

METHOD FOR AUTONOMOUS LEARNING OF MODELS FOR PREDICTIVE CONTROL IN PROCESS INDUSTRY

TECHNICAL FIELD

The present invention relates to the field of prediction technologies in the process industry, and in particular to a method for autonomous learning of models for predictive control in the process industry.

BACKGROUND

Since the late 20th century, the process industry has achieved rapid development, including the successful application of model predictive control in the process industry. Compared with conventional PID control, model predictive control is not only suitable for controlled objects that are difficult to control, such as controlled objects with multiple couplings and long delays, but also can model constraints like process conditions and economic indicators to complete control tasks in cooperation with optimization technologies. Therefore, model predictive control has a broad application prospect in the process industry. However, in actual production, dynamic characteristics of a process will change with various factors such as changes in production goals, changes in raw material ratios, equipment aging, and insufficient catalyst activity. Such changes will trigger a mismatch between a prediction model for predictive control and dynamic characteristics of an actual process. Therefore, if a matching degree between the prediction model and the dynamic characteristics of the actual process can be increased, a controller can achieve more accurate process output prediction, which further helps improve the control effect. At present, there are already some ideas or methods in the industry to deal with changing dynamic characteristics, such as nonlinear modeling, multi-condition modeling, and online identification. However, the existing methods have the following shortcomings in implementation.

First, for nonlinear modeling and multi-condition modeling: A large number of three-transfer and one-reaction (momentum transfer, heat transfer, mass transfer, and chemical reaction) processes in the process industry have increased the difficulty of mechanism modeling, and it is difficult for field operation data to cover all working conditions. Therefore, data-driven modeling methods can only be used to establish local models or partial-condition models. If operation is carried out under unknown conditions, a method for online model learning is still needed.

Second, for online identification methods: In one aspect, most of the existing methods use linear regression to model dynamic characteristics, parameters lack actual physical significance, and there is also a lack of reasonable constraints on the parameters during identification, resulting in excessive freedom in identifying the parameters; and in another aspect, dynamic characteristics of a process often change slowly, and a currently running prediction model should have guiding significance for new dynamic characteristic parameters. However, a technical method that can effectively take into account these two aspects has not yet been found.

SUMMARY

To overcome the defects of the above technologies, the present invention provides a method for autonomous learning of models for predictive control in the process industry.

The method is characterized in that: first, dynamic characteristics of a process can be autonomously learned online, and the method is applicable to a multi-input multi-output system through division of multi-input single-output subsystems, and certainly also applicable to other systems that are simpler than the multi-input multi-output system; second, types of to-be-learned parameters can be automatically selected based on preset model parameters, so that the to-be-learned parameters have actual physical significance, and during learning, a preset model has a constraint effect on model learning results to improve reliability of the model; third, the parameter learning results are tested, and prediction results of the model are evaluated to further improve the reliability of the model; and fourth, parameter dead zones are used to increase stability of the model parameters and reduce fluctuations in a control effect that are caused by frequent changes in the model parameters. The method in the present invention is an online learning method integrating model knowledge and process data, and is suitable for autonomous learning issues of models for predictive control in the process industry.

Description of terms:

1. Gain: an amplification factor.
2. Natural angular frequency: a physical quantity that describes a vibration speed of an object, and is related to inherent properties of a vibration system.
3. Damping coefficient: a parameter that characterizes a reduction of system energy during motion of a second-order or higher-order system.
4. Lognormal distribution: if a logarithm of a random variable follows a normal distribution, the random variable follows a lognormal distribution.
5. Prediction model: a quantitative relationship between things that is used for prediction in predictive control and described in a mathematical language or equation.

6. Parameter dead zone: when a parameter changes, if a change amplitude does not exceed this value, the parameter is considered unchanged.

The present invention uses the following technical solution to solve the technical problems:

A method for autonomous learning of models for predictive control in the process industry is implemented by using a multi-input single-output system as a basic subsystem, and includes the following:

Step 1: model setting and self-selection of to-be-learned parameters: Set all transfer function models, perform calculation on each transfer function model, and automatically select to-be-learned parameters.

The setting transfer function models specifically includes the following:

A transfer function model between each input variable and each output variable is set under a fixed transfer function model template combining a second-order system, first-order integration, and a dead time.

Assuming that the transfer function model has M input variables and P output variables, $M \times P$ transfer function models need to be set based on a template shown in equation (1), and an additional flag $\text{Int}_{p,m}$ needs to be set to indicate whether the transfer function model $G_{p,m}$ is additionally connected in series to a first-order integration link;

$$G_{p,m} = \frac{K_{p,m} e^{-\tau_{p,m} s}}{a_{p,m} s^2 + b_{p,m} s + 1}, a_{p,m} \geq 0, p = 1, 2, \dots, P, m = 1, 2, \dots, M \quad (1)$$

In the above equation, $K_{p,m}$ is a model gain, $\tau_{p,m}$ is a delay time, s is a Lagrangian operator, and $a_{p,m}$ and $b_{p,m}$ are respectively a second-order-term coefficient and a first-order-term coefficient of a characteristic polynomial of a transfer function.

