2009 American Control Conference
Hyatt Regency Riverfront, St. Louis, MO, USA
June 10-12, 2009

A new neural network-based approach for self-tuning control of nonlinear multi-input multi-output dynamic systems

Jose I. Canelon*, Leang S. Shieh, *Senior Member*, *IEEE*, Yongpeng Zhang, *Member*, *IEEE*, and Cajetan M. Akujuobi, *Senior Member*, *IEEE*

Abstract— **This paper presents a new neural network-based approach for self-tuning control of nonlinear MIMO dynamic systems. According to the approach, a neural network ARMAX (NN-ARMAX) model of the system is identified and continuously updated, using an online training algorithm. Control design is accomplished by solving an optimal discrete-time linear quadratic tracking problem using an observable block companion form Kalman innovation model, which is built from the parameters of a local linear version of the NN-ARMAX model. The state-feedback control law is implemented using the Kalman estimated state, which is calculated without estimating the noise covariance properties. The effectiveness of the proposed control approach is illustrated using a simulation example.**

I. Introduction

SELF-TUNING control refers to approaches in which the design of the control law is based on a model which is identified and continuously updated, using measured input-output data. Hence, self-tuning control strategies are well suited for systems whose model is unknown and/or time-varying systems.

In general, a self-tuning control approach includes three main steps [1]-[4]: (i) modeling, in which a mathematical model of the system is identified, using measured input-output data, (ii) design, in which a control design approach is used to synthesize the control law, based on the identified model, and (iii) implementation, which comprises the digital implementation of the control law. These steps are sequentially repeated at each control iteration.

In the 1970s, the increasingly available computing power boosted the research activity on self-tuning control. Several key papers were published, using different control design criteria (for a list of references, see [1]-[4]). Subsequent research focused on the theoretical convergence and stability properties of the algorithms, complemented with a significant number of engineering applications [1] on both single-input single-output (SISO) and multi-input multi-output (MIMO) systems.

In most of the reported work, the modeling task is accomplished with a linear model, which is updated using recursive least squares. This is mainly due, on one hand, to the great variety of available linear control design techniques and, on the other hand, to the convergence properties of recursive least squares. A detailed description of self-tuning control methods based on linear models can be found in [1]-[4]. In particular, Shieh et al. published various papers ([5]-[7], among others) related to their research work on self-tuning control of SISO and MIMO systems. For MIMO systems with integer observability index [6], their approach includes the execution of the following steps in each control iteration: (i) recursive identification of a linear autoregressive moving average with exogenous inputs (ARMAX) model of the system dynamics, (ii) construction of a linear Kalman innovation model, in observable block companion form, using the parameters of the ARMAX model, (iii) design, based on the linear state-space model, of a suboptimal state-feedback control law by approximately solving the steady-state Riccati equation, (iv) implementation of the control law using the state estimated by the Kalman innovation model. However, self-tuning approaches based on linear models may not show good performance in applications on highly nonlinear systems.

Since it was shown that neural networks are nonlinear models that can approximate any function with arbitrary degree of accuracy [8], there has been a growing number of applications reported in the literature that employ neural networks to control complex nonlinear processes. An extensive review of neural networks-based control approaches can be found in the surveys by Hunt et al. [9] and Agarwal [10]. Also based upon the modeling capabilities of neural networks, there has been research work on neural network-based self tuning control.

Rubaai et al. [11] proposed a scheme consisting of a model reference adaptive controller and a state estimation mechanism. The estimation mechanism involves five

Manuscript received September 22, 2008. This work was supported by the U.S. Army Research Office under grant W911NF-06-1-0507 and the National Science Foundation under grant NSF 0717860.

* Corresponding author

Jose I. Canelon is with the School of Electrical Engineering, Universidad del Zulia, Maracaibo, 4005, Venezuela (Phone: 58-261-7598844; fax: 58-261-7598748; e-mail: jcanelon@mail.uh.edu).

Leang S. Shieh is with the Department of Electrical and Computer Engineering, University of Houston, Houston, TX 77204-4005 USA (e-mail: lshieh@uh.edu).

Yongpeng Zhang is with the Center of Excellence for Communication Systems Technology Research and the Department of Engineering Technology, Prairie View A&M University, Prairie View, TX 77446 USA (e-mail: yongpeng_zhang@pvamu.edu).

Cajetan M. Akujuobi is with the Center of Excellence for Communication Systems Technology Research and the Department of Engineering Technology, Prairie View A&M University, Prairie View, TX 77446 USA (e-mail: c_akujuobi@pvamu.edu).

feedforward neural networks, trained online using the Levenberg-Marquardt algorithm. Each neural network estimates one state of the system at time instant k, based on previous measurements of all states and the control inputs.

