PSO assisted NURB neural network identification

X. Hong[1] and S. Chen[2]

[1] School of Systems Engineering, University of Reading, UK
[2] School of Electronics and Computer Science, University of Southampton, UK, and Faculty of Engineering, King Abdulaziz University, Jeddah 21589, Saudi Arabia.

Abstract. A system identification algorithm is introduced for Hammerstein systems that are modelled using a non-uniform rational B-spline (NURB) neural network. The proposed algorithm consists of two successive stages. First the shaping parameters in NURB network are estimated using a particle swarm optimization (PSO) procedure. Then the remaining parameters are estimated by the method of the singular value decomposition (SVD). Numerical examples including a linear pole assignment controller are utilized to demonstrate the efficacy of the proposed approach.

Keywords: B-spline, NURB neural networks, De Boor algorithm, Hammerstein model, pole assignment controller, particle swarm optimization, system identification.

1 Introduction

The Hammerstein model, comprising a nonlinear static functional transformation followed by a linear dynamical model, has been widely researched [3,21,2,12]. The model characterization/representation of the unknown nonlinear static function is fundamental to the identification of Hammerstein model. Various approaches have been developed in order to capture the *a priori* unknown nonlinearity by use of both parametric [22,6] and nonparametric methods [17,11,7]. The special structure of Hammerstein models can be exploited to develop hybrid parameter estimation algorithms [2,1,6].

Both the uniform/nonrational B-spline curve and the non-uniform/rational B-spline (NURB) curve have also been widely used in computer graphics and computer aided geometric design (CAGD) [8]. These curves consist of many polynomial pieces, offering much more versatility than do Bezier curves while maintaining the same advantage of the best conditioning property. NURB is a generalization of the uniform, non-rational B-splines, and offers much more versatility and powerful approximation capability. The NURB neural network possesses a much more powerful modeling capability than a conventional non-rational B-spline neural network because of the extra shaping parameters. This motivates us to propose the use of NURB neural networks to model the nonlinear static function in the Hammerstein system.

The PSO [15,16] constitutes a population based stochastic optimisation technique, which was inspired by the social behaviour of bird flocks or fish schools. It has been successfully applied to wide-ranging optimisation problems [18,20]. This paper introduces a hybrid system identification consisting two successive stages. We note that the model output can be represented as a linear in the parameters model once the shaping parameters are fixed. This means that the mean squares error due to the shaping parameters can be easily obtained using the least squares method, without explicitly estimating the other parameters. In the proposed algorithm the shaping parameters in NURB neural networks are estimated using the particle swarm optimization (PSO) as the first step, in which the mean square error is used as the cost function.

A popular treatment of the control of the Hammerstein model is to remove the nonlinearity via an inversion [9,4,19]. In this study, the controller consists of computing the inverse of the nonlinear static function approximated by NURB, followed by a linear pole assignment controller. The linearization of the closed loop system is achieved by inserting the inverse of the identified static nonlinearity via the inverse of De Boor algorithm [14] which was introduced for the control of B-spline based Hammerstein systems.

2 The Hammerstein system

The Hammerstein system consists of a cascade of two subsystems, a nonlinear memoryless function $\Psi(\bullet)$ as the first subsystem, followed by a linear dynamic part as the second subsystem. The system can be represented by

$$\begin{aligned} y(t) &= \hat{y}(t) + \xi(t) \\ &= -a_1 y(t-1) - a_2 y(t-2) - ... - a_{n_a} y(t-n_a) \\ &\quad + b_1 v(t-1) + ... + b_{n_b} v(t-n_b) + \xi(t) \qquad (1) \\ v(t-j) &= \Psi(u(t-j)), \quad j = 1, ..., n_b \qquad (2) \end{aligned}$$

where $y(t)$ is the system output and $u(t)$ is the system input. $\xi(t)$ is assumed to be a white noise sequence independent of $u(t)$ with zero mean and variance of σ^2. $v(t)$ is the output of nonlinear subsystem and the input to the linear subsystem. a_j's, b_j's are parameters of the linear subsystem. n_a and n_b are assumed known system output and input lags. Denote $\mathbf{a} = [a_1, ..., a_{n_a}]^T \in \Re^{n_a}$ and $\mathbf{b} = [b_1, ..., b_{n_b}]^T \in \Re^{n_b}$. It is assumed that $A(q^{-1}) = 1 + a_1 q^{-1} + ... + a_{n_a} q^{-n_a}$ and $B(q^{-1}) = b_1 q^{-1} + ... + b_{n_b} q^{-n_b}$ are coprime polynomials of q^{-1}, where q^{-1} denotes the backward shift operator. The gain of the linear subsystem is given by

$$G = \lim_{q \to 1} \frac{B(q^{-1})}{A(q^{-1})} = \frac{\sum_{j=1}^{n_b} b_j}{1 + \sum_{j=1}^{n_a} a_j} \qquad (3)$$