Then, calculation is performed on each transfer function model and the to-be-learned parameters are automatically selected, where $*$ represents model parameter values specified before autonomous learning of the model. The following is specifically included:

Calculation is performed on each transfer function model, and then the following analysis is performed:

If a gain of a transfer function model is $K_{p,m}^* = 0$, the transfer function model $G_{p,m}$ does not participate in online learning of the models.

Or otherwise, pole locations of a second-order system are determined: (1) if a root with a

positive real part or a purely imaginary root exists, the transfer function model $G_{p,m}$ does not participate in the online learning of the models, either; (2) if two conjugate complex roots with negative real parts exist, a natural angular frequency $\omega_{n_{p,m}}^*$ and a damping coefficient $\zeta_{p,m}^*$ of the transfer function model $G_{p,m}$ are calculated, and a model gain $K_{p,m}$, a natural angular frequency $\omega_{n_{p,m}}$ and a damping coefficient $\zeta_{p,m}$ are used as to-be-learned parameters.

In addition to the above two cases (1) and (2), if a denominator polynomial $a_{p,m}^*s^2 + b_{p,m}^*s + 1$ of the transfer function model can undergo factorization of equation (2), a non-zero first-order-term coefficient $T_{1p,m}^*$, a non-zero first-order-term coefficient $T_{2p,m}^*$ and the model gain $K_{p,m}$ are used as to-be-learned parameters of the transfer function model $G_{p,m}$.

$$a_{p,m}^*s^2 + b_{p,m}^*s + 1 = (T_{1p,m}^*s + 1)(T_{2p,m}^*s + 1), T_{1p,m}^* \geq 0, T_{2p,m}^* \geq 0 \quad (2)$$

Step 2: parameter distribution modeling: For each transfer function model participating in learning, establish a lognormal distribution model for the to-be-learned parameters by using lognormal distribution. The following is specifically included:

When the transfer function model $G_{p,m}$ participates in learning, a lognormal distribution model is established for distribution of model parameter values by using lognormal distribution $p(\theta | \mu_\theta, \sigma_\theta^2)$, so that parameters of the transfer function model follow a lognormal distribution, where μ_θ and σ_θ^2 are distribution parameters regarding lognormal distribution of a parameter θ ; and based on the types of the to-be-learned parameters described in step 1, a user specifies a distribution shape-related coefficient $\{\beta_\theta | \beta_\theta > 1\}$, where $\theta \in \{K, T_1, T_2, \omega_n, \zeta\}$, $\theta_{p,m}^* \in \{K_{p,m}^*, T_{1p,m}^*, T_{2p,m}^*, \omega_{n_{p,m}}^*, \zeta_{p,m}^*\}$ is a parameter value specified for the transfer function model, and parameters for lognormal distribution are solved based on equation (3).

$$\begin{cases} E(\theta) = \beta_\theta \times \theta_{p,m}^* \\ \arg \max(p(\theta)) = \theta_{p,m}^* \end{cases} \quad (3)$$

Step 3: data collection and processing: Collect data of input variables and output variables, and quality codes corresponding to the input variables and the output variables online, and process a case in which the quality code has a bad value. The following is specifically included:

Process variables and quality codes of sensors are collected online. Specifically, the data of the input variables and the output variables, and the quality codes corresponding to the input variables and the output variables are collected, where the quality codes are used to indicate whether the collected input variables and output variables are valid, the input variables are denoted as $u_m(k)$, the output variables are denoted as $y_p(k)$, and k is a sampling moment.

If the quality code of $u_m(k)$ is bad, historical data of all the variables collected is cleared and data is re-collected.

If the quality code of $y_p(k)$ is bad, data is collected until y_p has a good value. If a quantity of consecutive bad values during the period is greater than or equal to preset N_{bad} , historical data of all the variables collected is cleared and data is re-collected; or otherwise, good-value data points at both ends of a bad data segment are used for linear interpolation of the bad data segment to complete historical data online and continue to collect data.

Step 4: learning of model parameters: Establish a mathematical optimization model for autonomous learning of the transfer function model based on the Bayesian optimization theory, and optimize parameters of the transfer function model by using a differential evolution algorithm. The following is specifically included:

When a data volume (the number of measurements for the output variables is always the same as that for the input variables) of the input variables or the output variables accumulates to a preset quantity N_s , the mathematical optimization model is established for autonomous learning of the transfer function model based on the Bayesian optimization theory for each multi-input single-output subsystem.

