Available online at www.sciencedirect.com
ScienceDirect
Applied Soft Computing 8 (2008) 371–382
Applied Soft Computing
www.elsevier.com/locate/asoc

Adaptive control of a nonlinear dc motor drive using recurrent neural networks

Khaled Nouri [a], Rached Dhaouadi [b,*], Naceur Benhadj Braiek [a]

[a] *Laboratoire d'Etude et de Commande Automatique de Processus (LECAP), Ecole Polytechnique de Tunisie, BP 743, 2078 La Marsa, Tunisia*
[b] *Department of Electrical Engineering, School of Engineering, American University of Sharjah, United Arab Emirates*

Received 7 May 2006; received in revised form 30 January 2007; accepted 4 March 2007
Available online 7 March 2007

Abstract

A model-following adaptive control structure is proposed for the speed control of a nonlinear motor drive system and the compensation of the nonlinearities. A recurrent artificial neural network is used for the online modeling and control of the nonlinear motor drive system with high static and Coulomb friction. The neural network is first trained off-line to learn the inverse dynamics of the motor drive system using a modified form of the decoupled extended Kalman filter algorithm. It is shown that the recurrent neural network structure combined with the inverse model control approach allows an effective direct adaptive control of the motor drive system. The performance of this method is validated experimentally on a dc motor drive system using a standard personal computer. The results obtained confirm the excellent disturbance rejection and tracking performance properties of the system.

Keywords: Recurrent neural network; Nonlinear system; Motor drive; Kalman filter; Inverse model

1. Introduction

High performance motor drive systems have become the preferred choice in many industrial applications and there is a strong interest to develop new control tools to further enhance their performance and their intelligence. In lieu of the recent advances in power electronics and microprocessors, digital control of motor drives has become increasingly popular. Micro-computer-based control systems with digital control strategies for dc/ac motors have gained the widest acceptance in high performance adjustable speed drives. However, conventional control schemes (e.g. proportional-integral controller) have certain limitations and cannot achieve a good speed response when the system parameters vary. Many other difficulties could be also encountered when dealing with highly nonlinear plants. To overcome these drawbacks, the control scheme should include an additional compensation scheme to eliminate the variations in the load disturbance and/or system parameters. New control algorithms such as adaptive, variable structure, and robust controllers, are therefore introduced. This however leads to an increasingly more complicated control algorithm requiring a complex design procedure and extensive computations for real-time implementation.

Recently, substantial research efforts were devoted to artificial neural networks and their application to control systems to deal with the problems of non-linearity and system parameters uncertainty [1–9]. Artificial neural networks (ANN) have proved to be very successful in different areas of engineering sciences. The fundamental characteristics of neural networks are their ability to produce good models of nonlinear systems; their highly distributed and paralleled structure, their simple implementation by software or hardware; and their ability to learn and adapt themselves to the behavior of any real process. ANN can also be trained using data taken from the system or when directly connected to the system, and require less processing time because of their parallel structure.

The design of an ANN starts by defining the structure of the graph, the synaptic coefficients and the activation function. The architecture of the network and the activation function can be fixed according to the task that should be realized; the synaptic coefficients are calculated during the learning phase. MultiLayers Perceptrons (MLP) are one of the most popular

* Corresponding author. Tel.: +971 6 5152927; fax: +971 6 5152979.
E-mail addresses: khaled.nouri@ept.rnu.tn (K. Nouri), rdhaouadi@aus.edu (R. Dhaouadi), naceur.benhadj@ept.rnu.tn (N. Benhadj Braiek).

1568-4946/$ – see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2007.03.002

artificial neural net structures. If the network has feedback connections, the obtained graph is called a recurrent network. The standard and most popular procedure used to design an ANN is to have a feed-forward neural network structure trained with the back-propagation algorithm [10–12]. A limiting characteristic however of the feed-forward neural network is the absence of internal or hidden state, so that the processing of input patterns does not depend upon the order in which these patterns are represented during training or recall. Thus, the representation and processing of sequential or temporal information is not an intrinsic capability of feed-forward neural networks, although tapped-delay lines can be used to encode explicitly a finite number of past events [13–15].

Today, interest has been also increasing in the use of multiplayer neural networks as elements in dynamical systems [16–19]. This has been motivated by two considerations. Since all physical systems involve dynamics, modeling a physical system as a "natural" neural network should realistically include dynamical elements. Furthermore, from a control point of view, dynamical elements have to be included to have well-posed problems.

The use of recurrent neural networks (RNN) as system identification networks and feedback controllers offers a number of potential advantages over the use of static layered networks. RNN provide a means for encoding and representing internal or hidden states, albeit in a potentially distributed fashion, which leads to capabilities that are similar to those of an observer in modern control theory. RNN provide increasing flexibility for filtering noisy inputs. RNN feedback controllers may be less sensitive and more robust than static feed-forward controllers to changes in plant dynamics and parameters [20–24].

In this paper, a new robust adaptive control scheme using only one recurrent neural network is proposed for the identification and adaptive control of nonlinear dc motor drives. The RNN is designed and trained first off-line to learn the inverse dynamic model of the considered system from the observation of the input–output data. After the training process is completed, the proposed adaptive control structure is applied to control the speed and suppress the disturbances dues to nonlinearities in the system. Such development is sought to address the control problem for systems with nonlinear dynamics, the problem of disturbance suppression and offset-free control in the presence of such disturbances, which is of high priority in motion control systems [13,14,25].

The different sections of this paper are organized as follows: in Section 2 we present the dynamic model of the considered motor drive system. The description of the recurrent neural network used for identification and adaptive control is presented in Section 3. Section 4 presents the proposed version of a parameter-based decoupled extended Kalman filter algorithm (DEKF) used for adapting the weights of the RNN. The discrete time models of the considered system and the neural identification of the inverse model are presented in Section 5. Section 6 provides the proposed adaptive neuro-control structure and the detailed computation process. Experimental results of the identification and control system are illustrated in Section 7, and a conclusion is drawn in Section 8.

2. Dynamic model of the nonlinear motor drive system

The system considered for our analysis is a dc motor drive system composed of a dc motor connected to a mechanical load through a gear and flexible couplings. The complete system can be represented by the schematic diagram shown in Fig. 1.

The design of a high performance motion control system requires an accurate knowledge of the electro-mechanical system dynamics including the linear and nonlinear transmission attributes of the system. Friction on the other hand is an inevitable characteristic of mechanical systems and is undesirable by control system designers. Its presence is often responsible for performance degradation and closed-loop bandwidth limitation, and should be therefore taken into consideration in the modeling and control design phases.

The nonlinear dynamic equations of the system can be derived using a permanent magnet dc motor model with a two mass model equivalent system:

$$v_a = R_a i_a + L_a \frac{di_a}{dt} + K_b \omega_m, \qquad \tau_m = K_t i_a \tag{1}$$

$$J_m \frac{d\omega_m}{dt} + B_m \omega_m + \tau_f + \frac{h(\theta)}{N} = \tau_m \tag{2}$$

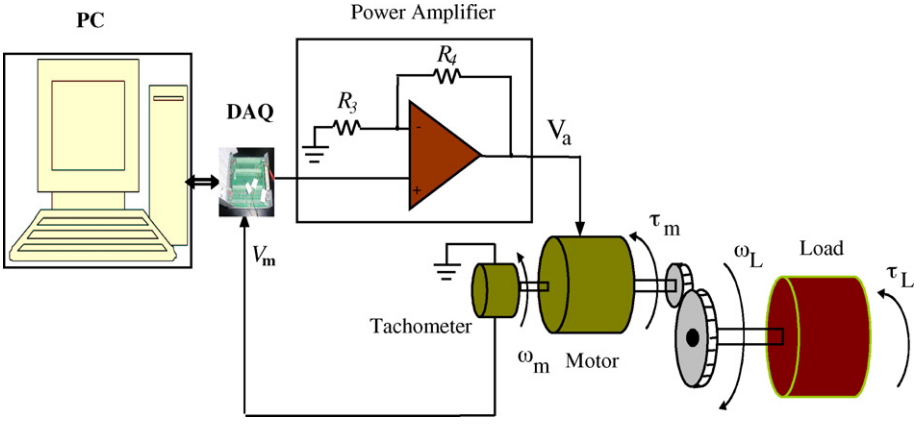

Fig. 1. Motor drive system.

$$J_l \frac{d\omega_l}{dt} + B_l\omega_l + \tau_L - h(\theta) = 0 \quad (3)$$

where R_a, L_a, K_b, K_t are the dc motor parameters, and v_a, i_a are the dc motor voltage and current. ω_m is the motor speed, ω_l the load speed, τ_m the motor torque, τ_f the friction torque, τ_L the load torque, J_m the motor inertia, J_l the load inertia, and B_m, B_l are the damping coefficients. $h(\theta)$ represents the nonlinear effects due to elasticity, and $\theta = (\theta_m/N) - \theta_l$ is the angular displacement. τ_f and τ_L represent the nonlinear friction and disturbance effects. N is the gear ratio.

