.3574$.



(a)

Figure 13: Continued.



(b)

Figure 13: The comparison between the method proposed in this paper and reference [14].

In order to intuitively display the control effect of each group of controller parameters in Table 1, the output response curves are shown in Figure 12.

As far as we know, this is the first work to use the machine learning technique to solve the MPC tuning problem considering parametric uncertainty and robust time-domain indices. Therefore, we can only compare with the most similar literature, [14] of this manuscript, which is a machine learning-based MPC tuning method developed for systems with no model uncertainty. According to [14], the MPC parameters are $q^* = 1.18$ and $r^* = 8.43$ when the controller is adjusted based on the nominal system without considering the model uncertainty. Assume that the actual model parameters of the system are

$$\begin{aligned} k_{11} &= -0.60, \\ k_{12} &= 0.44, \\ k_2 &= 1.05, \\ \tau_1 &= 93, \\ \tau_2 &= 197, \\ \theta_1 &= 30, \end{aligned} \tag{25}$$

and the nominal system is expressed by equation (19). Two groups of controllers are applied to the system, respectively, and the output image of the system is shown in Figure 13. The black curve represents the system output under the control of q^* and r^*, and its dynamic time-domain performance indices are overshoot and settling time of two outputs of the system, i.e., $[1.17\% \ 3.02\% \ 40s \ 35s]^T$. The red curve represents the system output under the control of the controller parameters $q = 8.57$ and $r = 4.54$ which tuned based on the worst-case time-domain performance indices of uncertain system that discussed in this paper, and its dynamic time-domain performance indices are $[1.08\% \ 2.86\% \ 22s \ 18s]^T$. It can be seen that the method proposed in this paper greatly enhances the robustness of parameter tuning of model predictive control for uncertain systems.

5. Conclusion

In this paper, an artificial neural network-based model predictive control (MPC) parameter tuning method for uncertain systems is proposed, which could solve the problem of low control accuracy caused by model mismatch and external disturbance effectively, so as to improve the robustness of the controlled system. To achieve such an objective, a novel method to compute the worst-case output performance in the time domain is developed through machine learning, and the potentially conflicted time-domain robust performance indices are transformed into a scalar performance label via a neural network formed based on expert's experience. At the same time, the parallel search ability of PSO is adopted to research the optimal tuning parameters efficiently. Finally, the method is verified in the process of pH regulation in the sewage treatment system.

At the present stage, the potential shortcoming of the method is that the considered two neural networks can only be obtained offline, and for some industrial systems, it is more desirable that the offline training stage can be avoided and the learning can be realized in an online manner. Thus, the main future research direction is to employ the online learning method to develop an efficient robust tuning algorithm considering model uncertainty as well as robust time-domain performance.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by NSFC of China (Grant no. 61903291), Special Scientific Research Project of Shaanxi Provincial Department of Education (19JK0459), and Basic Research Plan in Shaanxi Province of China (2020JQ-683).

References

[1] S. Zhang, H. Cao, Y. Zhang, L. Jia, Z. Ye, and X. Hei, "Data-driven optimization framework for nonlinear model predictive control," *Mathematical Problems in Engineering*, vol. 2017, Article ID 9402684, 15 pages, 2017.

[2] A. Garg, H. A. Abdulhussain, P. Mhaskar, and M. R. Thompson, "Handling constraints and raw material variability in rotomolding through data-driven model predictive control," *Processes*, vol. 7, no. 9, pp. 610–624, 2019.

[3] D. Shi, J. Wang, M. Forbes, J. Backström, and T. Chen, "Robust tuning of machine directional predictive control of paper machines," *Industrial & Engineering Chemistry Research*, vol. 54, no. 15, pp. 3904–3918, 2015.

[4] N. He, Y. Jiang, and L. He, "Analytical tuning method of mpc controllers for mimo first-order plus fractional dead time systems," *Processes*, vol. 8, no. 2, pp. 212–226, 2020.

[5] X. Yu-Geng, L. De-Wei, and L. Shu, "Model predictive control—status and challenges," *Acta Automatica Sinica*, vol. 39, no. 3, pp. 222–236, 2013.

[6] P. Bumroongsri and S. Kheawhom, "Off-line robust constrained mpc for linear time-varying systems with persistent disturbances," *Mathematical Problems in Engineering*, vol. 2014, Article ID 936093, 8 pages, 2014.

[7] K. Han, J. Zhao, and J. Qian, "A novel robust tuning strategy for model predictive control,"vol. 2, pp. 6406–6410, in *Proceedings of the 2006 6th World Congress on Intelligent Control and Automation*, vol. 2, pp. 6406–6410, IEEE, Dalian, China, June 2006.

[8] G. S. Sankar, R. C. Shekhar, C. Manzie, T. Sano, and H. Nakada, "Fast calibration of a robust model predictive controller for diesel engine airpath," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 4, pp. 1510–1519, 2019.

[9] N. He, D. Shi, J. Wang, M. Forbes, J. Backström, and T. Chen, "User friendly robust MPC tuning of uncertain paper-making processes," *IFAC-PapersOnLine*, vol. 48, no. 8, pp. 1021–1026, 2015.

[10] N. He, D. Shi, M. Forbes, J. Backström, and T. Chen, "Robust tuning for machine-directional predictive control of mimo paper-making processes," *Control Engineering Practice*, vol. 55, pp. 1–12, 2016.

[11] J. E. W. Santos, J. O. Trierweiler, and M. Farenzena, "Robust tuning for classical mpc through the multi-scenarios approach," *Industrial & Engineering Chemistry Research*, vol. 58, no. 8, pp. 3146–3158, 2019.

[12] G. N. Júnior, M. Martins, and R. Kalid, "A pso-based optimal tuning strategy for constrained multivariable predictive controllers with model uncertainty," *Isa Transactions*, vol. 53, no. 2, pp. 560–567, 2014.

[13] J. Oravec, M. Horváthová, and M. Bakošová, "Multivariable robust mpc design for neutralisation plant: experimental analysis," *European Journal of Control*, vol. 58, pp. 289–300, 2021.

[14] A. S. Ira, C. Manzie, I. Shames et al., "Tuning of multivariable model predictive controllers through expert bandit feedback," *International Journal of Control*, vol. 12, pp. 1–9, 2020.

[15] H. Moumouh, N. Langlois, and M. Haddad, "A novel tuning approach for mpc parameters based on artificial neural network," in *Proceedings of the 2019 IEEE 15th International Conference on Control and Automation (ICCA)*, pp. 1638–1643, IEEE, Edinburgh, Scotland, July 2019.

[16] P. T. Jardine, S. Givigi, and S. Yousefi, "Parameter tuning for prediction-based quadcopter trajectory planning using learning automata," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 2341–2346, 2017.

[17] L. A. Cheng, A. Dw, A. Yz, and B. Xm, "Model predictive control for path following and roll stabilization of marine vessels based on neurodynamic optimization - sciencedirect," *Ocean Engineering*, vol. 217, Article ID 107524, 2020.