Keywords: Particle Swarm Optimization, Neural Network Training, Subspace Identification, Static Nonlinearity, Dynamic Linearity, Radial Basis Function (RBF) Neural Networks.

Abstract: This paper presents a new method for modeling of Hammerstein systems. The developed identification method uses state-space model in cascade with radial basis function (RBF) neural network. A recursive algorithm is developed for estimating neural network synaptic weights and parameters of the state-space model. No assumption on the structure of nonlinearity is made. The proposed algorithm works under the weak assumption of richness of inputs. The problem of modeling is solved as an optimization problem and Particle Swarm Optimization (PSO) is used for neural network training. Performance of the algorithm is evaluated in the presence of noisy data and Monte-Carlo simulations are performed to ensure reliability and repeatability of the identification technique.

1 INTRODUCTION

The Hammerstein Model belongs to the family of block oriented models and is made up of a memoryless nonlinear part followed by a linear dynamic part as shown in Figure 1. It has been known to effectively represent and approximate several nonlinear dynamic industrial processes, for example pH neutralization process (Fruzzetti, K.P., Palazoglu A., McDonald, K.A., 1997), heat exchanger (Eskinat, E., Johnson, S.H., Luyben, W.L., 1991), nonlinear filters (Haddad, A.H., Thomas, J.B., 1968), and water heater (Abonyi, I., Nagy, L., Szeifert, E., 2000).

A lot of research has been carried out on identification of Hammerstein models. Hammerstein Systems can be modeled by employing either nonparametric or parametric models. Nonparametric models represent the system in terms of curves resulting from expansion of series such as the Volterra series or kernel regression. Parametric representations are more compact having fewer parameters. Notable parametric identification techniques can be found in (Narendra, K.S., Gallman, P., 1966), (Billings, S., 1980), (Al-Duwaish, H., 2001), (Vörös, J., 2002), (Wenxiao, Z., 2007) and in references therein. Nonparametric identification techniques can be found in several papers including, but not limited to those of (Greblicki, W., 1989), (Al-Duwaish, H., Nazmulkarim, M., Chandrasekar, V., 1997), (Zhao, W., Chen, H., 2006).

Recently, subspace identification has emerged as a well known method for identification of linear systems. It is computationally less complicated as compared to conventional prediction error methods (PEM), does not require initial estimate of a canonical model like PEM and, is easily extendable to systems having multiple inputs and outputs (Katayama, T., 2005). However, its use is restricted mostly to linear systems. To make use of this, attempts have been...
made to extend subspace linear identification to nonlinear systems such as Wiener and Hammerstein systems including use of static nonlinearity in the feedback path (Luo, D., Leonessa, A., 2002), assuming known nonlinearity structures (Verhaegen, M., Westwick, D., 1996), and using least squares support vector machines (Goethals, I., Pelckmans, K., Suykens, J.A.K., Moor, B.D., 2005).

In this paper, a new subspace based method is proposed for Hammerstein model identification, which uses radial basis function (RBF) network in cascade with a state-space model. A recursive algorithm is developed for parameter estimation of the two subsystems.

The paper is arranged as follows. Section 2 looks at the model structure proposed in this work. Section 3 takes a detailed look at the proposed identification scheme. Section 4 describes the proposed algorithm in detail and section 5 includes numerical examples, their results, and analysis.

Throughout this paper, the following convention is used for notations. Lower case variables represent scalars. Lower case bold variables represent vectors. Upper case bold letters denote matrices. The only exception to this convention is the choice of variable for the cost function, where a more conventional $J$ is used to define the cost function.