It is verified in practice that incidental nonlinearities such as saturation and nonlinear friction usually cause deterioration in the performance of systems otherwise linear. The combination of friction with gear compliance renders it difficult to achieve high precision speed, position and torque control in pointing and tracking servomechanisms. Fig. 2 shows the effect of these nonlinearities on our system. The speed response of the system to a sinusoidal control voltage is measured in open loop using a tachometer mounted on the load side. The flat regions of the speed signal are due to the nonlinear friction, mainly the static and Coulomb frictions, which are very large in the considered drive system. This makes it very hard to control the system in the low speed range.

The additional nonlinearity present in the system is the input voltage saturation which is due to the limited level of the D/A converter (±10 V). The power amplifier circuit presents also a saturation level limited by the dc power supply (±12 V). These effects should be taken into consideration when designing the closed-loop control system.

The accurate modeling of the complete motor drive system including the nonlinear effects presents therefore a difficult problem. In most of the literature, Coulomb and static friction are considered for modeling. Numerous models have been proposed also to represent either the general friction dynamics or some aspects of nonlinear friction and compliance effects [25]. The majority of these models have been either too complicated, with parameters that are difficult to determine, or too simple, assuming a linearized model and neglecting the nonlinear effects. The physical realities of the system have therefore limited the acceptance of these models for certain applications.

The successful design of any high performance control system will be therefore based on either an accurate knowledge of the dynamics of the plant including the nonlinearities, or on an effective control scheme for friction compensation. Recurrent neural networks are hereby introduced as a viable solution to model the complete nonlinear system and act as a controller that provides a direct compensation of the nonlinear effects.

3. Dynamic model of the recurrent neural network

The dynamic neural network used in our work is a recurrent multilayer perceptron (RMLP) consisting of nodes and layers arranged in feed-forward fashion as shown in Fig. 3. In addition, the hidden and output layers contain internal feedback connections, while the output is fed back to the input layer through tapped-delay lines (TDL). Only the nodes in the hidden layer contain sigmoid functions.

The weights and biases are represented by w^h and θ^h, between the input and the hidden layer, and by w^o and θ^o, between the hidden and output layer. The weights representing the feedback in the output layer are denoted by w^{ro}, while those in the hidden layer are denoted by w^{rh}.

Let u be the input to the neural network, and x, y be the output from the hidden layer and the output layer, respectively:

$$y(k) = w_{11}^{ro} y(k-1) + \sum_{j=1}^{n_h} w_{1j}^{o} x_j(k) + \theta_1^o \quad (4)$$

$$x_j(k) = \sigma\left(\sum_{i=0}^{n} w_{ji}^{h} u(k-i) + \sum_{l=1}^{m} w_{jl}^{h} y(k-l) + \sum_{i=1}^{n_h} w_{ji}^{rh} x_i(k-1) + \theta_j^h\right) \quad (5)$$

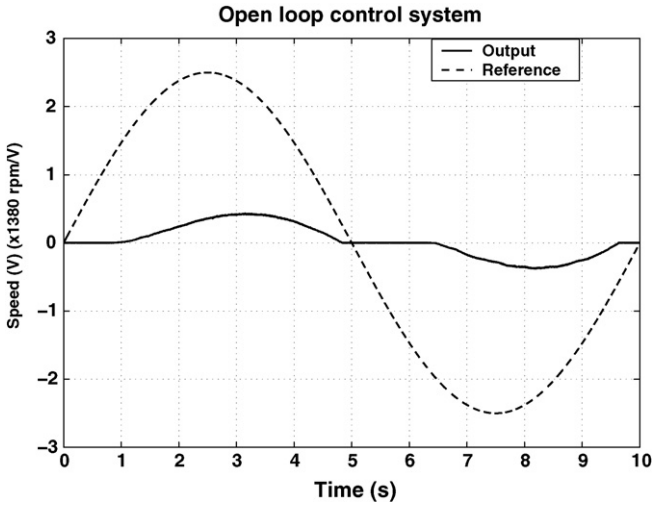

Fig. 2. Effet of nonlinear friction in open-loop control mode.

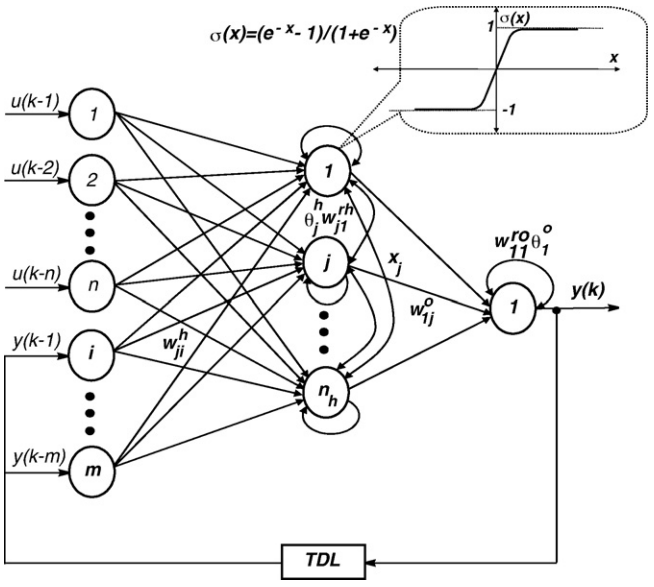

Fig. 3. Recurrent multilayer perceptron (RMLP) architecture.

The principal characteristic of the RMLP is the internal memory property, where the input along a recurrent connection is a function of the processing elements previous output, and hence a function of the processing elements entire past history. Feed-forward neural networks perform a static mapping of inputs to outputs, whereas RNN perform a dynamic mapping of inputs to outputs [5,8,9,16].

With a desired output d, the overall sum-squared error of the network over N training data can be written as

$$E = \frac{1}{2}\sum_{k=1}^{N} e_k^2 = \frac{1}{2}\sum_{k=1}^{N}(d(k) - y(k))^2 \tag{6}$$

The two most popular training algorithms for recurrent networks are the Back-Propagation Through Time (BPTT) and the Real-Time Recurrent Learning (RTRL) which are techniques based on gradient descent [22,23]:

$$w(k) = w(k-1) + \Delta w(k) \tag{7}$$

$$\Delta w(k) = -\alpha \frac{\mathrm{d}E(k)}{\mathrm{d}w} \tag{8}$$

where α is the learning rate.

For the purpose of training, the ''Back-Propagation Through Time'' (BPTT) algorithm treats a completely recurrent network that is executed for n time steps as an equivalent layered feed-forward network with n layers of nodes, where the weights of a given layer are identical to the weights of all other layers. On the other hand, for a completely recurrent network, the Real-Time Recurrent Learning (RTRL) algorithm needs the computation and storage of the partial derivatives of each processing element's output with respect to all weights of the network. This approach provides a mechanism for computing the appropriate gradients for training at each time step. The BPTT and RTRL have large memory requirement in comparison to the standard back-propagation algorithm. They exhibit long convergence times due to small learning rates that are required by these procedures, and they are often susceptible to local minima [22–24].

In order to increase the rate of convergence and assist in the avoidance of local minima, second-order gradient methods, such as the Extended Kalman Filter (EKF), have been proposed as training algorithms. The use of the EKF algorithm was first explored for application to signal processing problems. Later, many other approaches based on the EKF were proposed [10–12]. In their previous work [17], the authors presented a different variation of the decoupled extended Kalman filter (DEKF) proposed by Puskorius and Feldkamp [16]. The derivation of the DEKF is based upon a natural simplification of the EKF algorithm by ignoring the interdependence of mutually exclusive groups of weights.

In this paper, the RMLP is trained off-line to learn the inverse model dynamics of the considered plant from the input–output data observation. The weights of the RMLP will be adjusted by using the DEKF algorithm that will be described in the following section.

