B-spline neural networks based PID controller for Hammerstein systems

X. Hong[1], S. Iplikci[2] S. Chen[3], and K. Warwick[1]

[1] School of Systems Engineering, University of Reading, UK
[2] Pamukkale University, Department of Electrical and Electronics Engineering, Kinikli Campus, 20040, Denizli, Turkey
[3] School of Electronics and Computer Science, University of Southampton, UK, and Faculty of Engineering, King Abdulaziz University, Jeddah 21589, Saudi Arabia.

Abstract. A new PID tuning and controller approach is introduced for Hammerstein systems based on input/output data. A B-spline neural network is used to model the nonlinear static function in the Hammerstein system. The control signal is composed of a PID controller together with a correction term. In order to update the control signal, the multi-step ahead predictions of the Hammerstein system based on the B-spline neural networks and the associated Jacobians matrix are calculated using the De Boor algorithms including both the functional and derivative recursions. A numerical example is utilized to demonstrate the efficacy of the proposed approaches.

Keywords: B-spline neural network, Hammerstein model, PID controller, adaptive control, system identification.

1 Introduction

The proportional–integral–derivative (PID) controllers have been the most popular controller structures. Neural networks have been widely applied to model unknown dynamical processes and then used for PID parameter tuning [13,6,16]. Recently a novel predictive model-based PID tuning and control approach has been proposed for unknown nonlinear systems that are modelled using neural networks and support vector machines (SVMs) [11]. The work introduces a useful technique both for PID parameter tuning and for the correction of the PID output during control, which yields superior tracking and parameter convergence performance.

The Hammerstein model, comprising a nonlinear static functional transformation followed by a linear dynamical model, has been applied to nonlinear plant/process modelling in a wide range of biological/engineering problems [10,4,14,2]. Model based control for the Hammerstein system has been well studied [1,3,4]. The implementation of model based control for an *a priori* unknown Hammerstein model requires system identification including modelling and identification of the nonlinear static function. In [8], the closed loop system is linearized by inserting the inverse of the identified static nonlinearity, and the

nonlinear subsystems' inverse is calculated using the inverse of de Casteljau's algorithm.

Computationally efficient and numerically stable algorithms are in general desirable in nonlinear system identification and control. In this work a new PID controller is introduced for Hammerstein systems that are identified based on observational input/output data, in which the nonlinear static function in the Hammerstein system is modelled using a B-spline neural network. For system identification we used the Gauss-Newton algorithm subject to constraints as proposed in [9]. The predictive model-based PID tuning and controller approach in [11] was combined with the B-spline neural network based Hammerstein model. For this purpose, multistep ahead predictions of the B-spline neural networks based Hammerstein model are generated as well as the essential Jacobian matrix for updating the control signal, based on the De Boor recursion including both the functional and derivative recursions. The proposed model based on B-spline neural networks has a significant advantage over many other modeling paradigms in that this enables stable and efficient evaluations of functional and derivative values based on De Boor recursion, which is used for updating the PID control signals.

2 Modelling of the Hammerstein system based on B-spline functions

2.1 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

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

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.

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

Assumption 1: The persistence excitation condition is given by

$$\text{rank} \begin{pmatrix} u(n_a+n_b) & \cdots & u(n_a+1) & y(n_a+n_b) & \cdots & y(n_b+1) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u(N-1) & \cdots & u(N-n_b) & y(N-1) & \cdots & y(N-n_a) \end{pmatrix} = n_a + n_b \qquad (3)$$

Assumption 2: The gain of the linear subsystem is given by

$$G = \lim_{q \to 1} \frac{B(q^{-1})}{A(q^{-1})} = \frac{\sum_{j=1}^{n_b} b_j}{1 + \sum_{j=1}^{n_a} a_j} = 1 \tag{4}$$

Assumption 3: $\Psi(\bullet)$ is a one to one mapping, i.e. it is an invertible and continuous function.

Assumption 4: $u(t)$ is bounded by $U_{min} < u(t) < U_{max}$, where U_{min} and U_{max} are assumed known finite real values.

