Generalized Predictive Control with Online Least Squares Support Vector Machines

LI Li-Juan[1, 2] SU Hong-Ye[1] CHU Jian[1]

Abstract This paper proposes a practical generalized predictive control (GPC) algorithm based on online least squares support vector machines (LS-SVM) which can deal with nonlinear systems effectively. At each sampling period the algorithm recursively modifies the model by adding a new data pair and deleting the least important one out of the consideration on realtime property. The data pair deleted is determined by the absolute value of lagrange multiplier from last sampling period. The paper gives the recursive algorithm of model parameters when adding a new data pair and deleting an existent one, respectively, and thus the inversion of a large matrix is avoided and the memory can be controlled by the algorithm entirely. The nonlinear LS-SVM model is applied in GPC algorithm at each sampling period. The experiments of generalized predictive control on pH neutralizing process show the effectiveness and practicality of the proposed algorithm.

Key words Generalized predictive control, least squares support vector machines, fuzzy least squares support machines, online modeling, pH neutralizing process

1 Introduction

Generalized predictive control(GPC)[1], integrating self-adaptive control into predictive control, has been shown to be particularly effective. Linear predictive control approaches are well-established in control practice, and nonlinear model predictive control (NMPC) has also found its way in control practice[2, 3]. NMPC algorithms are based on various nonlinear models and the quality depends on the models. The popular nonlinear models are neural network ones which undoubtedly have solved many problems with complicated models. However, the theoretically perfect neural network sometimes performs poorly in practice due to local minimum and over-fitting problems.

Support vector machines (SVM) theory[4] has drawn much attention for the high generalization ability and global optimization property recently[5]. Analytical solutions can be obtained by solving linear equations instead of a quadratic programming (QP) problem in least squares support vector machines (LS-SVM)[6]. Fuzzy support vector machines (FSVM) is a modified SVM algorithm which treats the training data points differently according to their different degrees of importance[7]. The idea of FSVM is introduced into LS-SVM (Fuzzy least squares support vectors machines, FLS-SVM) in [8]. LS-SVM modeling is applied to GPC by Liu[9], but it is an offline method and could not express the dynamic behavior of plant realtime.

The contribution of this paper is to describe a GPC algorithm with practical online LS-SVM modeling which can avoid the process of inversion of large matrices, solve the computation, and memory problem of LS-SVM. In LS-SVM and its modified versions, inversion of a large matrix is included such that the computation is huge and it is generally applied in offline modeling. Reference [10] presented an iterative training algorithm for LS-SVM based on a conjugate gradient method. However, when the training set increases, the whole process of computation should be carried out once again, which limits its online applications in a sense. In [11], pruning algorithm is used for sparseness in which data are selectively omitted so that the training set is deduced and corresponding dimension of the matrix is decreased. It is a pity that the pruning algorithm is also an offline algorithm.

Received November 30, 2006; in revised form March 21, 2007
Supported by the National Creative Research Groups Science Foundation of P. R. China (NCRGSFC: 60421002) and National High Technology Research and Development Program of China (863 Program) (2006AA04 Z182)
1. State Key Laboratory of Industrial Control Technology, Institute of Advanced Process Control, Zhejiang University, Hangzhou 310027, P. R. China 2. College of Automation, Nanjing University of Technology, Nanjing 210009, P. R. China
DOI: 10.1360/aas-007-1182

2 Least squares support vector machines

2.1 Basic least squares support vector machines

Consider a given regression data set $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$, where N is the total number of training data pairs, $\boldsymbol{x}_i \in \mathbf{R}^n$ is the regression vector and $y_i \in \mathbf{R}$ is the output. According to SVM theory[4], the input space $\mathbf{R}^n$ is mapped into a feature space $\mathbf{Z}$ with the nonlinear function $\varphi(\boldsymbol{x}_i)$ being the corresponding mapping function. In the feature space,

$$y(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\varphi(\boldsymbol{x}) + b \quad \text{with} \quad \boldsymbol{w} \in \mathbf{Z}, b \in \mathbf{R} \tag{1}$$

is taken to estimate the unknown function, where vector $\boldsymbol{w}$ and scalar b are the parameters to be identified. The optimization problem is defined as follows.

$$\min_{\boldsymbol{w},\boldsymbol{e}} J(\boldsymbol{w}, \boldsymbol{e}) = \frac{1}{2}\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w} + \frac{\gamma}{2}\sum_{i=1}^{N} e_i^2, \qquad \gamma > 0 \tag{2}$$

$$\text{subject to} \quad y_i = \boldsymbol{w}^{\mathrm{T}}\varphi(\boldsymbol{x}_i) + b + e_i, \qquad i = 1, 2, \cdots, N \tag{3}$$

where e_i is the error between actual output and predictive output of the ith data.

The LS-SVM model of the data set can be given by

$$y(\boldsymbol{x}) = \sum_{i=1}^{N} \alpha_i K(\boldsymbol{x}, \boldsymbol{x}_i) + b \tag{4}$$

where $\alpha_i \in \mathbf{R}$ $(i = 1, 2, \cdots, N)$ are Lagrange multipliers and $K(\boldsymbol{x}, \boldsymbol{x}_i)$ $(i = 1, 2, \cdots, N)$ are any kernel functions satisfying the Mercer condition[5]. Analytical solutions of $\alpha_i \in \mathbf{R}$ $(i = 1, 2, \cdots, N)$ and b can be obtained from

$$\begin{bmatrix} b \\ \boldsymbol{\alpha} \end{bmatrix} = \Phi^{-1} \begin{bmatrix} 0 \\ \boldsymbol{Y} \end{bmatrix} \tag{5}$$