4. Training algorithm based on the extended Kalman filter

The derivation of the decoupled extended Kalman filter (DEKF) algorithm is based upon the assertion that the weights of the network can be grouped (or decoupled) such that the element's of the error covariance matrix $P(k)$ corresponding to the correlation between weights from different groups can be ignored [16,17]. Two forms of weight arrangement are examined. In the first form, the weights are grouped by node, referred to as node-decoupled extended Kalman filter (NDEKF). In the second, each weight of the network constitutes a group in itself. It is called fully-decoupled extended Kalman filter (FDEKF).

Given a network with M weights and n_o output nodes, partition the weights into g groups, with m_i weights in group i. The update for a training instance at time step k of the DEKF algorithm is given by

$$G(k) = [R(k) + \sum_{i=1}^{g} H_i^{\mathrm{T}}(k) P_i(k) H_i(k)]^{-1} \tag{9}$$

$$K_i(k) = P_i(k) H_i(k) G(k) \tag{10}$$

$$W_i(k+1) = W_i(k) + \alpha K_i(k)(d(k) - y(k)) \tag{11}$$

$$P_i(k+1) = P_i(k) - K_i(k) H_i^{\mathrm{T}}(k) P_i(k) + Q_i(k) \tag{12}$$

where the matrix $G(k)$ is a $n_o \times n_o$ global scaling matrix that contains correlation information between the plant's observable outputs. $R(k)$ is a diagonal $n_o \times n_o$ matrix whose diagonal components are equal to or slightly less than 1. $H_i(k)$ is a $m_i \times n_o$ gradient matrix containing the partial derivatives of the outputs of the plant's model with respect to the weights of group i. Note that the components of the matrix $H_i(k)$ are obtained with the RTRL algorithm [17,19,22–24]. $P_i(k)$ is a $m_i \times m_i$ matrix representing the uncertainty in the estimates of the weights of the group i defined as the approximate conditional error covariance matrix. $K_i(k)$ is a $m_i \times n_o$ matrix containing the Kalman gain for weights of group i. W_i is a vector of length m_i containing the weight values of group i. α is a scalar learning rate parameter. $d(k)$ is a desired output and $y(k)$ is the plant's observable output. Finally $Q_i(k)$ is a $m_i \times m_i$ diagonal covariance matrix of an artificial process noise. The presence of this matrix helps to avoid numerical divergence of the algorithm, and also helps the algorithm to avoid reaching poor local minima [16,20].

In our work, the considered RMLP weights are decoupled in a new form, which we refer to as layered decoupled extended Kalman filter (LDEKF): for each network layer, we regrouped the biases of the nodes in a group, the weights of the feedback connections in a second group, and the weights related to successive layer in another group. The weights of the network are therefore decoupled with respect to their type

of connection:

$$W_1 = [w_{11}^{\mathrm{ro}}], \qquad W_2 = [w_{11}^{\mathrm{o}}, w_{12}^{\mathrm{ro}}, \ldots, w_{1n_{\mathrm{h}}}^{\mathrm{o}}],$$

$$W_3 = \begin{bmatrix} w_{11}^{\mathrm{rh}} & w_{12}^{\mathrm{rh}} & \cdots & w_{1n_{\mathrm{h}}}^{\mathrm{rh}} \\ w_{21}^{\mathrm{rh}} & w_{22}^{\mathrm{rh}} & \cdots & w_{2n_{\mathrm{h}}}^{\mathrm{rh}} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n_{\mathrm{h}}1}^{\mathrm{rh}} & w_{n_{\mathrm{h}}2}^{\mathrm{rh}} & \cdots & w_{n_{\mathrm{h}}n_{\mathrm{h}}}^{\mathrm{rh}} \end{bmatrix},$$

$$W_4 = \begin{bmatrix} w_{11}^{\mathrm{h}} & w_{12}^{\mathrm{h}} & \cdots & w_{1n_i}^{\mathrm{h}} \\ w_{21}^{\mathrm{h}} & w_{22}^{\mathrm{rh}} & \cdots & w_{2n_i}^{\mathrm{h}} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n_{\mathrm{h}}1}^{\mathrm{h}} & w_{n_{\mathrm{h}}2}^{\mathrm{h}} & \cdots & w_{n_{\mathrm{h}}n_i}^{\mathrm{rh}} \end{bmatrix}, \qquad W_5 = [\theta_1^{\mathrm{o}}], \tag{13}$$

$$W_6 = [\theta_1^{\mathrm{h}}, \theta_2^{\mathrm{h}}, \ldots, \theta_{n_{\mathrm{h}}}^{\mathrm{h}}]$$

5. Inverse dynamic model identification

In this section, we present the design procedure of a recurrent neural network to learn the forward and inverse dynamics of the motor drive system as shown in Fig. 4. The proposed method requires the input–output data observation from the system. Therefore the system is considered as a black box and only the order of the system is assumed to be known. The order of the equivalent discrete time model is needed to design the architecture of the neural network, to select the minimum number of input/output nodes, and the required tapped-delay lines.

Referring to the system Eqs. (1)–(3), the nonlinear model of the system can be written in state space form as follows:

$$\dot{x} = f(u, x, t), \qquad y = g(u, x, t) \tag{14}$$

where $x = [\, i_a \quad \omega_{\mathrm{m}} \quad \omega_{\mathrm{l}} \quad \theta \,]$ is the state vector, and $y = \omega_{\mathrm{l}}$ is the measured speed.

This represents the equations of a fourth-order system. If the system is assumed to be rigid, i.e. the stiffness of the gear and couplings are assumed to be relatively high, the system can be then assumed to be equivalent to that of a second-order system.

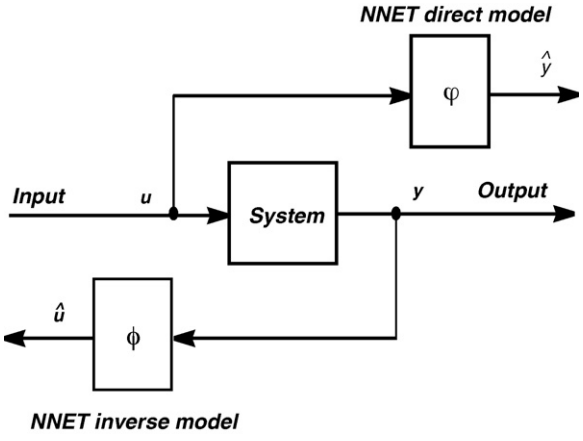

Fig. 4. NNET modeling of the system.

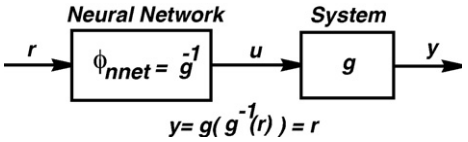

Fig. 5. Direct control using the inverse dynamic model.

The following model is therefore assumed for the NNET direct model:

$$\hat{y}(k+1) = \varphi[\hat{y}(k), \hat{y}(k-1), u(k), u(k-1)] \tag{15}$$

The NNET inverse model is derived from Eq. (15) to be

$$\hat{u}(k) = \phi[y(k+1), y(k), y(k-1), \hat{u}(k-1)] \tag{16}$$

where k is the discrete-time index. $u(k)$ and $y(k)$ denote, respectively, the system input (τ_{m}), and output (ω_{l}). $\varphi(\cdot)$ and $\phi(\cdot)$ are nonlinear maps assumed to be differentiable functions.

Consider now a RMLP with a four dimension input vector x_{c}, a single output u and an input–output relationship represented by

$$\hat{u}(k) = \phi_{\mathrm{nnet}}(x_{\mathrm{c}}) \tag{17}$$

where $\phi_{\mathrm{nnet}}(\cdot)$ denotes the input–output mapping of the neural network and $x_{\mathrm{c}}(k) = [y(k+1), y(k), y(k-1), u(k-1)]$. If $\hat{u}(k)$ approximates $u(k)$ for the corresponding inputs, RMLP can be thought of as an inverse model of the system. The equivalent inverse model can be then used as a direct controller as shown in Fig. 5.

Fig. 6 shows the off-line training scheme of the RMLP. At time k, the RMLP input vector is formed from the current and previous input–output samples. The network is updated in order to minimize the error function defined by

$$E = \frac{1}{2}\sum_{k=1}^{N}(u(k) - \hat{u}(k))^2 \tag{18}$$

where N is the number of input–output patterns.

