Dynamic Neural Controllers for Induction Motor

Mietek A. Brdyś, *Member, IEEE,* and Grzegorz J. Kulawski

***Abstract*—The paper reports application of recently developed adaptive control techniques based on neural networks to the induction motor control. This case study represents one of the more difficult control problems due to the complex, nonlinear, and time-varying dynamics of the motor and unavailability of full-state measurements. A partial solution is first presented based on a single input–single output (SISO) algorithm employing static multilayer perceptron (MLP) networks. A novel technique is subsequently described which is based on a recurrent neural network employed as a dynamical model of the plant. Recent stability results for this algorithm are reported. The technique is applied to multiinput–multioutput (MIMO) control of the motor. A simulation study of both methods is presented. It is argued that appropriately structured recurrent neural networks can provide conveniently parameterized dynamic models for many nonlinear systems for use in adaptive control.**

***Index Terms*— Dynamic neural networks, nonlinear adaptive control, learning, induction motor.**

I. Introduction

INDUCTION motor control is a difficult engineering problem, which is still not completely resolved. This is due to highly nonlinear dynamics of the machine and unavailability of measurements of some of the state variables and usually also of some of the controlled outputs, in a typical hardware configuration. In addition to that, some of the machine parameters, mainly the rotor resistance, exhibit strong variations due to changing thermal conditions. A description of the induction motor dynamics is given in Section II. Many different approaches for control of the motor have been proposed. Two schemes, which are able to provide high dynamic performance of the system, are the field oriented control, first proposed in [1], and the input–output linearizing control introduced in [2], an adaptive version of which has been recently presented in [3]. Both of them, however, require measurements of the rotor flux vector, which is typically not available.

In this paper, application to the induction motor control of two recently developed adaptive control techniques utilizing neural networks is presented. The first, single input–single output (SISO), controller employs static multilayer perceptron (MLP) networks, while the second, multiinput–multioutput (MIMO), approach is based on a recurrent network used as a dynamic model of the controlled system.

From the point of view of algorithm design, universal approximation properties of static networks, shown for example in [4]–[6], make them a potentially useful tool for nonlinear control and identification problems. In addition to that, their massively parallel structure combined with simplicity of single neurons brings benefits related to implementation issues such as speed of computations and fault tolerance due to natural redundancy. Realization of these facts sparked recently a great deal of research activity in the control community for which the early paper [7] takes much of the credit. A number of different control structures utilizing neural networks have been subsequently proposed, while MLP and radial basis function (RBF) networks became the most popular neural architectures used.

After the early enthusiasm and wealth of ideas, the challenge was to put neural networks more firmly into the control engineering context and rigorously address such essential issues as guarantees of performance, robustness or stability. Control algorithms, in which neural networks are trained off-line prior to their application in the on-line operation, when no further weights adjustments are made, are relatively simpler from the viewpoint of mathematical analysis. This approach was for example taken in two papers [8], [9], where a number of control methods using MLP trained off-line are presented and supported by solid mathematical treatment.

Design of provably stable adaptive neural controllers posed more difficult theoretical questions. One of the early adaptive schemes with local stability results has been presented in [10]. A globally stable nonlinear adaptive controller using RBF networks was proposed in [11]. The fact that an output of a RBF network is linear in the adjustable parameters proved important for the design of a stable parameter update law. Finally, global stability has also been shown for an adaptive controller using a single hidden layer MLP network, in which both output and hidden weights are tuned, first in [12] and later further developed in [13]. These developments are very much based on results from the theory of adaptive control of linear plants (see for example [14]). In case of linearly parameterized RBF networks the existing adaptive schemes could be applied in a relatively straightforward manner, while nonlinear parameterization of MLP networks was a significant obstacle. It was then realized that, due to the specific properties of sigmoid activation functions, control laws, with approximations of unknown functions realized by MLP networks, can be fit into a framework which allows use of methods from adaptive control for linear systems.

In Section II a SISO control scheme for the induction motor is presented together with a simulation study. The adaptive neural algorithm it employs is based on [10].

Manuscript received November 20, 1997; revised July 16, 1998 and November 30, 1998.

M. A. Brdyś is with the School of Electronic and Electrical Engineering, University of Birmingham, Edgbaston, Birmingham B15 2TT, U.K.

G. J. Kulawski is with Shell International Exploration and Production B.V., Research and Technical Services, Volmerlaan 8, 2280 AB Rijswijk, The Netherlands.

Publisher Item Identifier S 1045-9227(99)02036-6.

While, as briefly described above, significant progress has been achieved in applications of static neural networks for control, these algorithms are limited to the case when full state of the controlled plant is available for measurement. The output feedback control (when the full state is not measurable) and especially adaptive output feedback control for nonlinear systems remains one of the outstanding problems which are, at present, very actively researched. Systematic design techniques have only been developed for specific classes of nonlinear systems, e.g., [15] and [16].

Lack of state measurements makes the problem particularly difficult, as some kind of dynamic model of the system needs to be used for control law synthesis. It appears that recurrent neural networks can offer such models for use in nonlinear adaptive output feedback control. Although some work along these lines has been done, until very recently most of the schemes were of rather heuristic nature without comprehensive convergence analysis. A control technique based on a continuous time dynamic network which is used as a state-space model of the unknown object was described in [17] and two approaches to the stability analysis were later presented in [18] and [19]. This algorithm is described in Section IV and its application to MIMO control of the induction motor together with a simulation study is presented in Section V.

II. Induction Motor Model

A three-phase squirrel-cage induction motor is considered. u_a, u_b, and u_c are the three input voltages and i_a, i_b, and i_c are the respective currents. A few standard simplifying assumptions are made; the air gap between rotor and stator is uniform and constant, and saturation and hysteresis are neglected. It is further assumed that the stator supply neutral point is isolated so there can be no zero-sequence stator currents i.e. $i_a+i_b+i_c=0$ and $u_a+u_b+u_c=0$. In this case, the three-phase stator windings (sA, sB, sC) can be transformed, using the Parks transformation (e.g. [20]), into equivalent quadrature-phase windings (sD, sQ). The dynamics of the motor are then given by a fifth-order nonlinear differential model [3]

$$\frac{d\omega_r}{dt} = \frac{n_p M}{J L_r}(\psi_{rd} i_{sq} - \psi_{rq} i_{sd}) - \frac{t_L}{J} \tag{1}$$

$$\frac{di_{sd}}{dt} = \frac{M R_r}{\sigma L_s L_r^2}\psi_{rd} + \frac{n_p M}{\sigma L_s L_r}\omega_r \psi_{rq} - \frac{M^2 R_r + L_r^2 R_s}{\sigma L_s L_r^2} i_{sd} + \frac{1}{\sigma L_s} u_{sd} \tag{2}$$

$$\frac{di_{sq}}{dt} = -\frac{n_p M}{\sigma L_s L_r}\omega_r \psi_{rd} + \frac{M R_r}{\sigma L_s L_r^2}\psi_{rq} - \frac{M^2 R_r + L_r^2 R_s}{\sigma L_s L_r^2} i_{sq} + \frac{1}{\sigma L_s} u_{sq} \tag{3}$$

$$\frac{d\psi_{rd}}{dt} = -\frac{R_r}{L_r}\psi_{rd} - n_p \omega_r \psi_{rq} + \frac{R_r}{L_r} M i_{sd} \tag{4}$$

$$\frac{d\psi_{rq}}{dt} = n_p \omega_r \psi_{rd} - \frac{R_r}{L_r}\psi_{rq} + \frac{R_r}{L_r} M i_{sq} \tag{5}$$

complemented by a number of static relationships

$$i_{rd} = \frac{1}{L_r}(\psi_{rd} - M i_{sd}) \tag{6}$$

$$i_{rq} = \frac{1}{L_r}(\psi_{rq} - M i_{sq}) \tag{7}$$

$$\psi_{sd} = L_s i_{sd} + M i_{rd} \tag{8}$$

$$\psi_{sq} = L_s i_{sq} + M i_{rq} \tag{9}$$

where i, u, ψ denote current, voltage, and flux linkage, respectively. Subscripts r and s stand for rotor and stator. ω_r is the rotor speed. d and q denote ("direct" and "quadrature") components of the vectors with respect to the fixed stator reference frame (sD, sQ). L and R are the autoinductances and resistances, M is the mutual inductance and $\sigma = 1 - (M^2/L_s L_r)$. t_L is the load torque.

The machine is assumed to be supplied by a voltage-source inverter and thus the controller has to generate a required reference voltage for the inverter.

III. Adaptive Control of the Induction Motor Using Static Networks

The SISO control technique for the induction motor presented in this section utilizes the neural control algorithm of [10]. Therefore, this algorithm is first briefly described.

A. Algorithm (by Chen and Khalil [10])

In [10], the following discrete-time SISO plant with a general relative degree $d \geq 1$ is considered:

$$\begin{aligned} y(k+d) &= F(y(k), \cdots, y(k-n), u(k-1), \cdots, u(k-m)) \\ &\quad + G(y(k), \cdots, y(k-n), \\ &\quad u(k-1), \cdots, u(k-m))u(k) \end{aligned} \tag{10}$$

where y is the output, u is the input and F and G are unknown nonlinear functions of past inputs and outputs up to the latest available measurements $y(k)$ and $u(k-1)$, so that calculation of causal control is possible. It is assumed that $G(\cdot)$ is bounded away from zero and its sign is known.

