Support Vector Machine Identification of Output Error Hammerstein Models

Mujahed Al Dhaifallah ∗ and David T. Westwick **

* King Fahd University of Petroleum and Minerals, Systems Engineering Department, Box 5067, Dhahran 31261, Saudi Arabia (e-mail: mujahed@kfupm.edu.sa).
** Department of Electrical and Computer Engineering, Schulich School of Engineering, Calgary, Canada (e-mail: dwestwic@ucalgary.ca)

Abstract: The ARX structure is often used in linear and nonlinear system identification because it is compact and linear in the variables. However, the ARX structure includes a noise model which shares the same poles as the deterministic system which is not always appropriate. In this paper, we consider the extension of an SVM based identification technique for Hammerstein models with ARX linear dynamics proposed by Dhaifallah and Westwick (2008 IFAC World Congress, pp:4999–5004) to include the output-error class of linear system models. The presented algorithm will be compared to the previous Hammerstein ARX approach using simulations.

Keywords: Hammerstein, Output Error, Identification, Support Vector Machines

1. INTRODUCTION

System identification is the art of finding mathematical tools and algorithms that build an appropriate mathematical model of a system from measured input and output data (Sinha, 2000). The goal of such experimental system analysis is to develop a mathematical model which describes the static and dynamic behavior of the system in a sufficiently accurate manner (Vapnik, 1998).

Models range from complex, continuous time, distributed, nonlinear, and time varying models to simple, discrete time, lumped, linear, and time invariant ones. Having an accurate system model is important in many applications but is not easy to identify. Since the model is an approximation to the true system, there is a trade-off between the complexity of the model, and the accuracy of its predictions. A linear model can often be used to produce good predictions of a system’s behavior, particularly if that system is restricted to operating within a narrow region. If the model is required to cover a broader operating region, then a nonlinear model may be required (Schoukens et al., 2005).

Block structured models, cascades of static nonlinearities and dynamic linear systems, are often a good trade-off as they can represent some dynamic nonlinear systems very accurately, but are nonetheless quite simple. Moreover, the extensive knowledge about LTI system representations can be applied to the dynamic linear blocks. On the other hand, finding an effective representation for the nonlinearity is an active area of research. Polynomials are simple, easy to estimate, but cannot deal with many common nonlinearities (saturation, threshold, deadzone, etc.). Some of these nonlinearities can be approximated by spline functions. However, spline functions are defined by a series of knots which must either be chosen a-priori, or treated as model parameters and included in the (non-convex) optimization. Neural networks are another tool to approximate nonlinear functions. Their powerful approximation abilities make them attractive. However, the need to specify the neural network topology in terms of the number of nodes and layers, and the need to solve non-convex optimization complicate their implementation. Recently, support vector machines (SVMs) and least squares support vector machines (LS-SVMs) have demonstrated powerful abilities in approximating linear and nonlinear functions (Vapnik, 1998), (Suykens et al., 2002). In contrast with other approximation methods, SVMs do not require a-priori structural information. Furthermore, there are well established methods with guaranteed convergence (ordinary least squares, quadratic programming) for fitting LS-SVMs and SVMs (Boyd and Vandenberghe, 2004).

Many algorithms have been proposed to identify Hammerstein systems. Although the earliest algorithms assumed an output error (OE) model nonlinear structure (Narendra and P., 1966; Chang and Luus, 1971), most of the recent algorithms are based on the assumption that the linear dynamic component of the system being identified can be represented by an ARX model, an equation-error model structure that is linear in the variables, but where the input and the noise transfer functions have a common denominator. If the plant and noise models do not share common poles, this can result in bias to both models. In such situations, the resulting bias can be avoided by choosing a model structure where the plant and noise models are independently parametrized (Heuberger et al., 2005). The simplest such structure is the the OE error model. Several techniques have been suggested to deal with linear OE models; instrumental variables (IV), subspace methods, and algorithms based on the Steiglitz-McBride...
(S-M) iterative method (Ljung, 1999). The IV method is simple to use and gives consistent estimates under most conditions. However, its extension to SVM regression is not easy since its main idea is based on modifying the normal equation of the ordinary linear regression to obtain unbiased and consistent estimates. Extensions of subspace methods to SVM and LS-SVM regressions are possible (see Goethals et al., 2005b), but they are computationally expensive.