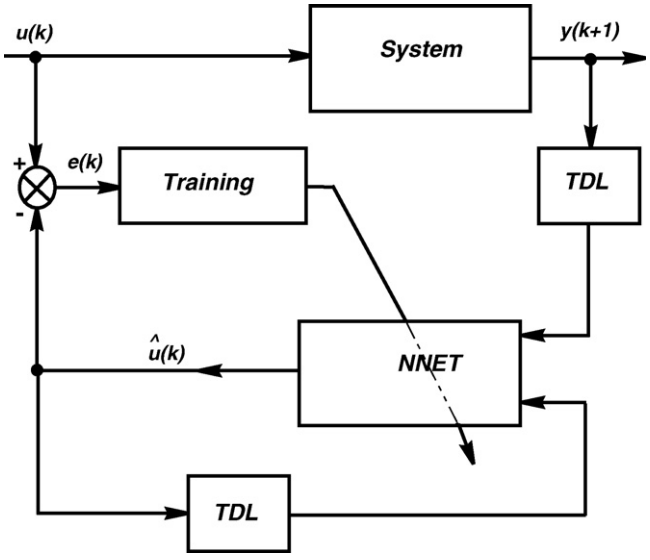

Fig. 6. Off-line learning of the inverse dynamic model.

The training data set is used cyclically by the LDEKF algorithm and the RMLP training is carried out until the output error e is kept small below a certain desirable level.

The weight/bias updating rules of the RMLP are shown as follows:

$$W_i(k) = W_i(k-1) + \Delta W_i(k) \tag{19}$$

$$\Delta W_i(k) = \alpha K_i(k)(u(k) - \hat{u}(k)) \tag{20}$$

$$H_i(k) = \frac{\mathrm{d}\hat{u}(k)}{\mathrm{d}W_i} \tag{21}$$

6. Adaptive neuro-control system

In this section, we propose a model-following adaptive control structure based on a recurrent multilayer perceptron (RMLP) that is used to compensate for the parameters variation and nonlinear effects on the system performance. Linear compensation may be effective in correcting some of the deficiencies introduced by these problems, but in other cases, performance is best improved by judicious use of adaptive compensation.

The proposed closed-loop speed control system is based only on one RMLP type recurrent neural network as shown in Fig. 7. The RMLP is first trained off-line to learn the inverse dynamics of the system from the observation of the input–output data. The model-following adaptive control approach is performed after the training process is achieved. The LDEKF approach is used to adjust the RMLP weights so that the neural model output follows the desired one.

The closed-loop system is described by the following equations:

$$e(k) = r_{\mathrm{m}}(k) - y(k) \tag{22}$$

$$u(k) = h[e(k), u(k), y(k)] \tag{23}$$

$$y(k) = g[u(k)] \tag{24}$$

where $r_{\mathrm{m}}(k)$ is the reference model signal, $e(k)$ the error signal, $u(k)$ the control signal applied to the input of the system, $h(\cdot)$ the inverse dynamic model function, $g(\cdot)$ the nonlinear system direct model, and $y(k)$ is the system output. The RMLP consists

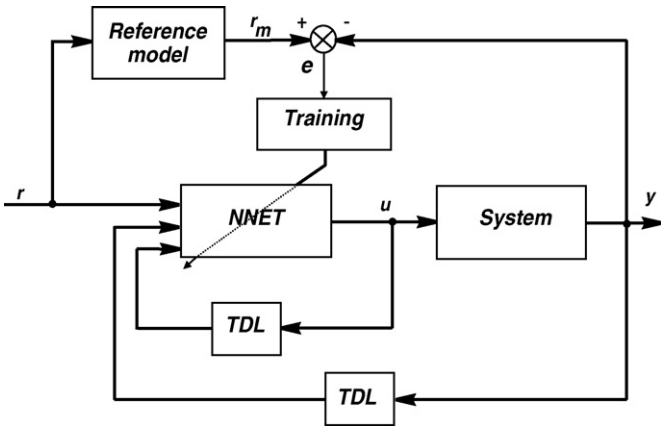

Fig. 7. Model-following adaptive control of the motor drive system.

of an input layer with four neurons, one hidden layer with sigmoidal neurons and a linear output layer with one neuron as shown in Fig. 3. The LDEKF algorithm is used to adjust the weights of the network. The initial values of the weights are those calculated by prior off-line learning of the inverse dynamics of the system. Consequently the RMLP learns continuously on-line and performs the adaptive speed control of the system.

Starting from the instant k, the different steps for training the RMLP controller with the LDEKF algorithm can be described as follows:

- *Step 1*. Decouple the weights in g groups: for the considered RMLP, the weights are decoupled in six groups.
- *Step 2*. Initialization: the weights are initialized by the values obtained during the prior off-line learning phase and the covariance matrix elements are set to random values.
- *Step 3*. Present the vector input and specify the desired output.
- *Step 4*. Calculate the output of the RMLP: this output represents the control signal of the system, obtained by propagating the vector input through the network.
- *Step 5*. Measure the output of the system.
- *Step 6*. Adapt the weights: the weights are updated in order to minimize the error function $e(k) = (r_{\mathrm{m}}(k) - y(k))$ using the LDEKF algorithm. For each group of weights, first we compute the elements of $H_i(k)$, which are used to determine $G(k)$ and $K_i(k)$. Note that to find the dynamic derivation of error with respect to the weights of the network, we need the knowledge of the system Jacobian [1].

The system Jacobian consists in calculating the derivative of the system output with respect to the input by using the assumed discrete direct and inverse models:

$$y(k+1) \simeq \hat{y}(k+1) = g[\hat{y}(k), \hat{y}(k-1), u(k), u(k-1)] \tag{25}$$

and

$$u(k) = h[r(k+1), y(k), y(k-1), u(k-1), W_i] \tag{26}$$

The derivative of the output with respect to a network weight W_i is obtained using the chain rules:

$$\frac{\mathrm{d}y(k+1)}{\mathrm{d}W_i} = \sum_{j=0}^{1}\left(\frac{\partial y(k+1)}{\partial y(k-j)}\frac{\mathrm{d}y(k-j)}{\mathrm{d}W_i}\right) + \sum_{j=0}^{1}\left(\frac{\partial y(k+1)}{\partial u(k-j)}\frac{\mathrm{d}u(k-j)}{\mathrm{d}W_i}\right) \tag{27}$$

$$\frac{\mathrm{d}u(k)}{\mathrm{d}W_i} = \frac{\partial h(k)}{\partial W_i} + \sum_{j=1}^{1}\left(\frac{\partial u(k)}{\partial u(k-j)}\frac{\mathrm{d}u(k-j)}{\mathrm{d}W_i}\right) + \sum_{j=0}^{1}\left(\frac{\partial u(k)}{\partial y(k-j)}\frac{\mathrm{d}y(k-j)}{\mathrm{d}W_i}\right) \tag{28}$$

Since the RMLP uses the delayed signals of u and y, the terms $\partial u(k)/\partial u(k-j)$ and $\partial u(k)/\partial y(k-j)$ will be computed through the RMLP. The $\mathrm{d}y(k-j)/\mathrm{d}W_i$ and $\mathrm{d}u(k-j)/\mathrm{d}W_i$ are, respectively, the delayed values of $\mathrm{d}y(k)/\mathrm{d}W_i$ and $\mathrm{d}u(k)/\mathrm{d}W_i$,

and are therefore computed recursively via Eqs. (27) and (28). Finally, $\partial y(k+1)/\partial u(k-j)$ and $\partial y(k+1)/\partial y(k-j)$ can be approximated, respectively, by the following equations:

$$\frac{\partial y(k+1)}{\partial u(k-j)} = \frac{\partial y(k+1)}{\partial u(k)} \frac{\partial u(k)}{\partial u(k-j)} \tag{29}$$

$$\frac{\partial y(k+1)}{\partial y(k-j)} = \frac{\partial y(k+1)}{\partial u(k)} \frac{\partial u(k)}{\partial y(k-j)} \tag{30}$$

The Jacobian terms $\partial y(k+1)/\partial u(k)$ are computed by using the following numerical approximation:

$$\frac{\partial y(k+1)}{\partial u(k)} \simeq \frac{y(k+1)-y(k)}{u(k)-u(k-1)} \tag{31}$$

In summary, the main features of the proposed control scheme are as follows:

- Proposed a new neural adaptive control structure using only one recurrent neural network and appropriate time delays (TDL) for system input/output feedback. The neural network represents the inverse dynamic model of the system.
- Systematic procedure to determine the system Jacobians needed to update on-line the RNN weights.
- Proposed control structure is very robust to external disturbance as validated by the experimental results.
- The RNN weights are updated by the layered decoupled extended Kalman filter (LDEKF) algorithm witch is a proposed modified version of the decoupled extended Kalman filter algorithm.

