INTERNATIONAL JOURNAL OF ADAPTIVE CONTROL AND SIGNAL PROCESSING
Int. J. Adapt. Control Signal Process. 2014; **28**:412–428
Published online 25 April 2012 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/acs.2293

A model-based PID controller for Hammerstein systems using B-spline neural networks

X. Hong [1,*,†], S. Iplikci [2], S. Chen [3,4] and K. Warwick [1]

[1]*School of Systems Engineering, University of Reading, Reading, RG6 6AY, 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, Southampton SO17 1BJ, UK*
[4]*Faculty of Engineering, King Abdulaziz University, Jeddah 21589, Saudi Arabia*

SUMMARY

In this paper, a new model-based proportional–integral–derivative (PID) tuning and controller approach is introduced for Hammerstein systems that are identified on the basis of the observational input/output data. The nonlinear static function in the Hammerstein system is modelled using a B-spline neural network. The control signal is composed of a PID controller, together with a correction term. Both the parameters in the PID controller and the correction term are optimized on the basis of minimizing the multistep ahead prediction errors. In order to update the control signal, the multistep ahead predictions of the Hammerstein system based on B-spline neural networks and the associated Jacobian matrix are calculated using the de Boor algorithms, including both the functional and derivative recursions. Numerical examples are utilized to demonstrate the efficacy of the proposed approaches. Copyright © 2012 John Wiley & Sons, Ltd.

Received 30 August 2011; Revised 8 December 2011; Accepted 19 March 2012

KEY WORDS: B-spline neural network; de Boor algorithm; Hammerstein model; PID controller; adaptive control; multistep ahead prediction; system identification

1. INTRODUCTION

Because of their simplicity and robust performance, proportional–integral–derivative (PID) controllers have been the most popular controller structures. A major issue associated with a PID controller design however is that its parameters need to be tuned properly. There exist many tuning and adaptation schemes for linear plants in the literature [1–3]. However, it is a challenging task to obtain good PID controller parameters if the plant to be controlled exhibits nonlinear behaviour. In order to tune and adapt the PID parameters for controlling a nonlinear unknown plant, it is necessary to obtain a nonlinear or a linearized model of the plant. For systems that can be described by the controlled autoregressive integrated moving average (CARIMA) models, PID parameters can be tuned by using model-predictive control methods based on the CARIMA model of the plant [4, 5].

Neural networks have been widely applied to model unknown dynamical processes and then used for PID parameter tuning. In [6], the parameters of a PID controller are tuned online by minimizing the multistep ahead prediction errors on the basis of a dynamic model of the plant obtained using a recurrent neural network. The predicted error gradients are back-propagated through the network in order to find the modifications in the controller parameters. In [7], a neural network model of the process is obtained and then is linearized around some instantaneous operating points, similar to the gain scheduling approach. On the basis of a linearized model corresponding to the operating point,

*Correspondence to: X. Hong, School of Systems Engineering, University of Reading, Reading, RG6 6AY, UK.
†E-mail: x.hong@reading.ac.uk

the PID parameters are then tuned online by a generalized minimum variance approach. In [8], a radial basis function (RBF) neural network model is used to tune PID parameters for controlling a nonlinear time-varying system. In a combined method [9], a neural network model of the plant is initially obtained, followed by tuning the PID parameters offline based on the neural network using the genetic algorithm optimization. The support vector machine (SVM) has also been used in the tuning and adaptation of PID parameters for nonlinear systems. Similar to [7], a model of a nonlinear system is obtained by an SVM and then is linearized to extract the instantaneous linear model for a self-tuning PID controller [10]. In [11], a least squares SVM model is used to tune PID parameters to control a nonlinear time-varying system. 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 SVMs [12]. 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 a nonlinear plant/process modelling in a wide range of biological/engineering problems [13–16]. For example, it is a suitable model for signal processing applications involving any nonlinear distortion followed by a linear filter, as in modeling of the human heart in order to regulate the heart rate during treadmill exercises [17] and in the modeling of hydraulic actuator friction dynamics [18]. The Hammerstein model in itself has also been widely researched [19–28].

The model characterization/representation of the unknown nonlinear static function is fundamental to the identification of a good Hammerstein model. Various approaches have been developed in order to capture the a priori unknown nonlinearity by the use of both parametric [27,28] and nonparametric methods [22,25,26]. It has been shown that the Bernstein basis is the best conditioned and the most stable among any other polynomial basis [29]. The inverse of de Casteljau's algorithm was introduced to identify the Bezier–Bernstein neural network by busing the Bernstein approximation and from observational data [30]. Recently, a new identification algorithm [31] for the Hammerstein model has been introduced on the basis of the Bezier–Bernstein approximation and the inverse of de Casteljau's algorithm. Alternatively, the special structure of Hammerstein models can be exploited to develop hybrid parameter estimation algorithms [21,28,32]. Similar to the Bezier curve, the B-spline curve has also been widely used in computer graphics and computer-aided geometric design [33]. B-spline curves consist of many polynomial pieces, offering much more versatility than Bezier curves while maintaining the same advantage of the best conditioning property. Early work on the construction of the B-spline curve was mathematically involved and found to be numerically unstable [34]. However, the de Boor algorithm uses recurrence relations and is numerically stable [34]. The B-spline basis functions for nonlinear system modelling have since been widely applied [35–37].

Model-based control for the Hammerstein system has been well studied [14, 38, 39]. A popular treatment for handling the Hammerstein model is to remove the nonlinearity via an inversion [39–41]. This enables the celebrated self-tuning control methods to be readily applicable [42]. 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. Different nonlinear model representations result in variations in the controller design algorithms. For example, in [38,43], the nonlinear static function is based on an explicit polynomial function of the input. The optimal control law satisfies a polynomial equation of the input, which is then found via root solving. In [44], 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 [45], the control of a Hammerstein system, as characterized by the B-spline neural network, is investigated through the removal of the nonlinearity via the inversion of a B-spine neural network and the application of a pole assignment controller. The work introduced a new algorithm referred to as the inverse of the de Boor algorithm which computes the inverse efficiently. Despite the advantages of linearization methods for controlling Hammerstein systems, its performance is heavily dependent upon the assumption that the nonlinearity is removed via the inserted functional inversion. However, this assumption may not be appropriate in some circumstances, leading to the deterioration in controller performance because of modeling/tracking errors especially during the transient phase of systems.