The main contribution of this paper is to extend the S-M iterative algorithm (Steiglitz and McBride, 1965) to the identification of OE Hammerstein models using support vector machine regression. The S-M method was chosen for extension to SVM regression because easy to implement and involves a reasonable computational cost.

2. Identification of Output Error Hammerstein Models

![Block diagram of OE Hammerstein cascade.](image)

Fig. 1. Block diagram of OE Hammerstein cascade. The investigator is assumed to have access to the input, \( u_t \), and the output, \( y_t \), but not the intermediate signals, \( x_t \) and \( s_t \) or the innovation, \( e_t \).

The OE Hammerstein system consists of a static nonlinear element followed by a linear OE model, as shown in Fig. 1. The linear dynamics are represented by an OE model:

\[
\begin{align*}
    s_t &= B(z) x_t + \sum_{j=0}^{m} b_j z^{-j} x_t \\
    y_t &= s_t + e_t = B(z) A(z) + e_t
\end{align*}
\]

(1)

where \( x_t, s_t, y_t \in \mathbb{R} \) are the output of the nonlinear block, the unmeasurable noise-free output, and the measured output signals, respectively, for \( t = 1, \ldots, N \). The innovation \( e_t \) is assumed to be white. The static nonlinearity is represented by a SVM, defined as follows:

\[
f(x) = w^T \varphi(x) + d_0
\]

(2)

Then, the output of the OE Hammerstein model may be computed from

\[
\begin{align*}
    s_t &= B(z) x_t + \sum_{j=0}^{m} b_j z^{-j} x_t \\
    y_t &= s_t + e_t = B(z) A(z) + d_0 + e(t)
\end{align*}
\]

(3)

Compared with the NARX problem considered in our previous work (Al Dhaifallah and Westwick, 2008), the only change, here, is that the output is nonlinear in the coefficients of \( A(z) \), which makes the identification problem difficult to solve. As before, the solution makes use of an overparameterized model

\[
y_t = W(z) A(z) + d + e(t)
\]

(4)

where

\[
W(z) = \left( w_0^T + w_1^T z^{-1} + w_2^T z^{-2} + \cdots + w_m^T z^{-m} \right)
\]

\[
w_j = b_j \ w, \ d = d_0 \sum_{j=0}^{m} b_j / \sum_{j=1}^{m} a_j
\]

It is clear that \( y_t \) is linear in the parameters of the overparameterized numerator in (4), but this model class is more general than the Hammerstein model, which it includes as a special case (when \( w_j = b_j \ w \) for \( j = 1, m \)). Similar to the approach adopted in (Goethals et al., 2005a; Al Dhaifallah and Westwick, 2008), the overparameterized model (4) will be identified first, and then a low-rank projection will be used to force the estimated model to be a Hammerstein cascade. To identify the parameters of model (4) using SVM regression, solve the following optimization problem

\[
\min_{w_j, a, d, \xi} \frac{1}{2} \sum_{j=0}^{m} w_j^T w_j + \frac{1}{2} \sum_{i=1}^{n} a_i^2 + \frac{\epsilon}{\sum_{t=r}^{N}} \left( (\xi_t)^k + (\xi'_t)^k \right)
\]

(5)

subject to

\[
\sum_{t=1}^{N} w_j T \varphi(u_t) = 0, \ j = 0, \ldots, m
\]

(6)

\[
y_t - W(z) A(z) \varphi(u_t) - d \leq \epsilon + \xi_t
\]

(7)

\[
W(z) A(z) \varphi(u_t) + d - y_t \leq \epsilon + \xi'_t
\]

(8)

where the \( \xi_t \) and \( \xi'_t \) are slack variables that allow some points to fall outside of the epsilon tube. Note that (5) is a standard SVM objective function, consisting of a weighted average, with the weighting controlled by the parameter \( c \), of the 2 norm of the parameters \( (w, a) \) and the Vapnik \( \epsilon \)-insensitive cost function applied to the residuals. Constraints (8) are identical to the constraints used in the ARX versions, which marginalize the effect of errors smaller than \( \epsilon \), a user selected tuning parameter, on the cost function. Again, constraints (6) were added to force the nonlinear functions \( w_j \varphi(\cdot) \), \( j = 0, \ldots, m \), to be zero-mean over the training set (Goethals et al., 2005a). Constraints (7) are derived by modifying the constraints of the standard SVM to include the dynamics of the OE model. Note that, unlike similar algorithms based on ARX dynamics, the constraints (7) are nonlinear in the parameters \( a \). Therefore, the optimization problem (5)-(8) is not a standard quadratic programming problem was the case with ARX dynamics. However, one can carry out the minimization (5)-(8) iteratively, using a procedure based on the iteration proposed by Steiglitz and McBride (1965) for linear OE models.