If the two functions F and G were known, a feedback linearizing control input could be calculated as

$$u(k) = -\frac{F(\cdot)}{G(\cdot)} + \frac{y_{\text{ref}}(k+d)}{G(\cdot)}$$

to achieve perfect tracking of the desired reference trajectory y_{ref}. Since F and G are unknown, two static neural networks are employed to approximate them. MLP networks with hyperbolic tangent activations are used for this purpose. Measurements of past values of plant output and input, which are the arguments of F and G, are the inputs to the networks. The approximations of F and G realized by the networks are denoted by $\hat{F}(\cdot, \hat{\mathbf{w}})$ and $\hat{G}(\cdot, \hat{\mathbf{v}})$, respectively. $\hat{\mathbf{w}}$ and $\hat{\mathbf{v}}$ denote the weights vectors of the two networks. Control input to the plant is calculated, using the network outputs in place of the unknown functions F and G, as

$$u(k) = -\frac{\hat{F}(\cdot, \hat{\mathbf{w}}(k))}{\hat{G}(\cdot, \hat{\mathbf{v}}(k))} + \frac{y_{\text{ref}}(k+d)}{\hat{G}(\cdot, \hat{\mathbf{v}}(k))}.$$

The output estimate, that the neural model based on $\hat{F}$ and $\hat{G}$ and with the current values of weights would have produced, can be obtained as

$$\hat{y}(k+1) = \hat{F}(\cdot, \hat{\mathbf{w}}(k)) + \hat{G}(\cdot, \hat{\mathbf{v}}(k))u(k-d+1).$$

The current network output error $e(k+1)$ is then defined as

$$e(k+1) = \hat{y}(k+1) - y(k+1).$$

A full parameter vector is defined as $\hat{\mathbf{\Theta}} = [\hat{\mathbf{w}}^T, \hat{\mathbf{v}}^T]^T$. The updating rule is formulated as

$$\hat{\mathbf{\Theta}}(k+1) = \hat{\mathbf{\Theta}}(k) - \frac{1}{r(k)} e_\Delta(k+1)\nabla e(k+1)$$

where $\nabla e(k+1)$ denotes the gradient of $e(k+1)$ with respect to the full parameter vector $\hat{\mathbf{\Theta}}$

$$\nabla e(k+1) = \begin{bmatrix} \dfrac{\partial \hat{F}(\cdot, \hat{\mathbf{w}}(k))}{\partial \hat{\mathbf{w}}} \\ \dfrac{\partial \hat{G}(\cdot, \hat{\mathbf{v}}(k))}{\partial \hat{\mathbf{v}}} u(k-d+1) \end{bmatrix}. \qquad (11)$$

The learning rate is adjusted "automatically" according to

$$r(k) = 1 + \nabla^T e(k+1)\nabla e(k+1).$$

$e_\Delta(k+1)$ denotes the error $e(k+1)$ passed through a dead zone. Thus, adaptation of network parameters consists of a gradient descent with respect to $\frac{1}{2}e^2(k+1)$, modified by application of the dead zone.

A detailed stability proof of this scheme was published in [21]. Stability is local with respect to the initial parameter values $\hat{\mathbf{w}}(0)$ and $\hat{\mathbf{v}}(0)$, which have to be close enough to the "ideal" values $\mathbf{w}$ and $\mathbf{v}$. These "ideal" weights are defined, for the purpose of analysis, as those giving the smallest approximation error over a certain compact set. Tracking error convergence is shown there via the convergence of the parameter error, which is due to adaptation. Analysis of the parameter error convergence is essentially based on the local linearization of the error $e(k+1)$ as a function of weights. Errors resulting from the functional reconstruction errors and higher order terms of the Taylor expansion are treated as a "disturbance" to the local linear parameterization and dealt with using the dead zone. Because the dead zone needs to accommodate higher order terms, its required size is proportionally related to the initial parameter mismatch $||\hat{\mathbf{\Theta}}(0) - \mathbf{\Theta}||$, which is unfortunately quite conservative.

B. Stator-Oriented Control Scheme

A typical technique for control synthesis purposes is to use a description of the machine in rotating reference frames (x, y), most commonly fixed either to rotor flux vector $\boldsymbol{\psi}_r$ or stator flux vector $\boldsymbol{\psi}_s$, instead of the stationary one (sD, sQ). Use of such rotating reference frames has the benefit of simplifying the model of the motor from the point of view of controller design. Different rotating frames are described in detail in [20]. In this section, a control algorithm for the induction motor, utilizing the stator-oriented reference frame, will be derived.

In the stationary reference frame (sD, sQ), the stator flux linkage phasor, described in complex notation as

$$\boldsymbol{\psi}_s = \psi_{sd} + j\psi_{sq} = |\boldsymbol{\psi}_s|e^{j\rho_s}$$

is rotating at a speed

$$\omega_{ms} = \frac{d\rho_s}{dt}$$

where ρ_s denotes its angle.

The rotating frame used here is fixed to the stator flux phasor $\boldsymbol{\psi}_s$ having its real axis (X) coaxial with this vector, such that $\psi_{sx} \equiv |\boldsymbol{\psi}_s|$ and $\psi_{sy} \equiv 0$. Stator terminal voltages in the rotating frame, u_{sx} and u_{sy}, can be calculated from the voltages in the stationary frame, u_{sd} and u_{sq}, using a simple transformation

$$u_{sx} + ju_{sy} = (u_{sd} + ju_{sq})e^{-j\rho_s}$$

which gives

$$\begin{bmatrix} u_{sx} \\ u_{sy} \end{bmatrix} = \begin{bmatrix} \cos\rho_s & \sin\rho_s \\ -\sin\rho_s & \cos\rho_s \end{bmatrix} \begin{bmatrix} u_{sd} \\ u_{sq} \end{bmatrix}.$$

A reverse transformation is quite straightforward. Calculation of i_{sx}, i_{sy} from i_{sd}, i_{sq} and in the reverse direction is performed similarly.

The electromagnetic torque produced by the machine is proportional to the magnitude of the stator flux linkage phasor and quadrature-axis component of the stator currents phasor

$$t_e = \frac{3}{2}n_p|\boldsymbol{\psi}_s|i_{sy} = \frac{3}{2}n_p\psi_{sx}i_{sy}.$$

In the proposed architecture, the first control objective is to maintain the stator flux magnitude $|\boldsymbol{\psi}_s|$ at a prescribed level which ensures high efficiency of motor operation and yet does not allow magnetic saturation. Varying load torque is compensated by changes in i_{sy}, which is drawn by the motor from supply in order to maintain a prescribed rotor speed ω_r, which is the second control objective. Rotor speed is controlled indirectly by controlling the rotating speed of the stator flux ω_{ms}. Angular slip frequency, which is a difference between the two

$$\omega_{sl} = \omega_r - \omega_{ms}$$

is small for nominal load torques and therefore the rotor speed can follow a desired value with high accuracy when this technique is used. It has been shown in [22], that in a steady-state, ω_{ms} is given by the expression

$$\omega_{ms} = \frac{R_s}{L_s}\frac{u_{sy}}{u_{sx}} - \frac{R_s M^2}{R_r L_s^2}\omega_{sl}.$$

Thus, for nominal load torques steady-state value of ω_{ms} can be accurately approximated by

$$\omega_{ms} \cong \frac{R_s}{L_s}\frac{u_{sy}}{u_{sx}}.$$

An exact formula is also available which uses additionally $|\boldsymbol{\psi}_s|$. The approach of [22] is adopted here and the rotor speed ω_r is controlled in an open-loop manner by keeping the ratio $\frac{u_{sy}}{u_{sx}}$ at an appropriate value. The proposed structure with u_{sx} as the control input, stator flux magnitude as the output, and with an open-loop control of speed, is shown in Fig. 1. It is assumed that the stator flux phasor is known. Estimates of the flux phasor required by the controller may be obtained

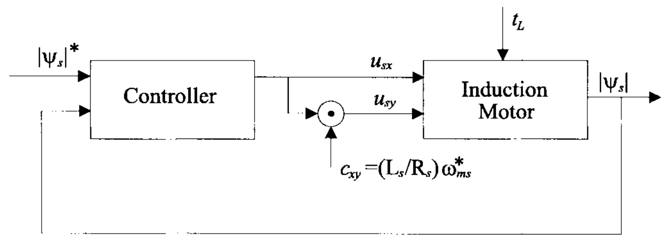


Fig. 1. Stator flux-oriented control structure for the induction motor.

from a stator flux observer. Although this control structure is applicable for small angular slip frequencies, it is very attractive as it enables us to solve the two input two output control problem by using the SISO design. It can be viewed as a nontrivial extension of the so-called V/f (stator voltage magnitude/input frequency) control scheme which is used in many practical applications where high dynamic performance and steady-state accuracy are not required. The V/f scheme is completely open loop. By controlling the flux in closed-loop robustness and steady-state error are improved. By applying neural networks to control the flux instead of a simple PI for example, the performance is also improved as the nonlinear controller can better accommodate the nonlinear dynamics of the plant.

C. Neural Controller for the Induction Motor

Digital control of the motor is considered. A hypothesis is made, that a single-input/single-output feedback linearisable model of the motor, with input voltage u_{sx} as the input and stator flux modulus $|\psi_s|$ as the output, can precisely enough approximate real complex dynamical relations involving these two variables.

Number of past output and input values appearing in the model functions and relative degree of the model have been chosen after a qualitative analysis of the motor dynamics. First, in the stator flux oriented frame, the stator flux-linkage magnitude ψ_{sx}, can be described in terms of the so-called stator magnetizing current i_{ms} as

$$\psi_{sx} = M i_{ms}$$

where

$$i_{ms} = \frac{L_s}{M} i_{sx} + i_{rx}.$$

Since in the nominal conditions

$$i_{ms} \approx \frac{L_s}{M} i_{sx}$$

i_{sx} is effectively the stator flux producing current.

The following two differential equations, governing the behavior of currents i_{sx} and i_{sy}, can be derived:

$$\frac{di_{sx}}{dt} = -\left(\frac{R_s}{L'_s} + \frac{1}{T'_r}\right) i_{sx} + \left(\omega_{sl} - \frac{R_s}{L_s T'_r}\frac{1}{\omega_{ms}}\right) i_{sy} + \frac{1}{L'_s} u_{sx} + \frac{1}{L_s T'_r}\frac{1}{\omega_{ms}} u_{sy} \quad (12)$$

$$\frac{di_{sy}}{dt} = -\omega_{sl} i_{sx} - \left(\frac{1}{T'_r} + \frac{R_s}{L'_s}\frac{\omega_{sl}}{\omega_{ms}}\right) i_{sy} + \frac{1}{L'_s}\frac{\omega_{sl}}{\omega_{ms}} u_{sy} \quad (13)$$

where $L'_s = \sigma L_s$ is the stator transient inductance and $T'_r = \sigma T_r = \sigma L'_r / R_r$ is the transient rotor time constant of the machine.

Based on the above, the model for control purposes is chosen as

$$y(k+1) = F(u(k-1), y(k)) + G(u(k-1), \cdots, u(k-4))u(k) \quad (14)$$

where $y = |\psi_s|$ and $u = u_{sx}$. Functions F and G are certainly nonlinear and, furthermore, they are dependent on the operating point of the machine.