The objective of the work is controller design for the system based on observational data. 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. In this work B-spline basis functions are adopted in order to model $\Psi(\bullet)$. Specifically, the B-spline basis functions are initially formed by using the De-Boor algorithm [5] for the input data sets.

2.2 Modelling of $\Psi(\bullet)$ using B-spline function approximation

Univariate B-spline basis functions are parameterized using 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

$$U_1 < U_2 < U_k = U_{min} < U_{k+1} < U_{k+2} < \cdots < U_d < U_{max} = U_{d+1} < \cdots < U_{d+k}. \tag{5}$$

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

$$\mathcal{B}_j^{(0)}(u) = \begin{cases} 1 \text{ if } U_j \le u < U_{j+1} \\ 0 \quad \text{otherwise} \end{cases} \tag{6}$$

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

$$\mathcal{B}_j^{(i)}(u) = \frac{u - U_j}{U_{i+j} - U_j} \mathcal{B}_i^{(i-1)}(u) + \frac{U_{i+j+1} - u}{U_{i+j+1} - U_{j+1}} \mathcal{B}_{j+1}^{(i-1)}(u), \quad \left.\begin{array}{l} \\ j = 1, \cdots, (d + k - i) \end{array}\right\} i = 1, \cdots, k \tag{7}$$

The first order derivatives of the B-spline function have a similar recursion

$$\frac{d}{du}\mathcal{B}_j^{(k)}(u) = \frac{k}{U_{k+j} - U_j} \mathcal{B}_j^{(k-1)}(u) - \frac{k}{U_{k+j+1} - U_{j+1}} \mathcal{B}_{j+1}^{(k-1)}(u), \; j = 1, \cdots, d \tag{8}$$

We model $\Psi(\bullet)$ as a B-spline neural network [7], in the form of

$$\Psi(u) = \sum_{j=1}^{d} \mathcal{B}_j^{(k)}(u) \omega_j \tag{9}$$

where ω_j's are weights to be determined. Denote $\boldsymbol{\omega} = [\omega_1, \cdots, \omega_d]^T \in \Re^d$. 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 $O(k^2)$. The evaluation of the first order derivatives can be regarded as a byproduct, with the additional computational cost in the order of $O(k)$.

2.3 The system identification algorithm

With the B-spline approximation, the model predicted output $\hat{y}(t)$ in (1) can be written as

$$\begin{aligned}\hat{y}(t) = &-a_1 y(t-1) - a_2 y(t-2) - ... - a_{n_a} y(t-n_a) + \\ &b_1 \sum_{j=1}^{d} \omega_j \mathcal{B}_j^{(k)}(t-1) + ... + b_{n_b} \sum_{j=1}^{d} \omega_j \mathcal{B}_j^{(k)}(t-n_b).\end{aligned} \tag{10}$$

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

$$y(t) = [\mathbf{p}(\mathbf{x}(t))]^T \boldsymbol{\vartheta} + \varepsilon(t) \tag{11}$$

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 $(n_a + n_b)$, $\boldsymbol{\vartheta} = [\mathbf{a}^T, (b_1\omega_1), ..., (b_1\omega_d), ...(b_{n_b}\omega_1), ..., (b_{n_b}\omega_{n_b})]^T \in \Re^{n_a + d \cdot n_b}$,

$$\begin{aligned}\mathbf{p}(\mathbf{x}(t)) = &[-y(t-1), ..., -y(t-n_a), \mathcal{B}_1^{(k)}(t-1), ... \\ &..., \mathcal{B}_d^{(k)}(t-1), ...\mathcal{B}_1^{(k)}(t-n_b), ..., \mathcal{B}_d^{(k)}(t-n_b)]^T\end{aligned} \tag{12}$$

(11) can be rewritten in the matrix form as

$$\mathbf{y} = \mathbf{P}\boldsymbol{\vartheta} + \boldsymbol{\varepsilon} \tag{13}$$