The two objectives of the work are that of the system identification and the subsequent controller design for the identified model. The objective of system identification for the above Hammerstein model is that, given an observational input/output data set $D_N = \{y(t), u(t)\}_{t=1}^{N}$, to identify $\Psi(\bullet)$ and to estimate the parameters a_j, b_j in the linear subsystems. Note that the signals between the two subsystems are unavailable.

Without significantly losing generality the following assumptions are initially made about the problem.

Assumption 1: $\Psi(\bullet)$ is a one to one mapping, i.e. it is an invertible and continuous function.

Assumption 2: $u(t)$ is bounded by $U_{min} < u(t) < U_{max}$, where U_{min} and U_{max} are assumed known finite real values.

3 Modelling of Hammerstein system using NURB neural network

In this work the non-uniform rational B-spline (NURB) neural network is adopted in order to model $\Psi(\bullet)$. De Boor's algorithm is a fast and numerically stable algorithm for evaluating B-spline basis functions [5]. Univariate B-spline basis functions are parameterized by the order of a piecewise polynomial of order k, and also by a knot vector which is a set of values defined on the real line that break it up into a number of intervals. Supposing that there are d basis functions, the knot vector is specified by $(d+k)$ knot values, $\{U_1, U_2, \cdots, U_{d+k}\}$. At each end there are k knots satisfying the condition of being external to the input region, and as a result the number of internal knots is $(d-k)$. Specifically

$$\begin{aligned} U_1 < U_2 < U_k = U_{min} < U_{k+1} < U_{k+2} < \cdots < \\ U_d < U_{max} = U_{d+1} < \cdots < U_{d+k}. \end{aligned} \tag{4}$$

Given these predetermined knots, a set of d B-spline basis functions can be formed by using the De Boor recursion [5], given by

$$\mathcal{B}_j^{(0)}(u) = \begin{cases} 1 \text{ if } U_j \le u < U_{j+1} \\ 0 \quad \text{otherwise} \end{cases} \tag{5}$$

$$j = 1, \cdots, (d+k)$$

$$\left.\begin{aligned} \mathcal{B}_j^{(i)}(u) = \tfrac{u-U_j}{U_{i+j}-U_j}\mathcal{B}_i^{(i-1)}(u) \\ +\tfrac{U_{i+j+1}-u}{U_{i+j+1}-U_{j+1}}\mathcal{B}_{j+1}^{(i-1)}(u), \\ j = 1, \cdots, (d+k-i) \end{aligned}\right\} i = 1, \cdots, k \tag{6}$$

We model $\Psi(\bullet)$ as the NURB neural network in the form of

$$\Psi(u) = \sum_{j=1}^{d} \mathcal{N}_j^{(k)}(u)\omega_j \tag{7}$$

with

$$\mathcal{N}_j^{(k)}(u) = \frac{\lambda_j \mathcal{B}_j^{(k)}(u)}{\sum_{j=1}^{d} \lambda_j \mathcal{B}_j^{(k)}(u)} \tag{8}$$

where ω_j's are weights, $\lambda_j > 0$'s the shaping parameters that are to be determined. Denote $\boldsymbol{\omega} = [\omega_1, \cdots, \omega_d]^T \in \Re^d$. $\boldsymbol{\lambda} = [\lambda_1, \cdots, \lambda_d]^T \in \Re^d$. For uniqueness we set the constraint $\sum_{j=1}^{d} \lambda_j = 1$. Note that due to the piecewise nature of B-spline functions, there are only $(k+1)$ basis functions with nonzero values for any point u. Hence the computational cost for the evaluation of $\Psi(u)$ based on the De-Boor algorithm is determined by the polynomial order k, rather than the number of knots, and this is in the order of $O(k^2)$.

Our algorithm involves estimating the weights and the shaping parameters in the NURB model. Note that the proposed NURB neural network possesses a much more powerful modeling capability than a nonrational B-spline network because of the extra shaping parameters. This is advantageous because all the parameters are continuous variables that can be solved by nonlinear optimization, compared to presetting the knots by trial and error which does not yield to the optimum.