In this work, a new PID controller is introduced for Hammerstein systems that are identified on the basis of observational input/output data, in which the nonlinear static function in the Hammerstein system is modelled using a B-spline neural network. Note that, for the system identification of the resultant model representation, Bai's overparameterization approach is directly applicable [21]. In this paper, we used the Gauss–Newton algorithm subject to constraints, as proposed in [31]. The predictive model-based PID tuning and controller approach in [12] was combined with the B-spline neural network-based Hammerstein model. For this purpose, multistep ahead predictions of the B-spline neural network-based Hammerstein model are generated as well as the essential Jacobian matrix for updating the control signal, on the basis of the de Boor recursion including both the functional and derivative recursions. The motivation of the proposed methods is twofold. First, this extends the model-based PID controller [12] to accommodate the Hammerstein systems. Second, the proposed model based on the 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 on the basis of the de Boor recursion, which is used in updating the PID control signals.

This paper is organized as follows. Section 2 formulates the modelling of the Hammerstein system on the basis of the B-spline functions and describes the system identification algorithm for the system. Section 3 introduces the proposed model-based PID controller on the basis of the multistep ahead predictive control. This includes the PID controller parameter optimization, the corrector block for optimal control signal and the associated multistep ahead predictions and Jacobian calculations. In Section 4, we have presented the simulation results, and some conclusions are given in Section 5.

2. MODELLING OF THE HAMMERSTEIN SYSTEM BASED ON B-SPLINE FUNCTIONS

2.1. The Hammerstein system

The Hammerstein system, as shown in Figure 1, 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

$$\begin{aligned} y(t) = &-a_1 y(t-1) - a_2 y(t-2) - \ldots - a_{n_a} y(t-n_a) \\ &+ b_1 v(t-1) + \ldots + b_{n_b} v(t-n_b) + \xi(t) \end{aligned} \tag{1}$$

$$v(t-j) = \Psi(u(t-j)), \quad j = 1, \ldots, n_b \tag{2}$$

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 the nonlinear subsystem and the input to the linear subsystem. a_js and b_js are parameters of the linear subsystem. n_a and n_b are assumed known system output and input lags. Denote $\mathbf{a} = [a_1, \ldots, a_{n_a}]^T \in \Re^{n_a}$ and $\mathbf{b} = [b_1, \ldots, b_{n_b}]^T \in \Re^{n_b}$. It is assumed that $A(q^{-1}) = 1 + a_1 q^{-1} + \ldots + a_{n_a} q^{-n_a}$ and $B(q^{-1}) = b_1 q^{-1} + \ldots + b_{n_b} q^{-n_b}$ are coprime polynomials of q^{-1}, where q^{-1} denotes the backward shift operator.

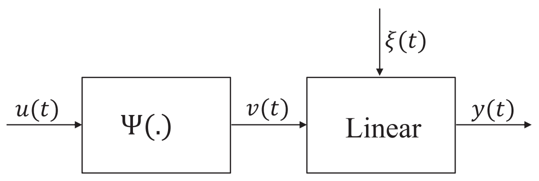


Figure 1. The Hammerstein system.

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

Assumption 1
The persistence excitation condition is given by

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

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; that is, 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.

Remark 1

- Assumption 1 is necessary irrespective of the model representation and identification algorithm. Persistent excitation is the essential condition for the input signal for the sake of identifiability.
- Because the signals between the two subsystems are unavailable, Assumption 2 is introduced for identifiability. Otherwise, any pair of $\{\rho[b_1,\ldots,b_{n_b}]^T,\ \Psi(\bullet)/\rho\}$, for $\rho \neq 0$, provides the identical input/output measurements.

The two objectives of the work are that of system identification and subsequent controller design for the identified model. The objective of system identification for the mentioned Hammerstein model is, 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 [34] 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 [34], given by

$$\mathcal{B}_j^{(0)}(u) = \begin{cases} 1 & \text{if } U_j \leqslant u < U_{j+1} \\ 0 & \text{otherwise} \end{cases} \\ j = 1, \cdots, (d+k) \tag{6}$$

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

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

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

We model $\Psi(\bullet)$ as a B-spline neural network [37] in the form of

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

where ω_js are weights to be determined. Denote $\boldsymbol{\omega} = [\omega_1, \cdots, \omega_d]^T \in \Re^d$. Note that because of 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 $\Psi(u)$, on the basis of 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

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

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

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

Equation (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. $\boldsymbol{\varepsilon} = [\varepsilon(1), \ldots, \varepsilon(N)]^T$, 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, that is $Rank(\mathbf{B}) = r < n_a + d \cdot n_b$, then performing the singular value decomposition, $\mathbf{BQ} = \mathbf{Q\Sigma}$, where $\mathbf{\Sigma} = \text{diag}[\sigma_1, \ldots \sigma_r, 0, \cdots, 0]$. $\mathbf{Q} = [\mathbf{q}_1, \cdots, \mathbf{q}_1, \cdots, \mathbf{q}_{n_a+d\cdot n_b}]$, 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. Clearly, information on $\hat{\mathbf{b}}$ and $\hat{\boldsymbol{\omega}}$ is contained in $\boldsymbol{\vartheta}_{LS}^{svd}$. Hence, it is straightforward to recover this on the basis of Bai's approach using singular value decomposition [21]. Alternatively, the parameter estimation for $\mathbf{b}$ and $\boldsymbol{\omega}$ can be obtained using our previous work [31]. This is outlined below and in the summary in Appendix A. Consider that a sequence $z(t)$ is generated, on the basis of the derived parameter estimates $\hat{\mathbf{a}}$, as an auxiliary model output sequence, given by

$$z(t) = y(t) + \hat{a}_1 y(t-1) + \hat{a}_2 y(t-2) + \ldots + \hat{a}_{n_a} y(t-n_a) \tag{17}$$

Then, consider approximating $z(t)$ by using the following model

$$\begin{aligned} \hat{z}(t) = &\ b_1 \sum_{j=1}^{d} \omega_j \mathcal{B}_j^{(k)}(t-1) + \ldots + b_{n_b} \sum_{j=1}^{d} \omega_j \mathcal{B}_j^{(k)}(t-n_b), \\ = &\ g(\mathbf{x}(t), \mathbf{b}, \boldsymbol{\omega}) \end{aligned} \tag{18}$$

By setting the objective function as $J_g(\mathbf{b}, \boldsymbol{\omega}) = \frac{1}{N}\sum_{t=1}^{N}[z(t) - g(\mathbf{x}(t)), \mathbf{b}, \boldsymbol{\omega})]^2$, subject to $\frac{\sum_{j=1}^{n_b} \hat{b}_j}{1+\sum_{j=1}^{n_a} \hat{a}_j} = 1$ (see Assumption 2), the Gauss–Newton algorithm, subject to constraints, as proposed in [31], is used in this work. For completeness, see Appendix A.