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

$$\mathbf{P} = \begin{bmatrix} p_1(\mathbf{x}(1)) & p_2(\mathbf{x}(1)) & \cdots & p_{n_a+d\cdot n_b}(\mathbf{x}(1)) \\ p_1(\mathbf{x}(2)) & p_2(\mathbf{x}(2)) & \cdots & p_{n_a+d\cdot n_b}(\mathbf{x}(2)) \\ \cdots & \cdots & \cdots & \cdots \\ p_1(\mathbf{x}(N)) & p_2(\mathbf{x}(N)) & \cdots & p_{n_a+d\cdot n_b}(\mathbf{x}(N)) \end{bmatrix} \tag{14}$$

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

$$\boldsymbol{\vartheta}_{LS} = \mathbf{B}^{-1}\mathbf{P}^T\mathbf{y} \tag{15}$$

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 singular value

decomposition (SVD) $\mathbf{BQ} = \mathbf{Q\Sigma}$, where $\mathbf{\Sigma} = \text{diag}[\sigma_1, ...\sigma_r, 0, \cdots, 0]$. $\mathbf{Q} = [\mathbf{q}_1, \cdots, \mathbf{q}_1, \cdots, \mathbf{q}_{n_a+d\cdot n_b}]$, followed by truncating the eigenvectors corresponding to zero eigenvalues, we have

$$\boldsymbol{\vartheta}_{LS}^{svd} = \sum_{i=1}^{r} \frac{\mathbf{y}^T \mathbf{P} \mathbf{q}_i}{\sigma_i} \mathbf{q}_i \tag{16}$$

This procedure produces our final estimate of $\hat{\mathbf{a}}$, which is simply taken as the subvector of the resultant $\boldsymbol{\vartheta}_{LS}^{svd}$, consisting of its first n_a elements. The parameter estimation for $\mathbf{b}$ and $\boldsymbol{\omega}$ can be obtained using our previous work [9].

3 The model based PID controller

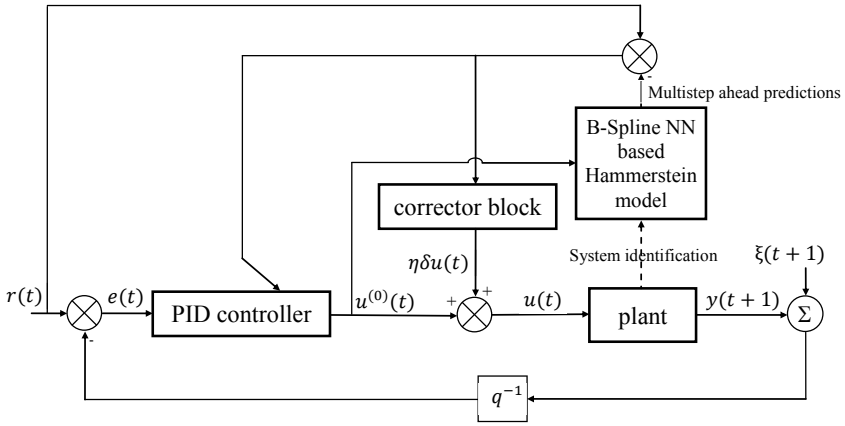


Fig. 1. Diagram of the model based PID controller

Figure 1 illustrates the proposed model based PID controller for Hammerstein systems using B-spline neural networks where $r(t)$ is the desired reference trajectory to be followed by the plant output $y(t)$, and $e(t)$ is the error between the desired and measured output at time index t. Both the PID controller parameters and the control signal are derived using the concept of predictive control, explained as follows. At each sampling time, consider that the control signal $u(t)$ is repeatedly applied to the plant exactly for consecutive K time steps and the resultant predictive output trajectory vector is denoted as $[\hat{y}(t+1|t), \hat{y}(t+2|t), \ldots, \hat{y}(t+K|t)]$. The optimal $u(t)$ is then derived such that

the sum of the squared K-step ahead prediction errors are minimized with minimum deviation in the control action. In other words, it is obtained by minimizing the objective function J given by

$$J(u(t)) = \frac{1}{2}\sum_{\kappa=1}^{K}[e(t+\kappa|t)]^2 + \frac{1}{2}\lambda\left(u(t) - u(t-1)\right)^2, \tag{17}$$