To the motor model of the form (14) the algorithm described in Section III-A can be applied. A modified version of this algorithm, which has been proposed in [23], is used here. The modification concerns the choice of learning rates. Two different learning rates for the two networks are used. These learning rates, once set at the beginning of the algorithm's operation, are then adjusted according to the average error calculated over a chosen number of time steps (called later period). If this average error increases comparing to the preceding period learning rates are decreased by 30%. If the average error decreases learning rates are increased by 5%. Although no formal stability proof has been derived, experiments show good performance of this method. As commented earlier, use of the dead zone in the way proposed in [10] is quite conservative. The rationale behind the technique used here is based on heuristic arguments concerning convergence of the modeling error function to a local minimum. If the learning rate is too big parameter updates "overshoot" the minimum increasing the error. If it is too small, the convergence is exceedingly slow. The periodic adaptations of the learning rate aim to avoid the instability, while maintaining reasonable error convergence. It has also been experimentally shown in [23], that use of such scheme is preferable to a learning rate based on the norm of instantaneous gradient (as in [10]), when measurement noise is present. In such case, learning rates adapted in accordance with averaged behavior of the system, rather than at each time step, have the benefit of smoothing out the effects of noise.

D. Simulation Results

MLP networks are used for both F and G approximation. The number of inputs to each of them is dictated by the choice of functions in the model (14). Both have two hidden layers with twenty neurons in each. The hidden neurons employ the hyperbolic tangent functions. Short preliminary off-line training of the networks needs to be performed to achieve initial weights values which give a first rough approximation of the target functions. This is done using data collected when motor, for obvious practical reasons, is controlled by some standard technique. These input–output data are not used to train the neural controller under design to mimic the standard one. Once the output errors are reasonably small, networks are employed in the on-line algorithm described above. Neural controller is expected to keep stator flux modulus at the desired

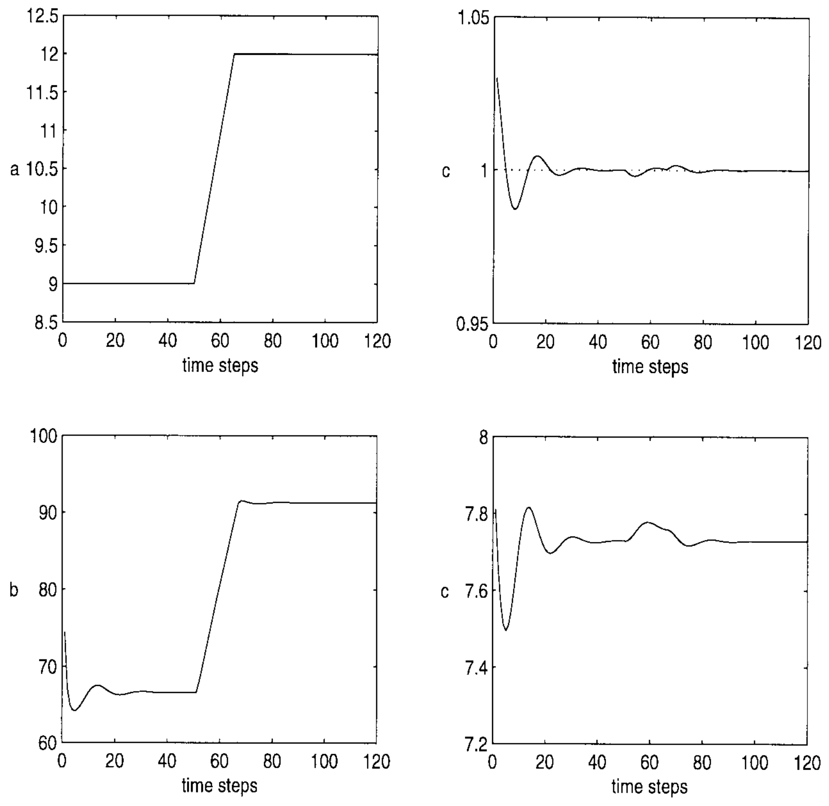


Fig. 2. Following of a speed profile. (a) Coefficient c_{xy}, (b) Rotor speed [rad/s], (c) Flux magnitude (solid), reference value $|\psi_s|^*$ (dotted) [Wb], (d) Control input u_{sx} [V].

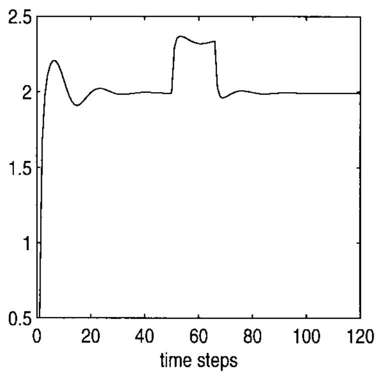


Fig. 3. Following of a speed profile. Current i_{sy} [A].

level $|\psi_s|^* = 1$ [Wb], while speed coefficient c_{xy} is changed and load torque t_L is varying. Sampling time is equal 0.1s. Initial learning rates for the networks in both reported below simulations are chosen as $\eta = \lambda = 0.000\,18$. Stator flux components are obtained directly from the simulation model without implementing the observer.

Example 1: In the first experiment, load torque is kept constant, $t_L = 2$ Nm, while fast ramp of c_{xy} is applied. Results are shown in Figs. 2 and 3. Some small oscillations at the beginning of the simulation are the result of initial conditions in the motor and adopting of the neural model by the controller. Then, an increase of the c_{xy} is closely followed by the rotor speed, while flux modulus shows only minor fluctuations. Extra electrical torque needed to accelerate the rotor is generated by the temporary increase of the current i_{sy}.

Example 2: In the second experiment, coefficient c_{xy} is kept at a constant value, $c_{xy} = 10$, while varying load torque is applied to the machine. Results are presented in Figs. 4 and 5. Strong and abrupt increase of the load torque brings about some small oscillations of the flux magnitude and a little decrease of the rotor speed due to the increased slip. Strong increase of the current i_{sy} produces the compensating electrical torque. After the load torque is completely removed, i_{sy} drops to zero and the speed is slightly increased.

It is interesting to analyze behavior of the weights of the network in course of the algorithm's operation. Behavior of two randomly chosen weights is presented in Fig. 6. As can be seen, changes of the weights correspond to the changes of machine's operating point, forced by the variations of the load torque. For each different operating point weights settle at a different value. This supports the earlier stated presumption that the derived model of the machine is dependent on the operating point and also means that controller adjusts its neural model according to the changing characteristics of the system. Thus, new local nonlinear approximation of the plant dynamics, has to be constructed with every change of the operating point. A similar control problem was investigated in [22] and later in [24], where a self-tuning predictive controller was proposed. The control input generated by such controller is based on the linearized model of the system which is

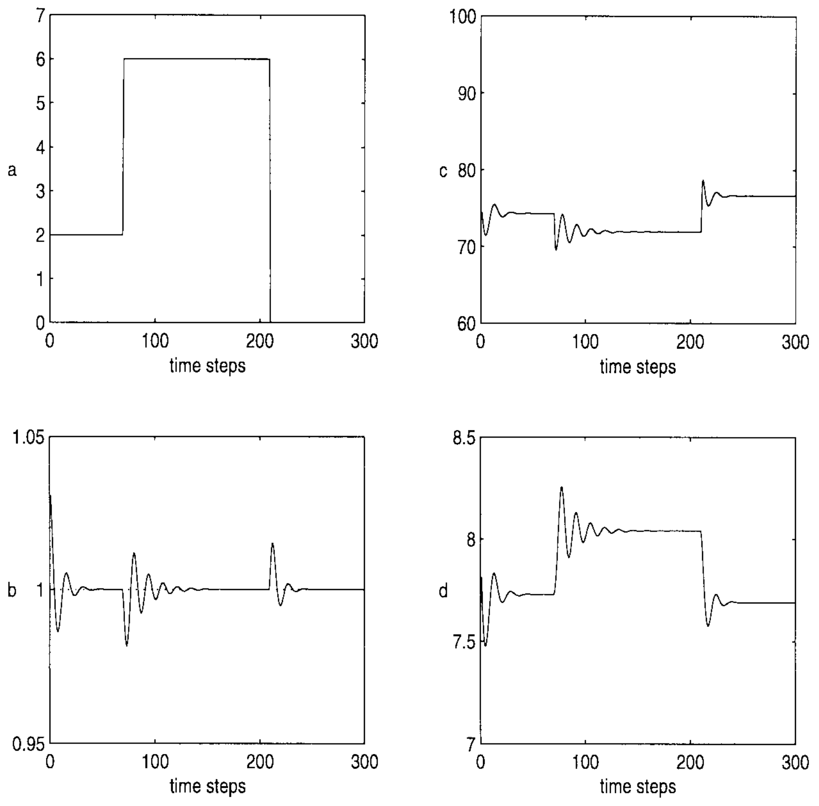


Fig. 4. Control under variations of the load torque t_L. (a) Load torque profile [Nm]. (b) Flux magnitude (solid), reference value $|\psi_s|^*$ (dotted) [Wb]. (c) Rotor speed [rad/s]. (d) Control input u_{sx} [V].

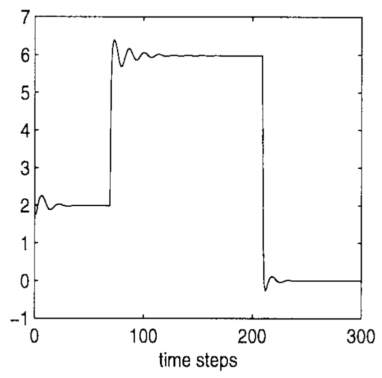


Fig. 5. Control under variations of the load torque t_L. Current i_{sy} [A].

recursively identified. The neural controller proposed here yields smoother transient response.

IV. Recurrent Networks for Adaptive Output Control

In this section the adaptive control technique utilizing dynamic neural networks proposed in [17] is presented. The development is motivated by nonlinear control problems, where first, state of the controlled plant is not available for measurement, and second, lack of precise knowledge about the dynamics and/or parameter variations call for adaptive character of the control system. Induction motor control is an example of such a problem. The section concludes with a summary of stability results from [18] and a discussion of assumptions on which this stability analysis is based.

A. Recurrent Networks as Dynamic Models for Adaptive Control

As already mentioned, despite some impressive results for specific classes of nonlinear systems, e.g. [15], [16], the output feedback control and especially the adaptive output feedback control for nonlinear systems remains one of the outstanding research problems. Lack of state measurements necessitates use of some kind of dynamic model of the system for control law synthesis. For linear systems with known parameters, thanks to the separation principle, the output feedback control design can be conveniently split into an independent design of stable Luenberger observer and stable-state feedback law. However, there is no similar separation principle in the adaptive context even for linear systems, let alone nonlinear ones. In case of adaptive control of linear plants, special, nonminimal in terms of state dimension, parameterizations of linear dynamic systems have been developed. These special models allowed formulation of stable adaptive control techniques, see, e.g., [14] and [25].