An impulse response sequence of the system is calculated based on the transfer function model, and a prediction sequence of the output variables of the system is calculated by using convolution of the impulse response sequence and the input variables of the system.

Learning of parameters of the transfer function model based on a differential evolution algorithm is repeated for L times, where $10 \leq L \leq 30$. A current situation is described as the r^{th} learning of model parameters of a P^{th} multi-input single-output subsystem, where N_{popu} is a population quantity specified by the user; a consecutive data segment with a length of N_{DS} is randomly selected, population parameters are randomly initialized depending on the types of the to-be-learned parameters, coefficients $a_{p,m}$ and $b_{p,m}$ are calculated based on the to-be-learned parameters and substituted into equation (1), discretization is performed by using

bilinear transformation to obtain an impulse response $\{g_{p,m,k}, k=1,2,\dots,N_{mdl}+d\}$ of the transfer function model $G_{p,m}$, where N_{mdl} is a phase length of an impulse response sequence, and d is a quantity of delay points, and a predicted value $\Delta\hat{y}_{p,m}(k)$ output by the transfer function model $G_{p,m}$ is calculated; if $\text{Int}_{p,m}=1$, a standard discretization calculation method for the first-order integration link is additionally used to update a sequence $\Delta\hat{y}_{p,m}(k)$; an output predicted value $\Delta\hat{y}_p(k)$ of the multi-input single-output subsystem is calculated based on equation (4), and an objective function value is calculated based on equation (5) depending on the types of the to-be-learned parameters, and is recorded in $\{J_{i_{popu}} | i_{popu}=1,\dots,N_{popu}\}$, where $\sigma^2 > 0$ is a noise level specified by the user, and $(\theta_1, \theta_2) \in \{(T_1, T_2), (\omega_n, \xi)\}$.

$$\Delta\hat{y}_p(k) = \sum_{m=1}^M \Delta\hat{y}_{p,m}(k) \quad (4)$$

$$J = \sum_{k=N_{mdl}+d}^{N_{DS}} \frac{(\Delta y_p(k) - \Delta\hat{y}_p(k))^2}{\sigma^2} - \sum_{m=1}^M \ln(p(|K_{p,m}|)p(\theta_{1p,m})p(\theta_{2p,m})) \quad (5)$$

Next, the following three steps in a standard differential evolution algorithm are repeatedly performed: population evolution, individual mutation, and individual selection, $\{J_{i_{popu}} | i_{popu}=1,\dots,N_{popu}\}$ and corresponding model parameters are recorded and updated based on a principle of minimizing an objective function; and finally, parameter learning results corresponding to optimal individuals are recorded in $\{K_{p,m,r}^{best}, T_{1p,m,r}^{best}, T_{2p,m,r}^{best} | m=1,\dots,M\}$ or $\{K_{p,m,r}^{best}, \omega_{np,m,r}^{best}, \zeta_{p,m,r}^{best} | m=1,\dots,M\}$, where $K_{p,m,r}^{best}, T_{1p,m,r}^{best}, T_{2p,m,r}^{best}, \omega_{np,m,r}^{best}, \zeta_{p,m,r}^{best}$ are respectively a model gain, two first-order-term coefficients, a natural angular frequency and a damping ratio of optimal individuals in the r^{th} learning of model parameters.

Step 5: parametric tests on models: Perform parametric tests including a normality test, parameter dead zone determining, and a mean test on each parameter of each transfer function model. The following is specifically included:

L learning results for each parameter of $\{G_{p,m} | p=1,2,\dots,P, m=1,2,\dots,M\}$ are tested.

Link 1: normality test: The Shapiro-Wilk test in GB/T 4882-2001 is used for the normality test, and sample data of a feature x_r is calculated based on equation (6) depending on the types of the to-be-learned parameters. If the normality test fails, the parameter is kept

unchanged at $\theta_{p,m}^*$; or otherwise, link 2 is performed.

$$x_r = \ln(|\theta_{p,m,r}^{best}|) - \ln(|\theta_{p,m}^*|) \quad (6)$$

Link 2: parameter dead zone determining: A mean $\bar{x}$ of $\{x_r | r=1, \dots, L\}$ is calculated, and if $|\bar{x}| \leq \ln(1+\eta_1)$, the parameter is kept unchanged at $\theta_{p,m}^*$, where $\eta_1 \geq 0$ is a parameter dead zone range specified by the user; or otherwise, link 3 is performed.

Link 3: mean test: The user specifies a mean test coefficient $\eta_2 \in [0,1]$, and if $\bar{x} > 0$, a null hypothesis is that a population mean meets: $\mu \leq \eta_2 \bar{x}$, or on the contrary, if $\bar{x} < 0$, a null hypothesis is that a population mean meets: $\mu \geq \eta_2 \bar{x}$; and if the mean test fails, the parameter is kept unchanged at $\theta_{p,m}^*$, or otherwise, $x_r = \eta_2 \bar{x}$ is substituted into equation (6) to inversely calculate a parameter learning result.