2.1 The Method of Steiglitz-McBride

The Steiglitz-McBride (S-M) method for linear OE models (Steiglitz and McBride, 1965) can be summarized as follows. Consider the OE model

\[
y_t = B(z) A(z) + d + e_t
\]

(9)

First, find an estimate to the following ARX model

\[
A(z) y_t = B(z) u_t + e_t
\]

(10)
by solving the following linear regression problem

\[ \Phi^T \theta = y \]

where

\[
\Phi = \begin{bmatrix}
- y_{r-1} & - y_r & \cdots & - y_{N-1} \\
- y_{r-2} & - y_{r-1} & \cdots & - y_{N-2} \\
\vdots & \vdots & \ddots & \vdots \\
- y_{r-n} & - y_{r-n+1} & \cdots & - y_{N-n} \\
u_r & u_{r+1} & \cdots & u_N \\
u_{r-1} & \cdots & \cdots & \cdots \\
\vdots & \vdots & \ddots & \vdots \\
u_{r-m} & u_{r-m+1} & \cdots & u_{N-m}
\end{bmatrix}
\]

\[ \theta = [a_1 \cdots a_n \ b_0 \ b_1 \cdots b_m]^T \]

\[ y = [y_r \cdots y_N]^T \]

\[ r = \max(n, m) + 1 \]

Call these \( A^{(1)}(z) \), \( B^{(1)}(z) \). Then, filter the original data records using the filter \( \frac{1}{A^{(1)}(z)} \), yielding

\[ y^{(f)}_j = \frac{1}{A^{(1)}(z)} y_j, \quad u^{(f)}_t = \frac{1}{A^{(1)}(z)} u_t \]

The linear regression problem (10) is then solved using the prefiltered input-output records to get new estimates \( A^{(2)}(z) \), \( B^{(2)}(z) \)

\[ A^{(2)}(z) \frac{1}{A^{(1)}(z)} y_j = B^{(2)}(z) \frac{1}{A^{(1)}(z)} u_t + \epsilon_t \]

Use \( A^{(2)}(z) \) to filter the original data record and so forth. Repeat until convergence.

### 2.2 Application to OE Hammerstein Models

Now to extend the S-M iterative algorithm, reviewed in the last section, to the identification of OE Hammerstein models using support vector machine regression, one can fit a NARX Hammerstein model to the data using the algorithm proposed by Al Dhaiffalah and Westwick (2008) as an initial starting point. The objective is a "first estimate" of \( A(z) \), \( B(z) \), \( f(ut) \), denoted \( A^{(1)}(z) \), \( B^{(1)}(z) \), \( f^{(1)}(ut) \). Then a S-M inspired iteration is used. Thus at iteration \( \ell \), the previous denominator \( A^{(\ell-1)}(z) \) is used to pre-filter the input and outputs of the linear part. Then \( A^{(\ell)}(z) \), \( W^{(\ell)}(z) \) are found by solving the following minimization

\[
\min_{w_j, \alpha^{(\ell)}_i, d, \xi, \delta} \frac{1}{2} \sum_{j=0}^{m} w_j^{(\ell)} w_j^{(\ell)} + \frac{1}{2} \sum_{i=1}^{n} (a_i^{(\ell)})^2 + \epsilon \sum_{t=r}^{N} (\xi_t + \xi_t^*)
\]

subject to

\[ w_j^{(\ell)} \varphi(ut) = 0, \quad j = 0, \ldots, m \]  

\[ A^{(\ell)}(z) y^{(f)}_j = \frac{W^{(\ell)}(z)}{A^{(\ell-1)}(z)} \varphi(ut) - d^{(\ell)} \leq \epsilon + \xi_t \]

\[ \frac{W^{(\ell)}(z)}{A^{(\ell-1)}(z)} \varphi(ut) + d^{(\ell)} - A^{(\ell)}(z) y^{(f)}_j \leq \epsilon + \xi_t^* \]

\[ \xi_t, \xi_t^* \geq 0, \quad t = r, \ldots, N \]

where \( y^{(f)}_j = \frac{y_t}{A^{(\ell-1)}(z)} \).