with $\boldsymbol{Y} = [y_1\ y_2\ \cdots\ y_N]^{\mathrm{T}}$, $\boldsymbol{\alpha} = [\alpha_1\ \alpha_2\ \cdots\ \alpha_N]^{\mathrm{T}}$, and the supposed nonsingular matrix

$$\Phi = \begin{bmatrix} \mathbf{0} & \mathbf{1}^{\mathrm{T}} \\ \mathbf{1} & \Omega + \gamma^{-1} I \end{bmatrix} \tag{6}$$

where $\mathbf{1} = [1\ 1\ \cdots\ 1]^{\mathrm{T}}$, I is a $N \times N$ identity matrix and Ω is a $N \times N$ symmetric matrix with the elements

$$\Omega_{ij} = \varphi(\boldsymbol{x}_i)^{\mathrm{T}}\varphi(\boldsymbol{x}_j) = K(\boldsymbol{x}_i, \boldsymbol{x}_j), \qquad i, j = 1, 2, \cdots, N \tag{7}$$

2.2 Fuzzy least squares support vector machines

Given a data set $\{(\boldsymbol{x}_i, y_i, s_i)\}_{i=1}^{N}$, where N is the total number of training data pairs, $\boldsymbol{x}_i \in \mathbf{R}^n$ is input vector, $y_i \in \mathbf{R}$ is output signal, and s_i is a membership function which denotes the weightiness of the corresponding data point. The value of s_i can be obtained by

$$s_i = f(\alpha_i) = (1 - \delta)\left(\frac{|\alpha_i| - |\alpha_{\min}|}{|\alpha_{\max}| - |\alpha_{\min}|}\right) + \delta \tag{8}$$

where sufficient small parameter $\delta > 0$ is the minimum of membership function s_i, $\alpha_{\min}$ is the Lagrange multiplier with minimum absolute value, and $\alpha_{\max}$ is that of maximum[8].

Then, s_i is introduced into the second term of (2) and optimization problem is defined as

$$\min_{\boldsymbol{w},\boldsymbol{e}} J(\boldsymbol{w}, \boldsymbol{e}) = \frac{1}{2}\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w} + \frac{\gamma}{2}\sum_{i=1}^{N} s_i e_i^2, \qquad \gamma > 0 \tag{9}$$

$$\text{subject to} \quad y_i = \boldsymbol{w}^{\mathrm{T}}\varphi(\boldsymbol{x}_i) + b + e_i, \qquad i = 1, 2, \cdots, N \tag{10}$$

where parameters have the same meanings as those in Section 2.1.

The expression of regression model is the same as (4). And the final formulation of analytical solution is just as (5), but the matrix Φ is

$$\Phi = \begin{bmatrix} \mathbf{0} & \mathbf{1}^{\mathrm{T}} \\ \mathbf{1} & \Omega + (\gamma S)^{-1} \end{bmatrix} \tag{11}$$

with $S = \mathrm{diag}\{s_1, s_2, \cdots, s_N\}$.

It is easy to see that basic LS-SVM is obtained in case of each s_i is set to 1. In this sense, the basic LS-SVM is a peculiar case of FLS-SVM.

3 Online LS-SVM modeling algorithm

3.1 Adding a new data pair

The following theorem gives the recursive algorithm of LS-SVM model parameters when a new point is added.

Theorem 1. Considering the function regression problem in Section 2, let us suppose that $\Theta_N = [b \quad \boldsymbol{\alpha}]^{\mathrm{T}}$ is the parameter matrix obtained from the training set consisting of N pairs of data by (5). Let $P_N = \Phi_N^{-1}$, when another data pair $\{\boldsymbol{x}_{N+1}, y_{N+1}\}$ is added into the training set by which the recursive algorithm of Θ can be obtained by (18) and (19) where

$$\Psi_{N+1} = [1 \quad \varphi_1\varphi_{N+1} \quad \varphi_2\varphi_{N+1} \quad \cdots \quad \varphi_N\varphi_{N+1}] \tag{12a}$$

$$\zeta_{N+1} = (\varphi_{N+1}\varphi_{N+1} + 1/\gamma)^{-1} \tag{12b}$$

$$\eta_{N+1} = (\Psi_{N+1}P_N\Psi_{N+1}^{\mathrm{T}} - \zeta_{N+1}^{-1})^{-1} \tag{12c}$$

Proof. For the sake of convenience, we introduce the following lemmas firstly.

Lemma 1[12]**.** Assume that A, C, and $(A + BCD)$ are nonsingular matrices, then

$$(A + BCD)^{-1} = A^{-1} - A^{-1}B(DA^{-1}B + C^{-1})^{-1}DA^{-1} \tag{13}$$

holds.

Lemma 2. Assume that partitioned matrix

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \tag{14}$$

and that A_{11} and A_{22} are nonsingular matrices, then (20) holds.

Remark. Lemma 2 can be easily deduced by the matrix theory.