## 2 PROPOSED MODEL STRUCTURE

The proposed model structure in this work uses state-space model to estimate the linear dynamic part. The memoryless nonlinear part is modeled using an RBF network. An RBF network is an effective type of neural network that has proved useful in applications like function approximation and pattern recognition. A typical three layer RBF network is shown in Figure 2. The input layer connects the network to its environment. The second layer, known as the hidden layer, performs a fixed nonlinear transformation using basis functions. The output layer linearly weighs the response of the network to the output (Haykin, S., 1999). The external inputs to the system $u(t)$ are fed to the RBF network, which generates the outputs $v(t)$. Considering an RBF network having $q$ number of neurons in the hidden layer, the basis vector is

$$\phi(t) = [\phi(u(t) - c_1) \cdots \phi(u(t) - c_q)],$$

where $c_i$ is the chosen center for the $i^\text{th}$ neuron, $\| . \|$ denotes norm that is usually Euclidean, and $\phi_i$ is the nonlinear radial basis function for the $i^\text{th}$ neuron, given by

$$\phi_i(t) = \exp \left( -\frac{\|u(t) - c_i\|^2}{2\sigma^2} \right),$$

where $\sigma$ is the spread of the Gaussian function $\phi_i(t)$. If the set of output layer weights of the RBF network is given by

$$w = [w_1 \ w_2 \cdots w_q],$$

the RBF output $v(t)$ is given by

$$v(t) = w\phi^T(t). \tag{1}$$

Considering a system with a single input and output, the output of the RBF network $v(t)$ in turn acts as input to the state-space model translating it into final output $y(t)$. The equation for $y(t)$ is given by discrete time state-space equation

$$\begin{align*}
\mathbf{x}(t+1) &= A\mathbf{x}(t) + Bv(t) + \mathbf{w}(t), \tag{2} \\
y(t) &= C\mathbf{x}(t) + Dv(t) + z(t), \tag{3}
\end{align*}$$

where $v(t)$ and $y(t)$ are input and output of the state-space system at discrete time instant $(t)$, $z(t)$ and $\mathbf{w}(t)$ are the measurement and process noise.

## 3 PROPOSED IDENTIFICATION SCHEME

The problem of Hammerstein modeling is therefore formulated as follows. Given a set of $m$ measurements of noisy inputs $u(t)$ and outputs $y(t)$, the problem is reduced to finding the weights of the RBF network and the matrices of the state-space model.

For the estimation of state-space matrices, N4SID numerical algorithm for subspace identification

![Figure 2: An RBF neural network with q neurons in the hidden layer.](image-url)
Notes the number of inputs and index is given by

For the system described in equations (1)-(3), the cost function problem where a cost index is to be minimized.

System, and that of the estimated model be minimized that the error between the output of the actual

B.D., 1994).

ber of measurements of inputs and outputs generated by the unknown system of equations (2) and (3) is

of outputs of the system, without any prior knowledge of the structure of the system, given that a large num-

ber, covariance matrices $Q \in \mathbb{R}^{n \times n}$, $R \in \mathbb{R}^{1 \times 1}$, $S$

$\in \mathbb{R}^{n \times 1}$, and the Kalman gain matrix $K$, where $p$ denotes the number of inputs and $r$ denotes the number of outputs of the system, without any prior knowledge of the structure of the system, given that a large number of measurements of inputs and outputs generated by the unknown system of equations (2) and (3) is provided. In N4SID, Kalman filter states are first estimated directly from input and output data, then the system matrices are obtained (Overschee, P.V., Moor, B.D., 1994).

For Hammerstein identification problem, it is desired that the error between the output of the actual system, and that of the estimated model be minimized. Therefore, in a way this becomes an optimization problem where a cost index is to be minimized. For the system described in equations (1)-(3), the cost index is given by

$$J = \sum_{t=1}^{m} e^2(t) = \sum_{t=1}^{m} (y(t) - \hat{y}(t))^2,$$

where $y(t)$ and $\hat{y}(t)$ are the outputs of the actual and estimated systems at time instant $(t)$. The weights of the RBF network are therefore updated so as to minimize this cost index. For this purpose, particle swarm optimization (PSO) used.