Zhu et al. [12] present a self tuning approach to regulate the rotor speed and rotor flux in an induction motor. A linear model, identified with recursive least recursive least squares, is used to obtain a local approximation of the dynamics of the motor, around an operating point. Then, a neural network model is trained, using backpropagation with momentum, to represent the residuals of the linear model. The parameters of these models are used to implement a generalized minimum variance controller.

Some other self-tuning approaches propose the use of neural network models for tuning a PID controller. Ohnishi et al. [13] tune a PID controller based upon a first order plus time delay (FOPD) linear model, whose parameters are estimated from an autoregressive with external inputs neural network model (NN-ARX).

Cheng and Huang [14] propose an algorithm which uses the parameters of a linear model to tune a PID controller. Those parameters are determined by employing Taylor's approach for instantaneous linearization of a neural network model of the process. The neural network is trained off-line, to reduce the computational load of training. Off-line training may not show good performance because of its limitations to handle noise, and to rapidly capture changes in the dynamics of the process. In addition, Taylor's approach will yield an affine rather than a linear model.

The speed of training is the most important issue when implementing a neural network-based self-tuning control scheme. The standard backpropagation [15], or some of its variations [16], have been used as training algorithms in most of the reported work. However, these gradient-based algorithms may converge slowly because they generally require several iterations, and rely rather critically on training parameters that must be specified by the user, without well-established rules for selecting the values of such parameters for a given training task.

In this paper we extend the self-tuning control approach proposed by Canelon et al. [17] for SISO discrete-time nonlinear systems, to the case of nonlinear MIMO systems. This extension is based on [6]. The modeling task is accomplished by a neural-network ARMAX (NN-ARMAX) model, which is trained using the online sequential extreme learning algorithm (OS-ELM) [18]. Control design involves the solution of an optimal discrete-time linear quadratic tracking problem, using a linear equivalent of the NN-ARMAX model. A Kalman innovation filter is used to estimate the state of the system, for implementation of the state-feedback control law. The proposed control approach is tested on a simulation example. The rest of the paper presents a description of the proposed control approach.

II. Online Extreme Sequential Learning Algorithm

In this work, a NN_ARMAX model of the MIMO system is identified and recursively updated. The neural network is trained using the online sequential extreme learning algorithm (OS-ELM), proposed by Liang et al. [18] as a recursive version of the extreme learning machine (ELM), developed by Huang et al. [19]. The universal approximation capability of ELM has been analyzed in [20]. While Liang et al. [18] used recursive least squares, we use extended least-squares [21] (also called approximate maximum-likelihood method) to implement OS-ELM.

Consider a feedforward neural network with I input units, one layer of J hidden units, and O output units. For a given input vector $\boldsymbol{\delta} = [\delta_1,\ldots,\delta_I]^T$, the response of the jth hidden unit is given by

$$h_j = \rho\left(\boldsymbol{v}_j^T \hat{\boldsymbol{\delta}}\right),\ j = 1,\ldots,J \tag{1}$$

where $\rho(\cdot) = tanh(\cdot)$ is the activation function for the hidden units, $\hat{\boldsymbol{\delta}} = [1,\delta_1,\ldots,\delta_I]^T$ and $\boldsymbol{v}_j = \left[v_{j0}, v_{j1},\ldots,v_{jI}\right]^T$ is the weight vector for the jth hidden unit, where v_{j0} is the bias of such hidden unit, and v_{ji} is the weight that connects the ith input unit to the jth hidden unit. The response of the oth output unit is given by

$$\hat{\zeta}_o = \boldsymbol{h}^T \boldsymbol{w}_o,\ o = 1,\ldots,O \tag{2}$$

where $\boldsymbol{h} = [1,h_1,\ldots,h_J]^T$ and $\boldsymbol{w}_o = \left[w_{o0}, w_{o1},\ldots,w_{oJ}\right]^T$ is the weight vector for the oth output unit, with w_{o0} the bias of this output unit, and w_{oj} the weight connecting the jth hidden unit to the oth output unit.

After the number J of hidden units has been chosen, the on-line training algorithm involves two phases:

A. *Initialization*

An initial chunk of S_0 training vectors is used to initialize the learning according to the following procedure:

(i) Randomly assign the weights between the input and hidden layers. Random assignment (with any distribution) of these weights guarantees that the approximation error can be made arbitrarily small ([18] and [19]).

(ii) Compute the matrix

$$\boldsymbol{H}(0) = \begin{bmatrix} h_{11} & \cdots & h_{1J} \\ \vdots & & \vdots \\ h_{S_o 1} & \cdots & h_{S_o J} \end{bmatrix} \tag{3}$$

of size $S_0 \times J$, containing the responses of the J hidden units for the training vectors in the initial chunk.