3. THE MODEL-BASED PID CONTROLLER

Figure 2 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, which is 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

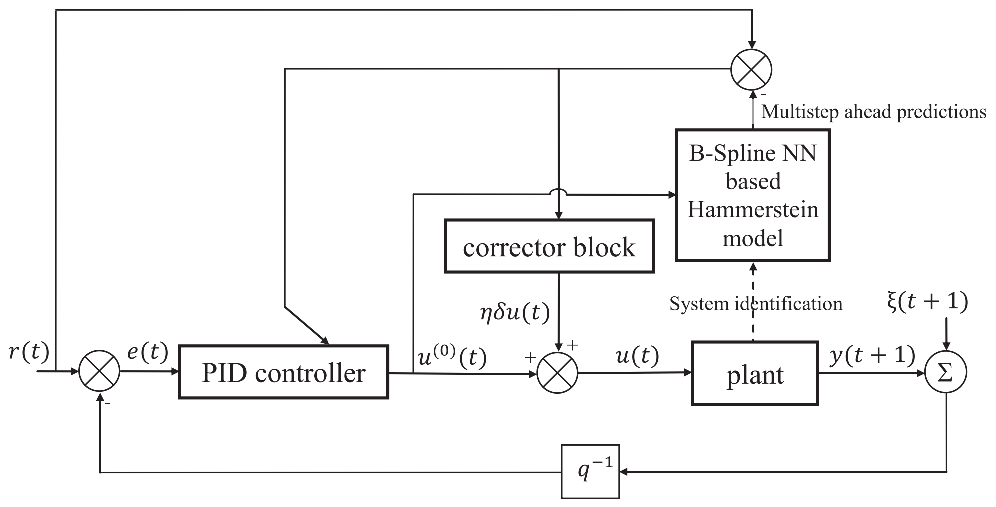


Figure 2. Diagram of the model-based PID controller.

$$J(u(t)) = \frac{1}{2}\sum_{\kappa=1}^{K}[e(t+\kappa|t)]^2 + \frac{1}{2}\lambda(u(t)-u(t-1))^2 \tag{19}$$

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

$$u(t) = u^{(0)}(t) + \eta\delta u(t) \tag{20}$$

which is obtained by a two-step procedure. (i) The PID controller parameters are initially optimized on the basis of minimizing (19) without the correction term ($\eta = 0$), followed by (ii) obtaining $\eta\delta u(t)$, minimizing (19) 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 following formula

$$u^{(0)}(t) = u(t-1) + K_P[e(t)-e(t-1)] + K_I e(t) + K_D[e(t)-2e(t-1)+e(t-2)], \tag{21}$$

is applied to the Hammerstein system, where K_P, K_I and K_D are the PID parameters to be optimized on the basis of the objective function $J(u^{(0)}(t))$. At the beginning of the control sequence, the PID parameters are set to zero. In the proposed scheme, we adopt the Levenberg–Marquardt rule (22) as the minimization algorithm, such that the PID parameters are updated at every time step according to

$$\begin{bmatrix} K_P^{new} \\ K_I^{new} \\ K_D^{new} \end{bmatrix} = \begin{bmatrix} K_P^{old} \\ K_I^{old} \\ K_D^{old} \end{bmatrix} - \alpha\left(\mathbf{J}^T\mathbf{J} + \mu\mathbf{I}\right)^{-1}\mathbf{J}^T\hat{\mathbf{e}} \tag{22}$$

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

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

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

$$\hat{\mathbf{e}} = \begin{bmatrix} e(t+1|t) \\ \vdots \\ e(t+K|t) \\ \sqrt{\lambda}(u(t)-u(t-1)) \end{bmatrix}_{|u(t)=u^{(0)}(t)} = \begin{bmatrix} r(t+1|t) - \hat{y}(t+1|t) \\ \vdots \\ r(t+K|t) - \hat{y}(t+\kappa|t) \\ \sqrt{\lambda}(u(t)-u(t-1)) \end{bmatrix}_{|u(t)=u^{(0)}(t)} \tag{24}$$

It can be seen that the Jacobian matrix (23) can be decomposed as the product of two different matrices by using the chain rule, as follows

$$\mathbf{J} = \left(- \begin{matrix} \frac{\partial \hat{y}(t+1|t)}{\partial u(t)} \\ \frac{\partial \hat{y}(t+2|t)}{\partial u(t)} \\ \vdots \\ \frac{\partial \hat{y}(t+K|t)}{\partial u(t)} \\ \sqrt{\lambda} \end{matrix} \times \begin{matrix} \frac{\partial u(t)}{\partial K_P} & \frac{\partial u(t)}{\partial K_I} & \frac{\partial u(t)}{\partial K_D} \end{matrix} \right)_{|u(t)=u^{(0)}(t)}$$
$$= \mathbf{J}_m \mathbf{J}_c \tag{25}$$

where

$$\mathbf{J}_m = - \begin{bmatrix} \frac{\partial \hat{y}(t+1|t)}{\partial u(t)} & \frac{\partial \hat{y}(t+2|t)}{\partial u(t)} & \dots & \frac{\partial \hat{y}(t+K|t)}{\partial u(t)} & \sqrt{\lambda} \end{bmatrix}^T_{|u(t)=u^{(0)}(t)} \tag{26}$$

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

$$\mathbf{J}_c = \begin{bmatrix} e(t) - e(t-1) \\ e(t) \\ e(t) - 2e(t-1) + e(t-2) \end{bmatrix}^T \tag{27}$$