PSO is a heuristic optimization algorithm which works on the principle of swarm intelligence (Kennedy, J., Eberhart, R., 2001). It imitates animals living in a swarm collaboratively working to find their food or habitat. In PSO, the search is directed, as every particle position is updated in the direction of the optimal solution. It is robust and fast and can solve complex problems. It generates globally optimum solutions and exhibits stable convergence characteristics. In this work, PSO is used to train the RBF network. Each particle of the swarm represents a candidate value for the weight of the output layer of RBF network. The fitness of the particles is the reciprocal of the cost index given in equation (4). Hence, the smaller the sum of output errors, the more fit are the particles. Based on this principle, PSO updates the position of all the particles moving towards an optimal solution for the weights of RBF neural network.

The $i^{th}$ particle of the swarm is given by a $k$-dimension vector $\hat{x}_i = [\hat{x}_{i1} \cdots \hat{x}_{ik}]$, where $k$ denotes the number of optimized parameters. Similar vectors $\hat{p}_i$ and $\hat{v}_i$ denote the best position and velocity of the $i^{th}$ particle respectively. The velocity of the $i^{th}$ particle is updated as

$$\hat{v}_i(t+1) = \chi \{ [w\hat{y}_i(t) + c_1r_1(t) \{ \hat{p}_i(t) - \hat{x}_i(t) \} + c_2r_2(t) \{ \hat{p}_g(t) - \hat{x}_i(t) \}]$$

and the particle position is updated as

$$\hat{x}_i(t+1) = \hat{x}_i(t) + \hat{v}_i(t+1).$$

In the above equations, $\hat{p}_g$ denotes global best positions, while $c_1$ and $c_2$ are the cognitive and social parameters respectively, and are both positive constants. Parameter $\omega$ is the inertia weight and $\chi$ is called the constriction factor (Eberhart, R., Shi, Y., 1998). The value of cognitive parameter $c_1$ signifies a particle’s attraction to a local best position based on its past experiences. The value of social parameter $c_2$ determines the swarm’s attraction towards a global best position.

4 TRAINING ALGORITHM

Given a set of $m$ observations of input and output, $u \in \mathbb{R}^{1 \times m}$ and $y \in \mathbb{R}^{1 \times n}$, a hybrid PSO/Subspace identification algorithm is proposed below based on minimization of output error given in equation (4).

1. Estimate state-space matrices $A_0$, $B_0$, $C_0$ and $D_0$ (initial estimate) from original non linear data using N4SID.

2. $\text{Iteration} = k = 1$.

3. Initialize PSO with random population of possible RBF network weights.

4. $w_k = \min_{w \in \mathbb{R}^{1 \times m}} J(A_{k-1}, B_{k-1}, C_{k-1}, D_{k-1}, w)$.

5. Estimate set of RBF neural network outputs $\hat{v}_k \in \mathbb{R}^{1 \times m}$

$$\hat{v}_k = w_k \Phi^T = \left[\begin{array}{c} w_{1k} \cdots w_{qk} \\
\vdots \\
\phi(1) \\
\phi(m)
\end{array} \right] \begin{bmatrix} \phi(1) & \cdots & \phi(1)_q \\
\vdots & \ddots & \vdots \\
\phi(m)_1 & \cdots & \phi(m)_q \\
\end{bmatrix}^T.$$

6. Estimate state space matrices $A_k$, $B_k$, $C_k$ and $D_k$ from $[\hat{v}_k, y]$. This estimate of state-space model would be an improvement on the previous estimate.

