Proceedings of the 17th World Congress
The International Federation of Automatic Control
Seoul, Korea, July 6-11, 2008

Adaptive Control of a Class of Nonlinear Discrete-Time Systems with Online Kernel Learning

Yanchen Gao[1,2], Yi Liu[1], Haiqing Wang[1], Ping Li[1]

1. State Key Laboratory of Industrial Control Technology, Institute of Industrial Process Control Zhejiang University, Hangzhou, 310027, P. R. China (Tel: 086-571-8795-1442-810; e-mail: hqwang@ iipc.zju.edu.cn).
2. Qingdao Mesnac Co., LTD., Qingdao, 266045, P. R. China (e-mail: gaoyc@mesnac.com).

Abstract: An Online Kernel Learning based Adaptive Control (OKL-AC) framework for discrete-time affine nonlinear systems is presented in this paper. A sparsity strategy is proposed to control the complexity of OKL identification model, meanwhile to make a trade-off between the demanded tracking precision and the complexity of the control law. The forward increasing and backward decreasing learning stages are performed, both incorporating efficient recursive updating algorithms. Owing to these advantages, the adaptive control law based on the OKL identification model is easily obtained and can be efficiently updated. Numerical simulations show that the proposed simple OKL-AC strategy has satisfactory performance, including good tracking performance and fast learning ability, in both deterministic and stochastic environments.

1. INTRODUCTION

Neural networks based adaptive control techniques for nonlinear processes have been intensively studied in the last decade (Chen and Kahlil, 1995; Hunt *et al*., 1992). However, there are still some drawbacks of neural networks, such as weak generalization ability. And there are still no guarantees of avoidance of local minima and the "over-fitting" phenomenon, meanwhile no general methods to choose the number of hidden units for common neural networks. Furthermore, a large number of samples should be required for training a neural network (Schölkopf and Smola, 2002).

Recently, support vector machine (SVM), which is a new powerful machine learning method based on statistical learning theory (SLT) and kernel learning (KL) technique, is gaining widespread attention in the field of machine learning, and nonlinear process modeling (Schölkopf and Smola, 2002; Suykens *et al*., 2002; Wang *et al*., 2006). Some SVM model based nonlinear control algorithms have been proposed just recently (Iplikci, 2006; Xu *et al*., 2005). However, there are some technical difficulties in these new control schemes, e.g., most of them are computational expensive. Furthermore, most of the SVM model based nonlinear control algorithms utilize the offline identification model, and fewer studies have focused on online SVM or KL identification methods. To our knowledge, the research of online SVM or KL identification techniques has little overt conceptual overlap with adaptive control.

A new Online Kernel Learning based Adaptive Control (OKL-AC) algorithm for discrete-time affine nonlinear systems in this paper. The structure is as follows. The basic framework of OKL-AC is formulated in Section 2. And a sparsity approach which can adaptively control the complexity for both nonlinear system identification and control issues is given in Section 3, following a forward increasing for incorporating a new node and a backward decreasing for forgetting an old one with quickly online updating are detailed discussed. In Section 4, the simulations are illustrated to show the performance of the new control law. The conclusions are drawn in the final Section.

This work was supported by the National Natural Science Foundation of China (No. 20576116) and Alexander von Humboldt Foundation, Germany (Dr. Haiqing Wang). Corresponding author: Haiqing Wang.

2. ONLINE KERNEL LEARNING BASED ADAPTIVE CONTROL

A class of single-input-single-output (SISO) nonlinear systems can be accurately represented by the following discrete model

$$y(k+1) = f\left[\mathbf{x}(k)\right] + g\left[\mathbf{x}(k)\right]u(k) \tag{1}$$

where k is the discrete time, and $\mathbf{x}(k)=[\mathbf{Y}(k), \mathbf{U}(k\text{-}1)]$, which can be considered as the regressor vector, consists of $\mathbf{Y}(k)=[y(k), \ldots, y(k\text{-}n_y\text{+}1)]$ and $\mathbf{U}(k)=[u(k\text{-}1), \ldots, u(k\text{-}n_u\text{+}1)]$. $y(k)$ and $u(k)$ represent the controlled output and the manipulated input, respectively, and n_y and n_u denote the process orders. $f(\cdot)$ and $g(\cdot)$ are smooth (i.e., infinitely differentiable) nonlinear functions, and $g(\cdot)$ is bounded away from zero (Chen and Kahlil, 1995).

Let $y_r(k)$ be the process desired output. Then the control law can be obtained as follows

$$u(k) = \frac{y_r(k+1) - f\left[\mathbf{x}(k)\right]}{g\left[\mathbf{x}(k)\right]} \tag{2}$$

To implement the above control law, quantitative of $f[\mathbf{x}(k)]$ and $g[\mathbf{x}(k)]$ must be calculated online. It is necessary to

estimate them efficiently. As one of the representative solutions, neural networks were used to provide them to formulate the control law (Chen and Kahlil, 1995).