Note that as \( A^{(\ell-1)}(z) \) approaches \( A^{(\ell)}(z) \), problem (11)-(14) approaches the NARX Hammerstein optimization problem. Problem (11)-(14) can be solved as follows. First, write the Lagrangian of (11)-(14)

\[
L \left( w_j^{(\ell)} , d^{(\ell)} , \xi, \xi^*, \alpha, \alpha^* \right) = \frac{1}{2} \sum_{j=0}^{m} w_j^{(\ell)^T} w_j^{(\ell)}
\]

\[ + \sum_{t=r}^{N} \frac{1}{2} \left( a_i^{(\ell)} \right)^2 + \epsilon \sum_{t=r}^{N} (\xi_t + \xi_t^*) \]

\[ - \sum_{j=0}^{m} \gamma_j \left( \sum_{t=1}^{N} w_j^{(\ell)^T} \varphi(ut) \right) - \sum_{t=r}^{N} \alpha_t \left( \frac{W^{(\ell)}(z)}{A^{(\ell-1)}(z)} \varphi(ut) - d^{(\ell)} \right) \]

\[ + \sum_{t=r}^{N} \sum_{t=1}^{N} \left( \beta \xi_t + \beta^* \xi_t^* \right) \]

where \( \alpha_i, \alpha^*_i, \beta_i, \beta^*_i \) are non-negative Lagrange multipliers and \( \gamma_j \in \mathbb{R} \).

The stationary point of the Lagrangian (15) can be found by setting its gradient to zero. Setting \( \frac{\partial L}{\partial w_j^{(\ell)}} \) to zero, for

\[ j = 1 \ldots m, \]

yields

\[ w_j^{(\ell)^T} \varphi(ut) = \gamma_j \sum_{t=1}^{N} \varphi(ut) + \sum_{t=r}^{N} (\alpha_t - \alpha^*_t) \left( \frac{1}{A^{(\ell-1)}(z)} \varphi(ut-j) \right) \]

Which leads to

\[ w_j^{(\ell)^T} \varphi(ut) = \gamma_j \sum_{t=1}^{N} \varphi(ut) + \sum_{t=r}^{N} (\alpha_t - \alpha^*_t) \left( \frac{1}{A^{(\ell-1)}(z)} \varphi(ut-j) \right) \]

\[ \varphi(ut) - \frac{W^{(\ell)}(z)}{A^{(\ell-1)}(z)} \varphi(ut) \]

(17)

Using the kernel trick, we have:

\[ w_j^{(\ell)^T} \varphi(ut) = \gamma_j \sum_{t=1}^{N} K(ut, ut) \]

\[ + \sum_{t=r}^{N} (\alpha_t - \alpha^*_t) K_f,0 (ut-j, ut) \]

Where \( K_f,0 \) is the result of filtering each column of the matrix \( K \) with the previous denominator, \( \frac{1}{A^{(\ell-1)}(z)} \).

Similarly, define \( K_{0,f} \) and \( K_{f,f} \) as the results obtained by filtering the rows, and rows and columns, of \( K \), respectively. Note that (18) involves the kernel, but does not use the nonlinear basis functions, \( \varphi \).

Similarly from the centering constraints (12), one can show that

\[ \gamma_j \sum_{t_2=1}^{N} \sum_{t_1=1}^{N} K(ut_2, ut_1) + \sum_{t=r}^{N} \sum_{t_1=1}^{N} (\alpha_t - \alpha^*_t) \]

\[ \times K_f,0 (ut-j, ut_1) = 0 \]
\[
\frac{\partial L}{\partial a^{(r)}_t} = 0 \Rightarrow a^{(r)}_t = \sum_{t=r}^{N} (\alpha_t - \alpha^*_t) y_{t-i}
\]
for \(i = 1 \ldots n\).

\[
\frac{\partial L}{\partial f_{j,t}} = 0 \Rightarrow \sum_{t=r}^{N} (\alpha_t - \alpha^*_t) = 0
\]

\[
\frac{\partial L}{\partial \xi_t} = 0 \Rightarrow \alpha_t + \beta_t = c, t = r, \ldots, N
\]

\[
\frac{\partial L}{\partial \xi^*_t} = 0 \Rightarrow \alpha^*_t + \beta^*_t = c, t = r, \ldots, N
\]