7. Regenerate $\hat{y}_k \in \mathbb{R}^{1 \times m}$.

8. If minimum goal is not achieved, iteration $= k + 1$. Repeat steps 3 to 7.
5 SIMULATION RESULTS

5.1 Example 1

The first example considers the following Hammerstein type nonlinear process whose static nonlinearity is given by

\[ v(t) = \text{sign}(u(t)) \sqrt{|u(t)|}. \]  (7)

The dynamic linear part is given by a third order discrete time state-space system

\[
\begin{bmatrix}
1.80 & 1 & 0 \\
-1.07 & 0 & 1 \\
0.21 & 0 & 0 \\
\end{bmatrix}
\begin{bmatrix}
v(t-1) \\
v(t) \\
\end{bmatrix}
+
\begin{bmatrix}
4.80 \\
1.93 \\
1.21 \\
\end{bmatrix}
\begin{bmatrix}
1 & 0 & 0 \\
\end{bmatrix}
\begin{bmatrix}
\lambda_0 \lambda_1 \lambda_2 \\
\end{bmatrix}
\begin{bmatrix}
\lambda_0 u(t-1) u(t) \\
\end{bmatrix}
\]

The eigen values of the linear subsystem lie at \(\lambda_1 = 0.7\), \(\lambda_2 = 0.6\), and \(\lambda_3 = 0.5\). Desired outputs are generated by exciting the process model with a rich set of uniformly distributed random numbers in the interval \([-1.75, 1.75]\). An RBF network of 10 neurons is initialized with random synaptic weights and centers uniformly distributed in the input interval. PSO social and cognitive parameters \(c_1\) and \(c_2\) are kept almost equal to each other with \(c_1\) slightly larger than \(c_2\) and \(c_1 + c_2 \geq 4\) as proposed in (Carlisle, A., Dozier, G., 2001). This allow trusting past experiences as well as ample exploration of the swarm for a global best solution. Constriction factor is kept close to 1 to enable slow convergence with better exploration. Number of particles amount to 10 for the synaptic weights of 10 neurons. A swarm population size of 50 is selected and the optimization process is run for 100 iterations.

The algorithm shows promising results and mean squared error between normalized outputs of actual and estimated systems converges to a final value of \(4 \times 10^{-3}\) in 30 iterations. Nonlinearity estimate is shown in Figure 6. The eigen values of the estimated system lie at \(\hat{\lambda}_1^{10dB} = 0.73\), \(\hat{\lambda}_2^{10dB} = 0.73\) and \(\hat{\lambda}_3^{10dB} = 0.58\).

The results presented above are obtained from a single run of estimation algorithm. To further ensure the reliability and repeatability of the algorithm, Monte-Carlo simulation is carried out and ensemble statistics are tabulated in Table 1. The statistics show encouraging convergence of normalized output squared error and estimation of linear subsystem. Parameters of the nonlinearity cannot be compared because of the nonparametric nature of estimation. At best, the estimates of nonlinearity can be judged from the shapes of estimated nonlinear function as presented in Figures 3 and 6.

5.2 Example 2

The second example considers the following Hammerstein type nonlinear process whose static nonlinearity is given by

\[ v(t) = \tanh[2u(t)] \quad 1.5 \geq u(t), \]
\[ v(t) = \frac{\exp(u(t)) - 1}{\exp(u(t)) + 1} \quad 4 > u(t) > 1.5. \]

The dynamic linear part is given by the following second order discrete time state-space system

\[
\begin{bmatrix}
1.0 & 1.0 \\
-0.5 & 0.0 \\
\end{bmatrix}
\begin{bmatrix}
v(t-2) \\
v(t-1) \\
\end{bmatrix}
\begin{bmatrix}
1 \\
0.5 \\
\end{bmatrix}
\begin{bmatrix}
\lambda_0 \lambda_1 \lambda_2 \\
\end{bmatrix}
\begin{bmatrix}
\lambda_0 u(t-2) u(t-1) \\
\end{bmatrix}
\]

The eigenvalues of the linear subsystem have eigen values at \(\hat{\lambda}_{1,2} = 0.5 \pm 0.5i\). Desired outputs are generated by
Table 1: Monte-Carlo simulation statistics for example 1

