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

A Recursive Method of Identification of Hammerstein Model Based on Least Squares Support Vector Machines

Kun Chen, Haiqing Wang*, Zhihuan Song

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)

Abstract: In the domain of industrial process modeling and control, Hammerstein model has been used widely to describe a class of nonlinear systems. Goethals et al. (2005) proposed a method based on Least Squares Support Vector Machines (LSSVM) to identify the input-output relationship of the Hammerstein model. Unfortunately, as the data points grow, this kernel learning approach costs much time correspondingly. Besides, Goethals's technique is not suitable for the on-line identification. To this end, a recursive nonlinear identification method is proposed in this paper. The basic idea is to get the recursive form of the parts of the high-dimensional matrix arisen from the optimization derivation, and get the estimation with the trick of sub-inverse matrix. With this new LSSVM approach, the Hammerstein model can be obtained recursively and much quickly, which is crucial to industrial applications that require on-line estimation and prediction. The simulation illustrates the validity and feasibility of the developed online identification method.

1. INTRODUCTION

Throughout the last few decades, the identification methods of the linear systems have been explored intensively. But in the field of nonlinear systems, the modelling issue is more complicated and harder. As one of common solutions, the linearization technique near the working point was utilized to transform the nonlinear system to a linear one, with the loss of the some accuracy. Some special nonlinear model structures such as Hammerstein model and Wiener model were also proposed, which are generally, composed of a nonlinear static and a linear dynamic subsystems. Here our topic is concerned with online identification problem of Hammerstein model, using the Kernel learning theory (Vapnik, 1995; Suykens et al., 2002). Related researches have shown that the Hammerstein systems can be used successfully for modelling biological process (Westwick and Kearney, 2000), chemical engineering (Eskinat et al., 1991)(for example, distillation column, heat exchanger and PH neutralization), bio-fermentation(Golden and Ydsite, 1989) and so on.

The structure of Hammerstein model is shown in Figure 1. It contains two parts: one is a static nonlinear system which is described using a nonlinear, function ***f***, and the other is a dynamic linear system. In this issue, we use discrete-time transfer function ***G(z)*** to express the linear part.

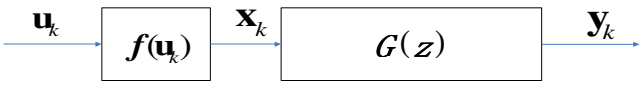


Fig. 1 The structure of Hammerstein model

This work was supported by the National Natural Science Foundation of China (No. 20776128) and Alexander von Humboldt Foundation, Germany (Dr. Haiqing Wang). *Corresponding author: Haiqing Wang.

In Fig. 1,

$$\mathbf{x}_k = \boldsymbol{f}\left(\mathbf{u}_k\right) \tag{1}$$

$$\boldsymbol{G}(z) = \frac{\mathbf{y}_k}{\mathbf{x}_k} = \frac{\boldsymbol{B}(z^{-1})}{\boldsymbol{A}(z^{-1})} \tag{2}$$

$$\mathrm{A}(z^{-1}) = 1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_n z^{-n} \tag{3}$$

$$\mathrm{B}(z^{-1}) = b_1 z^{-1} + b_2 z^{-2} + \cdots + b_m z^{-m} \tag{4}$$

where the symbols m and n are the order of numerator and denominator of the transfer function, respectively; vector elements $a_i, i = 1, \cdots, n$, $b_j, j = 1, \cdots, m$ are the corresponding parameters.

Since linear identification algorithms and standard toolboxes are easily available, the researchers focus their attention to the modelling of the nonlinear part. Known approaches include the expansion of the nonlinearity as a low-order polynomials (Eskinat et al., 1991), a sum of orthogonal (or non-orthogonal) basis functions (Narendra and Gallman,1966), a finite number of cubic spline functions (Dempsey and Westwick, 2004), piecewise linear functions (Van Pelt and Bernstein, 2000) and neural networks (Al-Duwaish and Karim, 1997). Most of them treat the linear part as a linear ARX model (AutoRegressive with eXogeneous input). No matter what method is chosen, there are some disadvantages:

- When there are only small datasets, these methods will get poor performances.
- The methods using low-order polynomials or piecewise linear functions, obviously have to sacrifice some precision and thus only get relative poor accuracy.
- Most are based on maximum-likelihood principle, resulting in a non-convex optimization problem. Most times, one can't get the global optimum estimation except that there is a proper initialization.