Over time, the PID parameters are expected to converge. However, mostly in the transient-state and to some extent in the steady-state, the converged PID parameters may not be good enough to produce an acceptable control action; that is, the control action $u^{(0)}(t)$ may not be adequate to force the plant output toward the desired trajectory because of some modeling inaccuracies and external disturbances, which lead 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 (20) by minimizing the objective function $J(u(t))$ given by (19). More specifically, the corrector block tries to minimize the objective function J with respect to $\delta u(t)$ on the basis of the second-order Taylor approximation of the objective function J as follows:

$$\begin{aligned} J(u(t)) &= J(u^{(0)}(t) + \delta u(t)) \\ &\approx J(u^{(0)}(t)) + \frac{\partial J(u(t))}{\partial u(t)}_{|u(t)=u^{(0)}(t)} \delta u(t) \\ &+ \frac{1}{2}\frac{\partial^2 J(u(t))}{\partial u(t)^2}_{|u(t)=u^{(0)}(t)} (\delta u(t))^2 \end{aligned} \tag{28}$$

Because 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

$$\frac{\partial J(u(t))}{\partial u(t)}_{|u(t)=u^{(0)}(t)} + \frac{\partial^2 J(u(t))}{\partial u(t)^2}_{|u(t)=u^{(0)}(t)} \delta u(t) = 0 \tag{29}$$

so

$$\delta u(t) = -\frac{\frac{\partial J}{\partial u(t)}_{|u(t)=u^{(0)}(t)}}{\frac{\partial^2 J}{\partial u(t)^2}_{|u(t)=u^{(0)}(t)}} \tag{30}$$

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 [46]. At this point, it seems that we need to calculate the gradient and Hessian terms, that is, the first-order and second-order derivatives of the objective function with respect to $u(t)$. However, in order to avoid calculating the time-consuming second-order derivatives, we can employ the well-known Jacobian approximation which suggests that the $(K+1)\times 1$ Jacobian matrix $\mathbf{J}_m$ can represent the gradient vector exactly and the Hessian matrix approximately as

$$\frac{\partial J(u(t))}{\partial u(t)}_{|u(t)=u^{(0)}(t)} = 2\mathbf{J}_m^T\hat{\mathbf{e}} \quad \text{and} \quad \frac{\partial^2 J(u(t))}{\partial u(t)^2}_{|u(t)=u^{(0)}(t)} \approx 2\mathbf{J}_m^T\mathbf{J}_m \tag{31}$$

Thus, we can compute the correction term as

$$\delta u(t) = -\mathbf{J}_m^T\hat{\mathbf{e}}/\mathbf{J}_m^T\mathbf{J}_m \tag{32}$$

In this way, we need only the first-order derivatives. It is obvious that the Jacobian matrix $\mathbf{J}_m$ plays a very crucial role in the proposed structure, as it allows us to obtain the Jacobian matrix (25) to update the PID parameters and also to calculate the correction term by means of (32).

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 [47]. This algorithm directly evaluates $J(u^{(n)}(t))$ 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 (see Appendix B).

Note that the PID controller parameter updating formula (22) requires the calculation of multistep ahead predictions and the Jacobian $\mathbf{J}_m$. Moreover, the subsequent control signal correction term, given by (20) and (32) via the golden section algorithm (Appendix B), 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 [12], this controller scheme has been applied to unknown nonlinear systems that are modelled using neural networks and 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

$$\begin{aligned}\hat{y}(t+\kappa|t) = & -a_1\hat{y}(t+\kappa-1|t) - a_2\hat{y}(t+\kappa-2|t) - \ldots - a_{n_a}\hat{y}(t+\kappa-n_a|t) \\ & + b_1 v(t+k-1) + \ldots + b_{n_b} v(t+k-n_b)\end{aligned} \tag{33}$$

in which each term in the right-hand side of (33) is computed by

$$\hat{y}(t+\kappa-i|t) = \begin{cases} \hat{y}(t+\kappa-i|t) & \text{if } (\kappa-i) > 0 \\ y(t+\kappa-i) & \text{otherwise} \end{cases}$$
$$i = 1, \cdots, n_a, \;\; \kappa = 1, \cdots, K \tag{34}$$

and

$$v(t+\kappa-i) = \begin{cases} \sum_{j=1}^{d} \mathcal{B}_j^{(k)}(u(t))\omega_j & \text{if } (\kappa-i) \geqslant 0 \\ \sum_{j=1}^{d} \mathcal{B}_j^{(k)}(u(t+k-i))\omega_j & \text{otherwise} \end{cases}$$
$$i = 1, \cdots, n_b, \;\; \kappa = 1, \cdots, K \tag{35}$$

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

$$\begin{aligned}\frac{\partial \hat{y}(t+\kappa|t)}{\partial u(t)} = & -a_1\frac{\partial \hat{y}(t+\kappa-1|t)}{\partial u(t)} - a_2\frac{\partial \hat{y}(t+\kappa-2|t)}{\partial u(t)} - \ldots - a_{n_a}\frac{\partial \hat{y}(t+\kappa-n_a|t)}{\partial u(t)} \\ & + b_1\frac{\partial v(t+\kappa-1|t)}{\partial u(t)} + \ldots + b_{n_b}\frac{\partial v(t+\kappa-n_b|t)}{\partial u(t)}\end{aligned} \tag{36}$$

in which each term in the right-hand side of (36) is computed by

$$\frac{\partial \hat{y}(t+\kappa-1|t)}{\partial u(t)} = \begin{cases} \frac{\partial \hat{y}(t+\kappa-1|t)}{\partial u(t)} & \text{if } (\kappa-i) > 0 \\ 0 & \text{otherwise} \end{cases}$$
$$i = 1, \cdots, n_a, \;\; \kappa = 1, \cdots, K \tag{37}$$

and

$$\frac{\partial v(t+\kappa-i|t)}{\partial u(t)} = \begin{cases} \sum_{j=1}^{d} \omega_j \frac{d}{du(t)}\mathcal{B}_j^{(k)}(u(t)) & \text{if } (\kappa-i) \geqslant 0 \\ 0 & \text{otherwise} \end{cases}$$
$$i = 1, \cdots, n_b, \;\; \kappa = 1, \cdots, K \tag{38}$$

Note that in calculating (33)–(39), the de Boor algorithm (6)–(8) is applied in evaluating the associated entries. In particular, we point out that the term $\frac{d}{du(t)}\mathcal{B}_j^{(k)}(u(t))$, in (39), 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 the de Boor recursion. Specifically, at each time step, the proposed algorithm requires (33) to be evaluated $(n_{max} + 1)$ times, on the basis of $u^{(n)}(t)$, $n = 1, \cdots, n_{max}$, where n_{max} is the maximum number of iterations set in the golden section algorithm. Equation (36) is, however, only evaluated once for the calculation of $\mathbf{J}_m$. Hence, the two main parts of the computational cost are, first, due to the PID parameter updates in the order of $O(9\times(K+1)+3^3)$ and, second, due to the iterative multistep ahead predictions mainly in the golden section algorithm. This is of the 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 function-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 next into consideration while attempting to find their proper values by trial and error.