Consider the function regression problem in Section 2.1. According to (5), when $\{\boldsymbol{x}_{N+1}, y_{N+1}\}$ is added to the training set, the parameter matrix $\Theta_{N+1} = [b \quad \boldsymbol{\alpha}_{N+1}]^{\mathrm{T}}$ can be expressed as

$$\Theta_{N+1} = \Phi_{N+1}^{-1}\begin{bmatrix} 0 \\ \boldsymbol{Y}_{N+1} \end{bmatrix} \tag{15}$$

with $\boldsymbol{Y}_{N+1} = [y_1\ y_2\ \cdots\ y_N\ y_{N+1}]^{\mathrm{T}}$, $\boldsymbol{\alpha}_{N+1} = [\alpha_1\ \alpha_2\ \cdots\ \alpha_{N+1}]$, and

$$\Phi_{N+1} = \begin{bmatrix} \mathbf{0} & \mathbf{1}^{\mathrm{T}} \\ \mathbf{1} & \Omega_{N+1} + \gamma^{-1}I \end{bmatrix} \tag{16}$$

Assume

$$\Psi_{N+1} = [1 \quad \varphi_1\varphi_{N+1} \quad \varphi_2\varphi_{N+1} \quad \cdots \quad \varphi_N\varphi_{N+1}]$$

Applying (6), we can write (16) as

$$\Phi_{N+1} = \begin{bmatrix} \Phi_N & \Psi_{N+1}^{\mathrm{T}} \\ \Psi_{N+1} & \varphi_{N+1}\varphi_{N+1} + 1/\gamma \end{bmatrix} \tag{17}$$

Taking $A_{11} = \Phi_N$, $A_{12} = \Psi_{N+1}^{\mathrm{T}}$, $A_{21} = \Psi_{N+1}$, $A_{22} = \varphi_{N+1}\varphi_{N+1} + 1/\gamma$ in (14) and applying (20) yields (21) where $\zeta_{N+1} = (\varphi_{N+1}\varphi_{N+1} + 1/\gamma)^{-1}$ is a scalar.

$$\Theta_{N+1} = \begin{bmatrix} \Theta_N + \eta_{N+1}P_N\Psi_{N+1}^{\mathrm{T}}[y_{N+1} - \Psi_{N+1}\Theta_N] \\ \zeta_{N+1}\Psi_{N+1}[\eta_{N+1}P_N\Psi_{N+1}^{\mathrm{T}}\Psi_{N+1} - I]\Theta_N - y_{N+1}\eta_{N+1} \end{bmatrix} \tag{18}$$

$$P_{N+1} = \begin{bmatrix} P_N - \eta_{N+1}P_N\Psi_{N+1}^{\mathrm{T}}\Psi_{N+1}P_N & \eta_{N+1}P_N\Psi_{N+1}^{\mathrm{T}} \\ \zeta_{N+1}\Psi_{N+1}[\eta_{N+1}P_N\Psi_{N+1}^{\mathrm{T}}\Psi_{N+1}P_N - P_N] & -\eta_{N+1} \end{bmatrix} \tag{19}$$

$$A^{-1} = \begin{bmatrix} (A_{11} - A_{12}A_{22}^{-1}A_{21})^{-1} & -A_{11}^{-1}A_{12}(A_{22} - A_{21}A_{11}^{-1}A_{12})^{-1} \\ -A_{22}^{-1}A_{21}(A_{11} - A_{12}A_{22}^{-1}A_{21})^{-1} & (A_{22} - A_{21}A_{11}^{-1}A_{12})^{-1} \end{bmatrix} \tag{20}$$

$$P_{N+1} = \begin{bmatrix} [\Phi_N - \Psi_{N+1}^{\mathrm{T}}\zeta_{N+1}\Psi_{N+1}]^{-1} & -\Phi_N^{-1}\Psi_{N+1}^{\mathrm{T}}[\zeta_{N+1}^{-1} - \Psi_{N+1}\Phi_N^{-1}\Psi_{N+1}^{\mathrm{T}}]^{-1} \\ -\zeta_{N+1}\Psi_{N+1}[\Phi_N - \Psi_{N+1}^{\mathrm{T}}\zeta_{N+1}\Psi_{N+1}]^{-1} & [\zeta_{N+1}^{-1} - \Psi_{N+1}\Phi_N^{-1}\Psi_{N+1}^{\mathrm{T}}]^{-1} \end{bmatrix} \tag{21}$$

Applying Lemma 1 to $[\Phi_N - \Psi_{N+1}^{\mathrm{T}}\zeta_{N+1}\Psi_{N+1}]^{-1}$ in (21) gives (19).

In addition, let $\bar{\boldsymbol{Y}}_N = [0 \quad \boldsymbol{Y}_N]^{\mathrm{T}}$, then $\bar{\boldsymbol{Y}}_{N+1} = [0 \quad \boldsymbol{Y}_{N+1}]^{\mathrm{T}} = [\bar{\boldsymbol{Y}}_N \quad y_{N+1}]^{\mathrm{T}}$. Now (15) can be written as follows,

$$\Theta_{N+1} = P_{N+1}\begin{bmatrix}\bar{\boldsymbol{Y}}_N \\ y_{N+1}\end{bmatrix} \tag{22}$$

Substituting (19) into (22) and noticing that $\Theta_N = P_N\bar{\boldsymbol{Y}}_N$, we get (18). □

3.2 Deleting an existent data pair

It can be found that the number of elements of $\boldsymbol{\alpha}$ scales with that of data pairs in (5) and Φ^{-1} need to be saved each time for the next computation in Theorem 1. Upon that, with a new data added into the training set the number of data to be saved increase by $(2N-1)$ and the computation time also increases. For online case the new data are added continually, and the memory and computation would continually increase at each sampling period. Therefore, if the size of training set is enough for a relatively precise model after some recursive steps, we consider to remove a former data pair once a new data pair is added. Then the size of training set would be controlled, the computation time would not be longer, and the required memory would not increase endlessly.