where $e(t+\kappa|t) = r(t+\kappa) - \hat{y}(t+\kappa|t)$ is the κ-step ahead prediction error, $\kappa = 1, \cdots, K$, K is the predetermined prediction horizon and $\lambda > 0$ is a predetermined penalty term. The control signal $u(t)$ is designed to be composed of the PID output $u^{(0)}(t)$ plus a correction term $\eta\delta u(t)$, with η as an optimum step-length, given by

$$u(t) = u^{(0)}(t) + \eta\delta u(t), \tag{18}$$

which is obtained by a two step procedure; (i) the PID controller parameters are initially optimized based on minimizing (17) without the correction term ($\eta = 0$), followed by (ii) obtaining $\eta\delta u(t)$ minimizing (17) subject to the PID controller as derived in (i).

3.1 PID controller parameter optimization using predictive control

Initially consider that the output of the PID controller $u^{(0)}(t)$ in response to error $e(t)$ according to the formula given below:

$$u^{(0)}(t) = u(t-1)+K_P[e(t)-e(t-1)]+K_I e(t)+K_D[e(t)-2e(t-1)+e(t-2)], \tag{19}$$

is applied to the Hammerstein system, where K_P, K_I and K_D are the PID parameters to be optimized based on the objective function $J\left(u^{(0)}(t)\right)$. At the beginning of the control sequence, the PID parameters are set to zero. In the proposed scheme, we adopt the Levenberg-Marquardt (LM) rule (20) as the minimization algorithm, such that the PID parameters are updated at every time step according to

$$\begin{bmatrix} K_P^{new} \\ K_I^{new} \\ K_D^{new} \end{bmatrix} = \begin{bmatrix} K_P^{old} \\ K_I^{old} \\ K_D^{old} \end{bmatrix} - \alpha(\mathbf{J}^T\mathbf{J} + \mu\mathbf{I})^{-1}\mathbf{J}^T\hat{\mathbf{e}}, \tag{20}$$

where α is a small predetermined positive number. $\mu > 0$ is a parameter that yields a compromise between the steepest descent and the Gauss-Newton algorithms, $\mathbf{I}$ is the 3×3 identity matrix, $\mathbf{J}$ is the $(K+1) \times 3$ Jacobian matrix given by,

$$\mathbf{J} = -\begin{bmatrix} \frac{\partial\hat{y}(t+1|t)}{\partial K_P} & \frac{\partial\hat{y}(t+1|t)}{\partial K_I} & \frac{\partial\hat{y}(t+1|t)}{\partial K_D} \\ \frac{\partial\hat{y}(t+2|t)}{\partial K_P} & \frac{\partial\hat{y}(t+2|t)}{\partial K_I} & \frac{\partial\hat{y}(t+2|t)}{\partial K_D} \\ \vdots & \vdots & \vdots \\ \frac{\partial\hat{y}(t+K|t)}{\partial K_P} & \frac{\partial\hat{y}(t+K|t)}{\partial K_I} & \frac{\partial\hat{y}(t+K|t)}{\partial K_D} \\ \sqrt{\lambda}\frac{\partial u(t)}{\partial K_P} & \sqrt{\lambda}\frac{\partial u(t)}{\partial K_I} & \sqrt{\lambda}\frac{\partial u(t)}{\partial K_D} \end{bmatrix}_{|u(t)=u^{(0)}(t)}, \tag{21}$$

and $\hat{\mathbf{e}}$ is the vector of prediction errors and input slew given by,

$$\hat{\mathbf{e}} = \begin{bmatrix} e(t+1|t) \\ \vdots \\ e(t+K|t) \\ \sqrt{\lambda}(u(t)-u(t-1)) \end{bmatrix}_{|u(t)=u^{(0)}(t)} = \begin{bmatrix} r(t+1|t)-\hat{y}(t+1|t) \\ \vdots \\ r(t+K|t)-\hat{y}(t+\kappa|t) \\ \sqrt{\lambda}(u(t)-u(t-1)) \end{bmatrix}_{|u(t)=u^{(0)}(t)} . \tag{22}$$