No such general framework exists for nonlinear systems and there is a need for some kind of nonlinear dynamic model, which allows synthesis of control input for the plant and is at the same time "manageable" enough, allowing analytical treatment and incorporation of mechanisms ensuring stability. Already in [7] a possibility of using dynamic neural models

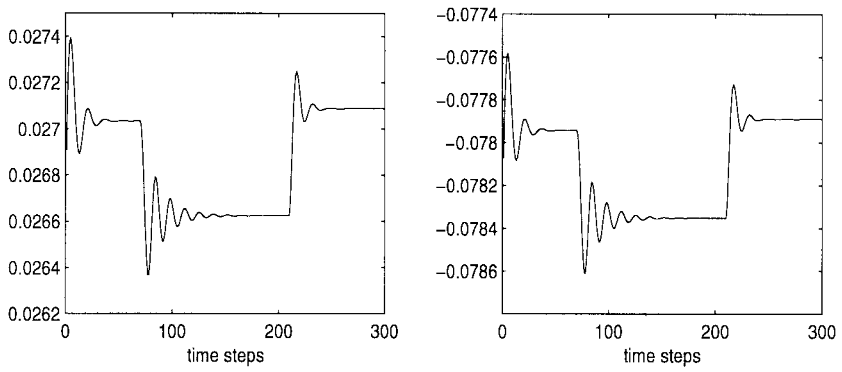


Fig. 6. Control under variations of the load torque t_L. Behavior of two randomly chosen weights.

combining nonlinearities with dynamics for the identification of nonlinear plants was demonstrated in simulations. Still, however, relatively little has been done as far as application of such models for control, and particularly adaptive control, is concerned. Most papers present heuristic approaches, e.g., [26]–[28], and comprehensive convergence analysis of adaptive schemes is still missing.

Stability of dynamic networks in control, in a nonadaptive context, has been rigourously addressed in recent publications [29], [30], [31], [32] using an extension of H_∞ theory. While use of H_∞ techniques raises questions about conservativeness of the stability results, this is certainly an important contribution.

Models of dynamic systems can be constructed using neural networks in different ways, for example by closing a feedback loop around a static MLP network. Modeling capabilities of such dynamic structures derive from approximation properties, for static functions, of static MLP networks. The ability of Hopfield-type recurrent neural networks to approximate dynamic systems, in continuous and discrete time, has been shown respectively in [33] and [34]. It is our belief that recurrent neural networks of this form have the potential to provide useful models for adaptive control of many nonlinear systems. Recurrent networks of the Hopfield type have relatively simple structure and the hyperbolic tangent function $\tanh(\cdot)$ has advantageous properties like smoothness, boundedness and being monotone. Loosely speaking it is a kind of "mild" nonlinearity. Usefulness of these features was apparent in the stability analysis of [18] and [19].

A model of the controlled system based on a recurrent network can be interpreted as a state-space model, usually with a nonminimal state dimension. Use of this type of model for adaptive control of nonlinear systems seems quite natural, as first, there is no general equivalence, for nonlinear systems, between state-space and input–output models while for most physical systems, state-space models based on first principles are a natural way of describing dynamics. Second, even when the input–output models exist, like, for example, a SISO system

$$y^{(d)} = F\big(y^{(d-1)}, \cdots, y\big) + G\big(y^{(d-1)}, \cdots, y\big)\, u$$

practical utilization of such models, when only output y but not its derivatives $y^{(d-1)}, \cdots, \dot{y}$ are measured, still requires constructing of a dynamical state-space model. If a dynamic neural model can be trained, so that in some region its input–output behavior is close to that of the unknown system, then we should be able to obtain some kind of equivalent information about state of the unknown plant from the state of the neural network. Thus, if a controller based on such a model can constructed, it should be able to achieve a good dynamic performance due to the information about the state of the actual plant without the explicit use of the state and parameter observer.

In the method presented here a linearizing feedback control law is computed analytically for the network and applied to the plant while parameters of the network are updated on-line. Such approach can be classified as indirect adaptive control, as it is the parameters of the plant model that are adapted and control is computed based on the current model, rather than directly adapting the controller parameters. The reason for such choice is that, as opposed to linear systems, question of existence of direct dynamic controllers for nonlinear systems is a very difficult one and use of a heuristicly chosen direct adaptive dynamic controller, as for example in [35], renders the overall scheme extremely difficult for analytical treatment.

A similar general motivation appears to underlie recent schemes of [36], [37] and [38], in which, either continuous- or discrete-time recurrent network respectively, is used as a model of the unknown system and control law utilizes state of the network.

B. Control Algorithm

We seek to design a control law for a continuous time nonlinear system, which is at this point formulated quite generally as

$$\begin{aligned} \dot{\mathbf{q}} &= f(\mathbf{q}, \mathbf{u}) \\ \mathbf{y} &= h(\mathbf{q}) \end{aligned} \tag{15}$$

where $\mathbf{q} \in \mathbf{R}^s$ is a state vector, $\mathbf{u} \in \mathbf{R}^m$ is an input vector, $\mathbf{y} \in \mathbf{R}^m$ is an output vector. The objective is to make system outputs track a vector of specified trajectories $\mathbf{y}_{\text{ref}} \in \mathbf{R}^m$.

A recurrent neural network is used as a dynamic model of the system (15), based on which control law is synthesized.

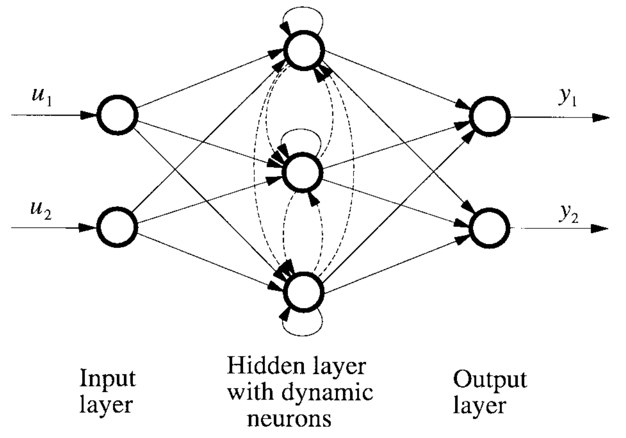


Fig. 7. Diagonal dynamic network.

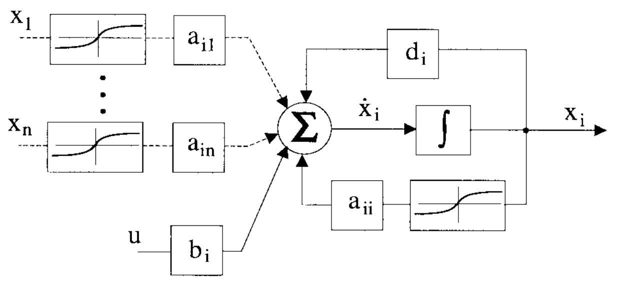


Fig. 8. A single dynamic neuron.

The neural model is defined by

$$\begin{aligned}\dot{\hat{\mathbf{x}}} &= \mathbf{D}\hat{\mathbf{x}} + \hat{\mathbf{A}}\mathbf{T}(\hat{\mathbf{x}}) + \mathbf{B}\mathbf{u} \\ \hat{\mathbf{y}} &= \mathbf{C}\hat{\mathbf{x}}\end{aligned} \tag{16}$$

where $\hat{\mathbf{x}} \in \mathbf{R}^n$ is a state vector, $\mathbf{u} \in \mathbf{R}^m$ is an input vector, $\hat{\mathbf{y}} \in \mathbf{R}^m$ is an output vector. A nonlinear operator $\mathbf{T}(\cdot)$ is defined as$\mathbf{T}(\hat{\mathbf{x}}) = [\tanh(\hat{x}_1) \ \cdots \ \tanh(\hat{x}_n)]^T$. $\mathbf{D} = \mathrm{diag}(d_1, \cdots, d_n)$ is a $n \times n$ diagonal matrix with negative entries, $d_i < 0$, $\hat{\mathbf{A}} = \mathrm{diag}(\hat{a}_1, \cdots, \hat{a}_n)$ is an $n \times n$ diagonal matrix, $\mathbf{B} \in \mathbf{R}^{n\times m}$ and $\mathbf{C} \in \mathbf{R}^{m\times n}$. An example of such network with two inputs and two outputs is shown in Fig. 7 and a single neuron in Fig. 8. The network can be regarded as a parsimonious version of the Hopfield-type network. It has a diagonal structure, that is, there is no interaction between dynamics of different neurons (these are shown as dashed lines on both figures).

In what follows, $\hat{\mathbf{a}}$ will denote a vector of parameters containing the diagonal elements of $\hat{\mathbf{A}}$, $\hat{\mathbf{a}} = [\hat{a}_1 \ \cdots \ \hat{a}_n]^T$.

The following assumption guaranteeing exponential stability of the neural model (as shown in Lemma 3 in [19]) is required.

Assumption 1: $\hat{a}_i < -d_i \ i = 1, \cdots, n$.

1) Control Synthesis: The evolution of the network output can be expressed as

$$\dot{\hat{\mathbf{y}}} = \mathbf{C}\dot{\hat{\mathbf{x}}} = \mathbf{C}\mathbf{D}\hat{\mathbf{x}} + \mathbf{C}\hat{\mathbf{A}}\mathbf{T}(\hat{\mathbf{x}}) + \mathbf{C}\mathbf{B}\mathbf{u}. \tag{17}$$

An assumption is made.

Assumption 2: $\mathbf{CB}$ is invertible.