With specified knots and over the estimation data set D_N, $\boldsymbol{\lambda}, \boldsymbol{\omega}, \mathbf{a}, \mathbf{b}$ may be jointly estimated via

$$\min_{\boldsymbol{\lambda}, \boldsymbol{\omega}, \mathbf{a}, \mathbf{b}} \{J = \sum_{t=1}^{N} (y - \hat{y}(t, \boldsymbol{\lambda}, \boldsymbol{\omega}, \mathbf{a}, \mathbf{b}))^2\} \tag{9}$$

subject to

$$\lambda_j \geq 0, \forall j, \ \boldsymbol{\lambda}^T \mathbf{1} = 1 \text{ and } G = 1 \tag{10}$$

in which $G = 1$ is imposed for unique solution. We point out that this is still a very difficult nonlinear optimization problem due to the mixed constraints, and this motivates us to propose the following hybrid procedure. It is proposed that the shaping parameters λ_j's are found using the PSO, as the first step of system identification, followed by the estimation of the remaining parameters.

4 The system identification of Hammerstein system based on NURB using PSO

4.1 The basic idea

Initially consider using NURB approximation with a specified shape parameter vector $\boldsymbol{\lambda}$, the model predicted output $\hat{y}(t)$ in (1) can be written as

$$\begin{aligned} \hat{y}(t) = & -a_1 y(t-1) - a_2 y(t-2) - ... \\ & -a_{n_a} y(t-n_a) + b_1 \sum_{j=1}^{d} \omega_j \mathcal{N}_j^{(k)}(t-1) + ... \\ & +b_{n_b} \sum_{j=1}^{d} \omega_j \mathcal{N}_j^{(k)}(t-n_b) \end{aligned} \tag{11}$$

Over the estimation data set $D_N = \{y(t), u(t)\}_{t=1}^N$, (1) can be rewritten in a linear regression form

$$y(t) = [\mathbf{p}(\mathbf{x}(t))]^T \boldsymbol{\vartheta} + \xi(t) \tag{12}$$

where $\mathbf{x}(t) = [-y(t-1), ..., -y(t-n_a), u(t-1), ..., u(t-n_b)]^T$ is system input vector of observables with assumed known dimension of $(n_a + n_b)$, $\boldsymbol{\vartheta} = [\mathbf{a}^T, (b_1\omega_1), ..., (b_1\omega_d), ...(b_{n_b}\omega_1), ..., (b_{n_b}\omega_{n_b})]^T \in \Re^{n_a+d\cdot n_b}$,

$$\begin{aligned}\mathbf{p}(\mathbf{x}(t)) = [&-y(t-1), ..., -y(t-n_a),\\ &\mathcal{N}_1^{(k)}(t-1), ..., \mathcal{N}_d^{(k)}(t-1), ...\mathcal{N}_1^{(k)}(t-n_b)\\ &, ..., \mathcal{N}_d^{(k)}(t-n_b)]^T\end{aligned} \tag{13}$$

(12) can be rewritten in the matrix form as

$$\mathbf{y} = \mathbf{P}\boldsymbol{\vartheta} + \boldsymbol{\Xi} \tag{14}$$

where $\mathbf{y} = [y(1), \cdots, y(N)]^T$ is the output vector. $\boldsymbol{\Xi} = [\xi(1), ..., \xi(N)]^T$, and $\mathbf{P}$ is the regression matrix

$$\mathbf{P} = \begin{bmatrix} p_1(\mathbf{x}(1)) & p_2(\mathbf{x}(1)) & \cdots & p_{n_a+d\cdot n_b}(\mathbf{x}(1)) \\ p_1(\mathbf{x}(2)) & p_2(\mathbf{x}(2)) & \cdots & p_{n_a+d\cdot n_b}(\mathbf{x}(2)) \\ \cdots & \cdots & \cdots & \cdots \\ p_1(\mathbf{x}(N)) & p_2(\mathbf{x}(N)) & \cdots & p_{n_a+d\cdot n_b}(\mathbf{x}(N)) \end{bmatrix} \tag{15}$$