7. Motor drive setup and experimental validation

In this section, we consider the practical implementation and validation of the neural network-based control system. The motor drive test setup considered for our analysis is an electro-mechanical system composed of a dc brush-type motor with a gear assembly module, an inertial load, a linear power amplifier module and a personal computer. It is represented schematically in Fig. 8. A data acquisition card mounted in the PC is interfaced with the assembly module to acquire the speed information using a tachometer mounted on the load side. The data acquisition and control in the PC is accomplished using C++ programming.

The main C program is developed to have an interactive user interface to set the desired speed and to store and display the system variables for monitoring. The control routine is dedicated to acquire the data from the speed sensor, implement the neural network, adapt its weights, and output the control signal to the motor. The program is designed as a software driven routine that at each sampling instant (1 ms in this work) performs the data acquisition and control. The motor speed may be observed in real-time through the oscilloscope to analyze the effectiveness of the neuro-control system.

For comparative analysis, a proportional controller and a PI controller are first designed to control the speed of the dc motor drive system. Fig. 9 shows the speed response of the system along with the corresponding control signal for a sinusoidal speed reference. The controller gains where optimized to minimize the error and avoid saturation of the motor input

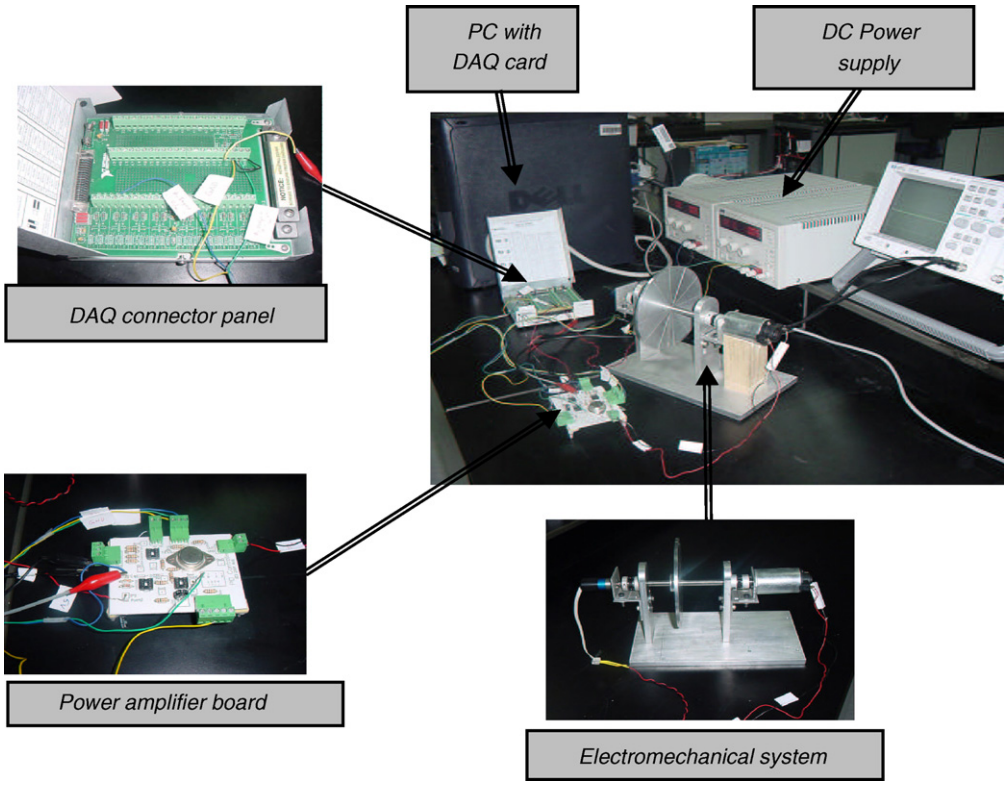

Fig. 8. View of the experimental setup.

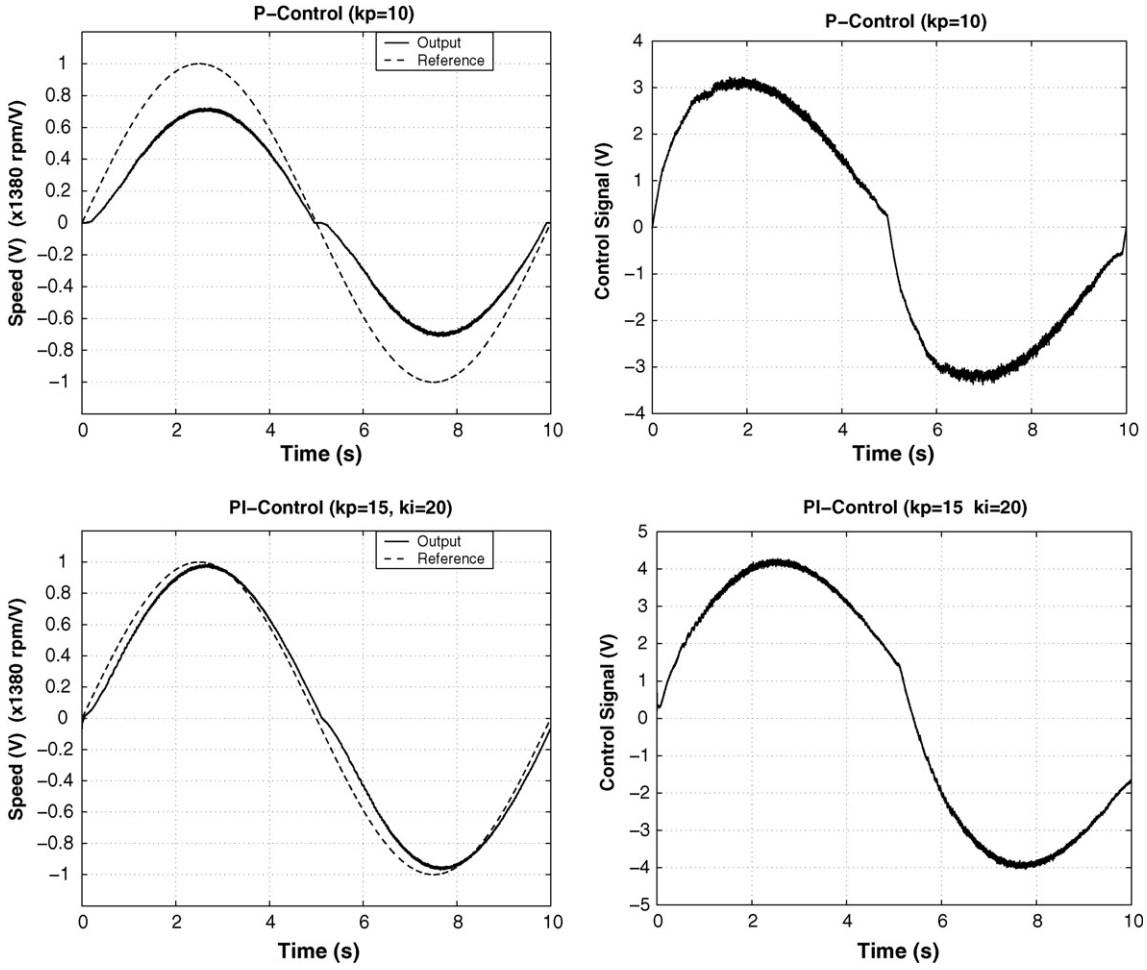

Fig. 9. Speed control with a P and PI controller.

voltage. For both cases, we can see that the friction effect around zero speed causes a distortion of the sine wave. Both controllers exhibit a steady state error. While this is expected from the proportional controller, the PI controller is also unable to compensate the friction nonlinearity.

A recurrent neural network is next designed to learn the inverse dynamics of the motor drive system. The training process is carried out for varying numbers of hidden neurons ranging from 6 to 20. The optimum RMLP consists of an input layer with four neurons, one hidden layer with eight sigmoidal neurons and an output layer with one neuron. Fig. 10 shows the input/output pattern used for training the RMLP.