The above assumption implies relative degree of the neural model equal one. The feedback linearizing control can be calculated as

$$\mathbf{u} = (\mathbf{CB})^{-1}(-\mathbf{CD}\hat{\mathbf{x}} - \mathbf{C}\hat{\mathbf{A}}\mathbf{T}(\hat{\mathbf{x}}) + \mathbf{v}) \tag{18}$$

which applied to the network results in it being decoupled and linear with respect to the new input vector $\mathbf{v}$

$$\dot{\hat{\mathbf{y}}} = \mathbf{v}. \tag{19}$$

The auxiliary control input $\mathbf{v}$ is designed as a simple linear pole placement

$$\mathbf{v} = \dot{\mathbf{y}}_{\mathrm{ref}} - \alpha(\hat{\mathbf{y}} - \mathbf{y}_{\mathrm{ref}}). \tag{20}$$

where α is a positive constant, $\alpha > 0$. Control input, as defined by (18) and (20), is applied to both plant and the neural model. The error between the output vector of the model and the reference output vector, $\hat{\mathbf{e}} = \hat{\mathbf{y}} - \mathbf{y}_{\mathrm{ref}}$, obeys

$$\dot{\hat{\mathbf{e}}} + \alpha\hat{\mathbf{e}} = 0$$

and converges exponentially to the origin with the rate α. Thus, we can assume, that after some initial transient, $\hat{\mathbf{e}}$ converges to zero and so the plant tracking error $\mathbf{e}_t = \mathbf{y}_{\mathrm{ref}} - \mathbf{y}$ is equivalent to the modeling error $\mathbf{e}_m = \hat{\mathbf{y}} - \mathbf{y}$

$$\mathbf{e}_t = \mathbf{y}_{\mathrm{ref}} - \mathbf{y} = \hat{\mathbf{y}} - \mathbf{y} = \mathbf{e}_m. \tag{21}$$

From now onwards, for simplicity of exposition, a SISO plant is considered.

2) On-Line Parameter Adaptation: On-line updates of the parameter vector $\hat{\mathbf{a}}$ are performed in discrete-time instants, with a period T, in the direction opposite to the gradient of the following error criterion:

$$E = \frac{1}{2}\int_{kT}^{kT+T} (\hat{y} - y)^2 \, d\tau \tag{22}$$

$$\hat{a}_i(kT + T) = \hat{a}_i(kT) - \lambda\frac{\partial E}{\partial \hat{a}_i} \tag{23}$$

where $\lambda > 0$ is the learning rate. The discrete time instants, when updates take place, are indexed by the positive integer k. Calculating the error function derivative as

$$\frac{\partial E}{\partial \hat{a}_i} = \int_{kT}^{(k+1)T} (\hat{y} - y)\frac{\partial \hat{y}}{\partial \hat{a}_i} \, d\tau \tag{24}$$

this derivative $\frac{\partial E}{\partial \hat{a}_i}$ is obtained by integrating over the time T the following two differential equations:

$$\frac{d}{dt}\frac{\partial E}{\partial \hat{a}_i} = (\hat{y} - y)\frac{\partial \hat{y}}{\partial \hat{a}_i} \tag{25}$$

$$\frac{\partial \hat{y}}{\partial \hat{a}_i} = c_i\frac{\partial \hat{x}_i}{\partial \hat{a}_i} \tag{26}$$

$$\frac{d}{dt}\frac{\partial \hat{x}_i}{\partial \hat{a}_i} = (d_i + \hat{a}_i \tanh'(\hat{x}_i))\frac{\partial \hat{x}_i}{\partial \hat{a}_i} + \tanh(\hat{x}_i) \tag{27}$$

starting with zero initial conditions: $\partial E/\partial \hat{a}_i(kT) = 0$, $\partial \hat{x}_i/\partial \hat{a}_i(kT) = 0$. Once the parameter updates are done initial conditions in (25)–(27) are reset to zero and the cycle is repeated. Differential equations (25)–(27) need to be integrated

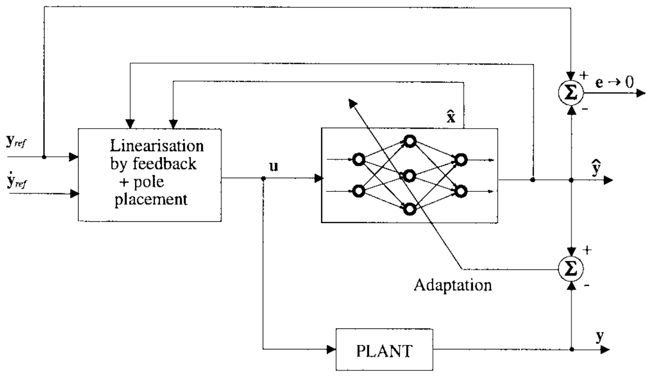


Fig. 9. Adaptive control using recurrent network.

in real time. Differential equation (27) is obtained by applying a partial derivative $\partial/\partial\hat{a}_i$ to the both sides of the differential state equation (16) (under standard smoothness conditions we can assume that d/dt and $\partial/\partial\hat{a}_i$ commute). Because of the diagonal structure of the network state equation we have

$$\frac{\partial\hat{x}_j}{\partial\hat{a}_i} = 0, \qquad \text{for } j \neq i.$$

Equation (27) describes the so-called sensitivity model, as in, e.g., [39]. Its derivation relies on the assumption that $\hat{a}_i$ is constant, which, in the case of the algorithm with periodic updates described above, is satisfied for each period of length T between consecutive parameter updates. As an alternative to (23), parameter updates could be performed continuously

$$\dot{\hat{a}}_i = -\lambda(\hat{y} - y)\frac{\partial\hat{x}_i}{\partial\hat{a}_i}c_i$$

with a very small learning rate so that this assumption would be "almost" true. $\partial\hat{x}_i/\partial\hat{a}_i$ would be then obtained from (27) integrated without resetting. We chose the periodic update method for it is more conceptually clear and makes the overall scheme more tractable analytically. In the adaptive literature, such schemes are referred to as hybrid adaptive control. It is pointed out in [14], that they exhibit better robustness, with respect to disturbances, compared with continuous parameter adaptation. Although results in [14] refer to control of linear systems, similar properties can be expected here as an additional nonlinearity of the tracking error due to nonlinear plant dynamics should not change the underlying argument.

The updating process has to be constrained in the way that Assumption 1 is satisfied at any point in time. The "raw" values of weights $\hat{a}_i$ being the result of (23) are projected onto the nearest point of the set

$$\mathcal{P}_i \equiv \{\hat{a}_i : \hat{a}_i \leq -d_i - \varepsilon\} \tag{28}$$

ε is a small positive number used to construct a closed set which is needed for a well-defined projection.

A diagram of the overall control scheme is shown in Fig. 9.

C. Stability Results

In the convergence analysis of [18] and [19] it is assumed that there exists a choice of network parameters $\hat{\mathbf{a}} = \mathbf{a}$, such that a recurrent neural network is capable of exactly modeling the plant. In other words the dynamical system

$$\begin{aligned}\dot{\mathbf{x}} &= \mathbf{D}\mathbf{x} + \mathbf{A}\mathbf{T}(\mathbf{x}) + \mathbf{B}u\\ y &= \mathbf{C}\mathbf{x}\end{aligned} \tag{29}$$

is a realization of the same input–output mapping $u(t) \to y(t)$ as (15). Systems (15) and (29) can have their state vectors of different size. System (29) has its state vector of different size than (15), always larger even if $\mathbf{D}$ and $\mathbf{A}$ were full matrices (see results in [33]). Hence, even if $\hat{\mathbf{a}} = \mathbf{a}$ vector $\mathbf{x}$ is not the same as $\mathbf{q}$. Increase in the state dimension is the price to be paid for the structure of (29). This can be seen as analogy with nonminimal parameterizations of linear dynamic systems for the purpose of adaptive control, as mentioned in the Section IV-A. We require that the plant (29) satisfies Assumption 1, implying open-loop stability of the original plant (15). The input–output mapping realized by (15) and (29) is certainly parameterized by the initial conditions of their respective state vectors. However, since in this analysis the plant is assumed stable, the influence of initial conditions should die out and we can treat (29) as a valid description of the controlled plant. From now on the vector $\mathbf{x}$ will serve as the state of the plant.

The ability of Hopfield type networks to approximate dynamic systems has already been mentioned. The simpler network structure is chosen here to allow analytical tractability and insight into the convergence issues. As a tradeoff, it may not have the full approximation capabilities of the Hopfield network, although this could probably be compensated to some extent by a further increase of the state vector dimension. Simulations reported in the following section seem to justify this conjecture. In any case, whether with diagonal network (16) or a fully interconnected one, some residual structure error will remain. By the structure error we mean that there is in fact no ideal set of parameters of (29) which would give a perfect model of (15). Although the assumption about perfect modeling capabilities made here is strong, it is not completely unreasonable. For example, many results in adaptive control assume a linear plant, although a perfectly linear plant is hardly ever the case. The stability analysis of [18] and [19] provides understanding of the general mechanisms governing behavior of such neural adaptive systems and explains interactions between plant and model dynamics on the one hand and the parameter adaptation on the other. Incorporation of the structure error into the analysis, which will most likely necessitate some modification of the adaptation laws, needs certainly to be the next step in this development.

The following assumptions are also required.

Assumption 3: Matrix

$$\mathbf{M} = \left(\mathbf{D} - \frac{1}{\mathbf{CB}}\mathbf{BCD} - \frac{\alpha}{\mathbf{CB}}\mathbf{BC}\right)$$

is Hurwitz (i.e., all its eigenvalues have negative real parts).

Assumption 4: Reference output signal y_{ref} and its derivative $\dot{y}_{\mathrm{ref}}$ are bounded, i.e. there exist positive constants d_1, d_2 such that $|\dot{y}_{\mathrm{ref}}| < d_1$, $|y_{\mathrm{ref}}| < d_2$.

Assumption 3 is related to the stability of the closed-loop dynamics of the neural model, that is, open-loop dynamics of the neural network (16) with the linearizing feedback control (18). Although Assumption 4 prevents theoretical step inputs to be considered, it does not pose significant restrictions in practice where prefiltering of sharp changes of reference inputs is a common approach. Under certain very technical conditions described in detail in [18] the main result of [18] is stated below.

Theorem: Under Assumptions 1, 2, 3, and 4, if the initial parameter mismatch $\|\hat{\mathbf{a}} - \mathbf{a}\|$ is small enough and T long enough, control (18) achieves ultimate boundedness of all signals in both plant (29) and model (16). There exists a positive number $\bar{\lambda}$ such that for $0 < \lambda \leq \bar{\lambda}$ the updating scheme (22)–(24) results in the bound on the tracking error $e_t = y_{\text{ref}} - y$ decreasing monotonously.