The parameter vector $\boldsymbol{\vartheta}$ can be found as the least squares solution of

$$\boldsymbol{\vartheta}_{LS} = \mathbf{B}^{-1}\mathbf{P}^T\mathbf{y} \tag{16}$$

provided that $\mathbf{B} = \mathbf{P}^T\mathbf{P}$ is of full rank. Alternatively if this condition is violated, i.e. $Rank(\mathbf{B}) = r < n_a + d \cdot n_b$, then performing the eigenvalue decomposition $\mathbf{BQ} = \mathbf{Q\Sigma}$, where $\boldsymbol{\Sigma} = \mathrm{diag}[\sigma_1, ...\sigma_r, 0, \cdots, 0]$ with $\sigma_1 > \sigma_2 > ... > \sigma_r > 0$. $\mathbf{Q} = [\mathbf{q}_1, \cdots, \mathbf{q}_{n_a+d\cdot n_b}]$, followed by truncating the eigenvectors corresponding to zero eigenvalues, we have

$$\boldsymbol{\vartheta}_{LS}^{svd} = \sum_{i=1}^{r} \frac{\mathbf{y}^T\mathbf{P}\mathbf{q}_i}{\sigma_i}\mathbf{q}_i \tag{17}$$

Thus the mean square error can be readily computed from

$$J(\boldsymbol{\lambda}) = [\mathbf{y} - \mathbf{P}\boldsymbol{\vartheta}_{LS}^{svd}]^T[\mathbf{y} - \mathbf{P}\boldsymbol{\vartheta}_{LS}^{svd}]/N. \tag{18}$$

for any specified $\boldsymbol{\lambda}$. Note that it is computationally simple to evaluate $J(\boldsymbol{\lambda})$ due to the fact that the model has a linear in the parameter structure for a given $\boldsymbol{\lambda}$. This is an important observation for simplifying the algorithm design. This suggests that we can optimize $\boldsymbol{\lambda}$ as the first task. The information of other models parameters are implicit in $\boldsymbol{\vartheta}_{LS}^{svd}$ and dependent on $\boldsymbol{\lambda}$. We point out that at this stage other models parameters are not estimated which would be much more computationally involved but unnecessary.

4.2 Particle swarm optimisation for estimating the shaping parameters $\boldsymbol{\lambda}_j$'s

In the following we propose to apply the PSO algorithm [15,16], and aim to solve

$$\boldsymbol{\lambda}_{\text{opt}} = \arg \min_{\boldsymbol{\lambda} \in \prod_{j=1}^{d} \Lambda_j} J(\boldsymbol{\lambda}), \quad \text{s.t.} \quad \boldsymbol{\lambda}^T \mathbf{1} = 1 \tag{19}$$

where $\mathbf{1}$ denotes a vector of all ones with appropriate dimension.

$$\prod_{j=1}^{d} \Lambda_j = \prod_{j=1}^{d} [0,\ 1] \quad \text{s.t.} \quad \boldsymbol{\lambda}^T \mathbf{1} = 1 \tag{20}$$

defines the search space. A swarm of particles, $\{\boldsymbol{\lambda}_i^{(l)}\}_{i=1}^S$, that represent potential solutions are "flying" in the search space $\prod_{j=1}^{d} \Lambda_j$, where S is the swarm size and index l denotes the iteration step. The algorithm is summarised as follows.

a) Swarm initialisation. Set the iteration index $l = 0$ and randomly generate $\{\boldsymbol{\lambda}_i^{(l)}\}_{i=1}^S$ in the search space $\prod_{j=1}^{d} \Lambda_j$. These are obtained by randomly set each element of $\{\boldsymbol{\lambda}_i^{(l)}\}_{i=1}^S$ as *rand*() (denoting the uniform random number between 0 and 1), followed normalizing them by

$$\boldsymbol{\lambda}_i^{(0)} = \boldsymbol{\lambda}_i^{(0)} / \sum_{j=1}^{d} \boldsymbol{\lambda}_i^{(0)}|_j \tag{21}$$

where $\bullet|_j$ denotes the j^{th} element of $\bullet$, so that $\{\boldsymbol{\lambda}_i^{(0)}\}^T \mathbf{1} = 1$ is valid.

b) Swarm evaluation. The cost of each particle $\boldsymbol{\lambda}_i^{(l)}$ is obtained as $J(\boldsymbol{\lambda}_i^{(l)})$. Each particle $\boldsymbol{\lambda}_i^{(l)}$ remembers its best position visited so far, denoted as $\mathbf{pb}_i^{(l)}$, which provides the cognitive information. Every particle also knows the best position visited so far among the entire swarm, denoted as $\mathbf{gb}^{(l)}$, which provides the social information. The cognitive information $\{\mathbf{pb}_i^{(l)}\}_{i=1}^S$ and the social information $\mathbf{gb}^{(l)}$ are updated at each iteration:

```
For ($i = 1$; $i \leq S$; $i$++)
    If ($J(\boldsymbol{\lambda}_i^{(l)}) < J(\mathbf{pb}_i^{(l)})$)   $\mathbf{pb}_i^{(l)} = \boldsymbol{\lambda}_i^{(l)}$;
End for;
$i^* = \arg \min_{1 \leq i \leq S} J(\mathbf{pb}_i^{(l)})$;
If ($J(\mathbf{pb}_{i^*}^{(l)}) < J(\mathbf{gb}^{(l)})$)   $\mathbf{gb}^{(l)} = \mathbf{pb}_{i^*}^{(l)}$;
```