Remark 2

- The choice of λ. Because λ penalizes the difference between successive control actions, its choice is mainly based on the allowable input slew in the control loop, the level of the measurement noise and the change in the reference input. If there is no measurement noise and the reference signal is not changing abruptly, then λ can be chosen to be very small, even zero. Otherwise, it may slow the system response if it is chosen to be too large. If there is some measurement noise, then the control input will try to tolerate it by fluctuating around a nominal value, which will increase the performance index given by (19) when λ is too small. Therefore, while making a choice for λ, the physical constraints on the actuators, the level of measurement noise and the expected trajectory of the reference input should be taken into consideration.
- Choice of K. Prediction horizon K has a large impact on the stability and the rise time of the closed-loop system. Too small K values may lead to oscillations at the output or even instability, whereas too large values could cause long delays in the system response. In other words, the choice of K gives a compromise between the stability and the rise time of the closed-loop system.
- Stability conditions. The main goal of the controller is to force the system output to follow the reference trajectory. For this purpose, the controller tries to find a suboptimal solution to minimize the cost function given by (19). For linear noise-free systems, it is demonstrated in [48] that instead of finding the global solution, feasible, suboptimal solutions to this type of problem can yield a stabilizing controller. Moreover, as shown in [49], it is even not necessary to find a local minimum, and the only task during each sampling period is to find a solution that provides a sufficient decrement in the cost function. In this respect, the model-based PID controller provides sufficient stability and works for applications with noise-free or low noise cases. The stability conditions for noisy nonlinear systems are, however, theoretically difficult and remain as an open problem. In practice, in order to show the robustness of the proposed controller against the uncertainties caused by prediction errors, the noise level up to which the controller can provide satisfactory performance should be determined, together with other parameter settings by trial and error.

4. NUMERICAL EXAMPLES

The Hammerstein system is a suitable model for signal processing applications involving any nonlinear distortion followed by a linear filter, for example the modeling of hydraulic actuator friction dynamics [18] and liquid level control system for a nonconstant cross-sectional area tank [42].

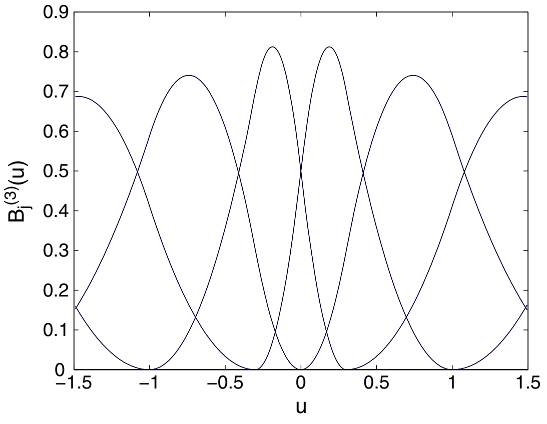


Figure 3. Eight B-spline basis functions used in the two Hammerstein systems.

Two Hammerstein systems are simulated, in which the linear subsystems are the same for both systems as $A(q^{-1}) = 1 - 1.2q^{-1} + 0.9q^{-2}$, $B(q^{-1}) = 1.7q^{-1} - q^{-2}$. The nonlinear subsystem, $\Psi(\bullet)$, is given in each case by

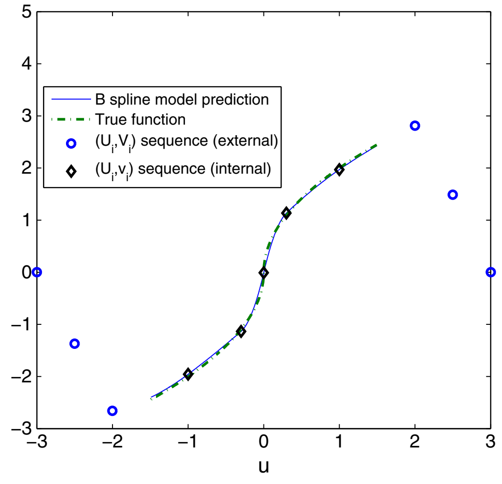
$$\text{System 1}: \Psi(u) = 2\text{sign}(u)\sqrt{|u|} \tag{39}$$

$$\text{System 2}: \Psi(u) = -2\text{sign}(u)u^2 \tag{40}$$

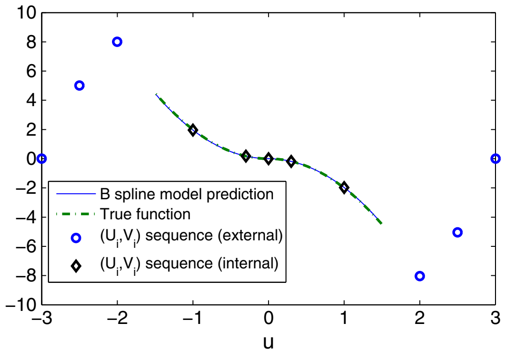
respectively. The variances of the additive noise to the system output are set as $\sigma^2 = 0.0001$ and 0.01, respectively. For each system, 1000 training data samples $y(t)$ were generated by using (1) and (2), where $u(t)$ was a uniformly distributed random variable $u(t) \in [-1.5, 1.5]$. The polynomial degree of the B-spline basis functions was set as $k = 2$ (piecewise quadratic). The knot sequence U_j was set as [-3, -2.5, -2, -1, -0.3, 0, 0.3, 1, 2, 2.5, 3]. The resultant 8 basis functions are plotted

Table I. Results of linear subsystem parameter estimation for two systems.

	a_1	a_2	b_1	b_2
True parameter	−1.2	0.9	1.7	−1
Estimate parameters (System 1, $\sigma^2 = 0.0001$)	−1.1986	0.8993	1.6886	−0.9880
Estimate parameters (System 1, $\sigma^2 = 0.01$)	−1.2003	0.9006	1.7115	−1.0112
Estimate parameters (System 2, $\sigma^2 = 0.0001$)	−1.1989	0.8994	1.6887	−0.9882
Estimate parameters (System 2, $\sigma^2 = 0.01$)	−1.1992	0.8999	1.6781	−0.9774



(a)



(b)

Figure 4. Modelling results for the nonlinear function $\Psi(u)$; (a) System 1 and (b) System 2.

in Figure 3. Initially, system identification was carried out using the modeling algorithm outlined in Appendix A for each system. Modelling results are shown in Table I, for the linear subsystem, and in Figure 4 (a) and (b), for the nonlinear subsystems.