For deleting a data pair, we consider the training set $\{\boldsymbol{x}_i, y_i\}_{i=1}^{N+1}$. Let $P_{N+1} = \Phi_{N+1}^{-1}$. For the sake of convenience, P_{N+1} in (19) is written as the form of block matrix

$$P_{N+1} = \left[\begin{array}{ccc:c} p_{11} & \cdots & p_{1(N+1)} & p_{1(N+2)} \\ \vdots & \vdots & \vdots & \vdots \\ p_{(N+1)1} & \cdots & p_{(N+1)(N+1)} & p_{(N+1)(N+2)} \\ \hdashline p_{(N+2)1} & \cdots & p_{(N+2)(N+1)} & p_{(N+2)(N+2)} \end{array}\right] = \begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22}\end{bmatrix} \tag{23}$$

When the last data pair $\{\boldsymbol{x}_{N+1}, y_{N+1}\}$ is removed from the training set the recursive algorithm of inversion $P_N = \Phi_N^{-1}$ can be obtained by

$$P_N = P_{11} - \frac{P_{12}P_{21}}{P_{22}} \tag{24}$$

which can be testified by substituting it back to (19)[13].

Equation (24) permits us to eliminate the last data pair. In LS-SVM, the order of training data pairs is approximately arbitrary, and any change of the order would not effect the modeling result. So, with any data pair $(\boldsymbol{x}_i, y_i)$ removed, equation (24) holds except that P_{11} is a $(N+1)\times(N+1)$ square matrix from P_{N+1} removing the $(i+1)$th row and $(i+1)$th column, P_{12} is the $(i+1)$th column of P_{N+1} without $p_{(i+1)(i+1)}$, P_{21} is the $(i+1)$th row of P_{N+1} without $p_{(i+1)(i+1)}$, and $P_{22} = p_{(i+1)(i+1)}$.

Contributions of different data pairs to the training of LS-SVM are not the same. Considering online algorithm, we try to delete the least important data pair at each sampling period. The value of $|\alpha_i|$ just denotes the importance of corresponding data point[11]. Thus, we have a method to eliminate the least important data pair and the one corresponding to minimum $|\alpha_i|$ is removed at each sampling period in our algorithm.

3.3 Procedure of online LS-SVM modeling

Consider the system based on input/output model

$$y(k) = f(\boldsymbol{x}(k))$$

where $y(k)$ is the output of the system at kth time , $f(\cdot)$ is the nonlinear function to be identified expressing the properties of the system, $\boldsymbol{x}(k)$ is the regression data vector

$$\begin{aligned}\boldsymbol{x}(k) = [x_k(1) \ \cdots \ x_k(n_u+n_y)] = \\ [u(k-1) \ \cdots \ u(k-n_u) \ y(k-1) \ \cdots \ y(k-n_y)]\end{aligned} \tag{25}$$

with n_u and n_y being the input and output orders, respectively. A general procedure of online LS-SVM modeling algorithm can be described as follows.

Step 1. Select an appropriate kernel function and the corresponding parameters. Determine the sampling period T_0, input order n_u, output order n_y, weight γ in (2), and size of training set NN.

Step 2. Sample input/output data $\{y_{k+1}, u_{k+1}\}$, construct the regression vectors $\boldsymbol{x}_{k+1}$ by (25), and form the updated training set $\{y_i, \boldsymbol{x}_i\}_{i=1}^{k+1}$.

Step 3. If the sampling number is not larger than NN, compute Ψ_{k+1}, ζ_{k+1} and η_{k+1} by (12) and apply (18), (19) to calculate the updated Θ_{k+1} and P_{k+1}, respectively. Thus the updated regression model is obtained by (24). Go to Step 2.

Step 4. Or else, if the sampling number is larger than NN, find the index I of minimum $|\alpha_i|$ from the result of last sampling period, delete α_I from $\boldsymbol{\alpha}$, delete $(\boldsymbol{x}_I, y_I)$ from the training set, set P_{11}, P_{12}, P_{21}, P_{22}, and calculate update P by (24).

Step 5. Compute Ψ_{k+1}, ζ_{k+1}, and η_{k+1} by (12) and apply (18), (19) to calculate the updated Θ_{k+1} and P_{k+1}, respectively. Thus the updated regression model is obtained by (4). Go to Step 2.

3.4 Consideration of FLS-SVM

The case of online fuzzy least squares support vector machines is much similar to that of LS-SVM except that a parameter s_i is introduced. The recursive algorithm is similar to (18) and (19), and we give the following theorem without proof.

Theorem 2. Consider the function regression problem in Section 2.2, and suppose that $\Theta_N = [b \quad \boldsymbol{\alpha}]^{\mathrm{T}}$ is the parameter matrix obtained from the training set consisting of N pairs of data by (5). Each membership function value s_i is determined by (8). Let $P_N = \Phi_N^{-1}$, when another data pair $\{\boldsymbol{x}_{N+1}, y_{N+1}\}$ is added into the training set the recursive algorithm of Θ is the same to LS-SVM (*i*.e. Theorem 1) except that (12b) is replaced by

$$\zeta_{N+1} = \left(\varphi_{N+1}\varphi_{N+1} + 1/(\gamma s_{N+1})\right)^{-1} \tag{26}$$

To apply Theorem 2, we should firstly compute the value of s_{N+1}. However, it can be seen from (8) that the parameter s_{N+1} is determined by α_{N+1} whereas it is unknown when the new data point is added. But, we have known that basic LS-SVM is obtained when all the s_i in FLS-SVR are set to 1. Thus on each recursive step, we could firstly work out $\alpha_i|_{i=1}^{N+1}$ by recursive LS-SVM (Theorem 1), and then calculate s_{N+1} by (8), and finally run the recursive FLS-SVM (Theorem 2).