The analysis presented in the proof of the Theorem highlights mechanisms governing stability and tracking error convergence in this neural adaptive control system. First, with bounded reference input and with constant values of the parameters the tracking error remains bounded. Then, decrease of the tracking error is achieved in the updating process via decrease of the network parameters distance to the "true" parameters of the plant, that is, the parameter values which result in a perfect modeling of the plant by the recurrent network. Consequently, two types of dynamics can be distinguished in this system: the fast continuous-time dynamics of the plant and neural controller and the slow discrete-time dynamics of learning. Learning dynamics binds together the evolution of parameter error and state error from one update step to another. The speed of the plant and neural model dynamics determines how fast the learning process can progress. This dependence is quantified by the length of the update period T, which must be long enough so that a certain technical relation holds and the gradient used gives a direction of improvement of the parameter error. The result of the Theorem does not offer prescriptive methods for choosing T and λ and in practice, as it is the case in the reported simulations, they have to be chosen experimentally. It provides however an insight into the convergence mechanisms which helps guide such process.

For a constant reference output, a stronger result of exponential stability is shown in [19] under slightly different assumptions.

V. Adaptive Control of the Induction Motor Using Dynamic Networks

The control technique for the induction motor presented here is based on the algorithm utilizing dynamic neural networks described in the previous section.

The control scheme was proposed in Section III was based on the SISO algorithm utilizing static neural networks. The same three-phase squirrel-cage and voltage-controlled machine is considered here, whose nonlinear dynamics are given by the (1)–(9). While the algorithm proposed in Section III involved closed-loop control of the stator flux magnitude only and rotor speed was controlled in an open-loop manner, in this section a full MIMO control is pursued, with closed-loop control of both flux and rotor speed.

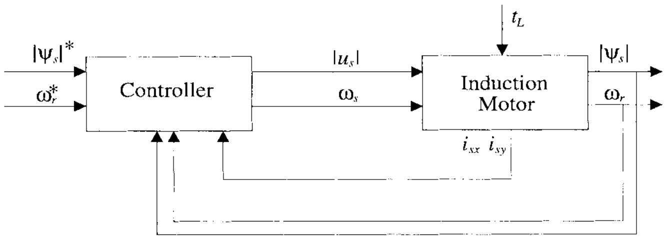


Fig. 10. MIMO control of the induction motor using dynamic networks.

The state-space description of the motor (1)–(5) uses a state vector $[\omega_r, i_{sd}, i_{sq}, \psi_{rd}, \psi_{rq}]$ consisting of rotor speed, stator current components, and rotor flux components. An alternative state-space description could be obtained in terms of rotor current and stator flux components by utilizing the static relationships (6)–(9). As mentioned before, the motor control problem is made difficult by the fact, that out of these different variables only stator currents and rotor speed, are normally available for measurement. In some control schemes for the induction motor, observers are used to obtain estimates of the unmeasurable variables. However, as the plant is nonlinear, the separation principle does not apply and stable-state feedback control combined with a stable observer do not imply stability of the closed-loop system. This is further complicated by significant parameter variations during operation. Due to these difficulties, no stability proof for observer-based induction motor control schemes has been shown so far.

General arguments for use of dynamic neural models for nonlinear adaptive control of systems with state, or part of state, unavailable for measurement have been already given the preceding section. Clearly, induction motor falls into such category and is in fact one of the case studies motivating the development of the scheme proposed in Section IV.

A. MIMO Control of the Motor

A general diagram of the proposed control structure is shown in Fig. 10. Rotor speed ω_r and amplitude of the stator flux $|\psi_s|$ are the controlled variables. Two inputs that the controller generates are the amplitude $|\mathbf{u}_s|$ and the rotating speed ω_s of the stator voltage phasor $\mathbf{u}_s$ in the stationary coordinates (sD, sQ). This rotating speed ω_s is equal to the angular frequency of stator supply voltage. Amplitude and frequency of the required sinewave are typically the inputs of a voltage-source inverter. Integration of ω_s yields the angle of $\mathbf{u}_s$, which allows calculation of u_{sd} and u_{sq}.

The controller requires an estimate of the stator flux magnitude $|\psi_s|$, which can be obtained from a stator flux observer. It has to be pointed out that, first, it is easier to estimate stator than rotor flux, and second, a need for the magnitude only is a weaker requirement than relying on estimates of the full stator phasor, that is, its both d-q components or, alternatively, angle and magnitude. Compared to methods which require good estimates of both coordinates of the stator phasor for conversion between different reference frames, sensitivity of the method presented here with respect to observer errors will be much lower. Furthermore, as the updating scheme of the neural algorithm applied here is based on the integral of the

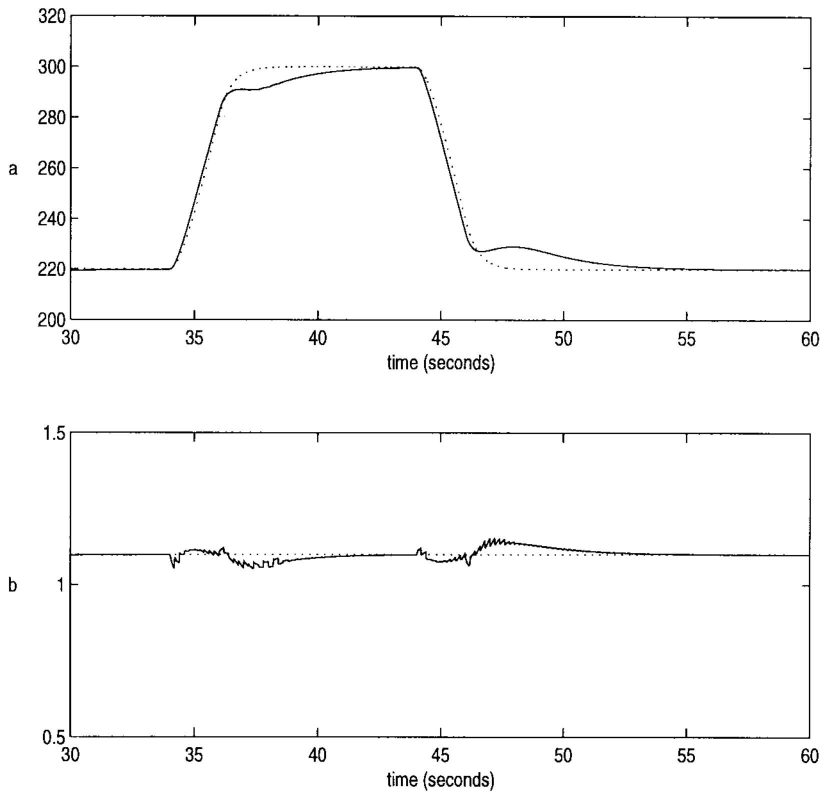


Fig. 11. Speed profile following. (a) Reference speed (dotted) and rotor speed (solid) [rad/s]. (b) Stator flux magnitude (solid) and reference value (dotted) [Wb].

output error, this will further provide for averaging out of the estimation errors.

Use of the measurements of the stator currents i_{sx}, i_{sy} in the control synthesis will be explained below.

B. Dynamic Neural Controller for the Induction Motor

The control algorithm utilizes the technique presented in Section IV, with a modification concerning structure of the neural model of the plant. A dynamic network serves as a model of the motor, with stator voltage magnitude $|\mathbf{u}_s|$ and angular supply frequency ω_s being the inputs and stator flux amplitude $|\psi_s|$ and rotor speed ω_r as the outputs. The neural model of the motor is constructed as

$$\begin{aligned}\dot{\hat{\mathbf{x}}} &= \mathbf{D}\hat{\mathbf{x}} + \mathbf{AT}(\hat{\mathbf{x}}) + \mathbf{F}_1 \tanh(i_{sx}) + \mathbf{F}_2 \tanh(i_{sy}) + \mathbf{Bu} \\ \hat{\mathbf{y}} &= \mathbf{C}\hat{\mathbf{x}} \end{aligned} \tag{30}$$

where, as before, $\hat{\mathbf{x}} \in \mathbf{R}^n$ is the state vector, $\mathbf{u} = [|\mathbf{u}_s|, \omega_s]^T$ is the input vector, and $\hat{\mathbf{y}} = [|\widehat{\psi}_s|, \hat{\omega}_r]^T$ is the output vector. $\mathbf{F}_1$ and $\mathbf{F}_2$ are matrices of adjustable weights. i_{sx} and i_{sy} denote components of the stator currents phasor with respect to a rotating reference frame fixed to the stator input voltage phasor $\mathbf{u}_s$. Measurements of stator currents are easily obtainable and since $\mathbf{u}_s$ is known, the conversion from (sD, sQ) coordinates can be easily performed.

Notice that the original structure (16) is augmented with terms containing measurements of stator currents to improve its modeling capabilities. As it has been already mentioned the proposed structure of the dynamic network (16) is a compromise between the full modeling capabilities and analytical tractability of the algorithm. It is shown here that this structure can be augmented by utilizing the available information and thus improve the modeling capabilities at no extra cost.

Since in normal operation, value of the rotor speed expressed in rad/s is about two orders of magnitude bigger than flux amplitude expressed in Wb, the speed output, that the neural model is supposed to model, is the measured value of rotor speed in rad/s scaled down by a factor of 0.01. This is done to improve conditioning of the error function used in parameter adaptation. Such problem scaling based on all *a priori* information is a commonly accepted engineering practice. Consequently, the speed setpoint for the motor (in rad/s) is also scaled by 0.01 for control generation purposes. Following general procedure described in Section IV, control input is calculated as

$$\begin{aligned}\mathbf{u} = (\mathbf{CB})^{-1}(&-\mathbf{CD}\hat{\mathbf{x}} - \mathbf{CAT}(\hat{\mathbf{x}}) - \mathbf{CF}_1 \tanh(i_{sx}) \\ &- \mathbf{CF}_2 \tanh(i_{sy}) + \mathbf{v})\end{aligned} \tag{31}$$

where, as previously

$$\mathbf{v} = \dot{\mathbf{y}}_{\text{ref}} - \alpha(\hat{\mathbf{y}} - \mathbf{y}_{\text{ref}})$$

and $\mathbf{y}_{\text{ref}}$ contains the reference values for flux magnitude and speed, $\mathbf{y}_{\text{ref}} = [|\psi_s|^*, \omega_r^*]^T$.

Parameter adaptation is performed as described in Section IV.