(iii) For each output unit, estimate the initial vector of weights connecting such unit and the hidden units as

$$\boldsymbol{w}_o(0) = \boldsymbol{P}(0)\boldsymbol{H}^T(0)\zeta_o(0), \tag{4}$$

where

$$\boldsymbol{P}(0)=\left[\boldsymbol{H}^T(0)\boldsymbol{H}(0)\right]^{-1} \tag{5}$$

and $\boldsymbol{\zeta}_o(0)$ is the vector containing the target values for the oth output in the initial chunk.

(iv) Set $k=1$.

B. Sequential learning

While there are more chunks of data

(i) Pick the kth chunk with S_k training vectors.

(ii) Calculate the partial hidden layer output matrix $\boldsymbol{H}(k)$, of size $S_k \times J$, containing the responses of the J hidden units for the training vectors in kth chunk.

(iii) For each output unit, update the vector of weights connecting such unit and all hidden units using the equation

$$\boldsymbol{w}_o(k)=\boldsymbol{w}_o(k-1)+\frac{\boldsymbol{P}(k-1)\boldsymbol{H}^T(k)}{\alpha(k)+\boldsymbol{H}(k)\boldsymbol{P}(k-1)\boldsymbol{H}^T(k)}\times \left[\boldsymbol{\zeta}_o(k)-\boldsymbol{H}(k)\boldsymbol{w}_o(k-1)\right], \tag{6}$$

where $\boldsymbol{\zeta}_o(k)$ is the vector containing the target values for the oth output in the kth chunk,

$$\boldsymbol{P}(k)=\frac{\boldsymbol{P}(k-1)}{\alpha(k)}\left[\boldsymbol{I}+\frac{\boldsymbol{H}^T(k)\boldsymbol{H}(k)\boldsymbol{P}(k-1)}{\alpha(k)+\boldsymbol{H}(k)\boldsymbol{P}(k-1)\boldsymbol{H}^T(k)}\right], \tag{7}$$

and $\alpha(k)$ is a forgetting factor given by the difference equation

$$\alpha(k)=\alpha_o\alpha(k-1)+(1-\alpha_o) \tag{8}$$

with initial condition $0.9<\alpha(0)<1$, and an updating factor $0<\alpha_o<1$. The use of a forgetting factor will give a better performance for time-varying systems. In addition, extended least squares is computationally more efficient since the update of $\boldsymbol{P}(k)$ requires no matrix inversion.

(iv) Set $k=k+1$.

Prior to the training, all entries of the input and target vectors in the training set are linearly normalized to span the interval $[-1,1]$, using the formula

$$\theta=-1+2\frac{\Theta-\Theta min}{\Theta max-\Theta min} \tag{9}$$

where Θ and θ represent the original and normalized values, respectively, while Θmin and Θmax are the corresponding lower and upper bounds, respectively.

From (9) we obtain the normalization function

$$\theta=nor(\Theta)=\frac{2}{\Theta max-\Theta min}\Theta-\frac{\Theta max+\Theta min}{\Theta max-\Theta min}. \tag{10}$$

and the denormalization function

$$\Theta=dnor(\theta)=\frac{\Theta max-\Theta min}{2}\theta+\frac{\Theta max+\Theta min}{2}. \tag{11}$$

If the normalization function is considered, $\hat{\boldsymbol{\delta}}$ in (1) can be expressed as $\hat{\boldsymbol{\delta}}=\left[1,nor_1(\Delta_1),nor_2(\Delta_2),\ldots,nor_I(\Delta_I)\right]$, where Δ_i and nor_i $(i=1,\ldots,I)$ represent the original value and the normalization function, respectively, corresponding to the ith input. On the other hand, the denormalized value of the oth output of the network is given by

$$\hat{Z}_o=dnor_o\left(\boldsymbol{h}^T\boldsymbol{w}_o\right),\ o=1,\ldots,O \tag{12}$$

where $dnor_o$ represent the denormalization function for the oth output. Following this normalization procedure, the neural network showed good learning performance with the random weights adjusted using a Gaussian distribution with zero mean and standard deviation one.

III. Optimal Linearization of a Neural Network Model

The neural network nonlinear model defined by (1) and (12) can be written in a simplified form as

$$\hat{Z}_o=\boldsymbol{f}_o(\boldsymbol{\Delta}) \tag{13}$$

where $\boldsymbol{\Delta}=[\Delta_1,\ldots,\Delta_I]^T$, and $\boldsymbol{f}_o:\mathbb{R}^I\to\mathbb{R}$ is the nonlinear vector function corresponding to the oth output. Given a particular operating point $\boldsymbol{\Delta}=\overline{\boldsymbol{\Delta}}$, it is desired to find a linear model of the form