The weights of the RMLP are decoupled into six groups ($g = 6$), and the initial conditions are specified for each group

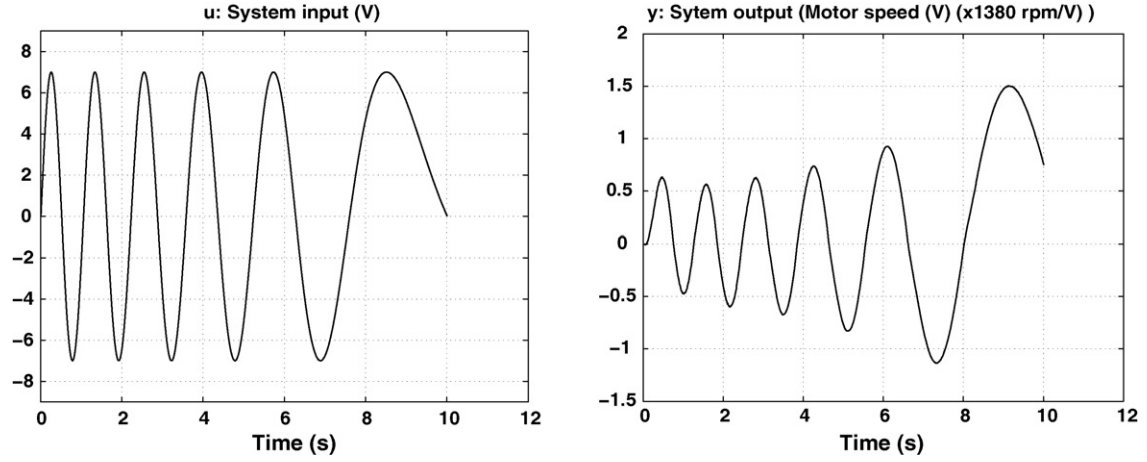

Fig. 10. Training data-pattern.

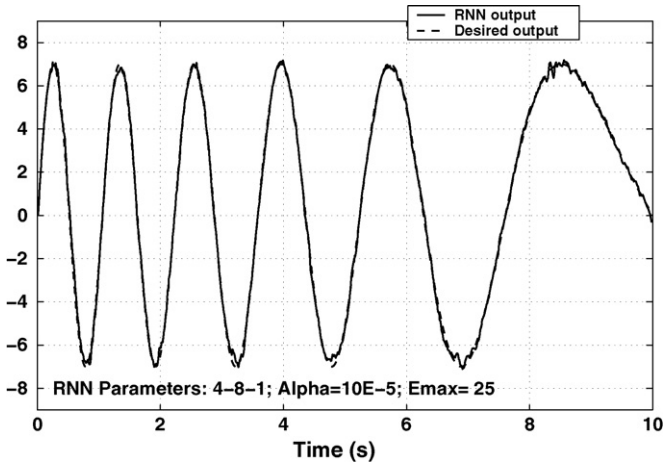

Fig. 11. RMLP output after training.

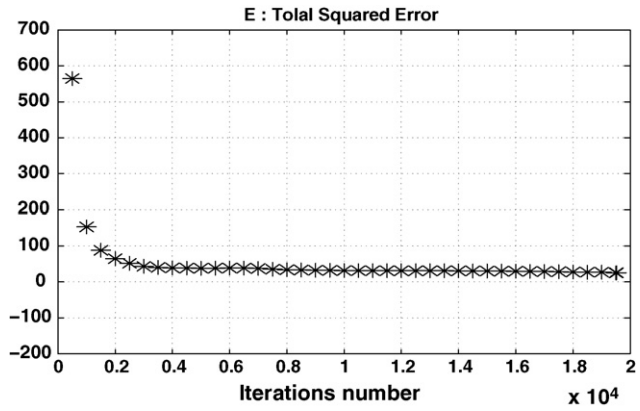

Fig. 12. Total squared error during training.

of weights. In the absence of any a priori information, the initial weight vector can be set randomly, and the P_i matrix can be set to some multiple of the identity matrix. At time step $k = 0$ the weight vector $W_i(0)$ and the covariance matrix $P_i(0)$ are initialized by random values in the range $[-10^{-3}, 10^{-3}]$ and $[10^2, 10^4]$, respectively. During training, the components of the artificial process noise $Q_i(k)$ were chosen to be random values in the range $[10^{-4}, 10^{-6}]$. The optimum learning rate used is $\alpha = 10^{-5}$.

Fig. 11 shows the RMLP output after good convergence is achieved. The graph in Fig. 12 shows the total squared error per trial as a function of iteration number. Each simulation cycle for t ranging from 0 to 10 s is called a trial.

After the training phase, the RMLP is used as a controller in the feed-forward adaptive control loop as illustrated in Fig. 7.

Fig. 13 shows the output of the dynamic system with direct adaptive neuro-control using the same reference signal $y(t)$ used in the open-loop case. The absence of the sine wave distortion around zero speed indicates the complete compensation of the nonlinearities and the good tracking performance of the system.

8. Performance analysis

To assess the performance of the proposed neuro-control scheme compared with the conventional P and PI controllers, three performance indices are used:

1. the mean of squares of errors (MSE) and the signal-to-noise ratio (SNR)

$$\mathrm{MSE} = \frac{1}{N}\sum_{k=1}^{N}(r(k) - y(k))^2, \tag{32}$$

$$\mathrm{SNR} = 10\log_{10}\left(\frac{\mathrm{MSE}}{\mathrm{MSE}_{\mathrm{ref}}}\right) \tag{33}$$

where $\mathrm{MSE}_{\mathrm{ref}}$ is the mean of squares of errors of the reference signal.

2. the power spectral density of the error signal (PSD), and
3. the sum of the modulus of the control signal (SMC):

$$\mathrm{SMC} = \sum_{k=1}^{N}|u(k)| \tag{34}$$

Fig. 14 shows the error signal $e = r - y$ and its PSD for the three controllers P, PI and RNN. The PSD shows the intensity of

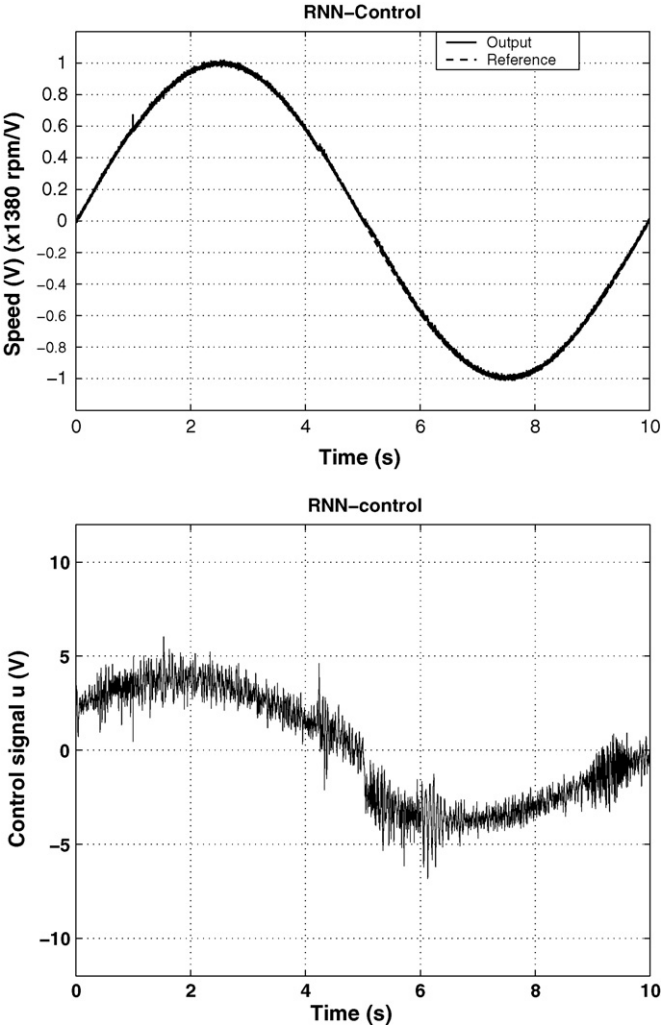

Fig. 13. Output of the neuro-control system using a sinusodal reference.