From (16)-(22), the Lagrangian function (15) can be rewritten as

\[
L(\alpha, \alpha^*, \gamma) = -\frac{1}{2} \sum_{t=r}^{N} \sum_{t=r}^{N} (\alpha_t - \alpha^*_t) (\alpha_t - \alpha^*_t) + \sum_{j=0}^{m} K_{f,f}(u_{t-j},u_{t-j}) + \sum_{i=1}^{n} y_{t-i} y_{t-i}^T
\]

\[
+ \frac{1}{2} \sum_{j=0}^{m} \sum_{t=r}^{N} \sum_{t'=t+1}^{N} K(u_{t},u_{t'}) + \sum_{t=r}^{N} (\alpha_t - \alpha^*_t) y_t - \sum_{t=r}^{N} \alpha_t \epsilon - \sum_{t=r}^{N} \alpha^*_t \epsilon
\]

where \(K_{f,f}\) is defined above. Hence, the dual optimization problem can be written as

\[
\min_{\alpha, \alpha^*, \gamma} \quad \frac{1}{2} \begin{bmatrix} \gamma^T & \alpha^T & \alpha^T \end{bmatrix} \begin{bmatrix} -S & I & 0 \\ I & 0 & -K \\ 0 & -K & K \end{bmatrix} \begin{bmatrix} \gamma \\ \alpha \\ \alpha^* \end{bmatrix} + \begin{bmatrix} 0 & -y_{r:N}^T & y_{r:N}^T \\ 0 & \epsilon \cdot \mathbf{1}^T \epsilon \cdot \mathbf{1}^T \end{bmatrix} \begin{bmatrix} \gamma \\ \alpha^* \end{bmatrix}
\]

subject to

\[
\sum_{t=r}^{N} (\alpha_t - \alpha^*_t) = 0
\]

\[
\gamma_j S + \sum_{t=r}^{N} (\alpha_t - \alpha^*_t) K^0(t, j) = 0, j = 0, \ldots, m
\]

\[
0 \leq \alpha^*_t, \alpha_t \leq c, t = r, \ldots, N
\]

with

\[
K(p, q) = \sum_{j=0}^{m} K_{f,f}(u_{p+r-j-1}, u_{q+r-j-1}) + \sum_{i=1}^{n} y_{q+r-j-1} y_{r+i-1}^T
\]

\[
S = \sum_{t_1=r}^{N} \sum_{t_2=r}^{N} K(u_{t_1}, u_{t_2})
\]

\[
K^0_j(t, j) = \sum_{t_1=r}^{N} K_{0,f}(u_{t_1}, u_{t+r-j-1})
\]

\[
K_{0,f}(u_{t_1}, u_{t_2}) = K(u_{t_1}, u_{t_2}) A^{1/(1-z)}
\]

Then, \(a^{(r)}_t\) is given by (20) and \(d^{(r)}\) can be computed based on the KKT conditions (Vapnik, 1998) as follows.

\[
d^{(r)} = y^T f(u) - \sum_{i=1}^{m} (\gamma_i \sum_{t=r}^{N} K_{0,f}(u_t, u_{i-j})) + \sum_{t=r}^{N} (\alpha_t - \alpha^*_t) K_{f,f}(u_{i-j}, u_{i-j}) - \sum_{h=1}^{n} a_h y_{t-h} \pm \epsilon
\]

Finally, \(d^{(r)}\) is calculated as

\[
d^{(r)} = \left( \frac{N-r}{N} \right) \left( 1 + \sum_{i=1}^{n} a^{(r)}_i \right)
\]

2.3 Separating Numerator and Nonlinearity Parameters

In this section, we follow the approach presented in (Goethals et al., 2005a) to build a matrix from which we can extract estimates of the numerator parameters and the output of nonlinear block. First and foremost, one should note that the solution has changed as a result of the difference between the ARX and OE formulations. So, it is clear that this is not going to be an exact repetition of the solution presented in (Goethals et al., 2005a). Recall

\[
f(u) = w^T \varphi(u) + d_0
\]

Substituting \(d_0 = \frac{1}{N} \sum_{t=1}^{N} f(u_t)\) into the last equation results in

\[
f(u) = w^T \varphi(u) + \frac{1}{N} \sum_{t=1}^{N} f(u_t)
\]