It can be seen that the Jacobian matrix (21) can be decomposed as the product of two different matrices by using the chain rule as follows:

$$\begin{aligned} \mathbf{J} &= \left(- \begin{bmatrix} \frac{\partial \hat{y}(t+1|t)}{\partial u(t)} \\ \frac{\partial \hat{y}(t+2|t)}{\partial u(t)} \\ \vdots \\ \frac{\partial \hat{y}(t+K|t)}{\partial u(t)} \\ \sqrt{\lambda} \end{bmatrix} \times \begin{bmatrix} \frac{\partial u(t)}{\partial K_P} & \frac{\partial u(t)}{\partial K_I} & \frac{\partial u(t)}{\partial K_D} \end{bmatrix} \right)_{|u(t)=u^{(0)}(t)} \\ &= \mathbf{J}_m \mathbf{J}_c, \end{aligned} \tag{23}$$

where

$$\mathbf{J}_m = - \begin{bmatrix} \frac{\partial \hat{y}(t+1|t)}{\partial u(t)} & \frac{\partial \hat{y}(t+2|t)}{\partial u(t)} & \dots & \frac{\partial \hat{y}(t+K|t)}{\partial u(t)} & \sqrt{\lambda} \end{bmatrix}^T_{|u(t)=u^{(0)}(t)} , \tag{24}$$

and $\mathbf{J}_c$ is a matrix of partial derivatives of $u^{(0)}(t)$ with respect to the PID parameters and can be written by using only the tracking errors as,

$$\mathbf{J}_c = \begin{bmatrix} e(t)-e(t-1) \\ e(t) \\ e(t)-2e(t-1)+e(t-2) \end{bmatrix}^T \tag{25}$$

However, mostly in the transient-state and to some extent in the steady-state, the obtained PID parameters may not be good enough to produce an acceptable control action, which leadd to the necessity of a correction term $\eta\delta u(t)$ to be added to the control action.

3.2 The optimal control signal using corrector block

The aim of the corrector block is to produce a suboptimal correction term $\delta u(t)$ used in (18) by minimizing the objective function $J(u(t))$ given by (17). More specifically, the corrector block tries to minimize the objective function J with respect to $\delta u(t)$ based on the second-order Taylor approximation of the objective function J as follows:

$$\begin{aligned} J(u(t)) &= J\Big(u^{(0)}(t)+\delta u(t)\Big) \\ &\approx J(u^{(0)}(t)) + \frac{\partial J(u(t))}{\partial u(t)}_{|u(t)=u^{(0)}(t)} \delta u(t) \\ &\quad + \frac{1}{2}\frac{\partial^2 J(u(t))}{\partial u(t)^2}_{|u(t)=u^{(0)}(t)} (\delta u(t))^2. \end{aligned} \tag{26}$$

Since we wish to find the $\delta u(t)$ that minimizes the objective function, if we take the derivative of the approximate J with respect to $\delta u(t)$ and equate it to zero, we obtain,

$$\frac{\partial J(u(t))}{\partial u(t)}_{|u(t)=u^{(0)}(t)} + \frac{\partial^2 J(u(t))}{\partial u(t)^2}_{|u(t)=u^{(0)}(t)} \delta u(t) = 0 \tag{27}$$

so

$$\delta u(t) = -\frac{\frac{\partial J}{\partial u(t)}|u(t)=u^{(0)}(t)}{\frac{\partial^2 J}{\partial u(t)^2}|u(t)=u^{(0)}(t)}, \tag{28}$$

which corresponds to the Newton direction that provides a quadratic convergence to the local minimum if the scalar second-order term (Hessian) in the Taylor expansion is positive and the higher-order terms are negligible [12]. In order to avoid calculating the time-consuming second-order derivatives, we can employ the well-known Jacobian approximation that suggests that the $(K+1)\times 1$ Jacobian matrix $\mathbf{J}_m$ can represent the gradient vector exactly and the Hessian matrix approximately as,

$$\frac{\partial J(u(t))}{\partial u(t)}_{|u(t)=u^{(0)}(t)} = 2\mathbf{J}_m^T\hat{\mathbf{e}} \quad \text{and} \quad \frac{\partial^2 J(u(t))}{\partial u(t)^2}_{|u(t)=u^{(0)}(t)} \approx 2\mathbf{J}_m^T\mathbf{J}_m. \tag{29}$$