However, neural networks are generally not parsimonious and hence any adaptive control schemes based on them have to deal with the issue of updating a large number of weights. On the contrary, a KL identification model can describe the nonlinear system well and exhibit good generalization ability, especially with few samples (Schölkopf and Smola, 2002).

The basic idea of SVM and KL framework for function approximation is, at first, to project the input vectors into a high-dimensional feature space, the so called *Reproducing Kernel Hilbert Space*, by a nonlinear mapping and then to perform a linear regression in this feature space. The "*kernel trick*" is adopted to encounter the curse-of-dimensionality problem. (Schölkopf and Smola, 2002).

Thus an OKL method is presented to identify $f[\mathbf{x}(k)]$ and $g[\mathbf{x}(k)]$, meanwhile the control law can be obtained. Based on SLT and KL theory, $f[\mathbf{x}(k)]$ and $g[\mathbf{x}(k)]$ can be formulated by KL identification models, respectively

$$f_m\left[\mathbf{x}(k)\right]=\left\langle \mathbf{w}_f(k),\phi_f\left[\mathbf{x}(k)\right]\right\rangle \tag{3}$$

$$g_m\left[\mathbf{x}(k)\right]=\left\langle \mathbf{w}_g(k),\phi_g\left[\mathbf{x}(k)\right]\right\rangle \tag{4}$$

where the nonlinear operators ϕ_f and ϕ_f are potentially infinite dimensional feature maps; $\mathbf{w}_f(k)$ and $\mathbf{w}_g(k)$ are the corresponding parameter vectors at time k. Then the one-step-ahead prediction of $y(k+1)$ and the control law $u(k)$ can be obtained below

$$y_m(k+1)=\left\langle \mathbf{w}_f(k),\phi_f\left[\mathbf{x}(k)\right]\right\rangle+\left\langle \mathbf{w}_g(k),\phi_g\left[\mathbf{x}(k)\right]\right\rangle u(k) \tag{5}$$

$$u(k)=\frac{y_r(k+1)-\left\langle \mathbf{w}_f(k),\phi_f\left[\mathbf{x}(k)\right]\right\rangle}{\left\langle \mathbf{w}_g(k),\phi_g\left[\mathbf{x}(k)\right]\right\rangle} \tag{6}$$

The error of plant output and the KL model output at time k is defined as

$$e(k)=y(k)-y_m(k) \tag{7}$$

Based on the philosophy of SLT and KL methods, the following optimization problem, which uses Tihonov regularization (Schölkopf and Smola, 2002), is proposed here to get the solutions $f_m[\mathbf{x}(k)]$ and $g_m[\mathbf{x}(k)]$ in (3) and (4)

$$J=\frac{1}{2}\left\|\mathbf{e}(k)\right\|^2+\gamma\left[\Omega\left(\|f\|\right)+\Omega\left(\|g\|\right)\right] \tag{8}$$

$$\begin{aligned}&\text{s.t. } y(i)=\left\langle \mathbf{w}_f(k),\phi_f\left[\mathbf{x}(i-1)\right]\right\rangle\\&+\left\langle \mathbf{w}_g(k),\phi_g\left[\mathbf{x}(i-1)\right]\right\rangle u(i-1)+e(i),\ i=1,\cdots,k\end{aligned} \tag{9}$$

where $\mathbf{e}(k)=\left[e(1),\cdots,e(k)\right]^T$; γ>0 is the regularization parameter to control the smoothness of the solution and $\left[\Omega\left(\|f\|\right)+\Omega\left(\|g\|\right)\right]$ is the regularization term (also referred as the penalty term), which is chosen to be convex, e.g., $\left(\left\|\mathbf{w}_f(k)\right\|^2+\left\|\mathbf{w}_g(k)\right\|^2\right)/2$. Notice that only a single learning machine is formulated here, both $f_m[\mathbf{x}(k)]$ and $g_m[\mathbf{x}(k)]$ can be simultaneously obtained by solving (8). Thus it is more simple than neural networks based method, where two neural networks have to be designed to get the solutions of $f_m[\mathbf{x}(k)]$ and $g_m[\mathbf{x}(k)]$, respectively.