For online modeling problem of FLS-SVM, the procedure is similar to online LS-SVM in Section 3.3, and the differences lie on:

1) An additional parameter δ in (8) should be determined beforehand in online FLS-SVM;

2) At each sampling period the recursive step would be run twice in FLS-SVM: the first time computes the initial $\alpha_i|_{i=1}^{N+1}$ for the computation of $s_i|_{i=1}^{N+1}$, the second time runs according to Theorem 2 and gets the final result of Θ at this recursive step.

4 GPC algorithm with online LS-SVM

4.1 Linearization of the nonlinear LS-SVM model

As techniques in [9] dealing with nonlinear GPC, the nonlinear LS-SVM model (4) is linearized at each sampling period in our algorithm.

Consider the kth sampling period and $\boldsymbol{x}_k$ is the corresponding regression vector. The LS-SVM model is linearized as (36) where C is a constant. Substituting (25) into (36), we get

$$\begin{aligned} y(\boldsymbol{x}) = C + b_1 u(k-1) + \cdots + b_{n_u} u(k-n_u) - \\ a_1 y(k-1) - \cdots - a_{n_y} y(k-n_y) \end{aligned} \tag{27}$$

Let

$$\begin{aligned} A(q^{-1}) &= 1 + a_1 q^{-1} + \cdots + a_{n_y} q^{-n_y} \\ B(q^{-1}) &= b_1 + b_2 q^{-1} + \cdots + b_{n_u} q^{-(n_u-1)} \end{aligned} \tag{28}$$

where q^{-1} is the shift operator denoting the movement of the amount corresponding to a sampling period back. Then the linearized model (27) can be written as follows:

$$A(q^{-1})y(k) = B(q^{-1})u(k-1) + C \tag{29}$$

Under the random disturbance to the plant, a controlled auto-regressive integrated moving average (*i.e.* CARIMA) model is formulated as follows:

$$A(q^{-1})y(k) = B(q^{-1})u(k-1) + \frac{\xi(k)}{\Delta} \tag{30}$$

where $\xi(k)$ including the constant item C in (29) is an uncorrelated random sequence denoting some kind of random noise and $\Delta = 1 - q^{-1}$ is difference operator.

4.2 GPC control

The optimal performance criterion at kth period is defined as

$$\begin{aligned} \min J(k) = E\Bigg\{ \sum_{j=N_1}^{N_2} [y(k+j) - y_r(k+j)]^2 + \\ \sum_{j=1}^{N_u} \lambda(j)[\Delta u(k+j-1)]^2 \Bigg\} \end{aligned} \tag{31}$$

where E represents mathematical expectation, y_r is the expectation of output, N_1 and N_2 are the initial and final value of optimal horizon, respectively, N_u is the control horizon, and λ is the weight of control item.

According to the algorithm of Clarke[1], the incremental optimal control law can be solved as follows:

$$\Delta \boldsymbol{u} = (G^{\mathrm{T}}G + \lambda I)^{-1} G^{\mathrm{T}} [y_r - Fy(k) - H\Delta \boldsymbol{u}(k-1)] \tag{32}$$

where the matrices G, F, and H are obtained by recursively solving the Diophantine equations. The instant optimal control law can be computed by

$$\boldsymbol{u}(k) = \boldsymbol{u}(k-1) + \boldsymbol{g}^{\mathrm{T}} [y_r - Fy(k) - H\Delta \boldsymbol{u}(k-1)] \tag{33}$$

where $\boldsymbol{g}^{\mathrm{T}}$ is the first row of matrix $(G^{\mathrm{T}}G + \lambda \boldsymbol{I})^{-1} G^{\mathrm{T}}$.

The multistep predictive output can be obtained by

$$\hat{y} = G\Delta \boldsymbol{u} + Fy(k) + H\Delta \boldsymbol{u}(k-1). \tag{34}$$

5 Experiments

The presented GPC strategy is applied in the simulation of pH neutralizing process, which is a weak acid-strong base system and with strong nonlinearity, especially in the vicinity of pH= 9.

The physical model of a pH process in a continuously stirred tank reactor (CSTR) consists of two parts, a linear dynamical part followed by a nonlinear static part[14]. The dynamical model is given by

$$\begin{cases} V \dfrac{\mathrm{d}w_a}{\mathrm{d}t} = F_a C_a - (F_a + F_b) w_a \\ V \dfrac{\mathrm{d}w_b}{\mathrm{d}t} = F_b C_b - (F_a + F_b) w_b \end{cases} \tag{35}$$

where F_a and F_b denote the inlet flow-rate of acid and base (cm^3/min), respectively, C_a and C_b are the inlet concentrations of acid and base (mol/L), the volume of the content in the reactor is denoted by the constant $V(\mathrm{cm}^3)$, and w_a and w_b are the concentrations of acid and base after the process of dynamical model (mol/L). Simultaneously, w_a and w_b are the inputs of the static model (37).