All the difficulties listed above show that a new identification algorithm is needed in the engineering applications. The kernel learning approach is one of the methods which meet the demand. The kernel learning theory, including Ridge Regression, Support Vector Machines (SVM) and so on, is such an identification method of nonlinear systems under small samples. According to the kernel trick, the input data is mapped into a high dimensional feature space which may be infinite dimensional. A construction of the linear separating hyperplane (in the classification problem) or linear function estimation (in the regression problem) is done in the high-dimensional feature space. Often, the technique of mapping into the feature space such as Fisher Discriminant Analysis (FDA), may cause a problem named dimension disaster. But with the kernel trick, no explicit construction of the nonlinear mapping is needed. Computations are done in another space without dimension disaster. For example, in the case of SVM or Least Squares SVM (LSSVM), one starts from a formulation in the prime space with a high-dimensional feature space by applying the nonlinear mapping, and solves the problem in the dual space. For detail, please refer to the work of Vapnik (1995) and Suykens et al. (2002).

Till now, there are only few algorithms using the kernel method to identify the Hammerstein model. In the work of Goethals et al. (2005), a new method based on was proposed to identify the ARX model of the Hammerstein systems. LSSVM is especially suitable for identification issues with only small datasets available, and just need to solve a group of (high-dimensional) linear equations during the computation, so it does not suffer from the above mentioned problems. On the other hand, the technique from Goethals et al. (2005) still can not be applied to on-line estimation. In this paper, we explore an on-line recursive method to overcome this problem.

2. HAMMERSTEIN MODEL IDENTIFICATION

2.1 Least Squares Support Vector Machines for Function Estimation

Suykens, Van Gestel et al. proposed a modification named LSSVM (Suykens et al., 2002) to the SVM theory (Vapnik, 1995). Compare to SVM, LSSVM keeps the advantage such as great generalization, small datasets identification, and changes the inequation constrains to equation constrains. So LSSVM only need to solve a set of linear equations, much easier than quadratic programs in SVM. During the modelling process, we do not use classification at all, so our attention will focus on the function estimation problems.

Let $\{(\mathbf{u}_t, \mathbf{y}_t)\}_{t=1}^N$ be the set of given input/output training data with input $\mathbf{u}_t \in \mathbf{R}^n$ and output $\mathbf{y}_t \in \mathbf{R}$. Consider the regression model:

$$\mathbf{y}_t = \boldsymbol{f}(\mathbf{u}_t) + \mathbf{e}_t = \boldsymbol{\omega}^T \varphi(\mathbf{u}_t) + c + \mathbf{e}_t \tag{5}$$

where $\boldsymbol{\omega}$ and c are a weight vector and the bias respectively; $\mathbf{e}_t$ is the residual.

According to the statistical learning theory, it can be represented as the following optimization problem:

$$\min_{\omega,b,e} \quad \mathrm{J}(\boldsymbol{\omega}, \mathbf{b}, \mathbf{e}) = \frac{1}{2}\boldsymbol{\omega}^T\boldsymbol{\omega} + \frac{\gamma}{2}\sum_{t=1}^N \mathbf{e}_t^2 \tag{6}$$

$$s.t. \quad \mathbf{y}_t = \boldsymbol{\omega}^T\varphi(\mathbf{u}_t) + c + \mathbf{e}_t, \qquad t = 1, 2, \cdots, N$$

where γ is a positive real constant. The following Lagrangian should be considered then:

$$\mathrm{L}(\boldsymbol{\omega}, \mathbf{b}, \mathbf{e}; \boldsymbol{\alpha}) = \mathrm{J}(\boldsymbol{\omega}, \mathbf{b}, \mathbf{e}) - \sum_{t=1}^N \boldsymbol{\alpha}_t \left(\boldsymbol{\omega}^T\varphi(\mathbf{u}_t) + c + \mathbf{e}_t - \mathbf{y}_t\right) \tag{7}$$

with Lagrange multipliers $\boldsymbol{\alpha}_t \geq 0$ .The conditions for optimality are given as:

$$\frac{\partial \mathrm{L}}{\partial \boldsymbol{\omega}} = 0, \ \frac{\partial \mathrm{L}}{\partial c} = 0, \ \frac{\partial \mathrm{L}}{\partial \mathbf{e}_t} = 0, \ \frac{\partial \mathrm{L}}{\partial \boldsymbol{\alpha}_t} = 0$$

and get:

$$\begin{pmatrix} \mathbf{0} & \vec{\mathbf{1}}^T \\ \vec{\mathbf{1}} & \boldsymbol{\Omega} + \gamma^{-1}\mathbf{I} \end{pmatrix} \begin{bmatrix} c \\ \boldsymbol{\alpha} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{y} \end{bmatrix} \tag{8}$$