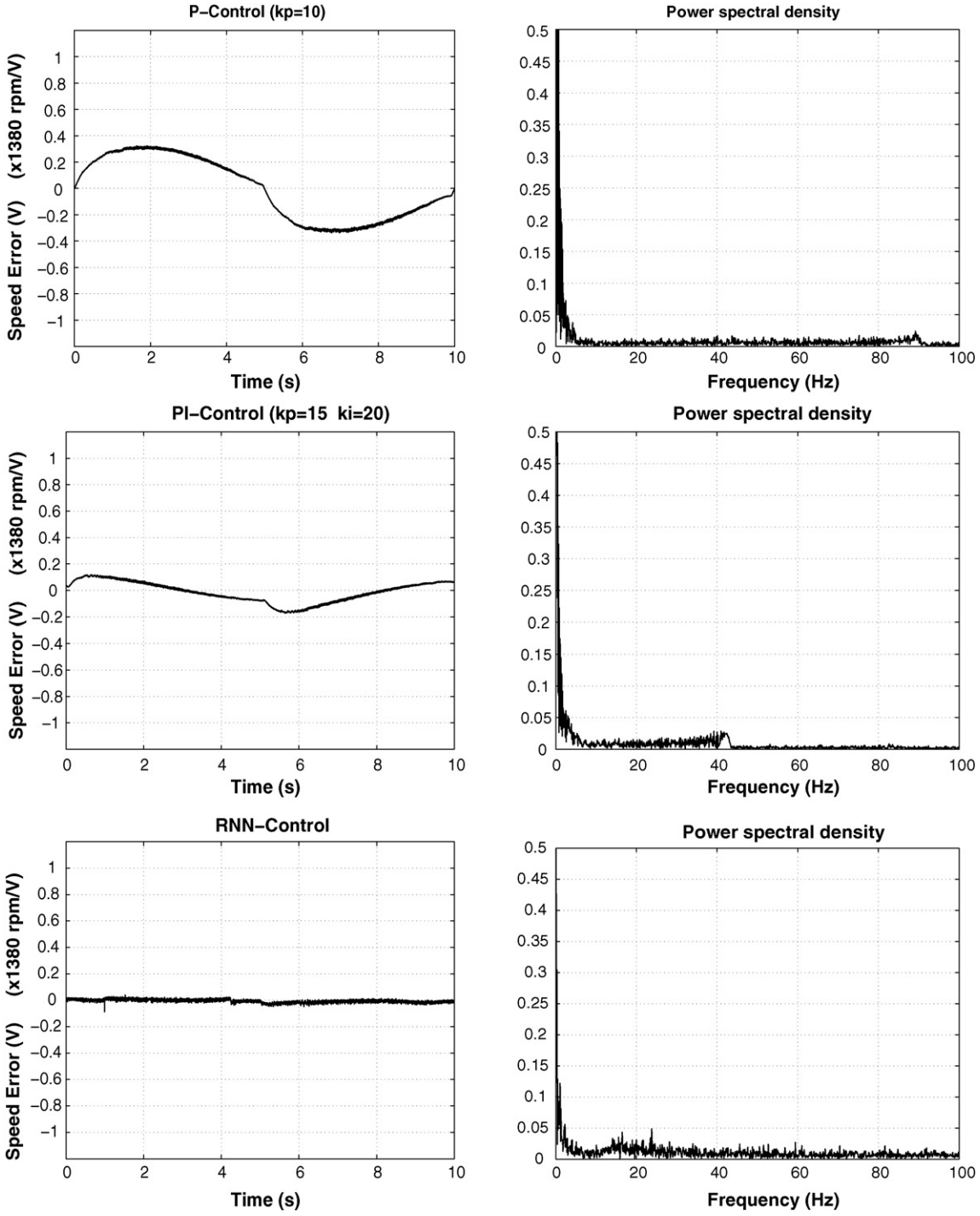

Fig. 14. Error speed signal and its PSD of P, PI and RNN controller.

measurement noise as well as the level of harmonics in the speed signal due to the nonlinearity. The high frequency signal represents for all cases the measurement noise introduced by the speed sensor. Table 1 shows the magnitude of the PSD for the first five harmonics. It can be observed that the P and PI controllers show high values for the odd harmonics (1st, 3rd, 5th,. . .). The RNN controller shows a much lower magnitude at all frequencies. Table 2 shows the MSE and (SNR) for each signal. It can be observed that the RNN controller resulted in an increase of 25.4 dB SNR compared to the P controller and 15.9 dB SNR compared to the PI controller.

To assess the efficiency of each controller, Table 3 shows the SMC for the three schemes for each case. Table 4 presents the RMS value of the speed signal. The results confirm again the efficiency of the neuro-control strategy which requires less control effort while compensating effectively the nonlinearity.

The success of this control method depend largely on the ability of the RMLP to generalize or learn to respond to inputs

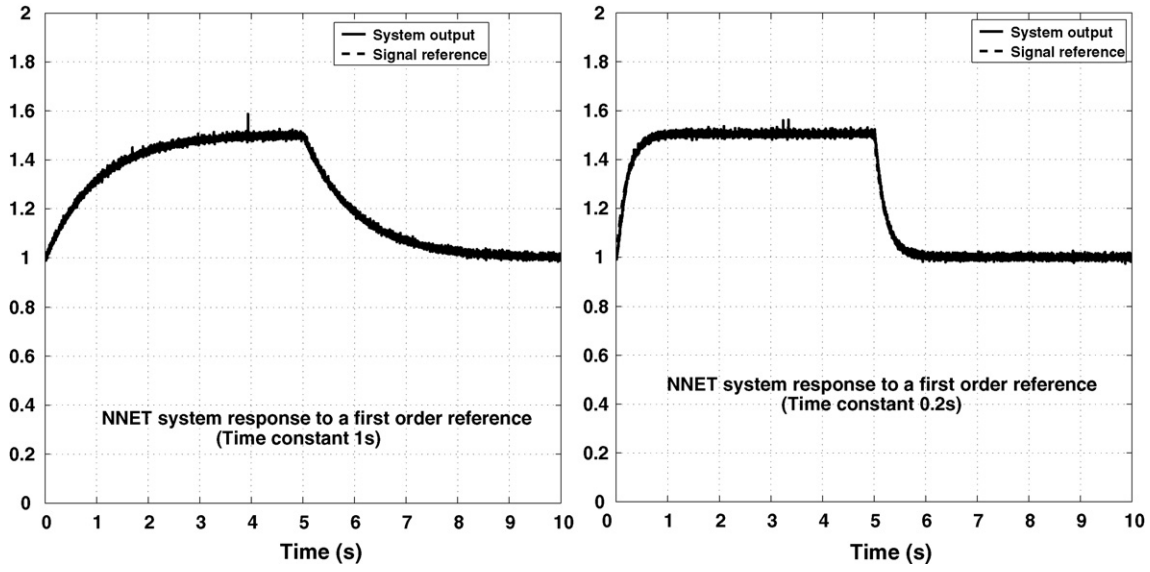

Fig. 15. Step response of neuro-control system.

that were not specifically used in the learning phase. In Fig. 15, we have illustrated the system response to a first-order reference signal, with the RMLP controller initialized by the weights acquired by prior off-line learning using the chirp signal. The results show that the system follows very closely the reference. The figure shows also the same type of response for a different reference model with a different time constant. These results have been obtained by letting the system run for a long period of time (more than 10 min) before saving the speed signal. This demonstrates the stability of the control algorithm.