The correction term is computed by

$$\delta u(t) = -\mathbf{J}_m^T\hat{\mathbf{e}}/\mathbf{J}_m^T\mathbf{J}_m. \tag{30}$$

Finally, once $\delta u(t)$ is determined, a line search is used to search for the optimum step-length η to further minimize the objective function. This is a typical one-dimensional optimization problem and can be solved by the golden section algorithm [15]. This algorithm directly evaluates $J\left(u^{(n)}(t)\right)$ for a sequence of control signals $u^{(n)}(t)$, $n = 1, 2, \cdots$, until this converges to the optimal $u(t)$ which is associated with the optimum step-length η.

Note that the PID controller parameter updating formula (20) requires the calculation of multistep ahead predictions and the Jacobian $\mathbf{J}_m$. Moreover the subsequent control signal correction term given by (18)&(30) via the golden section algorithm not only requires the Jacobian $\mathbf{J}_m$, but also the *iterative* calculation of multistep ahead predictions for the objective functional evaluations. Note also that the calculation of multistep ahead predictions and the Jacobian $\mathbf{J}_m$ are model specific. In [11], this controller scheme has been applied to unknown nonlinear systems that are modelled using neural networks and support vector machines (SVMs) respectively. In the following, the calculation of multistep ahead predictions and Jacobian $\mathbf{J}_m$ for the B-spline neural network based Hammerstein model are introduced.

3.3 The calculation of multistep ahead predictions and Jacobian $\mathbf{J}_m$

If a control signal $u(t)$ is repeatedly applied to the plant exactly K time steps, then the κ-step ahead predictions ($\kappa = 1, \cdots, K$) using the B-spline neural

network based Hammerstein model are given by

$$\begin{aligned}\hat{y}(t+\kappa|t) = &-a_1\hat{y}(t+\kappa-1|t) - a_2\hat{y}(t+\kappa-2|t) - ... - a_{n_a}\hat{y}(t+\kappa-n_a|t) \\ &+b_1 v(t+k-1) + ... + b_{n_b} v(t+k-n_b)\end{aligned} \tag{31}$$

in which each term in the right hand side of (31) is computed by

$$\hat{y}(t+\kappa-i|t) = \begin{cases} \hat{y}(t+\kappa-i|t) \text{ if } & (\kappa-i) > 0 \\ y(t+\kappa-i) & \text{otherwise} \end{cases}$$
$$i = 1, \cdots, n_a, \quad \kappa = 1, \cdots, K \tag{32}$$

and

$$v(t+\kappa-i) = \begin{cases} \sum_{j=1}^{d} \mathcal{B}_j^{(k)}(u(t))\omega_j & \text{if } (\kappa-i) \geq 0 \\ \sum_{j=1}^{d} \mathcal{B}_j^{(k)}(u(t+k-i))\omega_j & \text{otherwise} \end{cases}$$
$$i = 1, \cdots, n_b, \quad \kappa = 1, \cdots, K \tag{33}$$

Similarly the elements in $\mathbf{J}_m$, $\frac{\partial \hat{y}(t+\kappa|t)}{\partial u(t)}$, $\kappa = 1, \cdots, K$ are also computed recursively from

$$\begin{aligned}\frac{\partial \hat{y}(t+\kappa|t)}{\partial u(t)} = &-a_1\frac{\partial \hat{y}(t+\kappa-1|t)}{\partial u(t)} - a_2\frac{\partial \hat{y}(t+\kappa-2|t)}{\partial u(t)} - ... - a_{n_a}\frac{\partial \hat{y}(t+\kappa-n_a|t)}{\partial u(t)} \\ &+b_1\frac{\partial v(t+\kappa-1|t)}{\partial u(t)} + ... + b_{n_b}\frac{\partial v(t+\kappa-n_b|t)}{\partial u(t)}\end{aligned} \tag{34}$$