where $\boldsymbol{\Omega}(k,l) = \varphi(\mathbf{u}_k)^T \varphi(\mathbf{u}_l) = \mathrm{K}(\mathbf{u}_k, \mathbf{u}_l)$, $\boldsymbol{\alpha} = [\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \cdots, \boldsymbol{\alpha}_N]^T$, $\mathbf{y} = [\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_N]^T$, $\vec{\mathbf{1}} = [1, 1, \cdots, 1]^T$, with $\mathrm{K}(*,*)$ the positive definite kernel functions. According to (8), we are get [c; $\boldsymbol{\alpha}$] easily. The resulting LSSVM model for function estimation can be evaluated at a new point $\mathbf{u}_t$ as:

$$\hat{\mathbf{y}}_t = \left\langle \sum_{k=1}^N \boldsymbol{\alpha}_k \varphi(\mathbf{u}_k), \varphi(\mathbf{u}_t) \right\rangle + c = \sum_{k=1}^N \boldsymbol{\alpha}_k \, \mathrm{K}(\mathbf{u}_k, \mathbf{u}_t) + c \tag{9}$$

According to (8), we can find that a matrix inversion is needed. When the training samples increases, it will cost much more time. To overcome this problem, a recursive LSSVM is proposed in the issue (Chi and Ersoy, 2003).

2.2 Identification of Nonlinear ARX Hammerstein Models

The aim of Hammerstein identification is to model the nonlinearity and to estimate the parameters of the linear systems from the input/output measurements. We first consider SISO (single input-single output) systems, and then extend to MIMO (multiple input-multiple output) systems.

For the linear dynamic part, we will assume a model structure of the ARX form (Goethals et al., 2005):

$$y_t = \sum_{i=1}^n a_i y_{t-i} + \sum_{j=0}^m b_j \, \mathrm{f}\left(u_{t-j}\right) + e_t \tag{10}$$

with $\{(\mathbf{u}_t, \mathbf{y}_t)\}_{t=1}^N$ a set of given input/output measurements. We apply the LSSVM function estimation form (5) to (10):

$$\begin{aligned} y_t &= \sum_{i=1}^{n} a_i y_{t-i} + \sum_{j=0}^{m} \mathbf{b}_j \left(\boldsymbol{\omega}^T \varphi(u_{t-j}) + c \right) + e_t \\ &= \sum_{i=1}^{n} a_i y_{t-i} + \sum_{j=0}^{m} b_j \boldsymbol{\omega}^T \varphi(u_{t-j}) + \sum_{j=0}^{m} b_j c + e_t \\ &= \sum_{i=1}^{n} a_i y_{t-i} + \sum_{j=0}^{m} \boldsymbol{\omega}_j{}^T \varphi(u_{t-j}) + d + e_t \end{aligned} \tag{11}$$

with $\boldsymbol{\omega}_j^T = b_j \boldsymbol{\omega}^T$, $d = \sum_{j=0}^{m} b_j \mathbf{c}$.

Similar to (6), (7), (11) can be considered as an optimization problem (Goethals et al., 2005):

$$\min_{\boldsymbol{\omega}_j, \mathbf{a}, \mathbf{d}, \mathbf{e}} \; \mathrm{J}(\boldsymbol{\omega}_j, \mathbf{e}) = \frac{1}{2} \sum_{j=0}^{m} \boldsymbol{\omega}_j{}^T \boldsymbol{\omega}_j + \frac{\gamma}{2} \sum_{t=r}^{N} e_t^2 \tag{12}$$

$$s.t. \begin{cases} \sum_{i=1}^{n} a_i y_{t-i} + \sum_{j=0}^{m} \boldsymbol{\omega}_j \varphi(u_{t-j}) + d + e_t - y_t = 0 \\ \sum_{t=1}^{N} \boldsymbol{\omega}_j{}^T \varphi(u_t) = 0 \end{cases}$$

Also, resorting to a Lagrangian and solving the conditions for optimality, we get:

$$\begin{bmatrix} 0 & \mathbf{0} & 1^T & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{Y}_p & \mathbf{0} \\ 1 & \mathbf{Y}_p{}^T & \tilde{\mathbf{K}} + \gamma^{-1}\mathbf{I} & \mathbf{K} \\ 0 & 0 & \mathbf{K}^T & \left(\vec{1}^T \boldsymbol{\Omega} \vec{1}\right) \mathbf{I}_{m+1} \end{bmatrix} \begin{bmatrix} d \\ \boldsymbol{\alpha} \\ \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{0} \\ \mathbf{y}_f \\ \mathbf{0} \end{bmatrix} \tag{13}$$

with $r = \max(m,n)+1$, $\boldsymbol{\alpha} = [\alpha_r, \cdots, \alpha_N]^T$, $\boldsymbol{\beta} = [\beta_0, \cdots, \beta_m]^T$, $\boldsymbol{a} = [a_1, \cdots, a_n]^T$, $\boldsymbol{\Omega}_{pq} = \varphi(u_p)^T \varphi(u_q) = \mathrm{K}(u_p, u_q)$, $\mathbf{K}_{pq} = \sum_{t=1}^{N} \boldsymbol{\Omega}_{t,r+p-q}$, $\tilde{\mathbf{K}}_{pq} = \sum_{j=0}^{m} \boldsymbol{\Omega}_{p+r-j-1, q+r-j-1}$, $\mathbf{y}_f = [y_r \cdots y_N]^T$,