$$\hat{Z}_o=\boldsymbol{F}^T\boldsymbol{\Delta} \tag{14}$$

with $\boldsymbol{F}=[\phi_1\ \ \phi_2\ \ \cdots\ \ \phi_I]^T$, locally equivalent to (13) around $\overline{\boldsymbol{\Delta}}$. Taylor's linearization approach has been the most commonly used local linearization technique. However, this technique is not applicable in the vicinity of any operating point of the system. Even if that operating point is an equilibrium point, this approach will yield an affine rather than a linear model if such equilibrium is not the origin. Here we use the optimal linearization approach proposed by Teixeira and Żak [22] for local linearization of the neural network model. According to this approach, $\boldsymbol{F}^T$ is given by [17]

$$\boldsymbol{F}^T=\nabla_{\boldsymbol{\Delta}^T}\boldsymbol{f}_o(\overline{\boldsymbol{\Delta}})+\frac{\boldsymbol{f}_o(\overline{\boldsymbol{\Delta}})-\nabla_{\boldsymbol{\Delta}^T}\boldsymbol{f}_o(\overline{\boldsymbol{\Delta}})\overline{\boldsymbol{\Delta}}}{\overline{\boldsymbol{\Delta}}^T\overline{\boldsymbol{\Delta}}}\overline{\boldsymbol{\Delta}}^T. \tag{15}$$

where

$$\nabla_{\boldsymbol{\Delta}^T}\boldsymbol{f}_o(\overline{\boldsymbol{\Delta}})=\left[\frac{\partial f_o}{\partial\Delta_1}(\overline{\boldsymbol{\Delta}})\ \ \frac{\partial f_o}{\partial\Delta_2}(\overline{\boldsymbol{\Delta}})\ \ \cdots\ \ \frac{\partial f_o}{\partial\Delta_I}(\overline{\boldsymbol{\Delta}})\right] \tag{16}$$

where (according to (1), (12) and (13))

$$\frac{\partial f_o}{\partial\Delta_i}=\frac{d}{d\theta}dnor_o(\theta)\times \sum_{j=1}^{J}\left\{w_{oj}\cdot\left[\frac{d}{d\gamma}\rho(\gamma)\right]\Big|_{\gamma=\overline{\gamma}}\cdot v_{ji}\cdot\frac{d}{d\Theta}nor_i(\Theta)\right\}, \tag{17}$$

$$\overline{\gamma}=\sum_{i=0}^{I}v_{ji}\cdot nor_i(\overline{\Delta}_i), \tag{18}$$

and (from (10) and (11))

$$\frac{d}{d\Theta}nor_i(\Theta)=\frac{2}{\Theta max_i-\Theta min_i} \tag{19}$$

and

$$\frac{d}{d\theta} dnor_o(\theta) = \frac{2}{\Theta max_o - \Theta min_o}. \quad (20)$$

In (19) and (20), Θmax_i and Θmin_i represent the upper and lower bounds, respectively, for the ith input, and Θmax_o and Θmin_o represent the upper and lower bounds, respectively, for the oth output. The optimal linear model will have exactly the same dynamics of the original nonlinear model at the operating point, and minimum modeling error in the neighborhood of such operating point.

IV. Description of the Proposed Self-tuning Control Approach

The proposed control approach involves the execution of the following steps, at each operating point: (i) recursive identification of a discrete-time neural-network based nonlinear ARMAX (NN-ARMAX) model of the MIMO system, (ii) calculation of a linear ARMAX model, locally equivalent to the NN-ARMAX model, (iii) construction of an observable block companion form linear state-space Kalman innovation model, from the parameters of the linear ARMAX model, (iv) calculation of the control gains by solving an optimal control problem, based on the Kalman innovation model and (v) implementation of a state-feedback control law using the observed state.

A. *Identification of the discrete-time neural network-based nonlinear ARMAX model*

Consider a MIMO nonlinear dynamic system with m inputs and p outputs. The proposed approach includes the identification and continuous update of a discrete-time neural network-based nonlinear ARMAX model (NN_NARMAX) of the form

$$\hat{\boldsymbol{y}}(k+1) = \boldsymbol{F}^{(k)}[\boldsymbol{y}(k),\ldots,\boldsymbol{y}(k-n+1),\boldsymbol{u}(k),\ldots, \boldsymbol{u}(k-n+1),\boldsymbol{e}(k),\ldots,\boldsymbol{e}(k-n+1)] \quad (21)$$

where $\boldsymbol{y}(k) = \left[y_1(k) \cdots y_p(k) \right]^T$ represents the measured output vector, $\boldsymbol{u}(k) = \left[u_1(k) \cdots u_m(k) \right]^T$ is the input vector and $\boldsymbol{e}(k) = \left[e_1(k) \cdots e_p(k) \right]^T$ is the innovation error vector, all at time instant k, $\boldsymbol{F}^{(k)}$ represents the neural network model updated using k data vectors, $\hat{\boldsymbol{y}}(k+1)$ is the estimate of the output measured at time instant $k+1$ and n is the number of previous values of the output, input and innovation error vectors being considered in the model. The innovation error vector is defined as

$$\boldsymbol{e}(k) = \begin{bmatrix} y_1(k) - \hat{y}_1(k) \\ \vdots \\ y_p(k) - \hat{y}_p(k) \end{bmatrix}. \quad (22)$$

Note that the neural network model has p outputs and $2pn + mn$ inputs, and it is updated after a new output value is measured.