$$\begin{aligned} y(\boldsymbol{x}) =& y(\boldsymbol{x})|_{\boldsymbol{x}=\boldsymbol{x}_k} + \frac{\partial y}{\partial \boldsymbol{x}(1)}\bigg|_{\boldsymbol{x}=\boldsymbol{x}_k} [\boldsymbol{x}(1) - \boldsymbol{x}_k(1)] + \cdots + \frac{\partial y}{\partial \boldsymbol{x}(n_u+n_y)}\bigg|_{\boldsymbol{x}=\boldsymbol{x}_k} [\boldsymbol{x}(n_u+n_y) - \boldsymbol{x}_k(n_u+n_y)] = \\ & y(\boldsymbol{x})|_{\boldsymbol{x}=\boldsymbol{x}_k} + \frac{\partial y}{\partial \boldsymbol{x}(1)}\bigg|_{\boldsymbol{x}=\boldsymbol{x}_k} \boldsymbol{x}(1) + \cdots + \frac{\partial y}{\partial \boldsymbol{x}(n_u+n_y)}\bigg|_{\boldsymbol{x}=\boldsymbol{x}_k} \boldsymbol{x}(n_u+n_y) - \\ & \frac{\partial y}{\partial \boldsymbol{x}(1)}\bigg|_{\boldsymbol{x}=\boldsymbol{x}_k} \boldsymbol{x}_k(1) - \cdots - \frac{\partial y}{\partial \boldsymbol{x}(n_u+n_y)}\bigg|_{\boldsymbol{x}=\boldsymbol{x}_k} \boldsymbol{x}_k(n_u+n_y) = \\ & C + b_1 \boldsymbol{x}(1) + \cdots + b_{n_u} \boldsymbol{x}(n_u) - a_1 \boldsymbol{x}(n_u+1) - \cdots - a_{n_y} \boldsymbol{x}(n_u+n_y) \end{aligned} \tag{36}$$

$$w_b + 10^{-y} - 10^{y-14} - \frac{w_a}{1 + 10^{pK_a - y}} = 0 \qquad (37)$$

where y is the pH value of the effluent, K_a is the dissociation constant of the acetic acid with $K_a = 1.76 \times 10^{-5}$ and $pK_a = -\lg K_a$.

By fixing the flow-rate F_a at a specific value, the process is regarded as a single variable system with base flow-rate F_b and pH value y of the effluent being the input and the output, respectively. The simulating system uses the physical model with the parameter values given in Table 1. The sampling period is set to 0.5min.

Table 1 Parameter values used in pH neutralizing model

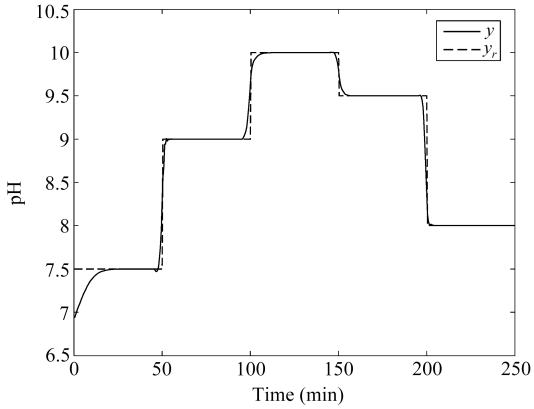
Parameter	Nominal value
F_a	81cm^3/min
C_a	0.32mol/L
C_b	0.05mol/L
V	1000cm^3
$w_a(0)$	0.0435mol/L
$w_b(0)$	0.0432mol/L

5.1 GPC with online LS-SVM and FLS-SVM modeling

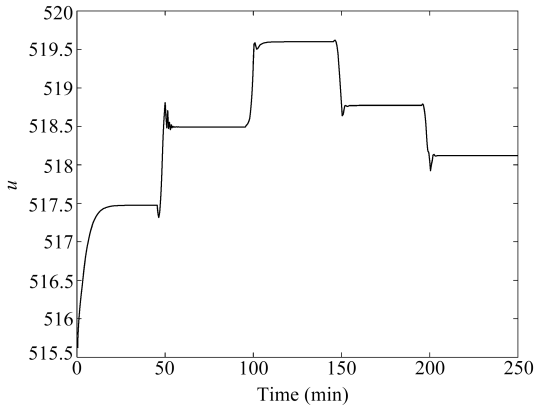
Radial Basis Function (RBF),

$$K(\boldsymbol{x}, \boldsymbol{x}_i) = \exp\{-\|\boldsymbol{x} - \boldsymbol{x}_i\|_2^2 / \sigma^2\} \qquad (38)$$

with $\sigma = 1.77$, is chosen as the kernel function in the experiment. Predictive horizon and control horizon are set to 10 and 5, respectively. Other parameters are regulated to $\gamma = 5$, $n_u = 3$, $n_y = 3$, and size of training set $NN = 150$ for the simulation. The expectation of output is the sum of some step signals. Fig. 1 shows the tracking results of LS-SVM of 500 sampling period where y is the actual output, y_r is the setting output, and u is the control imposed on the system.



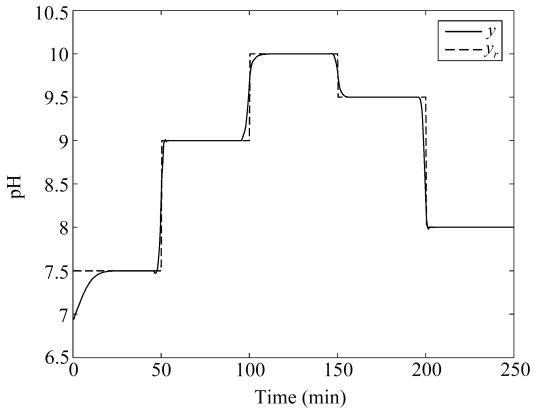
(a) Tracking curves



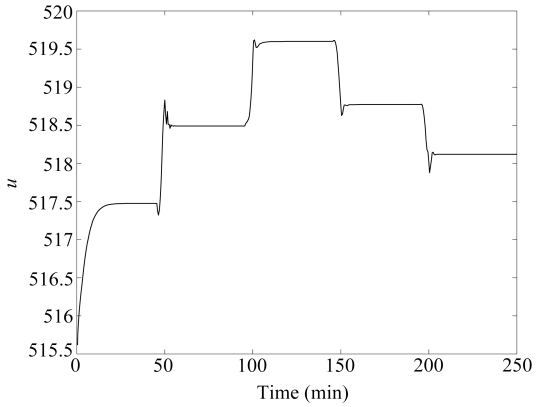
(b) Control output u

Fig. 1 The result of GPC based on online LS-SVM

GPC control with online FLS-SVM on the pH neutralizing process with the parameter $\delta = 0.1$ in (8) is also simulated in our experiments. The results are shown in Fig. 2 and the parameters in them have the same meaning as in Fig. 1.