Table 1
Power spectral density (PSD) of the speed-error

Frequency (Hz)	0	0.1	0.2	0.3	0.4	0.5
PSD ($error_P$)	0.5165	16.4865	0.0226	2.0217	0.0523	1.0023
PSD ($error_{PI}$)	1.1520	5.3846	0.5488	0.8514	0.2384	0.4825
PSD ($error_{RNN}$)	0.3543	0.5235	0.1950	0.3042	0.0240	0.1283

Table 2
The MSE and SNR for reference, P, PI and RNN signal

	MSE	SNR (dB)
Reference	0.5	0
P	0.0556	9.5428
PI	0.0063	19.0196
RNN	1.6009E−4	34.9461

Table 3
The SMC value of the control signal for the three controllers, P, PI and RNN

	SMC
P	2.1771E+004
PI	2.9052E+004
RNN	2.6822E+004

Table 4
The RMS value of the speed signal

	RMS
Reference	0.7071
P	0.4810
PI	0.6758
RNN	0.7007

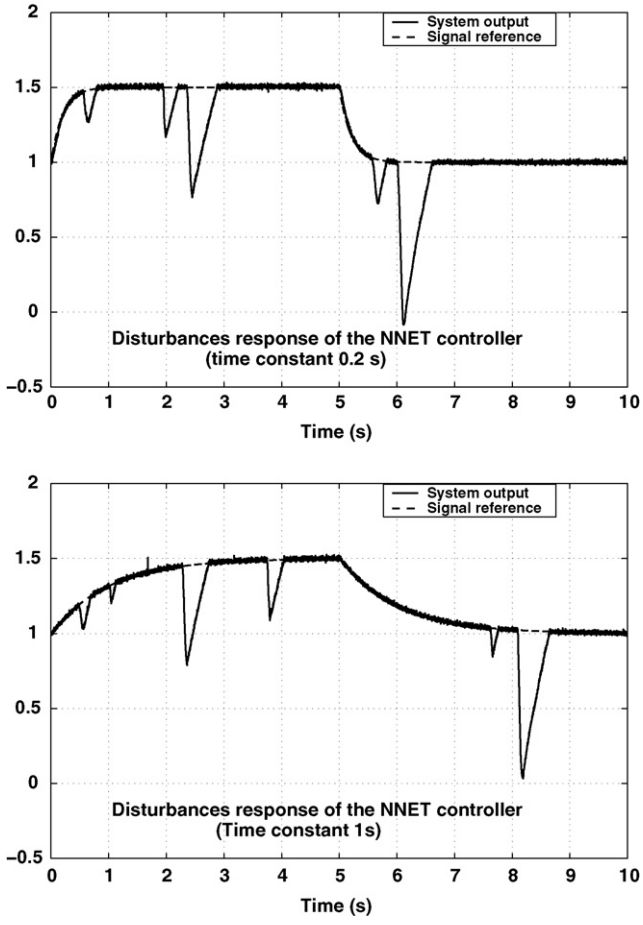

Fig. 16. Repetitive disturbance response of the neuro-control system.

To test the robustness of the system, random disturbances where applied to the system by sudden braking of the load inertia wheel. The disturbance response is illustrated in Fig. 16 for repetitive disturbances with different magnitudes. The system exhibits an excellent disturbance rejection property. The system output recovers quickly from the disturbance and follows the reference with good tracking performance.

9. Conclusion

In this paper, a model-following adaptive control method is developed for the speed control of a nonlinear motor drive system using recurrent neural networks. The neural network is first trained off-line to learn the inverse dynamics of the motor drive system using a modified form of the decoupled extended Kalman filter algorithm. The proposed neuro-control scheme is validated experimentally on a dc motor setup with highly nonlinear friction. The experimental results obtained showed the effectiveness of the recurrent network structure and its adaptation algorithm. The inverse dynamics approach combined with the recurrent neural network structure and its adaptation algorithm demonstrated a very good disturbance rejection and tracking performance.

References

[1] G.L. Plett, Adaptive inverse control of linear and nonlinear systems using dynamic neural networks, IEEE Trans. Neural Netw. 14 (2) (2003) 360–376.
[2] J.O. Jang, G.J. Jeon, A parallel neuro-controller for dc motors containing nonlinear friction, Neurocomputing 30 (2000) 233–248.
[3] X. Ouyang, C. Morgan, C. Nwagboso, Model-following control of nonlinear servo systems using neural networks, Simul. Tech. Article (2001) 263–272.
[4] X.M. Ren, A.B. Rad, P.T. Chan, W. Lun Lo, Identification and control of continuous-time nonlinear nystems via dynamic neural networks, IEEE Trans. Ind. Electron. 50 (3) (2003) 478–786.
[5] T.W.S. Chow, Y. Fang, A recurrent neural-network-based real-time learning control strategy applied to nonlinear systems with unknown dynamics, IEEE Trans. Ind. Electron. 45 (1) (1998) 151–161.
[6] K.C. Sio, C.K. Lee, Identification of a non-linear system with neural networks, in: Proceedings of the Advanced Motion Control'96, MIE, Japan, (1996), pp. 287–292.
[7] G.V. Puskorious, L.A. Feldkamp, Neuro-control of non linear dynamical systems with Kalman filter trained recurrent networks, IEEE Trans. Neural Netw. 5 (2) (1994) 279–297.
[8] S. Kumpati, F. Narendra, K. Parthasarathy, Identification and control of dynamical systems using neural networks, IEEE Trans. Neural Netw. 1 (1) (1990) 4–24.
[9] S. Kumpati, F. Narendra, K. Parthasarathy, Gradient methods for the optimisation of dynamical systems containing neural networks, IEEE Trans. Neural Netw. 2 (2) (1991) 4–24.
[10] S. Singhal, L. Wu, Training multi-layer perceptions with the extended Kalman algorithm, in: Proceedings of the Advances in Neural Information Proceedings Systems, vol. 1, Denver, (1988), pp. 133–140.
[11] D.W. Puck, S.K. Rogers, M. Kabrisky, P.S. Maybeck, M.E. Oxley, Comparative analysis of back-propagation and the extended Kalman filter for training multi-layers perceptrons, IEEE Trans. Pattern Anal. Mach. Intell. 14 (6) (1992) 686–691.
[12] L. Shah, F. Palmieri, MEKA-a fast, local algorithm for training feedforward neural networks, in: Proceedings of the International Joint Conference on Neural Networks, vol. III, San Dieago, (1990), pp. 41–45.
[13] R. Dhaouadi, K. Nouri, Neural network-based speed control of two-mass model system, J. Adv. Computat. Intell. 3 (5) (1999) 427–430.
[14] R. Dhaouadi, K. Nouri, Feedforward control of an elastic drive system using neural networks, Colloque Maghrebin sur les Modeles Numeriquesed l'Ingenieur (C2MMNI6) 12 (1) (1998) 187–192.
[15] K. Nouri, R. Dhaouadi, N. Benhadj Braiek, Adapting back-propagation algorithm for training recurrent multilayers neural networks, in: Proceedings of the Smart Systems and Devices, Hammamet, Tunisia, March, (2001), pp. 363–368.
[16] G.V. Puskorius, L.A. Feldkamp, Decoupled extended Kalman filter training of feed-forward layered neural networks, in: Proceedings of the IEEE International Joint Conference on Neural Networks, vol. 1, New York, (1991), pp. 771–777.
[17] K. Nouri, R. Dhaouadi, N. Benhadj Braiek, Identification of a nonlinear dynamic system using recurrent multilayers neural networks, in: Proceedings of the IEEE-SMC International Conference, Hammamet, Tunisia, March, 2002.
[18] G.V. Puskorious, L.A. FeldKamp, Model reference adaptive control with recurrent networks trained by the dynamic DEKF algorithm, in: Proceedings of the IEEE International Joint Conference on Neural Network, vol. II, New York, (1992), pp. 106–113.
[19] A.G. Parlos, S.K. Menon, A.F. Atiya, An algorithmic approach to adaptive state filtering using recurrent neural networks, IEEE Trans. Neural Netw. 12 (6) (2001) 1411–1432.
[20] G.V. Puskorious, L.A. Feldkamp, Recurrent network training with the decoupled extended Kalman filter algorithm, Proc. Sci. Artif. Neural Netw. 1710 (1992) 461–473 (SPIE).
[21] G.V. Puskorious, L.A. Feldkamp, Practical considerations for Kalman filter training of recurrent neural networks, in: Proceedings of the IEEE International Conference on Neural Networks, San Francisco, 1993.
[22] R.J. Williams, D. Ziper, A learning algorithm for continually running fully recurrent neural networks, Neural computat. 1 (1989) 270–280.
[23] R.J. Williams, J. Peng, An efficient gradient algorithm for on-line training of recurrent network trajectories, Neural Computat. 2 (1990) 490–501.
[24] K. Nouri, R. Dhaouadi, N. Benhadj Braiek, A comparative study of RTRL algorithm and DEKF approach for training recurrent multilayer perceptron, in: Proceedings of the International Arab Conference on Information Technology, ACIT 02, Doha, Qatar, (2002), pp. 643–648.
[25] B.A. Helouvry, P. Dupont, C. Canuda, A survey of models, analysis tools and compensation methods for the control of machines with friction, Automatica 30 (7) (1994) 1083–1138.