Let us derive the dual problem for solving this optimization problem. The Lagrangian for the problem is

$$\begin{aligned}&L\left(\mathbf{w}_f,\mathbf{w}_g,\mathbf{e},\boldsymbol{\alpha}(k)\right)=\left[\left\|\mathbf{e}(k)\right\|^2+\gamma\left(\left\|\mathbf{w}_f(k)\right\|^2+\left\|\mathbf{w}_g(k)\right\|^2\right)\right]/2\\&-\sum_{i=1}^{k}\alpha_i(k)\begin{Bmatrix}\left\langle \mathbf{w}_f(k),\phi_f\left[\mathbf{x}(i-1)\right]\right\rangle\\+\left\langle \mathbf{w}_g(k),\phi_g\left[\mathbf{x}(i-1)\right]\right\rangle u(i-1)+e(i)-y(i)\end{Bmatrix}\\&i=1,\cdots,k\end{aligned} \tag{10}$$

where $\alpha_i(k)$ are Lagrange multipliers. The conditions for optimality are given by

$$\begin{cases}\dfrac{\partial L}{\partial \mathbf{w}_f(k)}=0\rightarrow \mathbf{w}_f(k)=\dfrac{1}{\gamma}\sum_{i=1}^{k}\alpha_i(k)\phi_f\left[\mathbf{x}(i-1)\right]\\ \dfrac{\partial L}{\partial \mathbf{w}_g(k)}=0\rightarrow \mathbf{w}_g(k)=\dfrac{1}{\gamma}\sum_{i=1}^{k}\alpha_i(k)\phi_g\left[\mathbf{x}(i-1)\right]u(i-1)\\ \dfrac{\partial L}{\partial e(i)}=0\rightarrow \alpha_i(k)=e(i)\qquad i=1,\cdots,k\\ \dfrac{\partial L}{\partial \alpha_i(k)}=0\rightarrow \left\langle \mathbf{w}_f(k),\phi_f\left[\mathbf{x}(i-1)\right]\right\rangle+\left\langle \mathbf{w}_g(k),\phi_g\left[\mathbf{x}(i-1)\right]\right\rangle\\ \qquad\qquad u(i-1)+e(i)-y(i)=0\ \ i=1,\cdots,k\end{cases} \tag{11}$$

After elimination of the variables $\mathbf{w}_f(k)$ and $\mathbf{w}_g(k)$ and $\mathbf{e}(k)$, one gets the following solution

$$\left[\mathbf{Q}(k)/\gamma+\mathbf{I}_k\right]\boldsymbol{\alpha}(k)=\mathbf{y}(k) \tag{12}$$

where $\mathbf{I}_k\in \mathrm{R}^{k\times k}$ is the identity matrix, $\boldsymbol{\alpha}(k)=\left[\alpha_1(k),\cdots,\alpha_k(k)\right]^T$, $\mathbf{y}(k)=\left[y(1),\cdots,y(k)\right]^T$, and $\mathbf{Q}_{ij}(k)=\mathbf{K}_{f,ij}(k)+\mathbf{K}_{g,ij}(k)u(i-1)u(j-1)$. The "*kernel trick*" (Schölkopf and Smola, 2002) applied here is

$$\begin{aligned}&\mathbf{K}_{f,ij}(k)=\left\langle \phi_f\left(\mathbf{x}(i-1)\right),\phi_f\left(\mathbf{x}(j-1)\right)\right\rangle\\&\mathbf{K}_{g,ij}(k)=\left\langle \phi_g\left(\mathbf{x}(i-1)\right),\phi_g\left(\mathbf{x}(j-1)\right)\right\rangle,\ \forall i,j=1,\cdots,k\end{aligned} \tag{13}$$

Then the one-step-ahead estimation of KL identification model can be obtained

$$y_m(k+1)=\frac{1}{\gamma}\sum_{i=1}^{k}\alpha_i(k)\left[\mathbf{k}_f(k)+\mathbf{k}_g(k)u(k)\right] \tag{14}$$

where $\mathbf{k}_f(k)=\left\langle \phi_f\left(\mathbf{x}(i-1)\right),\phi_f\left(\mathbf{x}(k)\right)\right\rangle$ and $\mathbf{k}_g(k)=\left\langle \phi_g\left(\mathbf{x}(i-1)\right),\phi_g\left(\mathbf{x}(k)\right)\right\rangle$, i=1, …, k. The control law is consequently formulated below

$$u(k)=\frac{\gamma y_r(k+1)-\sum_{i=1}^{k}\alpha_i(k)\mathbf{k}_f(k)}{\sum_{i=1}^{k}\alpha_i(k)\mathbf{k}_g(k)} \tag{15}$$

In summary, the OKL method based adaptive control framework for SISO affine nonlinear system amounts to solving a set of linear equations in the high dimensional feature space introduced by the kernel transform. However, there still exist some adverse factors, as remarked by Wang *et al*. (2006), which make the solving of (12) infeasible.

1) The length of parameter vectors $\mathbf{w}_f(k)$ and $\mathbf{w}_g(k)$, which can be considered as the order of the KL identification model, are both equal to the number of the identification data used. It is similar with least squares SVM (LS-SVM) (Suykens *et al*., 2002). It means that, with the identification continuing online, the complexity of the model will be increasing steadily. Meanwhile, the computation scale of the control law will become larger gradually. That is computational impracticable.