Step 6: comparison of model prediction errors: Calculate a root mean square error of prediction by using an original transfer function model, calculate a root mean square error of prediction by using a post-learning transfer function model, decide whether to update the transfer function model of the multi-input single-output system based on a relationship between the errors, and then return to step 1 to continue the execution. The following is specifically included:

For each multi-input single-output subsystem, a root mean square error of prediction for a P^{th} multi-input single-output system is calculated based on parameters of an original transfer function model and denoted as $RMSE_p^1$; and a root mean square error of prediction for the P^{th} multi-input single-output system is calculated based on parameters of a post-learning transfer function model and denoted as $RMSE_p^2$.

If $RMSE_p^2 < RMSE_p^1$, the transfer function model of the multi-input single-output system is updated, or otherwise, the transfer function model of the multi-input single-output system is kept unchanged.

Then, step 1 is returned to, to continue the execution.

The present invention has the following beneficial effects:

1. The present invention is an autonomous learning technology for model parameters that support online operation, and can use online data to actively learn model parameters.
2. In the present invention, the transfer function template adopts a general form of a

second-order transfer function, by which a stable second-order system model with real eigenvalues can be learned, and a stable second-order system model with complex conjugate eigenvalues can also be learned, so that the to-be-learned parameters have clear physical significance.

3. In the present invention, the reliability of online learning results of the transfer function models is improved by using two methods: in a first method, prior knowledge of a process is integrated into a learning process of the transfer function models, and the prior knowledge includes structures, parameter values, gain signs of transfer function models of a controller being put into operation, and types of to-be-learned parameters of the transfer function models that are automatically calculated from coefficients and have physical significance; and a second method includes two steps: a parametric test, and comparison of prediction error results of the transfer function models. Only after the parameters have passed a hypothesis test and the transfer function models have passed the comparison of prediction error results, the parameters of the transfer function models will be automatically updated to post-learning results.

4. In the present invention, stability of the parameters of the transfer function models is increased by using a parameter dead zone method, and when changes in the parameters of the transfer function models are within a dead zone specified by a user, the parameters of the transfer function models do not change, thereby increasing the stability of the parameters of the transfer function models, which is beneficial to the stability of a control effect.

5. In the present invention, it is assumed that a parameter distribution of the transfer function models is a lognormal distribution. In one aspect, a value range of random variables of the lognormal distribution is a positive number domain, and a range of valid $|K|, T_1, T_2, \omega_n, \zeta$ is also a positive number domain, that is, the two ranges are consistent; and in another aspect, for gains of the transfer function models, a large gain will slow down control, but a small gain may make the system unstable, and therefore, it should be more cautious to reduce a gain. However, a curve of a probability density function of lognormal distribution is relatively steep on the left side of a maximum value, and a probability density is attenuated rapidly with decreasing gains. Therefore, this prior model is suitable for modeling risks caused by small gains.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 shows a block flowchart of a method for autonomous learning of models for predictive control in the process industry according to the present invention;

FIG. 2 shows a flowchart of three links in parametric tests on models: a normality test,

parameter dead zone determining, and a mean test;

FIG. 3 shows step response curves of learning results of a transfer function model with different η_2 and a step response curve of an original transfer function model; and

FIG. 4 shows step response curves of some learning results of a transfer function model without a parameter dead zone method and a step response curve of an original transfer function model.

DESCRIPTION OF EMBODIMENTS

For better understanding of the present invention by a person skilled in the art, the following further describes the present invention in detail with reference to the accompanying drawings and specific embodiments. The following descriptions are only examples and constitute no limitation on the protection scope of the present invention.

A method for autonomous learning of models for predictive control in the process industry according to the present invention is applicable to a multi-input multi-output system. Specifically, the method is implemented through division of multi-input single-output subsystems. Therefore, in specific implementations, multi-input single-output subsystems are used for detailed description.

A two-input single-output subsystem is used as an example for description below, but the present invention is not limited to the two-input single-output subsystem described in this embodiment, and other multi-input single-output subsystems are all applicable.

It is assumed that a current transfer function model of the two-input single-output system is represented by equation (7).

$$\begin{aligned} Y(s) &= G_{1,1}(s)U_1(s) + G_{1,2}(s)U_2(s) \\ &= \frac{2}{(5s+1)s}U_1(s) + \frac{3.5}{4s^2+0.8s+1}U_2(s) \end{aligned} \quad (7)$$

Step 1: model setting and self-selection of to-be-learned parameters: Set all transfer function models, perform calculation on each transfer function model, and automatically select to-be-learned parameters.