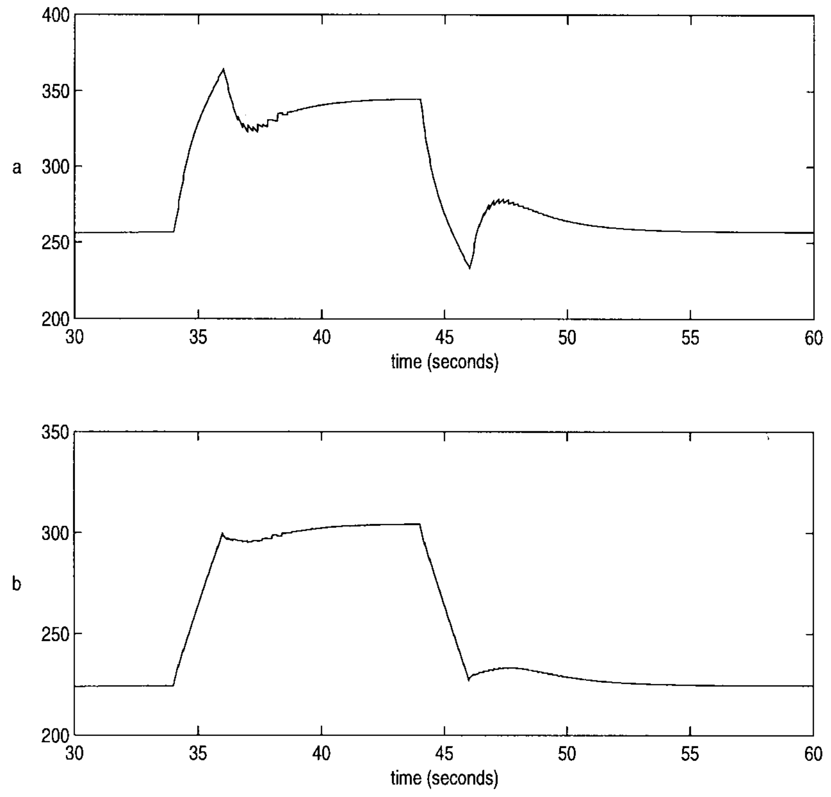


Fig. 12. Speed profile following—Control inputs. (a) Input voltage magnitude [V]. (b) Angular frequency of supply [rad/s].

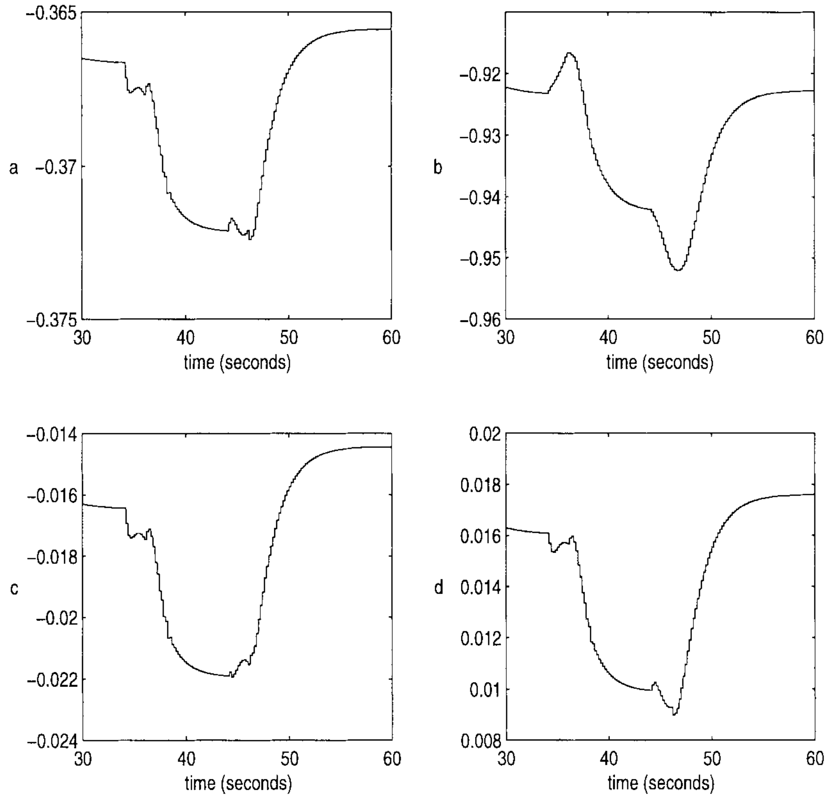


Fig. 13. Speed profile following—Changing weights in the neural model.

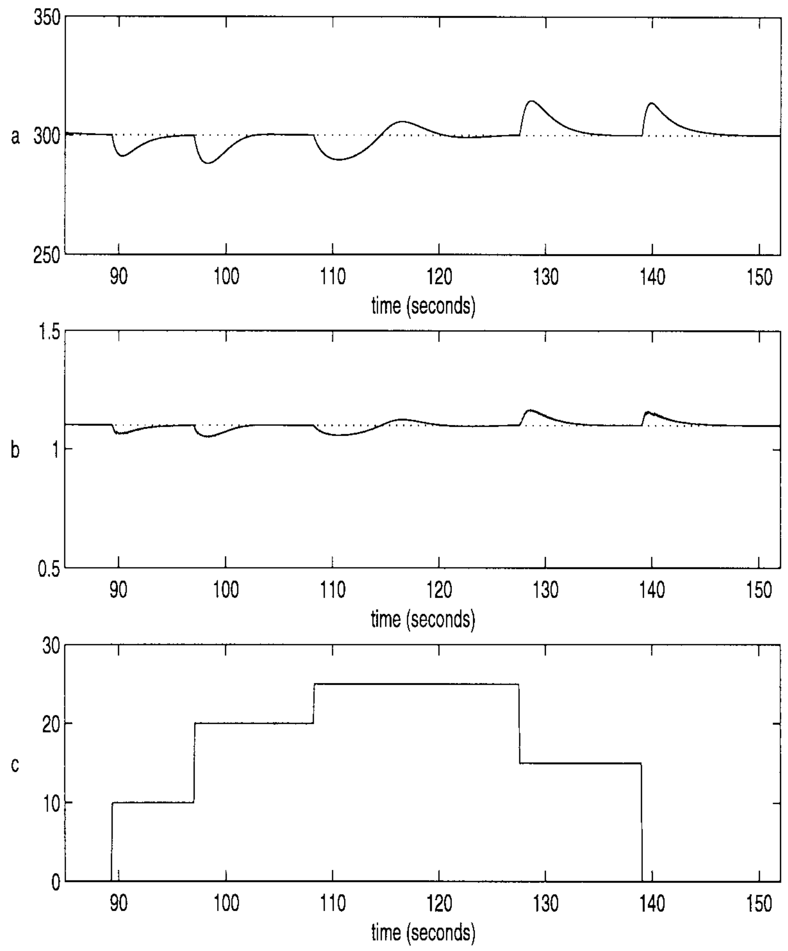


Fig. 14. Control under variations of load torque. (a) Rotor speed (solid) and reference value (dotted) [rad/s]. (b) Stator flux magnitude (solid) and reference value (dotted) [Wb]. (c) Load torque profile [Nm].

Although presence of the stator current measurements in the model (30) resembles an observer structure, it has to be pointed out that, first, as opposed to typical observers based for example on the extended Luenberger scheme [40], [41], here measurements enter nonlinearly and there is no explicit corrector term. Second, the control input is generated based on the model (30) in an integrated fashion without use of the explicit state and parameter observer and application of the certainty equivalence principle.

C. Simulation Results

For the simulation study reported here, a network of the structure (30) with 40 dynamic neurons was used. It was first trained in open-loop with externally generated control inputs applied to both motor and neural model. Weights were initialized as random numbers with uniform distributions, for $\mathbf{D}$, $\mathbf{A}$ in the interval [−1, 0], for $\mathbf{B}$, $\mathbf{C}$ in [−0.01, 0.01] and $\mathbf{F}_1$, $\mathbf{F}_2$ in [−0.05, 0.05]. The network was first trained in open-loop with externally generated control inputs applied to both motor and neural model. All the weights of the network were updated in this phase. To ensure that parameters d_i remain negative the projection algorithm was extended to constrain each d_i to the set $d_i \leq -\varepsilon$. This extended projection was used both for the preliminary training and the control. $\varepsilon = 0.0001$ was used. The size of the network and the initialization intervals were chosen after a few trials.

In the control experiment, the start-up of the motor was performed using an independent controller whose inputs were applied both to the motor and the neural model. In this start-up period learning was switched off. After the start-up, the control was switched to the neural controller described above. During the closed-loop control, only weight matrices $\mathbf{D}$, $\mathbf{A}$, $\mathbf{F}_1$ and $\mathbf{F}_2$ were adapted, with period $T = 0.2$ s and learning rate $\lambda = 10$. Stator flux magnitude was obtained directly from the simulation model without implementing the observer. Similarly as with the control scheme presented in Section III, two experiments are performed. The motor data used for simulations are given in the Appendix.

Example 3: In this first experiment, motor is subjected to constant load torque $t_L = 5$ Nm and is expected to follow a trajectory of reference rotor speed. This trajectory consists of an increasing (acceleration) and then decreasing (deceleration) ramp, passed through a stable linear filter. All the time, the reference value of stator flux magnitude is set to $|\psi_s|^* = 1.1$ Wb. Results are shown in Figs. 11 and 12. The controller is able to follow the speed profile quite well while maintaining the flux amplitude close to the desired value. At the same time

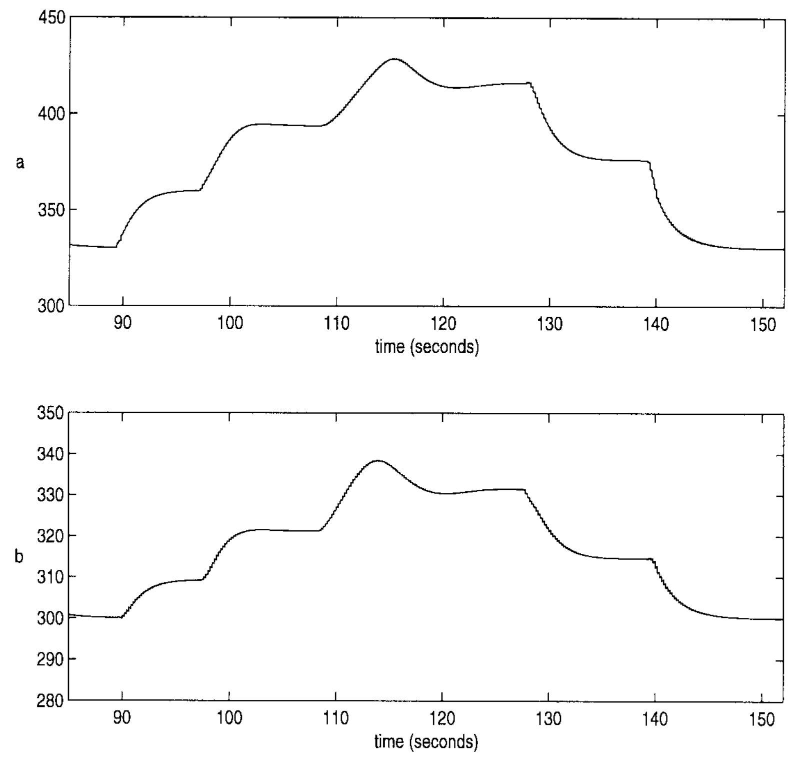


Fig. 15. Control under variations of load torque—Control inputs. (a) Input voltage magnitude [V]. (b) Angular frequency of supply [rad/s].