$\mathbf{Y}_p = \begin{bmatrix} y_{r-1} & y_r & \cdots & y_{N-1} \\ y_{r-2} & y_{r-1} & \cdots & y_{N-2} \\ \vdots & \vdots & \ddots & \vdots \\ y_{r-\mathrm{n}} & y_{r+1-n} & \cdots & y_{N-n} \end{bmatrix}$ and $\vec{1}$ is a column vector of length (N-r+1) with elements 1.

The projection of obtained model onto (10) goes as follows. The estimations for the parameters $\boldsymbol{a} = [a_1 \cdots a_n]^T$ can be directly obtained form (13). Furthermore, the estimation of the nonlinear part can be obtained from:

$$\begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_m \end{bmatrix} \begin{bmatrix} \hat{f}(u_1) \\ \hat{f}(u_2) \\ \vdots \\ \hat{f}(u_N) \end{bmatrix}^T = \begin{bmatrix} \alpha_N & \cdots & \alpha_r & & 0 \\ & \alpha_N & \cdots & \alpha_r & \\ & & \ddots & & \ddots \\ 0 & & & \alpha_N & \cdots & \alpha_r \end{bmatrix} \times \begin{bmatrix} \boldsymbol{\Omega}_{N,1} & \boldsymbol{\Omega}_{N,2} & \cdots & \boldsymbol{\Omega}_{N,N} \\ \boldsymbol{\Omega}_{N-1,1} & \boldsymbol{\Omega}_{N-1,2} & \cdots & \boldsymbol{\Omega}_{N-1,N} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{\Omega}_{r-m,1} & \boldsymbol{\Omega}_{r-m,2} & \cdots & \boldsymbol{\Omega}_{r-m,N} \end{bmatrix} + \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_m \end{bmatrix} \sum_{t=1}^{N} \begin{bmatrix} \boldsymbol{\Omega}_{t,1} \\ \boldsymbol{\Omega}_{t,2} \\ \vdots \\ \boldsymbol{\Omega}_{t,N} \end{bmatrix}^T \tag{14}$$

with $\boldsymbol{f}(u) = \hat{f}(u) + d \Big/ \sum_{j=0}^{m} b_j$. Using SVD decomposition to get $b_j, j = 0,1,\cdots,m$ and $\hat{f}(u)$, then we get the approximate form of the nonlinear part $\boldsymbol{f}(u)$.

For the MIMO systems, the structure can be assumed as:

$$\mathbf{y}_t = \sum_{i=1}^{n} \mathbf{A}_i \mathbf{y}_{t-i} + \sum_{j=0}^{m} \mathbf{B}_j \boldsymbol{f}(\mathbf{u}_{t-j}) + \mathbf{e}_t \tag{15}$$

with $\mathbf{y} \in R^{ny}$, $\mathbf{u} \in R^{nu}$, $\mathbf{A}_i \in R^{ny*ny}$, $\mathbf{B}_j \in R^{ny*nu}$, $\mathbf{e}_t \in R^{ny}$. The nonlinear mapping $\boldsymbol{f}: R^{nu} \to R^{nu}: \mathbf{u} \to \boldsymbol{f}(\mathbf{u}) = [\mathrm{f}_1(\mathbf{u})\ \mathrm{f}_2(\mathbf{u}) \cdots \mathrm{f}_{ny}(\mathbf{u})]^T$. Then (15) formula can be rewritten as:

$$\begin{aligned} \mathbf{y}_t(s) &= \sum_{i=1}^{n} \mathbf{A}_i(s,:) \mathbf{y}_{t-i} \\ &\quad + \sum_{j=0}^{m} \mathbf{B}_j(s,:) \mathrm{f}(\mathbf{u}_{t-j}) + \mathbf{e}_t(s) \quad s = 1, \cdots, ny \end{aligned} \tag{16}$$

As one can see, a MIMO system can be described as a set of MISO systems; meanwhile it is well known that LSSVM can treat MISO system straightforwardly as SISO case, thus the method for SISO systems can extend to MIMO systems as well. Please refer to the work of Goethals et al. (2005) for more details.