2) The projections of input samples at different time instances (e.g., when the system is in steady-state) might be linear dependent in the feature space and thus may cause the solving of $\boldsymbol{\alpha}(k)$ numerical unstable.

3. COMPLEXITY CONTROL AND RECURSIVE FORMS

3.1 Sparsity Strategy

To overcome the embarrassment mentioned above, the OKL framework above should be improved. An OKL that can adaptively control the complexity is formulated in this section for both nonlinear system identification and control issues. In this contribution, the samples used in the KL identification model are referred to as "nodes", just as proposed by Wang *et al*. (2006). The main motivation is to find as few nodes as possible, which can be utilized to identify an exact OKL model meanwhile with good generalization ability. That is to obtain a sparse OKL identification model.

There are two main strategies to obtain the sparsity: pre-sparsity and post-sparsity. The former is to control the complexity of the learning machine and is suitable for online learning; and the latter is to increase the speed/efficiency of later testing, which is always in batch learning (Wang *et al*., 2006). An interesting pre-sparsity strategy for online KL has been proposed recently by Wang *et al*. (2006), which uses a so-called "space angle index" to judge whether the mapped features are approximately linear independent. In this contribution, a simpler sparsity approach is proposed. The criterion for adding a new pair of sample $[\mathbf{x}(k), y(k+1)]$ to the learning machine is as follows:

$$\left|e(k+1)\right| = \left|y(k+1) - y_m(k+1)\right| > E_{tol} \tag{16}$$

where E_{tol} is a predefined small positive value. The basic idea of this complexity-controlled strategy for the learning machine (that is also the identification model) is of simplicity and intuition:

1) If $e(k+1) > E_{tol}$ holds, which means the approximation error between the actual output and prediction of the learning machine is significant, the OKL identification model is not accurate enough and should be improved, so $[\mathbf{x}(k), y(k+1)]$ will be introduced as a new node;

2) Otherwise, if $\left|e(k+1)\right| \le E_{tol}$ is true, which implies that the learning machine is always satisfied and there is not necessary to add the node in accordance with the well-known "*parsimony principle*".

The criterion for adding a new node proposed here does not adopt the colinearity concept as used in Wang *et al*. (2006). This method, however, directly utilizes the prediction error $e(k+1)$ as the criterion. Consequently, another advantage of this criterion is that it connects itself to the control problem close. Generally, if the tracking performance of a control target is required more precision, E_{tol} can be set smaller and vice versa. This is because the control task is to make the difference between the system output y and the reference trajectory y_r as small as possible. A smaller E_{tol} gets more nodes, and a larger one yields a more parsimonious but less precise learning machine. Thus, E_{tol} can be easily selected for the general identification and control issues.

This simple sparsity approach makes the complexity of the learning machine restrained, further, trades off the tracking performance of the controller and the learning machine. From a practical point of view, the computational scale is also very small. Note that this sparsity belongs to pre-sparsity approach (Wang *et al*., 2006) and is different from the basic idea of SVM, where the sparsity is obtained after optimization.

3.2 Forward Incremental Learning

Assume at time k the OKL identification model has N_k (at least one) nodes due to the sparsity criterion, and then gets the following equation

$$\left[\mathbf{Q}_{N_k}/\gamma + \mathbf{I}_{N_k}\right]\boldsymbol{\alpha}_{N_k} = \mathbf{y}_{N_k} \tag{17}$$

Note that the above equation has related terms similarly defined in (12), except that the nodes are different. For simplicity, the quantities are defined as $\mathbf{H}_{N_k} = \mathbf{Q}_{N_k}/\gamma + \mathbf{I}_{N_k}$ and $\mathbf{P}_{N_k} = \mathbf{H}_{N_k}^{-1}$, then yields the solution

$$\boldsymbol{\alpha}_{N_k} = \mathbf{P}_{N_k}\mathbf{y}_{N_k} \tag{18}$$

When a new node is added into the OKL model, (17) becomes

$$\begin{bmatrix} \mathbf{H}_{N_k} & \mathbf{V}_{N_{k+1}} \\ \mathbf{V}_{N_{k+1}}^T & v_{N_{k+1}} \end{bmatrix}\begin{bmatrix} \boldsymbol{\alpha}_{N_k} \\ \alpha_{N_{k+1}} \end{bmatrix} = \begin{bmatrix} \mathbf{y}_{N_k} \\ y_{N_{k+1}} \end{bmatrix} \tag{19}$$

where $\mathbf{V}_{N_{k+1}} = \left(\mathbf{V}_{f,N_{k+1}} + \mathbf{V}_{g,N_{k+1}}\right)/\gamma$,