(a) Tracking curves



(b) Control output u

Fig. 2 The result of GPC based on online FLS-SVM

The following expression is defined as the approximate performance in our experiments,

$$RMS = \frac{\|Y - Yr\|_2}{\sqrt{N}} \tag{39}$$

where Y is the vector of actual output, Yr is the expectation output vector, and N is the length of Y. The smaller value of RMS denotes the higher precision.

In our experiments, $\lambda(j)$ in (31) is regarded as a constant, and a group of values are tested both in the LS-SVM and FLS-SVM GPC methods. The comparison of RMS is shown in Table 2 and the mean computing time is shown in Table 3.

Table 2 Comparison of RMS

	$\lambda = 0.1$	$\lambda = 0.3$	$\lambda = 0.5$	$\lambda = 2.5$	$\lambda = 10$
LS-SVM	0.2036	0.1197	0.1327	0.1770	0.2025
FLS-SVM	0.1889	0.1220	0.1353	0.1773	0.1969

Table 3 Comparison of mean computing time

	$\lambda = 0.1$	$\lambda = 0.3$	$\lambda = 0.5$	$\lambda = 2.5$	$\lambda = 10$
LS-SVM(s)	0.025 9	0.028 4	0.027 7	0.031 0	0.031 6
FLS-SVM(s)	0.028 6	0.033 2	0.030 8	0.032 4	0.033 3

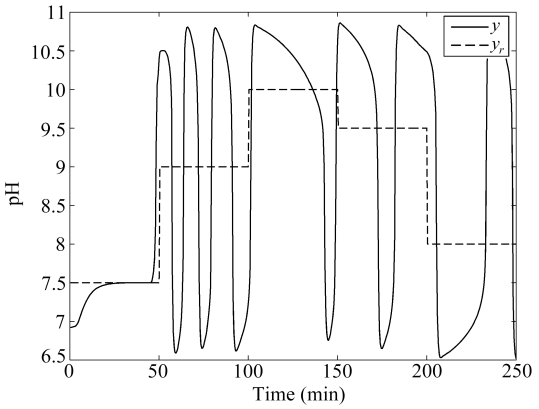
It can be seen from Figs. 1 and 2 that the online LS-SVM and FLS-SVM GPC control strategies both can track the set values quickly, stably, and accurately.

Tables 2 and 3 indicate that by selecting appropriate size of training set, the online modeling time is short enough for most industrial system and the tracking precision is satisfactory for such a strongly nonlinear process. Seen from Table 3, the computing time is longer in FLS-SVM than in LS-SVM for one more run of the recursive step in FLS-SVM. However, seeing from Table 2, we could find that, not as the effect in [8] which shows the advantage of offline FLS-SVM in modeling precision, the control precision of FLS-SVM is not so higher, but it is even lower at some experiments than that of LS-SVM. For online LS-SVM GPC algorithm, new information is continually added and the least important data is eliminated recursively, which works as the effect of weights in FLS-SVM, and consequently the advantage of FLS-SVM is not so distinct. In this sense, the FLS-SVM method is not so valuable in online GPC algorithm at the cost of computing time.

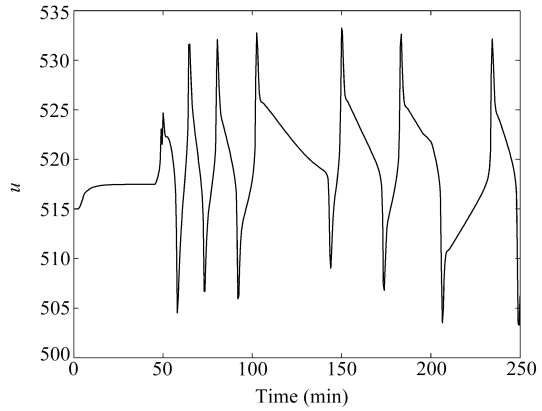
5.2 Comparison of GPC with online least squares modeling

We also do the experiment of generalized predictive control on pH neutralizing process with online least squares modeling. The control curves are shown in Fig. 3 where the parameters have the same meaning as in Fig. 1.

It can be seen from Fig. 3 that in the first 50 minutes the pH neutralizing process runs steadily while in the rest time the output oscillates severely. As having been explained, pH neutralizing process exhibits strong nonlinearity at the vicinity of pH = 9, which results in the severe oscillation of the output in the rest time after 50 min. Namely, GPC with online least squares can control the linear systems perfectly whereas it is incompetent once the output is in the neighborhood of pH = 9 (exhibiting nonlinearity).