3. ON-LINE RECURSIVE METHOD

Obviously, the method mentioned above is suitable to off-line identification. But in most industrial applications, there are only litter data points. Therefore, we should take full advantage of every point. According to (Goethals et al., 2005), when a new data point is obtained, the (n+m+1+N) dimensional matrix $\mathbf{L}$ of (13) should be reconstructed (with m and n, the order of numerator and denominator of the transfer function of the linear part respectively, and N the number of training datasets), and need to do a matrix inversion operation of L. Obviously, it costs too much time. To overcome this shortcoming, a new recursive updating method based on sub-matrix inverse calculation is proposed in this paper. During the recursive process, we will only need to do a matrix inverse calculation of a (m+1) dimensional square matrix instead of a high-dimensional matrix $\mathbf{L}$ (usually, $m \ll N$).

In the section, a SISO system is taken as an example, and the RBF kernel is chosen. When there are N training data points, the matrix $\mathbf{L_N}$ is:

$$\mathbf{L}_N = \begin{bmatrix} 0 & \mathbf{0} & \vec{\mathbf{1}}^T & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{Y}_p & \mathbf{0} \\ \mathbf{1} & \mathbf{Y}_p{}^T & \tilde{\mathbf{K}}+\gamma^{-1}\mathbf{I} & \mathbf{K} \\ \mathbf{0} & \mathbf{0} & \mathbf{K}^T & \left(\vec{\mathbf{1}}^T\boldsymbol{\Omega}_N\vec{\mathbf{1}}\right)\mathbf{I}_{m+1} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_N & \mathbf{S} \\ \mathbf{S}^T & \mathbf{T}_N \end{bmatrix} \tag{17}$$

with $\mathbf{S}=[\mathbf{0};\mathbf{0};\mathbf{K}]$ a column vector. Also, when a new data point is obtained, the matrix $\mathbf{L_{N+1}}$ will change to the form below:

$$\mathbf{L}_{N+1} = \begin{bmatrix} 0 & \mathbf{0} & \vec{\mathbf{1}}^T & 1 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{Y}_p & \mathbf{y}_1 & \mathbf{0} \\ \mathbf{1} & \mathbf{Y}_p{}^T & \tilde{\mathbf{K}}+\gamma^{-1}\mathbf{I} & \tilde{\mathbf{K}}_1 & \mathbf{K}_1 \\ 1 & \mathbf{y}_1{}^T & \tilde{\mathbf{K}}_1{}^T & \tilde{K}_2 & \mathbf{K}_2 \\ \mathbf{0} & \mathbf{0} & \mathbf{K}_1{}^T & \mathbf{K}_2{}^T & \left(\vec{\mathbf{1}}^T\boldsymbol{\Omega}_{N+1}\vec{\mathbf{1}}\right)\mathbf{I}_{m+1} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{N+1} & \mathbf{S}_{N+1} \\ \mathbf{S}_{N+1}{}^T & \mathbf{T}_{N+1} \end{bmatrix} \tag{18}$$

with $\mathbf{A}_{N+1}=\begin{bmatrix}\mathbf{A}_N & \mathbf{p}\\ \mathbf{p}^T & \mathbf{t}\end{bmatrix}$, $\mathbf{p}=\begin{bmatrix}1\\ \mathbf{y}_1\\ \tilde{\mathbf{K}}_1\end{bmatrix}$, $\mathbf{t}=\left[\tilde{K}_2\right]$, $\mathbf{S}_{N+1}=\begin{bmatrix}\mathbf{0}\\ \mathbf{0}\\ \mathbf{K}_1\\ \mathbf{K}_2\end{bmatrix}$,

$\mathbf{T}_{N+1}=\left[\left(\vec{\mathbf{1}}^T\boldsymbol{\Omega}_{N+1}\vec{\mathbf{1}}\right)\mathbf{I}_{m+1}\right]$.

According to (13), we can get the recursive form as follows:

$$\boldsymbol{\Omega}_1(j,l)=\mathrm{K}(u_{N-m+j},u_l)\quad l=1,\cdots,N+1,\ j=1,\cdots,m+1 \tag{19}$$

$$\mathbf{y}_1=\begin{bmatrix}y_N & y_{N-1} & \cdots & y_{N-n+1}\end{bmatrix}^T \tag{20}$$

$$\mathbf{T}_{N+1}=\mathbf{T}_N+\left[\left(2\times\boldsymbol{\Omega}_1(m+1,:)\times\vec{\mathbf{1}}-1\right)\mathbf{I}_{m+1}\right] \tag{21}$$

$$\tilde{K}_2=m+1+\frac{1}{\gamma} \tag{22}$$

$$\tilde{\mathbf{K}}_1(l,1)=\sum_{j=1}^{m+1}\boldsymbol{\Omega}_1(j,l+r+j-m-2)\quad l=1,\cdots,N-r+1 \tag{23}$$

$$\mathbf{K}_1(k,l)=\mathbf{K}(k,l)+\boldsymbol{\Omega}_1(m+1,r+k-l)\quad k=1,\cdots,N-r+1,l=1,\cdots,m+1 \tag{24}$$

$$\mathbf{K}_2(1,l)=\sum_{t=1}^{N+1}\boldsymbol{\Omega}_1(m+2-l,t)\quad l=1,\cdots,m+1 \tag{25}$$