<table>
<thead>
<tr>
<th>Estimation results without output noise</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Total number of runs</td>
<td>200</td>
</tr>
<tr>
<td>Magnitude of actual eigen value $\lambda_1$ of linear subsystem</td>
<td>0.7</td>
</tr>
<tr>
<td>Mean magnitude of estimated eigen value $\hat{\lambda}_1$ of linear subsystem</td>
<td>0.713</td>
</tr>
<tr>
<td>Variance of eigen value estimate $\hat{\lambda}_1$</td>
<td>$8 \times 10^{-4}$</td>
</tr>
<tr>
<td>Magnitude of actual eigen value $\lambda_2$ of linear subsystem</td>
<td>0.6</td>
</tr>
<tr>
<td>Mean magnitude of estimated eigen value $\hat{\lambda}_2$ of linear subsystem</td>
<td>0.62</td>
</tr>
<tr>
<td>Variance of eigen value estimate $\hat{\lambda}_2$</td>
<td>$4.5 \times 10^{-3}$</td>
</tr>
<tr>
<td>Magnitude of actual eigen value $\lambda_3$ of linear subsystem</td>
<td>0.5</td>
</tr>
<tr>
<td>Mean magnitude of estimated eigen value $\hat{\lambda}_3$ of linear subsystem</td>
<td>0.481</td>
</tr>
<tr>
<td>Variance of eigen value estimate $\hat{\lambda}_3$</td>
<td>$5.7 \times 10^{-3}$</td>
</tr>
<tr>
<td>Average number of iterations required for every run</td>
<td>8.26</td>
</tr>
<tr>
<td>Average mean squared output error (MSE)</td>
<td>$9.8 \times 10^{-4}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Estimation results with output SNR 10dB</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Total number of runs</td>
<td>200</td>
</tr>
<tr>
<td>Magnitude of actual eigen value $\lambda_1$ of linear subsystem</td>
<td>0.7</td>
</tr>
<tr>
<td>Mean magnitude of estimated eigen value $\hat{\lambda}_1$ of linear subsystem</td>
<td>0.75</td>
</tr>
<tr>
<td>Variance of eigen value estimate $\hat{\lambda}_1$</td>
<td>$1.6 \times 10^{-3}$</td>
</tr>
<tr>
<td>Magnitude of actual eigen value $\lambda_2$ of linear subsystem</td>
<td>0.6</td>
</tr>
<tr>
<td>Mean magnitude of estimated eigen value $\hat{\lambda}_2$ of linear subsystem</td>
<td>0.68</td>
</tr>
<tr>
<td>Variance of eigen value estimate $\hat{\lambda}_2$</td>
<td>$6 \times 10^{-3}$</td>
</tr>
<tr>
<td>Magnitude of actual eigen value $\lambda_3$ of linear subsystem</td>
<td>0.5</td>
</tr>
<tr>
<td>Mean magnitude of estimated eigen value $\hat{\lambda}_3$ of linear subsystem</td>
<td>0.4</td>
</tr>
<tr>
<td>Variance of eigen value estimate $\hat{\lambda}_3$</td>
<td>$6 \times 10^{-2}$</td>
</tr>
<tr>
<td>Average number of iterations required for every run</td>
<td>8.02</td>
</tr>
<tr>
<td>Average mean squared output error (MSE)</td>
<td>$6.6 \times 10^{-3}$</td>
</tr>
</tbody>
</table>
exciting the process model with a rich set of uniformly distributed random numbers in the interval [0,4]. An RBF network with 25 neurons is initialized with random synaptic weights and uniformly distributed centers chosen within the input interval. PSO social and cognitive parameters and constriction factor are kept similar to example 1. The number of particles is equal to the number of neurons and a population size of 50 gives good results again. The algorithm performs well and estimates the system in 20 iterations. The mean squared output error between normalized values of actual and estimated outputs converges to a final value of $8 \times 10^{-4}$. The estimate of nonlinearity shape is shown in Figure 7. Eigenvalues of the estimated system lie at $\hat{\lambda}_{1,2} = 0.497 \pm 0.5i$.