The performance of the proposed controllers for the two systems were tested on the basis of the models identified using data sets generated with $\sigma^2 = 0.0001$. 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 signals $r(t)$ were generated as a series of square waves resembling a staircase. Figure 5(a) and Figure 6(a) plot the computed control signal applied to each system, respectively. Figure 5(b) and Figure 6(b) plot the system output $y(t)$ together with the corresponding reference signal $r(t)$ for both systems, respectively, with a small noise ($\sigma^2 = 2.5 \times 10^{-5}$ for System 1 and $\sigma^2 = 9 \times 10^{-6}$ for System 2). From these figures, it is shown that the proposed method exhibits excellent results in terms of system identification, as well as the subsequent control for the identified systems for these particular examples.

In [45], the control is investigated through the removal of nonlinearity via the inversion of the same B-spine neural network and the application of a pole assignment controller. The performance is heavily dependent upon the assumption that the nonlinearity is removed via the inserted functional inversion. Our experiences on the same examples using pole assignment with linearization [45] have shown that, although they can perform equally well in cases of low noise level and when the reference signals $r(t)$ are well below the bounds, the proposed method can have a higher operating range in cases of noise-free or low noise level without performance deterioration.

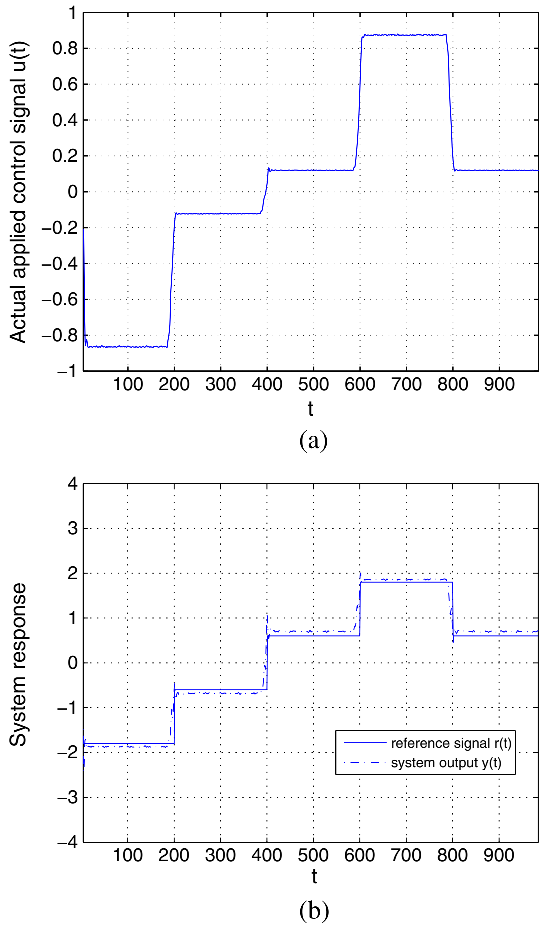
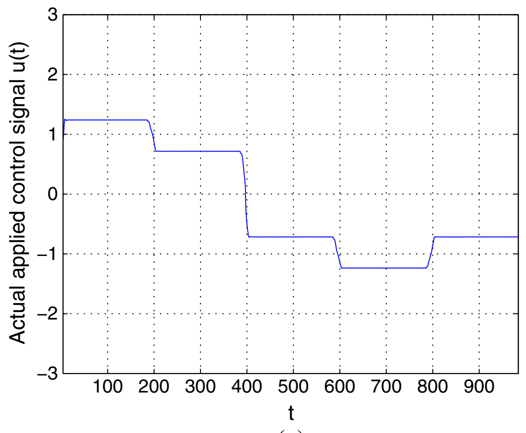
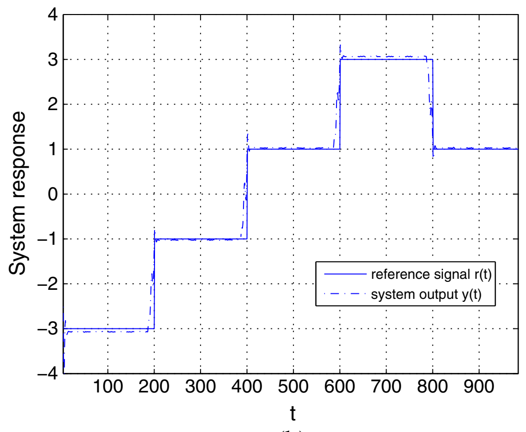


Figure 5. Results of the proposed PID controller for System 1.



(a)



(b)

Figure 6. Results of the proposed PID controller for System 2.

This is because in [45], the control signal can overshoot the range of $v(t)$, which has to be amplitude-limited before the inverse of the de Boor algorithm. On the other hand, in terms of robustness to noise level for stability, the proposed algorithm is worse than the pole assignment with linearization [45], probably, because of uncertainties caused by error propagation in multistep ahead predictions.

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 multistep ahead predictions of the B-spline neural network-based Hammerstein system and the associated Jacobian matrix are very efficiently computed on the basis of the de Boor algorithms, including both the functional and derivative recursions. The efficacy of the proposed approach has been demonstrated via two numerical examples.

The algorithm proposed here has been specifically aimed at using a practical, well-regarded PID structure to deal with a class of nonlinear systems that can be reasonably modelled using a Hammerstein approach. We have shown here how this can be achieved and have given a number of examples as to how computational cost can be minimized by employing recursive approximation techniques where applicable.