Then we can get $\mathbf{A}_{N+1}$ and $\mathbf{L}_{N+1}$ in turn. Based on the thought of sub-inverse matrix,

$$\mathbf{A}_{N+1}{}^{-1}=\begin{bmatrix}\mathbf{B}_{11} & \mathbf{B}_{12}\\ \mathbf{B}_{21} & \mathbf{B}_{22}\end{bmatrix} \tag{26}$$

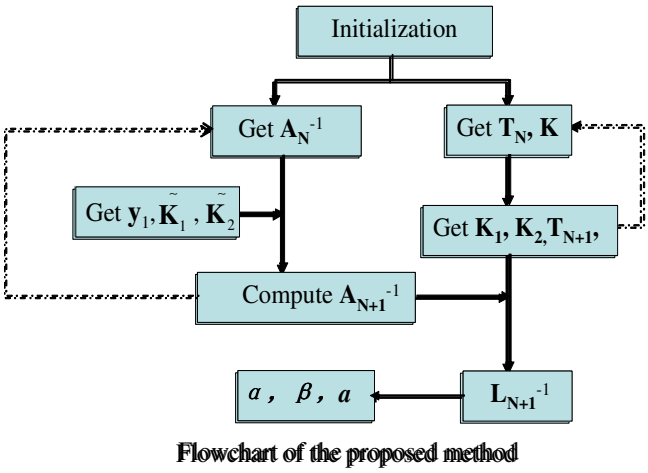
with $\begin{cases}\mathbf{B}_{11}=\mathbf{A}_N{}^{-1}+\mathbf{A}_N{}^{-1}\mathbf{p}\mathbf{B}_{22}\mathbf{p}^T\mathbf{A}_N{}^{-1}\\ \mathbf{B}_{12}=-\mathbf{A}_N{}^{-1}\mathbf{p}\mathbf{B}_{22}\\ \mathbf{B}_{21}=-\mathbf{B}_{22}\mathbf{p}^T\mathbf{A}_N{}^{-1}\\ \mathbf{B}_{22}=\left(\mathbf{t}-\mathbf{p}^T\mathbf{A}_N{}^{-1}\mathbf{p}\right)^{-1}\end{cases}$

and

$$\mathbf{L}_{N+1}{}^{-1}=\begin{bmatrix}\mathbf{L}_{11} & \mathbf{L}_{12}\\ \mathbf{L}_{21} & \mathbf{L}_{22}\end{bmatrix} \tag{27}$$

with $\begin{cases}\mathbf{L}_{11}=\mathbf{A}_{N+1}{}^{-1}+\mathbf{A}_{N+1}{}^{-1}\mathbf{S}_{N+1}\mathbf{L}_{22}\mathbf{S}_{N+1}{}^T\mathbf{A}_{N+1}{}^{-1}\\ L_{12}=-\mathbf{A}_{N+1}{}^{-1}\mathbf{S}_{N+1}L_{22}\\ L_{21}=-L_{22}\mathbf{S}_{N+1}{}^T\mathbf{A}_{N+1}{}^{-1}\\ L_{22}=\left(\mathbf{T}_{N+1}-\mathbf{S}_{N+1}{}^T\mathbf{A}_{N+1}{}^{-1}\mathbf{S}_{N+1}\right)^{-1}\end{cases}$

The final recursive algorithm for Hammerstein identification based on LSSVM can be summarized as follows:



Flowchart of the proposed method

Fig. 2. The flowchart of the proposed method

It should be noted that m, n, γ and the kernel parameter have an important effect on the precision of the result, and we should determine them by prior knowledge.

It is also noticed that the matrix should be full-rank when using a trick of sub-inverse. Obviously, the kernel method and the parameter γ would ensure the matrix non-singular on the theoretic aspect. But during the computation, it may be singular which will cause an additional error. A method using sparse LSSVM or re-initialization may be helpful.

4. ILLUSTRATIVE EXAMPLE

Consider the Hammerstein system in issue (Espinoza et al., 2004) as follows:

$$y_t=a_1y_{t-1}+a_2y_{t-2}+a_3y_{t-3}+b_0\sin c\left(u_t\right)+b_1\sin c\left(u_{t-1}\right) \tag{28}$$

with [a_1 ,a_2, a_3, b_0, b_1] = [0.6, 0.2, 0.1, 0.4, 0.2]. A white Gaussian input sequence u with length 200, zero mean and standard deviation 2 was generated and fed into the system.