B. *Calculation of the linear ARMAX model and construction of the observable block companion form linear Kalman innovation model*

Following the measurement of the output vector at time instant k, the optimal linearization approach is used to linearize the NN_NARMAX model at the operating point

$$[\boldsymbol{y}(k),\ldots,\boldsymbol{y}(k-n+1),\boldsymbol{u}(k),\ldots,\boldsymbol{u}(k-n+1), \boldsymbol{e}(k),\ldots,\boldsymbol{e}(k-n+1)] \quad (23)$$

to yield a linear ARMAX model of the form

$$\hat{\boldsymbol{y}}(k+1) = -\boldsymbol{A}_1^{(k)}\boldsymbol{y}(k) - \cdots - \boldsymbol{A}_n^{(k)}\boldsymbol{y}(k-n+1) + \boldsymbol{B}_1^{(k)}\boldsymbol{u}(k) + \cdots + \boldsymbol{B}_n^{(k)}\boldsymbol{u}(k-n+1) + \boldsymbol{D}_1^{(k)}\boldsymbol{e}(k) + \cdots + \boldsymbol{D}_n^{(k)}\boldsymbol{e}(k-n+1). \quad (24)$$

Then, we construct the following observable block companion form linear Kalman innovation model of the MIMO nonlinear system

$$\boldsymbol{x}_o(k+1) = \boldsymbol{A}_o^{(k)}\boldsymbol{x}_o(k) + \boldsymbol{B}_o^{(k)}u(k) + \boldsymbol{K}_o^{(k)}e(k) \quad (25a)$$

$$\hat{\boldsymbol{y}}(k) = \boldsymbol{C}_o^{(k)}\boldsymbol{x}_o(k) \quad (25b)$$

where $\boldsymbol{x}_o(k) \in \boldsymbol{R}^{pn\times 1}$, $\boldsymbol{u}(k) \in \boldsymbol{R}^m$, $\boldsymbol{e}(k) \in \boldsymbol{R}^p$ and $\hat{\boldsymbol{y}}(k) \in \mathbb{R}^p$ are the observed state vector, the input vector, the innovation error vector and the estimated output vector, respectively, at time instant k,

$$\boldsymbol{A}_o^{(k)} = \begin{bmatrix} -\boldsymbol{A}_1^{(k)} & \boldsymbol{I}_p & \boldsymbol{0}_p & \cdots & \boldsymbol{0}_p \\ -\boldsymbol{A}_2^{(k)} & \boldsymbol{0}_p & \boldsymbol{I}_p & \cdots & \boldsymbol{0}_p \\ \vdots & \vdots & \vdots & & \vdots \\ -\boldsymbol{A}_n^{(k)} & \boldsymbol{0}_p & \boldsymbol{0}_p & \cdots & \boldsymbol{0}_p \end{bmatrix} \in \mathbb{R}^{pn\times pn}, \quad (26)$$

with $\boldsymbol{I}_p \in \mathbb{R}^{p\times p}$ the identity matrix and $\boldsymbol{0}_p \in \mathbb{R}^{p\times p}$ a matrix of zeros,

$$\boldsymbol{B}_o^{(k)} = \begin{bmatrix} \boldsymbol{B}_1^{(k)} \\ \boldsymbol{B}_2^{(k)} \\ \vdots \\ \boldsymbol{B}_n^{(k)} \end{bmatrix} \in \mathbb{R}^{pn\times m}, \quad (27)$$

$$\boldsymbol{K}_o^{(k)} = \begin{bmatrix} \boldsymbol{D}_1^{(k)} - \boldsymbol{A}_1^{(k)} \\ \boldsymbol{D}_2^{(k)} - \boldsymbol{A}_2^{(k)} \\ \vdots \\ \boldsymbol{D}_n^{(k)} - \boldsymbol{A}_n^{(k)} \end{bmatrix} \in \mathbb{R}^{pn\times p} \quad (28)$$

and $\boldsymbol{C}_o$, which is fixed, is given by

$$\boldsymbol{C}_o = \begin{bmatrix} \boldsymbol{I}_p & \boldsymbol{0}_p & \cdots & \boldsymbol{0}_p \end{bmatrix} \in \mathbb{R}^{p\times pn}. \quad (29)$$

Note that the observability index is always an integer, equal to $pn/p = n$. Hence, the observable block companion form always can be constructed. Also note that knowledge of the exact order of the system is not required.