Subtracting \(\frac{1}{N} \sum_{t=1}^{N} f(u_t)\) from both sides and replacing \(f(u) - (1/N) \sum_{t=1}^{N} f(u_t)\) with \(f(u)\) gives

\[
\frac{1}{N} \sum_{t=1}^{N} f(u_t)
\]

Multiplying both sides by \(b_j\) gives

\[
b_j f(u) = b_j w^T \varphi(u)
\]

Recalling \(w_j = b_j w^T\), the last expression can be rewritten as

\[
b_j f(u) = w_j^T \varphi(u)
\]

After the final S-M iteration and by filtering each column of the matrix \(K\) with the final estimate of \(A(z)\) to obtain \(K_{f,0}\), (18) can be substituted into the last equation to get

\[
b_j f(u) = \gamma_j \sum_{t_1=1}^{N} K(u_{t_1}, u_t) + \sum_{t=r}^{N} (\alpha_t - \alpha^*_t) K_{f,0}(u_{t_1-j}, u_t)
\]

Based on the last expression, one can show that for the training input sequence \(\{u_1 \cdots u_K\}\), the following equality holds

\[
\begin{bmatrix} b_0 \\ \vdots \\ b_m \end{bmatrix} \begin{bmatrix} f(u_1) \\ \vdots \\ f(u_N) \end{bmatrix} = \begin{bmatrix} \gamma_0 \\ \vdots \\ \gamma_m \end{bmatrix} \sum_{t=1}^{N} \begin{bmatrix} K(t, 1) \\ \vdots \\ K(t, N) \end{bmatrix}
\]
\[
\begin{bmatrix}
\alpha_N - \alpha^*_N & \cdots & \alpha_r - \alpha^*_r \\
\alpha_N - \alpha^*_N & \cdots & \alpha_r - \alpha^*_r \\
& \ddots & \ddots \\
0 & \cdots & \alpha_N - \alpha^*_N & \cdots & \alpha_r - \alpha^*_r \\
\end{bmatrix}
\times (26)
\]

Thus, estimates for the numerator coefficients, \( b_j \), and the output of the static nonlinearity, \( f(u_1) \), can be obtained by approximating the right-hand side of (26) as a rank-1 matrix, via singular value decomposition. Once and the output of the static nonlinearity, \( f(u_1) \), can be obtained by approximating the right-hand side of (26) as a rank-1 matrix, via singular value decomposition. Once

\[
f(u) = f(u) + (1/N) \sum_{t=1}^{N} f(u_t)
\]

Now, using the training input sequence \([u_1 \cdots u_N]\) and the sequence of the nonlinearity responses to this input \([f(u_1) \cdots f(u_N)]\), we can train a support vector machine to represent the nonlinear part of the Hammerstein system, using a standard SVM regression algorithm (Vapnik, 1998).

### 2.4 Algorithm

The algorithm for identification of OE Hammerstein systems using support vector machines can be summarized follows

1. Compute “first estimate” of \( \Delta^0(z) \) using the algorithm described in (Al Dhaifallah and Westwick, 2008).
2. Obtain estimates for \( \alpha, \alpha^*, \gamma \) by solving (23).
3. Compute \( \alpha^0_l \) by (26).
4. Repeat steps 2 and 3 until convergence.
5. Compute \( d \) by (24).
6. Find the singular value decomposition of the right-hand side of (26) to obtain estimates for \( b \) and the static nonlinearity \( f \). After obtaining estimates \( b \), estimate for \( \sum_{t=1}^{N} f(u_t) \) can be obtained from (27).

Finally, \( f(u) \) can be computed by (28).

7. Use the input sequence \([u_1, u_2, \ldots, u_{n-1}]\) and the estimates of the response of the nonlinearity to this input \([f(u_1), f(u_2), \ldots, f(u_{n-1})]\), to train a SVM to approximate the nonlinear function \( f \).

### 3. ILLUSTRATIVE EXAMPLES

#### 3.1 Simulation Example 1: no output noise

In this section we will consider the artificial Hammerstein system identified in (Goethals et al., 2005a) which belongs to the class of ARX Hammerstein systems. The dynamic element was the ARX model:

\[
A(z) y = B(z) f(u) + \nu
\]

where

\[
B(z) = z^6 + 0.8z^5 + 0.3z^4 + 0.4z^3 \\
A(z) = (z - 0.98e^{j1}) (z - 0.98e^{j1.6}) (z - 0.97e^{j0.4})
\]

and \( f(u) = \text{sinc}(\alpha u) u^2 \) was used as the static nonlinearity.