$\mathbf{V}_{f,N_{k+1}} = \left[K\left(\mathbf{x}_{N_1}, \mathbf{x}_{N_{k+1}}\right), \cdots, K\left(\mathbf{x}_{N_k}, \mathbf{x}_{N_{k+1}}\right)\right]^T$ and

$\mathbf{V}_{g,N_{k+1}} = \left[K\left(\mathbf{x}_{N_1}, \mathbf{x}_{N_{k+1}}\right)u_{N_1}u_{N_{k+1}}, \cdots, K\left(\mathbf{x}_{N_k}, \mathbf{x}_{N_{k+1}}\right)u_{N_k}u_{N_{k+1}}\right]^T$

are corresponding kernel vectors of the new node, and $v_{N_{k+1}} = \left[1 + K\left(\mathbf{x}_{N_{k+1}}, \mathbf{x}_{N_{k+1}}\right)\right]u(k)u(k)/\gamma + 1$ is a scalar.

Applying the Sherman-Morrison-Woodbury formual (Golub and Van Loan, 1996) to (19) yields the following inverse relation between matrices computation of the inverse $\mathbf{P}_{N_{k+1}}$ and $\mathbf{P}_{N_k}$

$$\mathbf{P}_{N_{k+1}} = \begin{bmatrix} \mathbf{P}_{N_k} & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix} + \mathbf{r}_{N_{k+1}} \mathbf{r}_{N_{k+1}}^T z_{N_{k+1}} \tag{20}$$

where $\mathbf{r}_{N_{k+1}} = \left[\mathbf{V}_{N_{k+1}}^T \mathbf{P}_{N_{k+1}}, -1 \right]^T$ is a column vector and $z_{N_{k+1}} = 1 \Big/ \left(v_{N_{k+1}} - \mathbf{V}_{N_{k+1}}^T \mathbf{P}_{N_{k+1}} \mathbf{V}_{N_{k+1}} \right)$ is a scalar.

The recursive update algorithm of the forward incremental learning stage is efficient. Whenever a new node is available, the direct computation of the inverse of the matrix $\mathbf{H}_{N_{k+1}}$ requires about $O\left(N_{k+1}^3\right)$ operations, whereas the recursive update algorithms only need $O\left(N_{k+1}^2\right)$ operations. The improvement on computing speed is extremely noticeable when the number of nodes N_{k+1} becomes large.

3.3 Backward Decremental Learning

The aim of backward decremental learning, also referred as *pruning*, is to recursively delete the old information (Suykens *et al*., 2002; Wang *et al*., 2006). The issue of pruning methods in the batch learning of SVM has received a deal of attention (Suykens *et al*., 2002), however with little research for online learning. Let the symbol N as the memory length. Assume at time k the node growth of the OKL identification model is finished and N_{k+1} is larger than N. The simplest pruning approach is to delete the first node, for it is considered as the oldest one and with the least information for the learning machine (Wang *et al*., 2006). However, there is no guarantee on the rationality of this intuitional pruning approach.

From the optimality conditions in (11) one can infer that the nodes with small Lagrange multiplies also have the small error. Similar with the pruning method proposed by Suykens *et al*. (2002), the nodes with small Lagrange multiplies are deleted. However, Suykens *et al*. (2002) removed all the support vectors with small coefficients α_i (below some threshold value) and retrained the learning machine. Thus it is not suitable for online learning due to the intensive computations. In our approach, only one node is pruned at a time, furthermore, a recursive update algorithm is adopted to avoid the computation of the matrix inverse $\mathbf{P}_{N_{k+1}}$. The pruning procedure includes two steps. Once $N_{k+1} > N$ holds, first find out the smallest Lagrange multiple as follows

$$\arg\min \left| \alpha_{i,N_{k+1}} \right| \quad i = 1, \cdots, N, N_{k+1} \tag{21}$$

Then, when the l-th node is pruned from the OKL identification model, the update rule is formulated as

$$\begin{cases} P_{N,i,j} & \leftarrow P_{i,j} - P_{l,l}^{-1} P_{i,l} P_{l,j} \quad \forall i, j = 1, \cdots, l-1 \\ P_{N,i-1,j-1} & \leftarrow P_{i,j} - P_{l,l}^{-1} P_{i,l} P_{l,j} \quad \forall i, j = l+1, \cdots, N_{k+1} \end{cases} \tag{22}$$

where $P_{i,j}$ and $P_{N,i,j}$ stand for the items at the i-th row and j-th column of $\mathbf{P}_{N_{k+1}}$ and $\mathbf{P}_N$, respectively. If l equals one, only the second formulation of (22) is calculated. This updated procedure was developed for the incremental SVM algorithm (Cauwenberghs and Poggio, 2001). According to (22), $\mathbf{P}_N$ can be efficiently updated from $\mathbf{P}_{N_{k+1}}$ without explicitly computing the matrix inverse.