APPENDIX A: A SUMMARY OF THE SYSTEM IDENTIFICATION ALGORITHM

1. On the basis of the training data set and any prior knowledge on the system, predetermine the number of basis functions d, the polynomial order k and the input range $[U_{min}, U_{max}]$. Predetermine a set of $(d + k)$ knots according to (5).
2. Over the training data set $D_N = \{y(t), u(t)\}_{t=1}^{N}$, construct d basis functions $\mathcal{B}_j^{(d)}(u(t))$ on the basis of (6) and (7). Subsequently, apply the method described in Section 2.3 to find the parameter vector $\hat{\mathbf{a}}$ as the subvector of $\boldsymbol{\vartheta}_{LS}^{svd}$.
3. Construct the auxiliary model output sequence $z(t)$ by using (17).
4. Apply the Gauss–Newton algorithm, subject to the constraint $G = 1$ [31], to find $\hat{\mathbf{b}}$ and $\hat{\boldsymbol{\omega}}$.
5. On the basis of $\hat{\boldsymbol{\omega}}$, the underlying function $\Psi(\bullet)$ for any point within the range $[U_{min}, U_{max}]$ can be recovered by applying the de Boor algorithm (using (6)–(8)).

APPENDIX B: PSEUDOCODE OF THE GOLDEN SECTION ALGORITHM

```
Predetermine $\eta_{max}, \eta_{min}$. {*The search range for $\eta$ is determined by $\eta_{min} = 0$ and $\eta_{max}$, which is mapped from the control signal bounds via (20)*}
$\tau = 0.38197$;
Predetermine precision $\epsilon$.
$n_{max} = -2.078\log(\epsilon/(\eta_{max} - \eta_{min}))$ {*the number of iterations*}
$\eta_1 = (1-\tau)\eta_{min}+\tau\eta_{max}$; $u_{temp} = u^{(0)}(t)+\eta_1\delta u(t)$
Obtain $K$-step ahead predictions $(\hat{y}(t + \tau|t) \quad \tau = 1, ..., K)$ in response to $u_{temp}$
Calculate $J_1 = \sum_{\tau=1}^{K}[r(t + \tau) - \hat{y}(t + \tau|t)]^2 + \lambda[u_{temp} - u(t-1)]^2$
$\eta_2 = \tau\eta_{min}+(1-\tau)\eta_{max}$; $u_{temp} = u^{(0)}(t)+\eta_2\delta u(t)$
Obtain $K$-step ahead predictions $(\hat{y}(t + \tau|t) \quad \tau = 1, ..., K)$ in response to $u_{temp}$
Calculate $J_2 = \sum_{\tau=1}^{K}[r(t + \tau) - \hat{y}(t + \tau|t)]^2 + \lambda[u_{temp} - u(t-1)]^2$
**for** $n = 1 \rightarrow n_{max}$ **do**
  $n \leftarrow n + 1$
  **if** $J_2 < J_1$ **then**
    $\eta_{min} \leftarrow \eta_1$; $\eta_1 \leftarrow \eta_2$ ; $J_1 \leftarrow J_2$
    $\eta_2 = \tau\eta_{min}+(1-\tau)\eta_{max}$; $u_{temp} = u^{(0)}(t)+\eta_2\delta u(t)$
    Obtain $K$-step ahead predictions $(\hat{y}(t + \tau|t) \quad \tau = 1, ..., K)$ in response to $u_{temp}$
    Calculate $J_2 = \sum_{\tau=1}^{K}[r(t + \tau) - \hat{y}(t + \tau|t)]^2 + \lambda[u_{temp} - u(t-1)]^2$
  **end if**
  **if** $J_1 < J_2$ **then**
    $\eta_{max} \leftarrow \eta_2$; $\eta_2 \leftarrow \eta_1$ ; $J_2 \leftarrow J_1$
    $\eta_1 = (1-\tau)\mu_{min}+\tau\mu_{max}$; $u_{temp} = u^{(0)}(t)+\eta_1\delta u(t)$
    Obtain $K$-step ahead predictions $(\hat{y}(t + \tau) \quad \tau = 1, ..., K)$ in response to $u_{temp}$
    Calculate $J_1 = \sum_{\tau=1}^{K}[r(t + \tau) - \hat{y}(t + \tau|t)]^2 + \lambda[u_{temp} - u(t-1)]^2$
  **end if**
**end for**
$\eta_{optimal} = (\eta_1 + \eta_2)/2$
```

ACKNOWLEDGEMENTS

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

REFERENCES

1. Astrom KJ, Hagglund T, Hang CC, Ho HK. Automatic tuning and adaptation for PID controllers—a survey. *Control Engineering Practice* 1993; **1**(5):699–714.
2. Astrom KJ, Hagglund T. *PID Controllers: Theory, Design and Tuning*. ISA Research Triangle Park, NC, 1995.
3. O'Dwyer A. *Handbook of PI and PID Controller Tuning Rules*. Imperial College Press: London, 2003.