When calculating $\boldsymbol{x}_o(k+1)$, $\boldsymbol{u}(k)$ is the control input at

time instant k, and

$$\boldsymbol{e}(k) = \boldsymbol{y}(k) - \hat{\boldsymbol{y}}(k) = \boldsymbol{y}(k) - \boldsymbol{C}_o \boldsymbol{x}_o(k) \tag{30}$$

C. *Calculation of the control gains and implementation of the control law*

The control design approach used here is the optimal discrete-time linear quadratic tracker described in [23], extended for the case of MIMO systems. Given the linear Kalman innovation model of (25), it is desired to design and implement a control law of the form

$$\boldsymbol{u}(k) = -\boldsymbol{K}_c^{(k)} \boldsymbol{x}_o(k) + \boldsymbol{K}_I^{(k)} \boldsymbol{v}(k) \tag{31}$$

where the feedback gain $\boldsymbol{K}_c^{(k)} \in \mathbb{R}^{m \times pn}$ and the integral gain $\boldsymbol{K}_I^{(k)} \in \mathbb{R}^{m \times p}$ are the design parameters, $\boldsymbol{x}_o(k)$ is the observed-state and

$$\boldsymbol{v}(k) = \boldsymbol{v}(k-1) + \boldsymbol{r}(k) - \boldsymbol{y}(k) \tag{32}$$

where $\boldsymbol{r}(k)$ is the reference signal. Observe that this control design approach includes a state-feedback loop and an integrator in the feedforward path.

If we define

$$\hat{\boldsymbol{K}}^{(k)} = \begin{bmatrix} \boldsymbol{K}_c^{(k)} & -\boldsymbol{K}_I^{(k)} \end{bmatrix}, \tag{33}$$

it can be shown [23] that $\hat{\boldsymbol{K}}$ can be calculated as

$$\hat{\boldsymbol{K}}^{(k)} = \left[\boldsymbol{R} + \left(\hat{\boldsymbol{B}}^{(k)} \right)^T \boldsymbol{P} \hat{\boldsymbol{B}}^{(k)} \right]^{-1} \left(\hat{\boldsymbol{B}}^{(k)} \right)^T \boldsymbol{S} \hat{\boldsymbol{A}}^{(k)} \tag{34}$$

where $\boldsymbol{R}$ is a positive definite matrix,

$$\hat{\boldsymbol{A}}^{(k)} = \begin{bmatrix} \boldsymbol{A}_o^{(k)} & \boldsymbol{0} \\ -\boldsymbol{C}_o \boldsymbol{A}_o^{(k)} & \boldsymbol{I}_p \end{bmatrix}, \tag{35}$$

where $\boldsymbol{0} \in \mathbb{R}^{pn \times p}$ is a matrix of zeros,

$$\hat{\boldsymbol{B}}^{(k)} = \begin{bmatrix} \boldsymbol{B}_o^{(k)} \\ -\boldsymbol{C}_o \boldsymbol{B}_o^{(k)} \end{bmatrix} \tag{36}$$

and $\boldsymbol{S}$ is the solution of the Ricatti equation

$$\hat{\boldsymbol{A}}^T \boldsymbol{S} \hat{\boldsymbol{A}} - \boldsymbol{S} - \hat{\boldsymbol{A}}^T \boldsymbol{S} \hat{\boldsymbol{B}} \left(\boldsymbol{R} + \hat{\boldsymbol{B}}^T \boldsymbol{S} \hat{\boldsymbol{B}} \right)^{-1} \hat{\boldsymbol{B}}^T \boldsymbol{S} \hat{\boldsymbol{A}} + \boldsymbol{Q} = \boldsymbol{0} \tag{37}$$

with $\boldsymbol{Q}$ a positive definite or positive semidefinite matrix.

Following the computation of $\hat{\boldsymbol{K}}^{(k)}$, $\boldsymbol{K}_c^{(k)}$ and $K_I^{(k)}$ can be easily extracted and the control law can be readily implemented, according to (31).

V. Simulation Results

The proposed approach was used on the direct digital control of a permanent-magnet synchronous motor (PMSM), a nonlinear system described by the continuous-time state-space model

$$\begin{aligned} \frac{d}{dt} i_d &= -\frac{R}{L} i_d + \eta_p i_q \omega + v_d \\ \frac{d}{dt} i_q &= -\frac{R}{L} i_q - \eta_p i_d \omega - \frac{\psi_r}{L} \omega + v_q \\ \frac{d}{dt} \omega &= \frac{\psi_r}{L} i_q - \frac{\beta}{J} \omega \end{aligned} \tag{38}$$

where the state variables i_d, i_q and ω are the direct and quadrature current components and the motor angular frequency, respectively, the input variables v_d and v_q are the direct and quadrature voltage components, $R = 19.1388$ is the stator winding resistance, $L = 0.04$ is the direct and quadrature-axis stator inductors, $J = 4.1295 \times 10^{-4}$ is the polar moment of inertia, $\beta = 0.0013$ the viscous damping coefficient, $\psi_r = 0.1349$ the permanent-magnet flux and $n_p = 50$ the number of pole-pairs [24]. Under certain operating conditions, the PMSM can exhibit chaotic behavior [24].