Consequently, the OKL identification model can be efficiently updated, including both forward incremental learning and backward pruning. And these recursive algorithms avoid the direct computation of the inverse of the matrix; owing this feature, the adaptive control law (that is OKL-AC) is obtained with small computation scale.

4. SIMULATIONS

To verify the validity of the proposed OKL-AC algorithm, consider an unknown nonlinear discrete-time system

$$\begin{aligned} y(k+1) = & \frac{1.5\, y(k)\, y(k-1)}{1 + y^2(k) + y^2(k-1)} \\ & + 0.35 \sin\left[y(k) + y(k-1) \right] + 1.2u(k) \end{aligned} \tag{23}$$

This problem has been studied with neural networks based adaptive control strategy and LS-SVM based adaptive control (Chen and Kahlil, 1995; Xu *et al*., 2005). However, the sparseness of LS-SVM is lost, which will result in a computational load as mentioned previously.

The simulation environment is Matlab V7.1 with CPU main frequency 2.4GHz and 256M memory. The regressor vector is chosen as $\mathbf{x}(k)=[y(k), y(k\text{-}1)]$, and the Gaussian kernel is utilized in all simulations (Schölkopf and Smola, 2002)

$$K(\mathbf{x}_1, \mathbf{x}_2) = \exp[-\|\mathbf{x}_1 - \mathbf{x}_2\| / \sigma^2] \tag{24}$$

The simulations are composed of three parts. The first is a set-point tracking problem and different initial conditions are investigated. The second focuses on a sine wave tracking task and different values of E_{tol} are set to demonstrate its effect. The last part is to mimic an industrial environment, including both noise and disturbance.

4.1 Set point Tracking

In the first case, the set-point tracking ability of OKL-AC is investigated. There are only three parameters to be predefined, where the regularization parameter γ=0.001 and the kernel parameter σ^2=100 adopted here are chosen by simulation, and E_{tol}=0.1 is in accord with the precision of this control task. Note that there is no rigorous parameter selection theory aiming for industrial application available. Fortunately, the first two parameters both work well in a wide range.

A simple two steps instruction should be followed to choose them. First, a smaller γ is adopted when the system is in a relatively deterministic environment with less noise, and vice versa. Second, when the system to be identified is extremely nonlinear, σ^2 should be set smaller, and vice versa. Thus, the

parameters can be tuned easily, and the value provided here is just one of many parameter pairs that turn out satisfied results and no optimality is guaranteed.

The initial condition of the system is [$y(0)$, $y(-1)$]=[-3, -3]. As depicted in Fig. 1, the output of the system can track all of the different set points quickly, and the stable errors are zero or close to zero. Anyhow, the stable error is far less than E_{tol}. Another advantage of the proposed OKL-AC strategy also shown in Fig. 1 is that only 17 nodes are selected out, about 8.5% of the total training data. The running time of the whole procedure takes 0.391 s. The number of the nodes is likely to be increasing when the system is in the transient state or the set-point changes. It is important to point out that only these *key* samples are adopted in the OKL identification model, this is extraordinary different from the control law proposed by Xu *et al.* (2005), where all of the samples are fed into the learning machine, resulting a verbose identification model, a complex control law and an overload of computation.

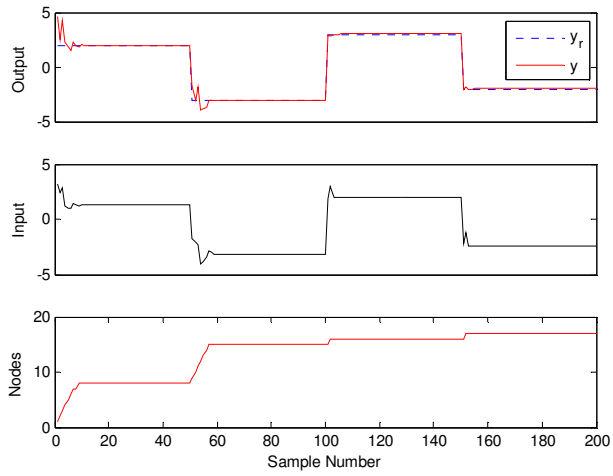

Fig. 1. Set point tracking performance of OKL-AC

Then we want to emphasize that the proposed control law can work well when initial conditions of the system are changed. Three different initial conditions are investigated: [$y(0)$, $y(-1)$]=[3, 3], [-1.5, -1.5], and [-3, -3]. To show the robustness of OKL-AC, the parameters adopted here are the same as the former case. Fig. 2 shows the system outputs of the first 50 time steps as shown in Fig. 1. Obviously, the tracking errors are observed to converge quickly, whenever any initial condition is used.