According to equation (1) and equation (7), parameters are set as follows for two transfer function models in this embodiment:

$$K_{1,1}^* = 2, a_{1,1}^* = 0, b_{1,1}^* = 5, \text{Int}_{1,1} = 1; K_{1,2}^* = 3.5, a_{1,2}^* = 4, b_{1,2}^* = 0.8, \text{Int}_{1,2} = 0$$

For a transfer function model $G_{1,1}$, a characteristic polynomial (excluding pure integration) thereof can be factorized into a form of a product of first-order polynomials in a real number

domain; and therefore, to-be-learned parameters of $G_{1,1}$ are $K_{1,1}$ and $T_{11,1}$ based on calculation in equation (2), and $K_{1,1}^* = 2$ and $T_{11,1}^* = 5$.

For a transfer function model $G_{1,2}$, a characteristic polynomial (excluding pure integration) thereof includes two conjugate complex roots; and therefore, to-be-learned parameters of $G_{1,2}$ are $K_{1,2}$, $\omega_{n1,2}$ and $\zeta_{1,2}$, and after the characteristic polynomial is transformed into a zero-pole form, it can be known that $K_{1,2}^* = 3.5$, $\omega_{n1,2}^* = 0.5$ and $\zeta_{1,2}^* = 0.2$.

Step 2: parameter distribution modeling: For each transfer function model participating in learning, establish a lognormal distribution model for the to-be-learned parameters by using lognormal distribution.

Because both transfer function models have non-zero gains, and real parts of eigenvalues thereof are non-positive and have no purely imaginary root, both transfer function models should be learned;

Assuming that $\beta_\theta = 1.1$ ($\theta \in \{K, T_1, T_2, \omega_n, \zeta\}$), the following can be obtained by calculation based on equation (3): $u_{K_{1,1}} \approx 0.756687$, $u_{T_{11,1}} \approx 1.672978$, $u_{K_{1,2}} \approx 1.316303$, $u_{\omega_{n1,2}} \approx -0.629607$, $u_{\zeta_{1,2}} \approx -1.545898$, and $\sigma_K^2 = \sigma_{T_1}^2 = \sigma_{\omega_n}^2 = \sigma_\zeta^2 \approx 0.06354$, where $(u_{K_{1,1}}, \sigma_K^2)$, $(u_{T_{11,1}}, \sigma_{T_1}^2)$, $(u_{K_{1,2}}, \sigma_K^2)$, $(u_{\omega_{n1,2}}, \sigma_{\omega_n}^2)$ and $(u_{\zeta_{1,2}}, \sigma_\zeta^2)$ are respectively distribution parameters of lognormal distributions of parameters $K_{1,1}$, $T_{11,1}$, $K_{1,2}$, $\omega_{n1,2}$ and $\zeta_{1,2}$.

Step 3: data collection and processing: Collect data of input variables and an output variable, and quality codes corresponding to the input variables and the output variable online, and process a case in which the quality code has a bad value.

The data of the input variables and the output variable, and the quality codes corresponding to the input variables and the output variable are collected online, and it is determined whether the data quality code is good or bad, where the input variables are denoted as $u_1(k)$ and $u_2(k)$, the output variable is denoted as $y_1(k)$, and k is a sampling moment.

If the quality code of $u_1(k)$ or $u_2(k)$ is bad, historical data of all the variables collected is cleared and data is re-collected.

If the quality code of $y_1(k)$ is bad, data is collected until y_1 has a good value. If a quantity of consecutive bad values during the period is greater than or equal to preset N_{bad} , historical data of all the variables collected is cleared and data is re-collected; or if a quantity

of consecutive bad values during the period is less than N_{bad} , good-value data points at both ends of a bad data segment are used for linear interpolation of the bad data segment to complete historical data online and continue to collect data. For example, $N_{bad} = 3$, $y_1(5)=1, y_1(6)=1, y_1(7)=1, y_1(8)=4$, quality codes at sample points 6 and 7 are bad, and quality codes at sample points 5 and 8 are good. In this case, when $k=8$, data points with $k=5$ and $k=8$ are used for linear interpolation for $k=6$ and $k=7$, that is, $y_1(6)=2, y_1(7)=3$.

Step 4: learning of model parameters: Establish a mathematical optimization model for autonomous learning of the transfer function model based on the Bayesian optimization theory, and optimize parameters of the transfer function model by using a differential evolution algorithm.

When a data volume of the input variables or the output variable accumulates to a preset quantity $N_s = 3000$, learning of model parameters based on a differential evolution algorithm is repeated for the two-input single-output system for $L = 20$ times.

A current situation is described as the r^{th} learning of model parameters of the two-input single-output system.