The controlled outputs are the states i_d and i_q, i.e. the output equation is

$$\boldsymbol{y}_1(t) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} i_d \\ i_q \\ \omega \end{bmatrix}. \tag{39}$$

Zero mean Gaussian noise, with variance $1 \cdot 10^{-4}$, was added to each of the states and the outputs.

The sampling time was chosen as $T = 0.05$ *sec*. The system has $m = 2$ inputs and $p = 2$ outputs, and $n = 2$ previous values of the outputs, inputs and innovation errors were considered. Hence, the neural network model has 2 outputs and $2pn + mn = 12$ inputs. In addition, 20 hidden units were chosen. The extended least-squares training parameters were $\alpha_o = 0.95$, $\alpha(0) = 0.95$ and $\boldsymbol{P}(0) = \boldsymbol{I}$. The upper and lower bounds were 20 and -20, respectively, for the outputs and innovation errors, and 1000 and -1000, respectively, for the inputs. Regarding the optimal discrete-time linear quadratic tracker, $\boldsymbol{Q} = 10^3 \boldsymbol{I}_6$ and $\boldsymbol{R} = 2\boldsymbol{I}_2$. The reference signal for both outputs was a unit step.

Fig. 1(a) is a plot of the output i_d (solid line) and the corresponding reference signal r (dotted line), while Fig. 1(b) shows the output i_q (solid line) and the corresponding reference signal r (dotted line). On the other hand, Fig. 2 shows the control inputs, v_d (dotted line) and v_q (solid line). It can be seen that, after approximately 5 sec, both outputs closely follow the reference signals.

VI. Conclusions

This paper presented a new approach for self-tuning control of nonlinear MIMO dynamic systems. Some advantages of the proposed approach are: (i) the training

algorithm requires small computational effort and shows the convergence properties of recursive least squares, (ii) since control design is based on a linear model, the great variety of available linear design techniques may be used, (iii) full-state measurement is not required due to the use of the Kalman filter, and (iv) it is not necessary to know or estimate the noise covariance properties.

The proposed control approach showed to be very effective in a simulation example using a permanent magnet synchronous motor, a two-input two-output continuous-time nonlinear systems that, under certain operating conditions may exhibit chaotic behavior. Further research topics are using more efficient digital control approaches, and real-time implementation of the proposed self-tuning approach.

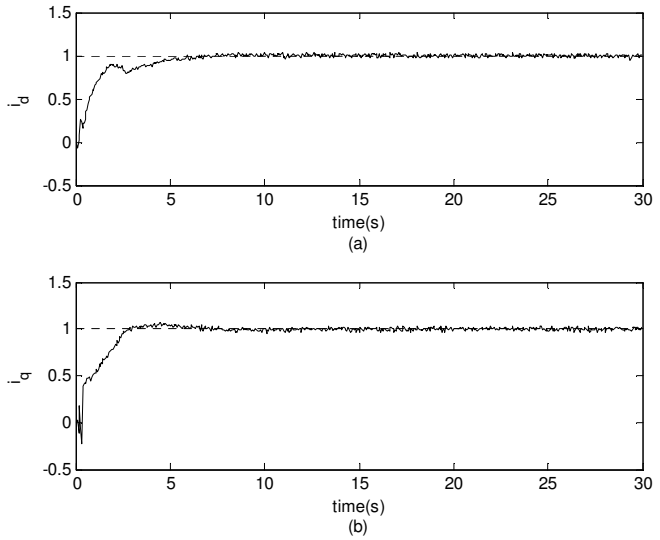

Figure 1. Outputs of the systems and reference signals during the control task: (a) i_d (solid line) and reference (dotted line), (b) i_q (solid line) and reference (dotted line).

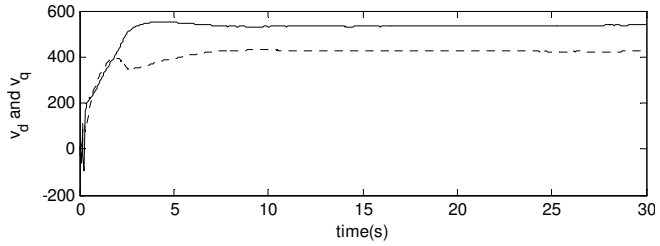

Figure 2. Control inputs during the control task: v_d (dotted line) and v_q (solid line).