4. Na MG. Auto-tuned PID controller using a model-predictive control method for the steam generator water level. *IEEE Transactions on Nuclear Science* 2001; **48**:1664–1671.
5. Xu M, Li S, Qi C, Cai W. Auto-tuning of PID controller parameters with supervised receding horizon optimization. *ISA Transactions* 2005; **44**:491–500.
6. Parlos AG, Parthasarathy S, Atiya AF. Neuro-predicitive process control using on-line controller adaptation. *IEEE Transactions on Control Systems Technology* 2001; **9**:741–755.
7. Chen J, Huang T. Applying neural networks to on-line updated PID controllers for nonlinear process control. *Journal of Process Control* 2004; **14**:211–230.
8. 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*, Beijing, China, 2005; 1854–1857.
9. Wu C. Genetic tuning of PID controllers using a neural network model: a seesaw example. *Journal of Intelligent and Robotic Systems* 1999; **25**:43–59.
10. Liu H, Liu D. Self-tuning PID controller for a nonlinear system based on support vector machines. *Kongzhi Lilun yu Yingyong / Control Theory and Applications* 2008; **25**(3):468–474.
11. Shen Y, Shang W, Zhao S. Adaptive PID controller based on online LSSVM identification. In *Proceedings of the IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, Xian, China, 2008; 694–698.
12. 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* 2010; **20**(13):1483–1501.
13. Hunter IW, Korenberg MJ. The identification of nonlinear biological systems: Wiener and Hammerstein cascade models. *Biological Cybernetics* 1986; **55**(2-3):135–144.
14. Bloemen HHJ, Van Den Boom TJ, Verbruggen HB. Model-based predictive control for Hammerstein–Wiener systems. *International Journal of Control* 2001; **74**(5):482–295.
15. Turunen J, Tanttu JT, Loula P. Hammerstein model for speech coding. *EURASIP Journal of Applied Signal Processing* 2003; **12**:1238–1249.
16. 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* 2001; **48**(3):645–655.
17. Su SW. Identification and control for heart rate regulation during treamill exercise. *IEEE Transactions on Biomedical Engineering* 2007; **54**(7):1238–1246.
18. Kwak B, Yagle AE, Levitt JA. Nonlinear system identification of hydraulic actuator friction dynamics using a Hammerstein model. In *Proceedings of the IEEE ASSP'98*, Seattle, WA, 1998; 1933–1936.
19. Billings SA, Fakhouri SY. Nonlinear system identification using the Hammerstein model. *International Journal of Systems Science* 1979; **10**:567–578.
20. Stoica P, Söderström T. Instrumental variable methods for identification of Hammerstein systems. *International Journal of Control* 1982; **35**:459–476.
21. Bai EW, Fu MY. A blind approach to Hammerstein model identification. *IEEE Transactions on Signal Processing* 2002; **50**(7):1610–1619.
22. Lang ZQ. A nonparametric polynomial identification algorithm for the Hammerstein system. *IEEE Transactions on Automatic Control* Oct 1997; **42**:1435–1441.
23. Greblicki W, Pawlak M. Identification of discrete Hammerstein systems using kernel regression estimate. *IEEE Transactions on Automatic Control* 1986; **AC-31**(1):74–77.
24. Greblicki W. Nonparametric orthogonal series identification of Hammerstein systems. *International Journal of Systems Science* 1989; **20**:2355–2367.
25. Greblicki W. Stochastic approximation in nonparametric identification of Hammerstein systems. *IEEE Transactions on Automatic Control* 2002; **47**(11):1800–1810.
26. Chen HF. Pathwise convergence of recursive identification algorithms for Hammerstein systems. *IEEE Transactions on Automatic Control* 2004; **49**(10):1873–1896.
27. Verhaegen M, Westwick D. Identifying MIMO Hammerstein systems in the context of subspace model identification. *International Journal of Control* 1996; **63**(2):331–349.
28. Chaoui FZ, Giri F, Rochdi Y, Haloua M, Naitali A. System identification based Hammerstein model. *International Journal of Control* 2005; **78**(6):430–442.
29. Farouki RT, Goodman TNT. On the optimal stability of the Bernstein basis. *Mathematics of Computation* 1996; **65**(216):1553–1566.
30. Hong X, Harris CJ. Generalised neurofuzzy network modelling algorithms using bezier bernstein polynomial functions and additive decomposition. *IEEE Transactions on Neural Networks* 2000; **11**:889–902.
31. Hong X, Mitchell RJ. A Hammerstein model identification algorithm using Bezier–Bernstein approximation. *IET Proc Control Theory and Applications* 2007; **1**(4):1149–1159.
32. Bai EW. An optimal two-stage identification algorithm for Hammerstein–Wiener nonlinear systems. *Automatica* 1998; **34**:333–338.
33. Farin G. *Curves and Surfaces for Comnputer-aided Geometric Design: A Practical Guide*. Academic Press: Boston, 1994.
34. de Boor C. *A Practical Guide to Splines*. New York: Spring Verlag, 1978.
35. Kavli T. ASMOD-an algorithm for adaptive spline modelling of observation data. *International Journal of Control* 1993; **58**(4):947–967.
36. Brown M, Harris CJ. *Neurofuzzy Adaptive Modelling and Control*. Prentice Hall: Hemel Hempstead, 1994.

37. Harris CJ, Hong X, Gan Q. *Adaptive Modelling, Estimation and Fusion from Data: A Neurofuzzy Approach*. Springer-Verlag: Berlin, 2002.
38. Anbumani K, Patnaik LM, Sarma IG. Self-tuning minimum variance control of nonlinear systems of the Hammerstein model. *IEEE Transactions on Automatic Control* 1981; **AC-26**(4):959–961.
39. Bloemen HH, van den Boom TJ, Verbruggen HB. Model based predictive control for Hammerstein systems. In *Proceedings of the 39th IEEE Conference on Decision and Control*, Sydney, Australia, 2000; 4963–4968.
40. Fruzzetti E, Palazoglu A, Mcdonald KA. Nonlinear model predictive control using Hammerstein models. *Journal of Process Cotnrol* 1997; **7**(1):31–41.
41. Patwardhan RS, Lakshminarayanan S, Shah SL. Contrained nonlinear MC using Hammerstein and Wiener model PLS framework. *AIChE Journal* 1998; **44**(7):1611–1622.
42. Astrom KJ, Wittenmark B. *Adaptive Control*. Addison Wesley: MS, 1989.
43. Zhu QM, Warwick K, Douce JL. Adaptive general predictive controller for nonlinear systems. *IEE Proceedings Control Theory and Applications* 1991; **138**(1):33–40.
44. Hong X, Mitchell RJ. A pole assignment controller for Bezier–Bernstein polynomial based Hammerstein model. In *Proceedings of International Control Conference (ICC) 2006*, Glascow, UK, 2006.
45. Hong X, Mitchell RJ, Chen S. Modelling and control of Hammerstein system using B-spline approximation and the inverse of de Boor algorithm. *International Journal of Systems Science* 2011. DOI: 10.1080/00207721.2011.564320.
46. Nocedal J, Wright SJ. *Numerical Optimization*. Springer: New York, 1999.
47. Venkataraman P. *Applied Optimization with MATLAB Programming*. Wiley-Interscience: New York, 2002.
48. Scokaert POM, Mayne DQ, Rawlings JB. Suboptimal model predictive controllers (feasibility implies stability). *IEEE Transactions on Automatic Control* 1999; **44**:648–654.
49. Maciejowski JM. *Predictive Control with Constraints*. Pearson Education Limited: Essex, UK., 2002.