A consecutive data segment with a length of $N_{DS} = 1000$ is randomly selected; a population quantity of $N_{popu} = 10$ is set, population parameters $\{K_{1,1,i_{popu}}, T_{1,1,i_{popu}} | i_{popu} = 1, \dots, 10\}$ and $\{K_{1,2,i_{popu}}, \omega_{n1,2,i_{popu}}, \zeta_{1,2,i_{popu}} | i_{popu} = 1, \dots, 10\}$ are randomly initialized, coefficients $a_{1,1}$, $b_{1,1}$, $a_{1,2}$ and $b_{1,2}$ are inversely calculated based on the to-be-learned parameters and substituted into a transfer function model template, discretization is performed by using bilinear transformation to obtain impulse responses (denoted as $\{g_{1,1,k}, k=1, 2, \dots, N_{mdl}+1\}$ and $\{g_{1,2,k}, k=1, 2, \dots, N_{mdl}+1\}$) of $G_{1,1}$ and $G_{1,2}$, predicted values $\Delta \hat{y}_{1,1}(k)$ and $\Delta \hat{y}_{1,2}(k)$ output by the transfer function models $G_{1,1}$ and $G_{1,2}$ are calculated based on equation (8), where Δ is a difference operator.

$$\Delta \hat{y}_{p,m}(k) = \sum_{i=1}^{N_{mdl}+d} g_{p,m,i} \Delta u_m(k-i), \Delta u_m(0) = 0, k = N_{mdl} + 1, \dots, 1000 \quad (8)$$

Because $\text{Int}_{1,1} = 1$, a standard discretization calculation method for a first-order pure integration link is used to update $\Delta \hat{y}_{1,1}(k)$, an output predicted value $\Delta \hat{y}_1(k)$ of the multi-

input single-output system is calculated based on equation (9), and an objective function value is calculated based on equation (10) and recorded in $\{J_{i_{popu}} | i_{popu} = 1, \dots, 10\}$, where $\sigma^2 > 0$ is a noise level specified by a user, and $(\theta_1, \theta_2) \in \{(T_1, T_2), (\omega_n, \xi)\}$.

$$\Delta \hat{y}_p(k) = \sum_{m=1}^2 \Delta \hat{y}_{p,m}(k) \quad (9)$$

$$J = \sum_{k=N_{mdl}+d}^{N_{ps}} \frac{(\Delta y_p(k) - \Delta \hat{y}_p(k))^2}{\sigma^2} - \sum_{m=1}^2 \ln(p(|K_{p,m}|)p(\theta_{1p,m})p(\theta_{2p,m})) \quad (10)$$

Next, the following three steps in a standard differential evolution algorithm are repeatedly performed: population evolution, individual mutation, and individual selection, $\{J_{i_{popu}} | i_{popu} = 1, \dots, 10\}$ and corresponding model parameters are recorded and updated based on a principle of minimizing an objective function; and finally, parameter learning results corresponding to optimal individuals are recorded in $\{K_{1,1,r}^{best}, T_{11,1,r}^{best}\}$ and $\{K_{1,2,r}^{best}, \omega_{n1,2,r}^{best}, \xi_{1,2,r}^{best}\}$.

Step 5: parametric tests on models: Perform parametric tests including a normality test, parameter dead zone determining, and a mean test on each parameter of each transfer function model.

As shown in FIG. 2, 20 learning results for each parameter of the transfer function models $G_{1,1}$ and $G_{1,2}$ are tested.

Link 1: normality test: Sample data of a feature x_r is calculated based on equation (6) depending on the types of to-be-learned parameters. For example, if a test result of a parameter $T_{1p,m}$ is unsatisfactory, the parameter $T_{1p,m}$ is kept unchanged at 5; and if other parameters are satisfactory, link 2 is performed.

Link 2: parameter dead zone determining: A mean $\bar{x}$ of $\{x_r | r = 1, \dots, 20\}$ is calculated. If $|\bar{x}| \leq \ln(1 + \eta_1)$, the parameter is kept unchanged at $\theta_{p,m}^*$. For example, $\eta_1 < 0.2$, if $|\bar{x}| = 0.15 \leq \ln(1 + 0.2)$ when parameter dead zone determining is performed on $K_{1,2}$, $K_{1,2}$ is kept unchanged at 3.5; and if other parameters are satisfactory, link 3 is performed.

Link 3: mean test: If $\bar{x} > 0$, a null hypothesis is that a population mean meets: $\mu \leq \eta_2 \bar{x}$. On the contrary, if $\bar{x} < 0$, a null hypothesis is that a population mean meets: $\mu \geq \eta_2 \bar{x}$. If the mean test fails, the parameter is kept unchanged at $\theta_{p,m}^*$. For example, when the mean test is

performed on a parameter $\omega_{n1,2}$ ($\eta_2=0.8$, $\bar{x}=0.2$), if a test result is that a hypothesis of $\mu \leq 0.16$ cannot be rejected, $\omega_{n1,2}$ is kept unchanged at 0.5; or otherwise, the null hypothesis is rejected, and $x_r = \eta_2 \bar{x}$ is substituted into equation (6) to inversely calculate a parameter learning result.