Based on the prior knowledge, we choose the parameters n=3, m=1, the initial training dataset of N_0 samples, and a total step N_t.

In this paper, following three techniques were used for identification:

1 The method in (Goethals et al., 2005): the initial dataset was used as training set, and the set of the later data points was used as testing set. No new data was added into the training set.

2 The method in (Goethals et al., 2005) and updating the training dataset: A matrix inverse operation was needed when any new data was added to the training set. We mentioned it as the naive method.

3 The recursive technique proposed in this paper.

In the naive implementation, one should identify the model N_t times, each time adding a new data to the training set. This implementation involves the solutions of N_t linear systems of dimensions $N_0+m+n-r+3$ to $N_t+N_0+n+m-r+3$ respectively. Note that the complexity of solving a linear system with dimensions d is $\frac{1}{3}d^3$ in general, the complexity of the naive method is $\frac{1}{3}\sum_{i=0}^{N_t}\left(N_0+m+n-r+3+i\right)^3$. On the other hand, the proposed algorithm involves one inverse of a $N_0+m+n-r+3$ square matrix, and N_t inverses of a (m+1) square matrix. Hence, the complexity of the proposed method is $\frac{1}{3}\left(N_0+m+n-r+3\right)^3+\frac{1}{3}N_t\left[\left(m+1\right)^3\right]$.

Obviously, the naive method is much less efficient than the proposed implementation. Here, we use the CPU time to indicate the complexity. Table 1 shows the indication of different steps N_t with N_0=10, while Table 2 lists the CPU time with different initial training dataset of a fixed pair of γ and σ. All the time listed in Tab. 1 and 2 includes the time of estimation of the parameters and the predictions. It is obtained from a computer with PIII-800MHz and 512M memory.

Table 1. The average CPU time (seconds) of the proposed and naive implementation of different steps with a initial dataset of 10 points for one pair of σ and γ

N_t	50	100	150	200	250
Naive	1.048	5.685	19.732	54.919	128.428
Proposed	0.270	1.028	3.175	7.905	16.310

Also, in this issue, the Root Mean Squared Errors (RMSE) of the prediction was calculated, with

$$RMSE=\sqrt{\sum_{k=1}^{N}\left(y_k-\tilde{y}_k\right)^2\Big/N} \qquad (29)$$

where $\tilde{y}_k$ is the estimation of y_k.

Table 2. The average CPU time (seconds) of 100 steps of the proposed and naive implementation with different initial data points for one pair of σ and γ

N_0	10	20	40	80	100
Naive	5.685	7.521	12.034	28.300	41.430
Proposed	1.028	1.289	1.973	4.326	6.025

Here, we choose a dataset of 210 data points, the first 10 of which used as the initial training set and the rest as the new coming data. The kernel parameter and the regularization constant are chosen according to RMSE on the initial training set. For each new data, we first predict the output, then add the data into the training set, retrain again to get new estimation of the parameters. The predictions and the predictive errors are showed in Fig. 2. The estimations of the static nonlinearity of different algorithm are showed in Fig. 3.

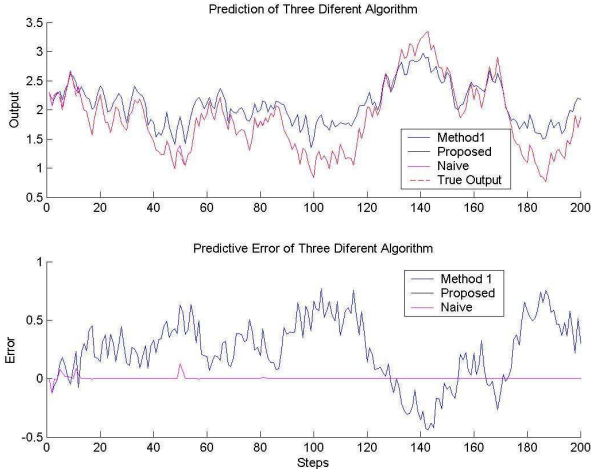


Fig. 3. The one-step-ahead prediction and predictive error of three different methods

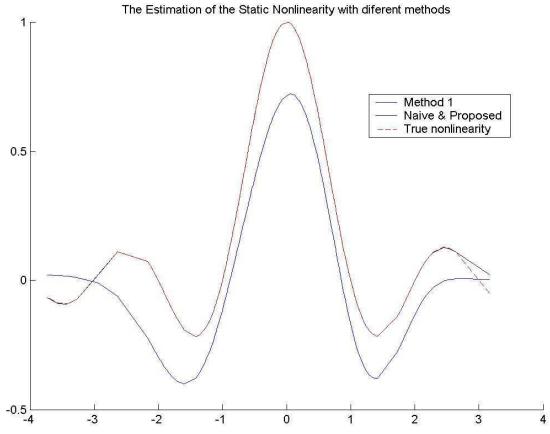


Fig. 4. The Estimations of the static nonlinearity with three different algorithms.