in which each term in the right hand side of (34) is computed by

$$\frac{\partial \hat{y}(t+\kappa-1|t)}{\partial u(t)} = \begin{cases} \frac{\partial \hat{y}(t+\kappa-1|t)}{\partial u(t)} \text{ if } & (\kappa-i) > 0 \\ 0 & \text{otherwise} \end{cases}$$
$$i = 1, \cdots, n_a, \quad \kappa = 1, \cdots, K \tag{35}$$

and

$$\frac{\partial v(t+\kappa-i|t)}{\partial u(t)} = \begin{cases} \sum_{j=1}^{d} \omega_j \frac{d}{du(t)}\mathcal{B}_j^{(k)}(u(t)) \text{ if } & (\kappa-i) \geq 0 \\ 0 & \text{otherwise} \end{cases}$$
$$i = 1, \cdots, n_b, \quad \kappa = 1, \cdots, K \tag{36}$$

Note that in calculating (31)-(37), the De Boor algorithm (6)-(8) is applied in evaluating the associated entries. In particular we point out the term $\frac{d}{du(t)}\mathcal{B}_j^{(k)}(u(t))$ in (37) is evaluated using (8) and gives exact derivative values at minimum extra computational cost, and this is an advantage specific to our proposed Hammerstein model using B-spline neural network with De Boor recursion. Specifically at each time step the proposed algorithm requires (31) to be evaluated $(n_{max}+1)$ times, based on $u^{(n)}(t)$, $n = 1, \cdots, n_{max}$, where n_{max} is the maximum number of iterations set in the golden section algorithm. Equation (34) is however

only evaluated once for the calculation of $\mathbf{J}_m$. Hence the two main parts of the computational cost are firstly due to the PID parameter updates in the order of $O(9\times(K+1)+3^3)$, and secondly due to the iterative multistep ahead predictions mainly in the golden section algorithm, and this is of order $O(k^2+(n_a+k\cdot n_b)K)$, which is further scaled by n_{max}. Effectively the proposed algorithm enables stable and efficient evaluations of the multistep ahead predictions and functional derivatives to be possible, which could be problematic for many other nonlinear representations including some spline functions based nonlinear models.

The optimal values for λ and K are mostly problem-dependent and thus there is no general analytical way of finding them. Still, it will be helpful to take some general facts given below into consideration while attempting to find their proper values by trial-and-error.

4 An illustrative example

An illustrative Hammerstein system is simulated, in which the linear subsystem is given by $A(q^{-1}) = 1-1.2q^{-1}+0.9q^{-2}$, $B(q^{-1}) = 1.7q^{-1}-q^{-2}$. The nonlinear subsystem, $\Psi(\bullet)$ is given by

$$\Psi(u) = -2\text{sign}(u)u^2 \tag{37}$$

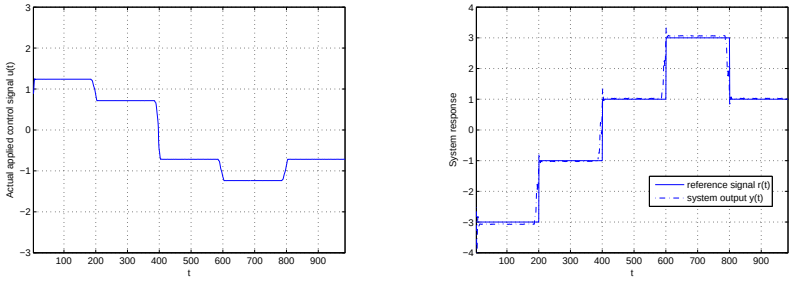
The variances of the additive noise to the system output are set as $\sigma^2 = 0.0001$. 1000 training data samples $y(t)$ were generated by using (1) and (2), where $u(t)$ was a uniformly distributed random variable $u(t) \in [-1.5, 1.5]$. The polynomial degree of B-spline basis functions was set as $k = 2$ (piecewise quadratic). The knots sequence U_j was set as

$$[-3,\ -2.5,\ -2,\ -1,\ -0.3,\ 0,\ 0.3,\ 1,\ 2,\ 2.5,\ 3].$$

Initially system identification was carried out. The parameters were empirically set at $\alpha = 0.2$, $\mu = 10^{-3}$, $K = 15$, $\lambda = 10$ for illustration only because it was found that the proposed approach is robust for a wide range of parameters. The reference signal $r(t)$ was generated as a series of square waves resembling a staircase. Figure 2(a) plot the applied computed control signal. Figure 2(b) plot the system output $y(t)$ together with the corresponding reference signal $r(t)$ with a small noise ($\sigma^2 = 9\times 10^{-6}$). It is shown that the proposed method exhibits excellent result.