Step 6: comparison of model prediction errors: Calculate a root mean square error of prediction by using an original transfer function model, calculate a root mean square error of prediction by using a post-learning transfer function model, decide whether to update the transfer function model of the multi-input single-output system based on a relationship between the errors, and then return to step 1 to continue the execution.

Specifically, an output predicted value of the two-input single-output system is calculated based on parameters of the original transfer function model, and the root mean square error of prediction is calculated based on equation (11) and denoted as $RMSE_1^1$; and an output predicted value of the two-input single-output system is calculated again based on parameters of the post-learning transfer function model, and the root mean square error of prediction is calculated based on equation (11) and denoted as $RMSE_1^2$.

$$RMSE_p = \sqrt{\frac{1}{N_s - (N_{mdl} + d) + 1} \sum_{k=N_{mdl}+d}^{N_s} (\Delta y_p(k) - \Delta \hat{y}_p(k))^2} \quad (11)$$

If $RMSE_1^2 < RMSE_1^1$, a model parameter $\{K_{1,m}^*, a_{1,m}^*, b_{1,m}^* \mid m=1, 2\}$ of the two-input single-output system is updated; or on the contrary, all models of the two-input single-output system are kept unchanged.

Then step 1 is returned to, to continue the execution.

Further description is provided with reference to another specific embodiment. The specific embodiment is a multi-input single-output system shown in equation (12). Input variables are specified values of water temperatures of two branches, and an output variable is a water temperature of a main line. A control scheme is to control the water temperature of the main line by using the two branches. Specific implementations of the multi-input single-output system are the same as those described above, and therefore, an implementation process is not repeatedly described in detail, but the effect of the present invention is emphasized.

$$Y(s) = G_{1,1}(s)U_1(s) + G_{1,2}(s)U_2(s) \quad (12)$$

It is assumed that a transfer function model of a controller being put into operation is represented by equation (13):

$$\begin{aligned}
Y(s) &= G_{1,1}^*(s)U_1(s) + G_{1,2}^*(s)U_2(s) \\
&= \frac{0.395e^{-75s}}{300s+1}U_1(s) + \frac{0.312e^{-45s}}{480s+1}U_2(s)
\end{aligned} \tag{13}$$

$\eta_2 = 0.3$, $\eta_2 = 0.6$ and $\eta_2 = 0.8$ are set separately for autonomous learning of model parameters, and step response curves of learning results of $G_{1,1}(s)$ and a step response curve of $G_{1,1}^*(s)$ are plotted in FIG. 3. When a value of η_2 is 0.3, the step response curve of the learning result of $G_{1,1}(s)$ is closer to the step response curve of $G_{1,1}^*(s)$, and the model parameters change slightly; and when a value of η_2 is 0.8, the step response curve of the learning result of $G_{1,1}(s)$ is farther away from the step response curve of $G_{1,1}^*(s)$, and the model parameters change greatly. To sum up, η_2 will affect the learning results of the transfer function model $G_{1,1}(s)$. In an iterative process of autonomous learning of the model, a user can set different η_2 to control a change rate of parameters for autonomous learning of the model, so as to increase the reliability of the model in uncertain working conditions, and prevent impact of great changes of the model on stability of the process.

FIG. 4 shows step response curves of learning results of $G_{1,2}(s)$ and a step response curve of $G_{1,2}^*(s)$ without a parameter dead zone method. A relative difference between the parameters of the transfer function model in these learning results and the parameters of the transfer function model of $G_{1,2}^*(s)$ is less than 10%. In this case, because a predictive controller has a feedback correction link, the predictive controller has robustness in some range. Therefore, to reduce slight fluctuations in a transfer function model of the predictive controller, a parameter dead zone range $\eta_1 = 0.1$ can be set, so that learning results of the model will keep $G_{1,2}^*(s)$ unchanged. To sum up, a user can set different η_1 to increase the stability of parameters of the transfer function models through the robustness of the controller, which is beneficial to the stability of a control effect.

The foregoing only describe the basic principle and preferred implementations of the present invention. A person skilled in the art can make many changes and improvements based on the foregoing descriptions, and these changes and improvements shall fall within the protection scope of the present invention.