Also, the mean of the estimation of the parameters a, the RMSE were listed in Table 3.

Table 3. The average of the estimations by three different methods. With a 10 initial training data points and running 200 steps. The final estimation by the naive and proposed algorithm are $a_1 = 0.6001$, $a_2 = 0.1998$ and $a_3 = 0.1000$.

	System	Method 1	Naive	Proposed
a_1	0.6	0.8117	0.6129	0.6129
a_2	0.2	-0.2453	0.1832	0.1832
a_3	0.1	-0.0517	0.0995	0.0995
RMSE		0.3605	0.0164	0.0164

From Table 1 and 2, it seems that the proposed algorithm is approximately 4~8 times as efficient as the naive method. It is also interesting to note that the ratio increases with increasing N_t. It seems to be more efficient with more training dataset.

Also, from Figure 3 and Figure 4, the prediction of method 1 is much worse than that of the naive, which means it is necessary to adding new samples when there is litter data. With the updating of training set, the obtained prediction by either of the naive method or the proposed algorithm will be more accurate. It is noted that the prediction of some points are no so accurate, just because the training points near the new data are not enough.

On the other hand, though the naive method can get an accurate estimation as the proposed, it costs too much times because of the matrix inverse operation. So, it needs a much faster algorithm, just like the proposed. According to the proposed method, *the inverse of the whole matrix* $\boldsymbol{L}$ *can be obtained from an inverse of a (m+1) squared matrix and some multiplications, which can save much time.* The simulation shows that it does get a good approximation of the Hammerstein model and can be applied in the on-line identification.

5. CONCLUSIONS

In this paper, a recursive algorithm based on LSSVM is developed for the identification of Hammerstein ARX systems. The proposed method uses the trick of sub-inverse matrix and got a recursive form of the matrix $\mathbf{L}$, which can reduce the complexity of the computation and save much time. By this manner, an on-line nonlinear identification approach is derived and both SISO and MIMO cases are explored.

To illustrate its performance, the method was compared to the result of two existed algorithms. It is obviously observed that the proposed one could get a more accurate estimation than the method without adding new samples, and was much faster than the naive algorithm. In a word, the proposed method gets the best performance among the three. But there are still some difficulties such as the adaptive choice of the kernel parameters, lack of sparsity, recursive estimation of the nonlinear part for the on-line identification and deserves further studies.

REFERENCES

AL-DUWAISH, H. and M.N. KARIM (1997). A new method for the Identification of Hammerstein Model. In: Automatica, **33**(10): 1871-1875.

Chi, H.M., O.K. Ersoy (2003). Recursive update algorithm for least squares support vector machines. In: Neural Processing Letters, **17**(2): 165-173.

Dempsey, E.J. and D.T. Westwick (2004). Identification of Hammerstein models with cubic spline nonlinearities. In: IEEE Transactions on Biomedical Engineering, **51**(2): 237-245.

Eskinat, E., S.H. Johnson and W.L. Luyben (1991). Use of Hammerstein models in identification of nonlinear systems. In: AICHE Journal. **37**(2): 255-268.

Espinoza, M., J.A.K. Suykens, and B. De Moor (2004). Partially linear models and least squares support vector machines. In: 43rd IEEE Conference on Decision and Control. Atlantis, Paradise Island, Bahamas.

Goethals, I., K. Pelckmans, J.A.K. Suykens et al. (2005). Identification of MIMO Hammerstein models using least squares support vector machines. In: Automatica, **41**(7): 1263~1272.

Golden, M.P. and B.E. Ydsite (1989). Adaptive extremum control using approximate process models. In: AICHE Journal, 35(7): 1157-1169

Narendra, K. and P. Gallman (1966). An iterative method for the identification of nonlinear systems using a Hammerstein model. In: IEEE Transactions on Automatic Control , **11**(3): 546-550.

Suykens, J.A.K, T. Van Gestel, J. De Brabanter et al. (2002). Least squares support vector machines. Singapore: World Scientific.

Van Pelt, T.H. and D.S. Bernstein (2000). Nonlinear system identification using Hammerstein and nonlinear feedback models with piecewise linear static maps. 1. Theory. In: American Control Conference, Chicago.

Vapnik, V. (1995). The nature of statistical learning theory. New York: Spring.

Westwick, D. T. and R. Kearney (2000). Identification of a Hammerstein model of the stretch reflex EMG using separable least squares. In: Proceeding pf the World Congress on Medicine Physics and Biomedical Engineering, Chicago.