Step response of linear subsystem is shown in Figure 8. The squared error convergence plot is shown in Figure 9.

Estimation is then carried out in noisy environment, with zero mean Gaussian additive noise included at the output of the system such that the signal to noise ratio (SNR) is 10dB. The algorithm performs well in noisy environment. The final mean squared error converges to $1.8 \times 10^{-3}$ in 30 iterations. Nonlinearity estimate is shown in Figure 10. The eigenvalues of the estimated system lie at $\hat{\lambda}_{1,2}^{10dB} = 0.493 \pm 0.499i$.

Table 2 shows ensemble statistics of Monte-Carlo simulation for example 2. The statistics show encouraging convergence of normalized output squared error and estimation of linear subsystem. As mentioned before, parameters of the nonlinearity cannot be compared due to the nonparametric nature of estimation. At best, the estimates of nonlinearity can be judged from the shapes of estimated nonlinear function as presented in Figures 7 and 10.

6 CONCLUSION

The PSO/Subspace algorithm is basically a combination of PSO and Subspace N4SID algorithm, and hence its convergence properties are directly related to the convergence properties of PSO and Subspace algorithms. PSO has been usually known to perform better than most evolutionary algorithms (EA)s in finding global optimum provided its parameters are tuned properly according to the application. The subspace algorithm is also known for having no convergence problems. Its numerical robustness is guaranteed because of well understood linear algebra techniques like QR decomposition and singular value decomposition (SVD). As a consequence, it does not experience problems like lack of convergence, slow con-
Table 2: Monte-Carlo simulation statistics for example 2

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Estimation results without output noise</strong></td>
<td></td>
</tr>
<tr>
<td>Total number of runs</td>
<td>200</td>
</tr>
<tr>
<td>Magnitude of actual eigen values $\lambda_{1,2}$ of linear subsystem</td>
<td>0.7071</td>
</tr>
<tr>
<td>Mean magnitude of estimated eigen values $\hat{\lambda}_{1,2}$ of linear subsystem</td>
<td>0.7071</td>
</tr>
<tr>
<td>Variance of eigen value estimates</td>
<td>$2 \times 10^{-5}$</td>
</tr>
<tr>
<td>Average number of iterations required for every run</td>
<td>9</td>
</tr>
<tr>
<td>Average mean squared output error (MSE)</td>
<td>$7 \times 10^{-4}$</td>
</tr>
<tr>
<td><strong>Estimation results with output SNR 10dB</strong></td>
<td></td>
</tr>
<tr>
<td>Total number of runs</td>
<td>200</td>
</tr>
<tr>
<td>Magnitude of actual eigen values $\lambda_{1,2}$ of linear subsystem</td>
<td>0.7071</td>
</tr>
<tr>
<td>Mean magnitude of estimated eigen values $\hat{\lambda}_{1,2}$ of linear subsystem</td>
<td>0.7079</td>
</tr>
<tr>
<td>Variance of eigen value estimates</td>
<td>$6 \times 10^{-5}$</td>
</tr>
<tr>
<td>Average number of iterations required for every run</td>
<td>21</td>
</tr>
<tr>
<td>Average mean squared output error (MSE)</td>
<td>$2 \times 10^{-3}$</td>
</tr>
</tbody>
</table>

Figure 7: Estimate of tangent-hyperbolic nonlinearity of example 2.

Figure 8: Step response of linear dynamic part of example 2.

ACKNOWLEDGEMENT

The authors would like to acknowledge the support of King Fahd University of Petroleum & Minerals, Dhahran, Saudi Arabia.

REFERENCES


Al-Duwaish, H. (2001). A genetic approach to the identification of linear dynamical systems with static nonlin-
Figure 9: Mean squared error for example 2.

Figure 10: Estimate of tangent-hyperbolic nonlinearity of example 2 with output SNR 10dB.