c) Swarm update. Each particle $\boldsymbol{\lambda}_i^{(l)}$ has a velocity, denoted as $\boldsymbol{\gamma}_i^{(l)}$, to direct its "flying". The velocity and position of the ith particle are updated in each iteration according to

$$\begin{aligned} \boldsymbol{\gamma}_i^{(l+1)} &= \mu_0 * \boldsymbol{\gamma}_i^{(l)} + rand() * \mu_1 * (\mathbf{pb}_i^{(l)} - \boldsymbol{\lambda}_i^{(l)}) \\ &\quad + rand() * \mu_2 * (\mathbf{gb}^{(l)} - \boldsymbol{\lambda}_i^{(l)}), \end{aligned} \tag{22}$$

$$\boldsymbol{\lambda}_i^{(l+1)} = \boldsymbol{\lambda}_i^{(l)} + \boldsymbol{\gamma}_i^{(l+1)}, \tag{23}$$

where μ_0 is the inertia weight, μ_1 and μ_2 are the two acceleration coefficients. In order to avoid excessive roaming of particles beyond the search space [13], a velocity space

$$\prod_{j=2}^{d} \varUpsilon_j = \prod_{j=2}^{d} [-\varUpsilon_{j,\max},\ \varUpsilon_{j,\max}] \tag{24}$$

is imposed on $\boldsymbol{\gamma}_i^{(l+1)}$ so that

$$\begin{aligned}&\text{If } (\boldsymbol{\gamma}_i^{(l+1)}|_j > \varUpsilon_{j,\max}) \quad \boldsymbol{\gamma}_i^{(l+1)}|_j = \varUpsilon_{j,\max};\\ &\text{If } (\boldsymbol{\gamma}_i^{(l+1)}|_j < -\varUpsilon_{j,\max}) \quad \boldsymbol{\gamma}_i^{(l+1)}|_j = -\varUpsilon_{j,\max};\end{aligned}$$

Moreover, if the velocity as given in equation (22) approaches zero, it is reinitialised proportional to $\varUpsilon_{j,\max}$ with a small factor ν

$$\text{If } (\boldsymbol{\gamma}_i^{(l+1)}|_j == 0)\ \boldsymbol{\gamma}_i^{(l+1)}|_j = \pm rand() * \nu * \varUpsilon_{j,\max}; \tag{25}$$

In order to ensure each element of $\boldsymbol{\lambda}_i^{(l+1)}$ that it satisfies the constraint and stays in the space, we modified constraint check in the PSO as follows;

If $(\boldsymbol{\lambda}_i^{(l+1)}|_j < 0) \quad \boldsymbol{\lambda}_i^{(l+1)}|_j = 0;$

then

$$\boldsymbol{\lambda}_i^{(l+1)} = \boldsymbol{\lambda}_i^{(l+1)} / \sum_{j=1}^{d} \boldsymbol{\lambda}_i^{(l+1)}|_j \tag{26}$$

Note that the normalization step that we introduced here does not affect the cost function value, rather it effectively keeps the solution stay inside the bound.

d) Termination condition check. If the maximum number of iterations, $I_{\max}$, is reached, terminate the algorithm with the solution $\mathbf{gb}^{(I_{\max})}$; otherwise, set $l = l + 1$ and go to Step *b)*.

Ratnaweera and co-authors [20] reported that using a time varying acceleration coefficient (TVAC) enhances the performance of PSO. We adopt this mechanism, in which μ_1 is reduced from 2.5 to 0.5 and μ_2 varies from 0.5 to 2.5 during the iterative procedure:

$$\begin{aligned}\mu_1 &= (0.5 - 2.5) * l/I_{\max} + 2.5,\\ \mu_2 &= (2.5 - 0.5) * l/I_{\max} + 0.5.\end{aligned} \tag{27}$$

The reason for good performance of this TVAC mechanism can be explained as follows. At the initial stages, a large cognitive component and a small social component help particles to wander around or better exploit the search space, avoiding local minima. In the later stages, a small cognitive component and a large social component help particles to converge quickly to a global minimum. We use $\mu_0 = rand()$ at each iteration.