A 1000 point sequence of zero mean, white Gaussian noise with variance 2 was generated and used as the input, \( u_t \), to this system. The output noise sequence \( \{\nu_t\}_{t=1}^{N-1} \) was chosen to be zero mean Gaussian white noise such that a signal to noise ratio of 20 dB was obtained at the output signal.

This system was identified using both the ARX and OE versions of the SVM estimation procedures based on linear \( \varepsilon \)-insensitive loss function discussed in (Al Dhaifallah and Westwick, 2008) and this paper, respectively. The hyperparameters were set based on validation testing, resulting in the use of an RBF kernel with variance parameter \( \sigma = 1 \), and \( c = 50 \) in the cost function specified in (5).

<table>
<thead>
<tr>
<th>Method</th>
<th>MAE</th>
<th>Number of SV</th>
<th>Computation Time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARX Algorithm ( \varepsilon = 0.001 )</td>
<td>0.0299</td>
<td>193</td>
<td>3.9955</td>
</tr>
<tr>
<td>OE Algorithm ( \varepsilon = 0.001 )</td>
<td>0.0516</td>
<td>192</td>
<td>4.6463</td>
</tr>
</tbody>
</table>

Fig. 2. True nonlinearities (solid) and estimated ones (dashed) identified using the Linear-SVM OE algorithm (top-left) and the Linear-SVM ARX algorithm (lower-left) for the ARX Hammerstein system. The true Transfer functions (solid) and estimated (ones) are displayed on the right, for the Linear-SVM OE algorithm (top-right) and the Linear-SVM ARX algorithm (lower-right).

It is clear from Table 1 and Fig. 2 that the ARX algorithm outperformed the OE algorithm, but that the OE algorithm performance was reasonably satisfactory. Also, the added complexity of the OE algorithm resulted in additional computation time, and inferior results.
3.2 Simulation Example 2: including output noise

The two algorithms were then compared on the same system, with the exception that the linear dynamics were replaced with an OE model with the same plant transfer function. Thus, (29) was replaced with

\[ A(z)(y + \nu) = B(z)f(u) \]  

(30)

The rest of the simulation remained unchanged. Again, an RBF-kernel with \( \sigma = 1 \) and a regularization parameter \( c = 50 \) were used.

Figure 3 shows the elements of the OE Hammerstein system identified using both the ARX and OE versions of the SVM estimation procedures based on linear \( \varepsilon \)-insensitive loss function (\( k = 1 \)). One can see that the OE algorithm succeeded in identifying the linear and nonlinear elements of the system while the ARX algorithm performed very badly in this case.

Moreover, Table 2 highlights the superiority of the OE algorithm over the ARX algorithm in estimating system (30) with different values of \( \varepsilon \). The price paid for the increased accuracy is longer computation and a less sparse model.

Table 2. Mean absolute error between true and estimated nonlinearity in addition to the number of SVs needed to model the nonlinearity and Computation Time.

<table>
<thead>
<tr>
<th>Method</th>
<th>MAE</th>
<th>Number of SV</th>
<th>Computation Time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARX Algorithm ( \varepsilon = 0.001 )</td>
<td>0.8264</td>
<td>115</td>
<td>124.7071</td>
</tr>
<tr>
<td>OE Algorithm ( \varepsilon = 0.001 )</td>
<td>0.0207</td>
<td>329</td>
<td>334.683</td>
</tr>
</tbody>
</table>

4. CONCLUSION

In this paper, an identification algorithm for OE Hammerstein models consisting of a SVM nonlinearity followed by a linear OE model was developed. In order to test and compare the performance of the proposed algorithm against the algorithm developed to identify the ARX Hammerstein models developed in (Al Dhaifallah and Westwick, 2008), two simulation examples have been presented.

In the first simulation example, the OE algorithm developed in this paper performed adequately, even though the original system was an ARX Hammerstein system. In the second simulation example, an OE Hammerstein system was considered. It was clear from the results that the OE algorithm outperformed the ARX algorithm in this case. By comparing the computation time of the two approaches, one can see that the OE algorithm needs more time to converge.

REFERENCES