CLAIMS

What is claimed is:

1. A method for autonomous learning of models for predictive control in the process industry, wherein the method is implemented by using a multi-input single-output system as a basic subsystem, and comprises:

step 1: setting all transfer function models, performing calculation on each transfer function model, and automatically selecting to-be-learned parameters;

step 2: for each transfer function model participating in learning, establishing a lognormal distribution model for the to-be-learned parameters by using lognormal distribution;

step 3: collecting data of input variables and output variables, and quality codes corresponding to the input variables and the output variables online, and processing a case in which the quality code has a bad value;

step 4: establishing a mathematical optimization model for autonomous learning of the transfer function model based on the Bayesian optimization theory, and optimizing parameters of the transfer function model by using a differential evolution algorithm;

step 5: performing parametric tests comprising a normality test, parameter dead zone determining, and a mean test on each parameter of each transfer function model; and

step 6: for each multi-input single-output subsystem, calculating a root mean square error of prediction by using an original transfer function model, calculating a root mean square error of prediction by using a post-learning transfer function model, deciding whether to update the transfer function model of the multi-input single-output system based on a relationship between the errors, and then returning to step 1 to continue the execution.

2. The method for autonomous learning of models for predictive control in the process industry according to claim 1, wherein in step 1, the setting transfer function models specifically comprises:

setting a transfer function model between each input variable and each output variable under a fixed transfer function model template combining a second-order system, first-order integration, and a dead time.

3. The method for autonomous learning of models for predictive control in the process industry according to claim 2, wherein in step 1, a quantity of the transfer function models depends on

quantities of the input variables and the output variables, and if M input variables and P output variables are set, $M \times P$ transfer function models need to be set.

4. The method for autonomous learning of models for predictive control in the process industry according to claim 1, wherein in step 1, the performing calculation on each transfer function model, and automatically selecting to-be-learned parameters specifically comprises:

performing calculation on each transfer function model, and then performing the following analysis:

if a gain of a transfer function model is 0, the transfer function model does not participate in online learning of the models;

or otherwise, pole locations of a second-order system are determined: (1) if a root with a positive real part or a purely imaginary root exists, the transfer function model does not participate in the online learning of the models; (2) if two conjugate complex roots with negative real parts exist, a natural angular frequency and a damping coefficient of the transfer function model are calculated, and a model gain, a natural angular frequency and a damping coefficient are used as to-be-learned parameters; and in addition to the above two cases (1) and (2), a denominator polynomial of the transfer function model is factorized in a real number domain, and all non-zero first-order-term coefficients and model gains obtained after factorization are used as to-be-learned parameters.

5. The method for autonomous learning of models for predictive control in the process industry according to claim 1, wherein step 2 specifically comprises:

for each transfer function model participating in learning, calculating lognormal distribution parameters based on a distribution shape-related coefficient specified by a user, and modeling the to-be-learned parameters by using lognormal distribution.

6. The method for autonomous learning of models for predictive control in the process industry according to claim 1, wherein step 3 specifically comprises:

collecting the data of the input variables and the output variables, and the quality codes corresponding to the input variables and the output variables online, and performing the following determining and processing:

for the input variables, whenever there is bad quality code data, historical data of all the variables collected is cleared and data is re-collected;

for the output variables, when a quantity of consecutive pieces of occurring bad quality code data reaches a preset quantity N_{bad} or is above N_{bad} , historical data of all the variables collected is cleared and data is re-collected; or when a quantity of consecutive pieces of occurring bad quality code data is less than N_{bad} , good-value data points at both ends of a bad data segment are used to complete historical data online by using linear interpolation.

7. The method for autonomous learning of models for predictive control in the process industry according to claim 1, wherein step 4 specifically comprises:

when a data volume of the input variables or the output variables accumulates to a preset quantity, establishing the mathematical optimization model for autonomous learning of the transfer function model based on the Bayesian optimization theory for each multi-input single-output subsystem;

calculating an impulse response sequence of the system based on the transfer function model, and calculating a prediction sequence of the output variables of the system by using convolution of the impulse response sequence and the input variables of the system; and

optimizing the parameters of the transfer function model by using the differential evolution algorithm.

8. The method for autonomous learning of models for predictive control in the process industry according to claim 7, wherein the differential evolution algorithm comprises three steps: population evolution, individual mutation, and individual selection.

9. The method for autonomous learning of models for predictive control in the process industry according to claim 1, wherein step 5 specifically comprises:

sequentially performing the normality test, the parameter dead zone determining, and the mean test on each parameter of each transfer function model, wherein a parameter learning result is valid when all the three links are successful, or otherwise, the parameter learning result is invalid, and the original parameter is kept unchanged.

10. The method for autonomous learning of models for predictive control in the process industry according to claim 1, wherein step 6 specifically comprises:

for each multi-input single-output subsystem, calculating the root mean square error of prediction for the multi-input single-output system based on parameters of the original transfer

function model and denoting the root mean square error of prediction as $RMSE^1$, and calculating the root mean square error of prediction for the multi-input single-output system based on parameters of the post-learning transfer function model and denoting the root mean square error of prediction as $RMSE^2$;

if $RMSE^2 < RMSE^1$, updating the transfer function model of the multi-input single-output subsystem, or otherwise, keeping the transfer function model of the multi-input single-output subsystem unchanged; and

returning to step 1 to continue the execution.