4.2 Sine Wave Tracking

In this case, a sine wave tracking problem is investigated (Chen and Kahlil, 1995; Xu *et al.*, 2005). The initial condition in this case is [$y(0)$, $y(-1)$]=[3, 3]. The parameters adopted here are γ=0.001 and σ^2=9. Two scenarios, including no pruning and pruning, are both considered.

In the scenario of no pruning, different values of E_{tol} are set to demonstrate the effect of E_{tol} on the proposed control strategy. The integral of the absolute set-point tracking error (IAE) is used here to quantify the performance characteristic. Several performance indices, including IAE, are listed in Tab. 1. It is clearly showed that a smaller E_{tol} achieves a better tracking performance and more nodes of the identification model, just as previously mentioned. Even a relatively large E_{tol} is adopted (i.e. 0.5), the tracking precision is accepted to some extent. Only 6 nodes (The sparsity ratio is 3%.) are selected out, which means an extremely sparse identification model.

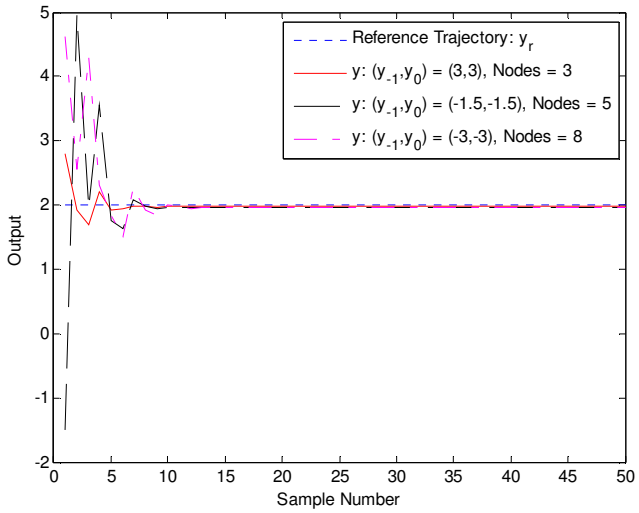

Fig. 2. Set point tracking with different initial conditions

Table 1. Control performance with different E_{tol} (no pruning)

E_{tol}	0.5	0.2	0.1	0.05
IAE	34.217	16.007	12.515	10.914
Time (s)	0.313	0.360	0.422	0.485
Nodes	6 (3%)	12 (6%)	20 (10%)	34 (17%)

For a smaller value of E_{tol}, e.g., 0.01, and a sine wave tracking task, the nodes selected out here are larger than the memory length N (Suppose N=50 in this case.). And in this scenario, the pruning stage, also referred as backward decremental learning, is performed. The OKL-AC parameters adopted here are not changed. Two kinds of pruning approaches are investigated, the former is to prune the node with the smallest Lagrange multiple, and the latter is to delete the first node (Xu *et al.*, 2005).

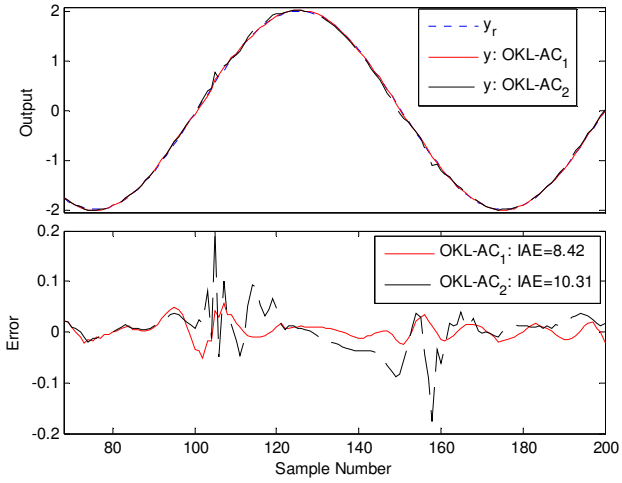

Fig. 3. Sine wave tracking and comparison with different pruning approaches

Fig. 3 depicts the details of the performance comparison of both pruning methods. For clarity, the input signals are not shown here and the plotting is only from the pruning stage. The IAE of our proposed approach is 8.42 and this procedure only takes 0.829 s, but the latter is 10.31 and takes 1.078 s.

That is because the proposed pruning method is more accurate, and there is no need to introduce a new node to the OKL identification model at all times. Thus, a conclusion can be drawn that the pruning approach proposed here is better, with smaller errors and more stable tracking performance.