The search space as given in equation (20) is defined by the specific problem to be solved, and the velocity limit $\varUpsilon_{j,\max}$ is empirically set. An appropriate value of the small control factor ν in equation (25) for avoiding zero velocity is empirically found to be $\nu = 0.1$ for our application.

4.3 Estimating the parameter vectors $\boldsymbol{\omega}, \mathbf{a}, \mathbf{b}$ using $\boldsymbol{\vartheta}_{LS}^{svd}$

In this section we describe the second stage of Bai's two stage identification algorithm [2] which can be used to recover $\boldsymbol{\omega}, \mathbf{a}, \mathbf{b}$ from $\boldsymbol{\vartheta}_{LS}^{svd}(\boldsymbol{\lambda}_{\text{opt}})$ based on the result of PSO above. Our final estimate of $\hat{\mathbf{a}}$, which is simply taken as the subvector of the resultant $\boldsymbol{\vartheta}_{LS}^{svd}(\boldsymbol{\lambda}_{\text{opt}})$, consisting of its first n_a elements.

Rearrange the $(n_a+1)^{th}$ to $(n_a+(d+1)\times n_b)^{th}$ elements of $\boldsymbol{\vartheta}_{LS}^{svd}(\boldsymbol{\lambda}_{\text{opt}})$ into a matrix

$$\mathbf{M} = \begin{bmatrix} b_1\omega_0 & b_1\omega_1 & \cdots & b_1\omega_d \\ b_2\omega_0 & b_2\omega_1 & \cdots & b_2\omega_d \\ \multicolumn{4}{c}{\dotfill} \\ b_{n_b}\omega_0 & b_{n_b}\omega_1 & \cdots & b_{n_b}\omega_d \end{bmatrix} = \mathbf{b}\boldsymbol{\omega}^T \in \Re^{n_b\times(d+1)} \tag{28}$$

The matrix $\mathbf{M}$ has rank 1 and its singular value decomposition is of the form

$$\begin{aligned} \mathbf{M} &= \boldsymbol{\Gamma} \begin{bmatrix} \delta_{\mathbf{M}} & 0 \cdots 0 \\ 0 & 0 \cdots 0 \\ \multicolumn{2}{c}{\dotfill} \\ 0 & 0 \cdots 0 \end{bmatrix} \boldsymbol{\Delta}^T \\ &= \boldsymbol{\Gamma} \begin{bmatrix} \delta_{\mathbf{M}} \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 1\, 0 \ldots\, 0 \end{bmatrix} \boldsymbol{\Delta}^T \end{aligned} \tag{29}$$

where $\boldsymbol{\Gamma} = [\boldsymbol{\Gamma}_1, ..., \boldsymbol{\Gamma}_{n_b}] \in \Re^{n_b\times n_b}$ and $\boldsymbol{\Delta} = [\boldsymbol{\Delta}_1, ..., \boldsymbol{\Delta}_{d+1}] \in \Re^{(d+1)\times(d+1)}$, where $\boldsymbol{\Gamma}_i$ $(i = 1, .., n_b)$ and $\boldsymbol{\Lambda}_i$ $(i = 1, ..., (d+1))$ are orthonormal vectors. $\delta_{\mathbf{M}}$ is the sole non-zero singular value of $\mathbf{M}$. $\mathbf{b}$ and $\boldsymbol{\omega}$ can be obtained using

$$\begin{aligned} \hat{\mathbf{b}} &= \delta_{\mathbf{M}}\Gamma_1 \\ \hat{\boldsymbol{\omega}} &= \boldsymbol{\Delta}_1 \end{aligned} \tag{30}$$

followed by

$$\begin{aligned} \hat{\mathbf{b}} &\longleftarrow \beta\hat{\mathbf{b}} \\ \hat{\boldsymbol{\omega}} &\longleftarrow \hat{\boldsymbol{\omega}}/\beta \end{aligned} \tag{31}$$

where $\beta = (1 + \sum_{j=1}^{n_a} \hat{a}_j)/(\sum_{j=1}^{n_b} \hat{b}_j)$.

Note that the standard Bai's approach as above may suffer a serious numerical problem that the matrix $\mathbf{M}$ turns out to have rank higher than one, resulting in the parameters estimator far from usable. This issue was discussed in [10], in which the modified SVD approach was proposed to address the problem. The more stable modified SVD approach [10] is used in our simulations.