(a) Tracking curves



(b) Control output u

Fig. 3 The result of GPC with online least squares

Let us fall back on the linearizing formula (36) of LS-SVM modeling. With RBF kennel function (38) and $n_u + n_y = n$ (for the sake of writing), (36) can be written as

$$y(\boldsymbol{x}) = C + \frac{\partial y}{\partial \boldsymbol{x}(1)}\bigg|_{\boldsymbol{x}=\boldsymbol{x}_k} \boldsymbol{x}(1) + \cdots + \frac{\partial y}{\partial \boldsymbol{x}(n)}\bigg|_{\boldsymbol{x}=\boldsymbol{x}_k} \boldsymbol{x}(n) =$$

$$C - 2\sum_{i=1}^{N} \frac{\alpha_i K(\boldsymbol{x}_k, \boldsymbol{x}_i)[\boldsymbol{x}_k(1) - \boldsymbol{x}_i(1)]}{\sigma^2} \boldsymbol{x}(1) - \cdots -$$

$$2\sum_{i=1}^{N} \frac{\alpha_i K(\boldsymbol{x}_k, \boldsymbol{x}_i)[\boldsymbol{x}_k(n) - \boldsymbol{x}_i(n)]}{\sigma^2} \boldsymbol{x}(n) =$$

$$C + b_1\boldsymbol{x}(1) + \cdots + b_{n_u}\boldsymbol{x}(n_u) - a_1\boldsymbol{x}(n_u + 1) - \cdots - a_{n_y}\boldsymbol{x}(n) \tag{40}$$

It is easy to find from (40) that parameters $\{b_1, \cdots, b_{n_u}, a_1, \cdots, a_{n_u}\}$ are obtained by mapping the inputs into the feature space with much higher dimension and thus they are provided with more freedom and generalization than least squares algorithm to depict the training data. Therefore, GPC with online LS-SVM is more capable of dealing with nonlinearity than online least squares algorithm.

6 Conclusion

This paper proposes a practical online GPC algorithm based on LS-SVM and FLS-SVM which can online model and control the nonlinear system effectively. By recursively adding new information and deleting useless data of the system, the presented algorithm can construct nonlinear model with high precision and satisfy the demand of realtime property. The paper gives the recursive algorithm of model parameters and corresponding deducing process. The nonlinear model is linearized at each sampling period and the whole online LS-SVM GPC method is formed.

The proposed algorithm is applied to the experiments of generalized predictive control of pH neutralizing process. In this paper, the tracking and control curves of the result is shown, the contrast between LS-SVM and FLS-SVM in precision and computing time is discussed, and also the comparison of GPC with LS-SVM to GPC with least squares is given. The experimental results show the effectiveness and practicality of the presented algorithm.

References

1 Clarke D W, Mohtadi C, Tuffs P S. Generalized predictive control part I: the basic algorithm. *Automatica*, 1987, **23**(2): 137∼148

2 Qin S J, Badgwell T A. A survey of industrial model predictive control technology. *Control Engineering Practice*, 2003, **11**(7): 733∼764

3 Darryl D, Martin G. A new real-time perspective on nonlinear model predictive control. *Journal of Process Control*, 2006, **16**(6): 635∼643

4 Vapnik V. *Statistical Learning Theory*. New York: John Wiley, 1998

5 Alex J S, SchÄokopf B. A tutorial on support vector regression [Online], available: http://www.neurocolt.com/tech reps/1998/98030.ps.gz, September 10, 2005

6 Suykens J A K, Vandewalle J. Least squares support vector machines classiffers. *Neural Processing Letters*, 1999, **9**(3): 293∼300

7 Lin C, Wang S. Fuzzy support vector machines. *IEEE Transactions on Neural Networks*, 2002, **13**(2): 464∼471

8 Li L, Su H, Chu J. *Lectur Notes in Computer Science*. Chongqing, China: Springer, 2006. 1275∼1281

9 Liu Bin, Su Hong-Ye, Chu Jian. Predictive control algorithm based on least squares support vector machines. *Control and Decision*, 2004, **19**(12): 1439∼1402 (in Chinese)

10 Suykens J A K, Lukas L, Dooren V, Moor B D, Vandewalle J. Least squares support vector machines classiffers: a large scale algorithm. In: Proceedings of European Conference on Circuit Theory and Design. Italy: Stresa, 1999. 839∼842

11 Suykens J A K, Lukas L, Vandewalle J. Sparse approximation using least squares support vector machines. In: Proceedings of IEEE International Symposium on Circuits and Systems. Geneva, Swizerland: IEEE, 2000. 757∼760

12 Ljung L. *System Identification Theory for the User*. Sweden: Prentice Hall PTR, 1999. 364∼364

13 Csato L. Gaussian Processes-iterative Sparse Approximations [Ph. D. dissertation], University of Aston, 2002

14 Nie J H, Loh A P, Hang C C. Modeling pH neutralization process using fuzzy-neutral approaches. *Fuzzy Set and Systems*, 1996, **78**(1): 5∼22

LI Li-Juan Ph. D. candidate in Institute of Advanced Process Control at Zhejiang University. She received her bachelor and master degrees from Nanjing University of Technology in 1997 and 2004, respectively. Her research interest covers predictive control, nonlinear identification and modeling. She is also a lecturer in College of Automation at Nanjing University of Technology. Corresponding author of this paper. E-mail: ljli@iipc.zju.edu.cn

SU Hong-Ye Professor in Institute of Advanced Process Control at Zhejiang University. He received his bachelor degree from Nanjing University of Chemical Technology in 1990, master and Ph. D. degrees from Zhejiang University in 1993 and 1995, respectively. His research interest covers robust control, time-delay systems, nonlinear systems, DEDS, and advanced process control theory and application.
E-mail: hysu@iipc.zju.edu.cn

CHU Jian Professor in Institute of Advanced Process Control at Zhejiang University. He received his bachelor degree in chemical engineering from Zhejiang University in 1982 and Ph. D. degree from Kyoto University in 1989, respectively. His research interest covers time-delay system, nonlinear contro1, and robust control and application. E-mail: chuj@iipc.zju.edu.cn