REFERENCES

[1] P. E. Wellstead, and M. B. Zarrop, *Self-Tuning Systems: Control and signal processing*. Chichester, UK: John Wiley & Sons, 1991, pp. 1-38.

[2] D. W. Clarke, *Advances in Model-Based Predictive Control*. Oxford, UK: Oxford University Press, 1994, pp. 4-21.

[3] P. J. Gawthrop, *Continuous-Time Self-Tuning Control*. New York, USA: Research Studies Press, 1987, pp. 6.1-6.33.

[4] K. Warwick, "Self-tuning regulators-a state-space approach," *International Journal of Control*, vol. 33, pp. 839-858, May 1981.

[5] Y. T. Tsay, and L. S. Shieh, "State-space approach for self-tuning feedback control with pole assignment," *IEE Proceedings D - Control Theory and Applications*, vol. 128, pp. 93-101, May 1981.

[6] L. S. Shieh, C. T. Wang, and Y. T. Tsay, "Fast suboptimal state-space self-tuner for linear stochastic multivariable systems," *IEE Proceedings D - Control Theory and Applications*, vol. 130, pp. 143-154, July 1983.

[7] L. S. Shieh, Y. L. Bao, and F. R. Chang, "State-space self-tuning regulators for general multivariable stochastic systems," *IEE Proceedings D - Control Theory and Applications*, vol. 136, pp. 17-27, January 1989.

[8] K. Hornik, M. Stinchcombe, and H. White, H. "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, pp. 359-366, July 1989.

[9] K. J. Hunt, D. Sbarbaro, R. Żbikowski, and P. J. Gawthrop, "Neural networks for control systems – A survey," *Automatica*, vol. 28, pp. 1083-1112, November 1992.

[10] M. Agarwal, "A systematic classification of neural-network based control," *IEEE Control Systems Magazine*, vol. 17, pp. 75-93, April 1997.

[11] A. Rubaai, R. Kotaru, and M. D. Kankam, "Online training of parallel neural network estimators for control of induction motors," *IEEE Transactions on Industry Applications*, vol. 37, pp. 1512-1520, September/October 2001.

[12] Q. M. Zhu, L. Z. Guo, and Z. Ma, "Neural network enhanced optimal self-tuning controller design for induction motors," in *Global Optimization*, vol. 85, book series Nonconvex Optimization and Its Applicatons, New York, USA: Springer, 2006, pp. 529-546.

[13] Y. Ohnishi, T. Yamamoto, T. Yamada, I. Nanno, and M. Tanaka, "A design of nonlinear PID control systems with neural-net based system estimator," presented at *SICE*, Osaka, Japan, August 5-7, 2002, Paper sice02-0480, pp. 2272-2273.

[14] J. Chen, and T. C. Huang, "Applying neural networks to on-line updated PID controllers for nonlinear process control," *Journal of Process Control*, vol. 14, pp. 211-230, March 2004.

[15] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing: Exploration in the Microstructure of Cognition*, D. E. Rumelhart, and J. L. McClelland, Eds., Cambridge, USA: MIT Press, 1986, pp. 318-362.

[16] R. Battiti, "First- and second-order methods for learning: between steepest descent and Newton's method," *Neural Computation*, vol. 4, pp. 141-166, March 1992.

[17] J. I. Canelon, L. S. Shieh, and G. Song, "A new neural network-based approach for self-tuning control of nonlinear SISO discrete-time systems," *International Journal of Systems Science*, submitted for publication.

[18] N. Y. Liang, G. B. Huang, P. Saratchandran, and N. Sundararajan, "A fast and accurate online sequential learning algorithm for feedforward networks," *IEEE Transactions on Neural Networks*, vol. 17, pp. 1411-1423, November 2006.

[19] G. B. Huang, Q. Y. Zhu, and C. K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, pp. 489-501, December 2006.

[20] G. B. Huang, L. Chen, and C. K. Siew, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *IEEE Transactions on Neural Networks*, vol. 17, pp. 879-892, July 2006.

[21] L. Ljung, *System Identification: Theory for the user*. Upper Saddle River, NJ: Prentice Hall PTR, 1999, pp. 374-376.

[22] M. Teixeira, and S. Żak, "Stabilizing controller design for uncertain nonlinear systems using fuzzy models," *IEEE Transactions on Fuzzy Systems*, vol. 7, pp. 133-142, April 1999.

[23] K. Ogata, *Discrete-Time Control Systems*. Englewood Cliffs, NJ: Prentice Hall, 1995, pp. 596-609.

[24] Z. Li, J. B. Park, Y. H. Joo, B. Zhang, and G. Chen, "Bifurcations and chaos in a permanent-magnet synchronous motor," *IEEE Transactions on Circuits and Systems I, Fundamental Theory and Applications*, vol. 49, pp. 383-387, March 2002.