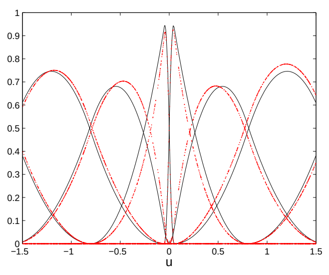
5 An illustrative example

A Hammerstein system is simulated, in which the linear subsystem is $A(q^{-1}) = 1 - 1.2q^{-1} + 0.9q^{-2}$, $B(q^{-1}) = 1.7q^{-1} - q^{-2}$, and the nonlinear subsystem

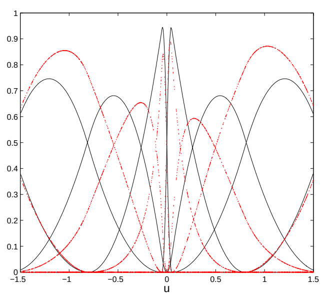
$\Psi(u) = 2\mathrm{sign}(u)\sqrt{|u|}$. The variance of the additive noise to the system output is set as 0.01. 1000 training data samples $y(t)$ were generated by using (1) and (2), where $u(t)$ was uniformly distributed random variable $u(t) \in [-1.5, 1.5]$. The polynomial degree of B-spline basis functions was set as $k = 2$ (piecewise quadratic). The knots sequence U_j is set as

$$[-3.2,\ -2.4,\ -1.6,\ -0.8,\ -0.05,\ 0,\ 0.05,\ 0.8,\ 1.6,\ 2.4,\ 3.2].$$

Initially the system identification was carried out. In the modified PSO algorithm, we set $S = 20$, $I_{\max} = 20$, $\Upsilon_{j,\max} = 0.025$. The resultant 8 NURB basis functions for the two data sets are plotted in Figure 1.



(a)



(b)

Fig. 1. The resultant B-spline (solid line) and NURB (dotted line) basis functions formed using PSO; (a) $\sigma^2 = 0.01$ and $\sigma^2 = 0.25$

The simulations of the pole assignment controller as shown in Figure 2, was experimented based on a given closed loop polynomial $T(q^{-1}) = 1 - 0.6q^{-1} + 0.1q^{-2}$. Under the assumption that the inverse of De Boor algorithm [14]) can cancel the nonlinearity in the system which is modeled by the identified NURB model, the required controller polynomials are estimated. The reference signals $r(t)$ are generated as a series of sinusoidal wave with its magnitude and frequency changing every 200 time steps. Figure 3(a) and (b) plot the resultant control signal and system response to the reference signal, respectively, when the output noise variance is set at 0.01. It can be concluded that the proposed method has excellent results in terms of system identification as well as the subsequent control for the identified systems.

6 Conclusions

This paper introduced a new system identification algorithm for the Hammerstein systems based on observational input/output data, using the non-uniform

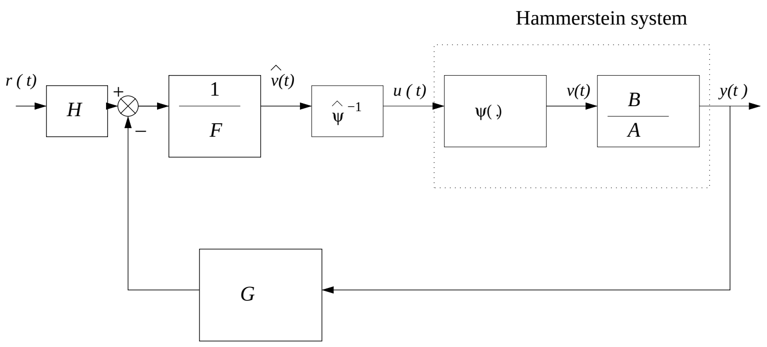


Fig. 2. The control of Hammerstein system using pole assignment and the inverse of De Boor algorithm.

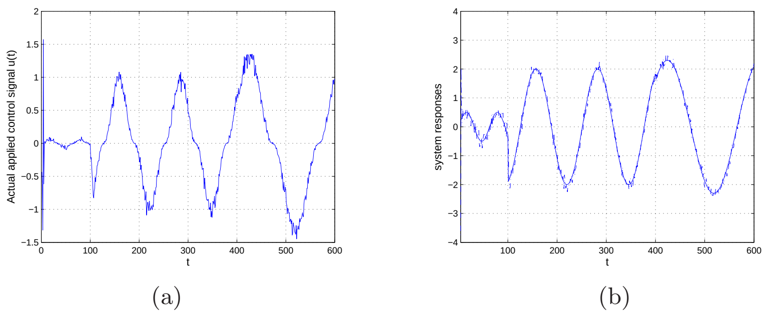


Fig. 3. The results of the pole assignment controller.

rational B-spline (NURB) neural network. We propose the PSO for the estimation of the shaping parameters in NURB neural networks. In simulation, a model based controller using inverse of the nonlinear static function approximated by NURB neural network together with a linear pole assignment controller are utilised to demonstrate the effectiveness of the proposed approaches.