Detailed performance indices of the pruning approach are provided in Tab. 2. If let memory length N=200, there is unnecessary to prune. Compared with no-pruning scenario, the pruning strategy proposed here can apparently improve the control performance, including both the tracking precision and the response time. When E_{tol}=0.005, a high tracking precision is obtained. And we can see that if E_{tol}=0, in despite of N=50 or N=200 in this scenario, the control performance becomes a bit worse. It demonstrates that a suitable E_{tol} can control the complexity of the identification model, and then make the control law more feasible.

Table 2. Control performance with Different E_{tol} (pruning)

E_{tol}	0.02		0.01		0.005		0	
N	50	200	50	200	50	200	50	200
IAE	8.89	8.98	8.42	8.57	**8.33**	8.43	8.55	8.69
Time (s)	0.70	0.78	0.83	0.97	1.12	1.24	1.20	1.77
Nodes	50	78	50	120	50	152	50	200

4.3 Reject Noise and Disturbance

To simulate the industrial environment, the output signal is corrupted by independent Gaussian noise with a variance of 0.1, furthermore, some disturbances are added into the output signal when at time k=40, 80, 120, 160, the values being 0.5 or –0.5 random. The initial condition in this case is [y(0), y(-1)]=[3, 3]. The parameters adopted here is γ=1, σ^2=9 and E_{tol}=0.5. Compared with the former cases in the deterministic environment, γ is set larger here, just as previously analyzed. To get rid of the effect caused by noise and disturbance, E_{tol} should be also set a relatively larger value.

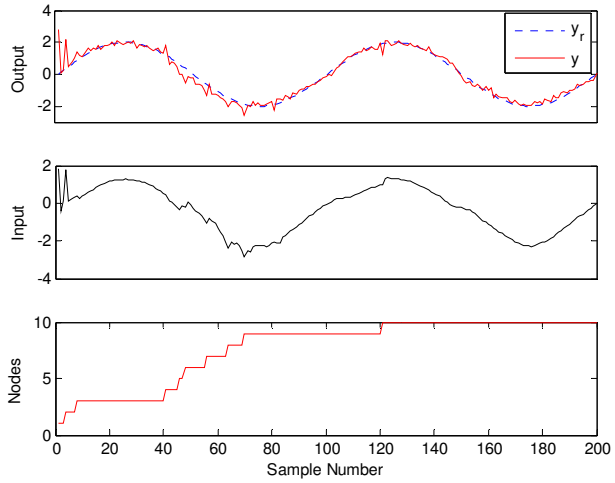

Fig. 4. Noise and disturbance rejection of OKL-AC

The result is shown in Fig. 4, in which 10 nodes are selected out. It clearly demonstrates a very concise identification model and control law, and the tracking performance is satisfactory under this noise and disturbance environment.

5. CONCLUSIONS

Recently the application of SVM and KL methods for modelling of nonlinear systems has attracted much interest since it allows one to easily obtain nonlinear control algorithms just as the considered plant was "linear". In this paper, an OKL framework that can adaptively control its complexity is addressed for both nonlinear system identification and control issues. Then the proposed OKL-AC strategy is applied to control a class of discrete-time affine nonlinear systems, with recursively incremental and decremental learning algorithms. The OKL identification model can describe the nonlinear system well and has good generalization ability using small sample set, therefore it is unsurprising that OKL based control techniques will be extended to general nonlinear systems.

REFERENCES

Cauwenberghs G. and Poggio T. (2001). Incremental and decremental support vector machine learning. *Advances Neural Information Processing Systems (NIPS 2000)*. Cambridge, MA: MIT Press.

Chen F.C. and Khalil H.K. (1995). Adaptive control of a class of nonlinear discrete-time systems using neural networks. *IEEE Trans. Autom. Control*, **40**(5): 791-801.

Golub G.H. and Van Loan C.F. (1996). *Matrix Computations*. (3rd ed.), Baltimore: The John Hopkins Univ. Press.

Hunt K.J., Sbarbaro D., Zbikowski R. et al. (1992). Neural networks for control systems—a survey. *Automatica*, **28**(6): 1083-1112.

Iplikci S. (2006). Support vector machines-based generalized predictive control. *Int. J. Robust Nonlinear Control*, **16**: 843-862.

Schölkopf B. and Smola A.J. (2002). *Learning with Kernels*. Cambridge, MA: MIT Press.

Suykens J.A.K., Van Gestel T., De Brabanter J. et al. (2002). *Least Squares Support Vector Machines*. Singapore: World Scientific.

Wang H.Q., Li P., Gao F.R. et al. (2006). Kernel classifier with adaptive structure and fixed memory for process diagnosis. *AIChE J.*, **52**(10): 3515-3531.

Xu J.Q., Wang J.J., Zhu J. et al. (2005). Adaptive control of nonlinear discrete-time system by least square support vector machine. *Proceedings of the Fourth International Conference on Machine Learning and Cybernetics*, **1-9**: 544-548.