5 Conclusions

This paper has introduced a new effective PID control method for Hammerstein systems based on observational input/output data. Modeling of the nonlinear static function in the Hammerstein system is based on B-spline function approximation. By minimizing the multistep ahead prediction errors the PID controller parameters are updated and then corrected to generate the control signal. The



(a) (b)

Fig. 2. Results of the proposed PID controller.

multistep ahead predictions of the B-spline neural networks based Hammerstein system and the associated Jacobians matrix are very efficiently computed based on the De Boor algorithms including both the functional and derivative recursions. The efficacy of the proposed approach has been demonstrated via an illustrative example.

Acknowledgement

The authors would like to thank the financial support from UK EPSRC and the Council of Higher Education in Turkey.

References

1. Anbumani, K., Patnaik, L.M., Sarma, I.G.: Self-tuning minimum variance control of nonlinear systems of the Hammerstein model. IEEE Transactions on Automatic Control AC-26(4), 959–961 (1981)
2. Balestrino, A., Landi, A., Ould-Zmirli, M., Sani, L.: Automatic nonlinea aut-tuning method for Hammerstein modelling of electrical drives. IEEE Transactions on Industrial Electronics 48(3), 645–655 (2001)
3. 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)
4. Bloemen, H.H.J., Boom, T.J.V.D., Verbruggen, H.B.: Model-based predictive control for Hammerstein-Wiener systems. International Journal of Control 74(5), 482–295 (2001)
5. de Boor: A Practical Guide to Splines. New York: Spring Verlag (1978)
6. Chen, J., Huang, T.: Applying neural networks to on-line updated PID controllers for nonlnear process control. Journal of Process Control 14, 211–230 (2004)
7. Harris, C.J., Hong, X., Gan, Q.: Adaptive Modelling, Estimation and Fusion from Data: A Neurofuzzy Approach. Springer-Verlag (2002)

8. Hong, X., Mitchell, R.J.: A pole assignment controller for bezier-bernstein polynomial based hammerstein model. In: Proceedings of International Control Conference (ICC) 2006. Glascow, UK (2006)
9. Hong, X., Mitchell, R.J.: A Hammerstein model identification algorithm using bezier-bernstein approximation. IET Proc Control Theory and Applications 1(4), 1149–1159 (2007)
10. Hunter, I.W., Korenberg, M.J.: The identification of nonlinear biological systems: Wiener and Hammerstein cascade models. Biological Cybernetics 55(2-3), 135–144 (1986)
11. Iplikci, S.: A comparative study on a novel model-based PID tuning and control mechanism for nonlinear systems. International Journal of Robust and Nonlinear Control 20(13), 1483–1501 (2010)
12. Nocedal, J., Wright, S.J.: Numerical Optimization. Springer, New York (1999)
13. Parlos, A.G., Parthasarathy, S., Atiya, A.F.: Neuro-predicitive process control using on-line controller adaptation. IEEE Transactions on Control Systems Technology 9, 741–755 (2001)
14. Turunen, J., Tanttu, J.T., Loula, P.: Hammerstein model for speech coding. EURASIP Journal of Applied Signal Processing 12, 1238–1249 (2003)
15. Venkataraman, P.: Applied Optimization with MATLAB Programming. Wiley-Interscience, New York (2002)
16. Zhang, M., Li, W., Liu, M.: Adaptive PID control strategy based on RBF neural network identification. In: Proceedings of the ICNNB International Conference on Neural Networks and Brain. pp. 1854–1857. Beijing, China (2005)