References

1. Bai, E.W.: An optimal two-stage identification algorithm for Hammerstein-wiener nonlinear systems. Automatica 34, 333–338 (1998)
2. Bai, E.W., Fu, M.Y.: A blind approach to Hammerstein model identification. IEEE Transactions on Signal Processing 50(7), 1610–1619 (2002)
3. Billings, S.A., Fakhouri, S.Y.: Nonlinear system identification using the Hammerstein model. International Journal of Systems Science 10, 567–578 (1979)
4. Bloemen, H.H., van den Boom, T.J., Verbruggen, H.B.: Model based predictive control for Hammerstein systems. In: Proc. of the 39th IEEE Conference on Decision and Control. pp. 4963–4968. Sydney, Australia (2000)

5. de Boor: A Practical Guide to Splines. New York: Spring Verlag (1978)
6. Chaoui, F.Z., Giri, F., Rochdi, Y., Haloua, M., Naitali, A.: System identification based Hammerstein model. International Journal of Control 78(6), 430–442 (2005)
7. Chen, H.F.: Pathwise convergence of recursive identification algorithms for Hammerstein systems. IEEE Trans. on Automatic Control 49(10), 1873–1896 (2004)
8. Farin, G.: Curves and Surfaces for Comnputer-aided Geometric Design: a Practical Guide. Academic Press, Boston (1994)
9. Fruzzetti, E., Palazoglu, A., Mcdonald, K.A.: Nonlinear model predictive control using Hammerstein models. Journal of Process Cotnrol 7(1), 31–41 (1997)
10. Goethals, I., Pelckmans, K., Suykens, J.A.K., Moor, B.D.: Identification of MIMO Hammerstein models using least squares support vector machines. Automatica 41, 1263–1272 (2005)
11. Greblicki, W.: Stochastic approximation in nonparametric identification of Hammerstein systems. IEEE Transactions on Automatic Control 47(11), 1800–1810 (2002)
12. Greblicki, W., Pawlak, M.: Identification of discrete Hammerstein systems using kernel regression estimate. IEEE Transactions on Automatic Control AC-31(1), 74–77 (1986)
13. Guru, S.M., Halgamuge, S.K., Fernando, S.: Particle swarm optimisers for cluster formation in wireless sensor networks. In: Proc. 2005 Int. Conf. Intelligent Sensors, Sensor Networks and Information Processing. pp. 319–324. Melbourne, Australia (Dec 5-8, 2005)
14. Hong, X., Mitchell, R.J., Chen, S.: Modeling and control of Hammerstein system using B-spline approximation and the inverse of De Boor algorithm. International Journal of Systems Science p. In Press (2011)
15. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proc. of 1995 IEEE Int. Conf. Neural Networks. vol. 4, pp. 1942–1948. Perth, Australia (Nov 27-Dec 1, 1995)
16. Kennedy, J., Eberhart, R.C.: Swarm Intelligence. Morgan Kaufmann (2001)
17. Lang, Z.Q.: A nonparametric polynomial identification algorithm for the Hammerstein system. IEEE Transactions on Automatic Control 42, 1435–1441 (Oct 1997)
18. van der Merwe, D.W., Engelbrecht, A.P.: Data clustering using particle swarm optimization. In: Proc. CEC 2003. pp. 215–220. Cabberra, Australia (Dec 8-12, 2003)
19. Patwardhan, R.S., Lakshminarayanan, S., Shah, S.L.: Contrained nonlienar mc using Hammerstein and Wiener model: PLS framework. AIChE Journal 44(7), 1611–1622 (1998)
20. Ratnaweera, A., Halgamuge, S.K., Watson, H.C.: Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. IEEE Trans. Evolutionary Computation 8, 240–255 (June 2004)
21. Stoica, P., Söderström, T.: Instrumental variable methods for identification of Hammerstein systems. International Journal of Control 35, 459–476 (1982)
22. Verhaegen, M., Westwick, D.: Identifying mimo Hammerstein systems in the context of subspace model identification. International Journal of Control 63(2), 331–349 (1996)