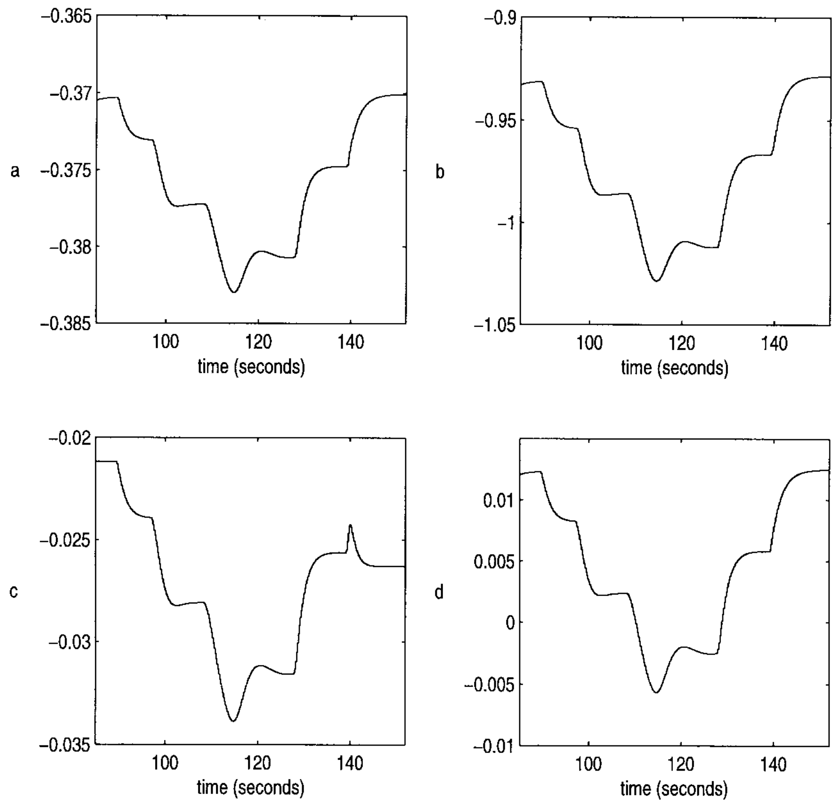


Fig. 16. Control under variations of load torque—Changing weights in the neural model.

the generated control inputs are "well behaved." Behavior of four randomly chosen weights of the neural model, during the course of this experiment, is shown in Fig. 13. It can be seen that speed variation over a wide range necessitates adaptation of the neural model, whose approximation capabilities are not global enough.

Example 4: In the second experiment, controller performance is tested in the presence of sharp changes of the load torque. Both speed and flux reference values are constant, $\omega_r^* = 300$ rad/s and $|\psi_s|^* = 1.1$ Wb. The results are shown in Figs. 14 and 15. Each step change of the load torque disturbs initially the controlled variables from their reference values, but the controller is able to drive them back to the required settings.

Load torque is an external disturbance of the system and is not directly included in any way in the model. Each change of the load torque changes in effect the dynamics of the model of the motor, which requires the controller to adapt its model of the machine. This point is illustrated in Fig. 16, in which behavior of a number weights is shown. After each change of the load torque, weights are tuned to settle at different values which provide good control for the particular operating conditions.

Remark: As it has already been mentioned, the analysis of the structure error is beyond the scope of the theoretical work presented in [18] and [19]. Control of the induction motor presented above is clearly a case where structure error is present. The recurrent network is not able to provide a global model of the complex dynamics of the motor and parameter adjustments are required if the operating point of the machine changes significantly. In this way parameter adaptation appears to give robustness with respect to the structure error. The issue of the structure error is related to the locality of the neural model and the question whether the network will "forget" a model obtained at a previous operating point. In the convergence analysis, which is presented for the case of no structure error, parameter updates result in the decrease of both the parameter error and the tracking error bound. Thus, the overall quality of the (global) model improves. The structural error will prevent achieving a good global model. Intuitively however, accurate enough local model is sufficient to generate suitable control action in the present operating region and consequently achieve maintaining decrease of the tracking error. This explains the observed robustness with respect to the structure error.

VI. Conclusion

In the paper, application of the recently developed adaptive control methods using neural networks to the induction motor control is presented. This case study represents one of the more difficult control problems due to the complex, nonlinear, and time-varying dynamics of the motor and unavailability of the full-state measurements.

Firstly, a partial solution is presented based on a SISO algorithm employing static MLP networks. Simulations showed good performance of this method. Since the nonlinear representation of the plant used in this scheme for control generation purposes is not global, the model of the plant needs to be updated when the operating point changes significantly. This is, however, still an improvement over more traditional techniques which rely on local linear approximation of the plant model.

In the second part of the paper an adaptive control scheme for the output feedback case is presented which uses dynamic neural networks. It is pointed out that in the case when state measurements are not available, similarly as it was the case in the successfully solved adaptive control problem for linear systems, a good parameterization of the controlled plant is of paramount importance. Recurrent networks of the Hopfield type have the ability to provide models of nonlinear dynamic systems and at the same time appear quite "manageable" allowing analytical treatment. Recent stability results for this scheme showed that recurrent networks seem to possess certain intrinsic features making them suitable for nonlinear adaptive control. It is clear from those results that the hyperbolic tangent activation function appears to be a very good choice of nonlinearity for constructing nonlinear dynamic models. Properties of the function $\tanh(\cdot)$, like smoothness, boundedness and being monotone, have been used many times in the stability proofs in a crucial way. Application of this neural algorithm, using a relatively simple network structure, to the difficult problem of the MIMO induction motor control reported in Section V points to the potential of such schemes for practical applications. It our belief that appropriately structured recurrent neural networks can provide conveniently parameterized dynamic models for many nonlinear systems for use in adaptive control.

Appendix
Machine Data Used in Simulations

ASEA 3–50 Hz(ABB) MBL 132 SB38-2 7.5 kW

n_p	number of pole pairs	1
R_s	stator resistance	2.19 Ω
R_r	rotor resistance	1.038 Ω
L_s	stator autoinductance	0.511 59 H
L_r	rotor autoinductance	0.511 59 H
M	mutual inductance	0.501 H
J	rotor inertia	0.35 kgm^2.

References

[1] F. Blaschke, "The principle of field orientation applied to the new transvector closed-loop control system for rotating field machines," *Siemens Rev.*, vol. 39, pp. 217–220, 1972.

[2] Z. Krzeminski, "Nonlinear control of induction motor," in *Proc. 10th IFAC World Congr.*, München, Germany, 1987, pp. 349–354.

[3] R. Marino, S. Peresada, and P. Valigi, "Adaptive input–output linearizing control of induction motors," *IEEE Trans. Automat. Contr.*, vol. 38, pp. 208–221, 1993.

[4] K. I. Funahashi, "On the approximate realization of continuous mappings by neural networks," *Neural Networks*, vol. 2, pp. 183–191, 1989.

[5] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Math. Control, Signals. Syst.*, vol. 2, pp. 303–314, 1989.

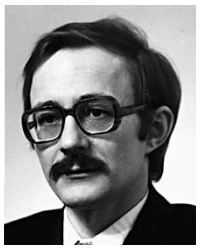
[6] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, pp. 359–366, 1989.

[7] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. Neural Networks*, vol. 1, pp. 4–27, 1990.

[8] A. U. Levin and K. S. Narendra, "Control of nonlinear dynamical systems using neural networks: Controllability and stabilization," *IEEE Trans. Neural Networks*, vol. 4, pp. 192–206, 1993.

[9] A. U. Levin and K. S. Narendra, "Control of nonlinear dynamical systems using neural networks—Part II: Observability, identification and control," *IEEE Trans. Neural Networks*, vol. 7, pp. 30–42, 1996.

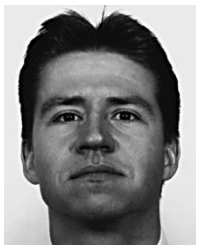
[10] F. C. Chen and H. K. Khalil, "Adaptive control of nonlinear systems using neural networks—A dead-zone approach," in *Proc. 1991 Amer. Contr. Conf.*, 1991, pp. 667–672.
[11] R. M. Sanner and J.-J. E. Slotine, "Gaussian networks for direct adaptive control," *IEEE Trans. Neural Networks*, vol. 3, pp. 837–863, 1992.
[12] F. L. Lewis, A. Yesildirek, and K. Liu, "Neural net robot controller: Structure and stability proofs," in *Proc. 32nd Conf. Decision Contr.*, San Antonio, TX, 1993, pp. 2785–2791.
[13] W. D. Chang, L. C. Fu, and J. H. Yang, "Adaptive robust neural-network based control for siso systems," in *Proc. 13th IFAC World Congr.*, San Francisco, CA, 1996, vol. F, pp. 163–168.
[14] K. S. Narendra and A. M. Annaswamy, *Stable Adaptive Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1989.
[15] R. Marino and P. Tomei, *Nonlinear Control Design: Geometric, Adaptive and Robust*. Englewood Cliffs, NJ: Prentice-Hall, 1995.
[16] M. Krstić, I. Kanellakopoulos, and P. Kokotović, *Nonlinear and Adaptive Control Design*. New York: Wiley, 1995.
[17] M. A. Brdyś, G. J. Kulawski, and J. Quevedo, "Recurrent networks for nonlinear adaptive control," in *Proc. 13th IFAC World Congr.*, San Francisco, 1996, vol. F, pp. 151–156.
[18] G. J. Kulawski and M. A. Brdyś, "Stable adaptive control with recurrent networks," in *Proc. 4th European Contr. Conf.*, Brussels, Belgium, 1997.
[19] M. A. Brdyś, G. J. Kulawski, and J. Quevedo, "Recurrent networks for nonlinear adaptive control," *Proc. Inst. Elect. Eng. Contr. Theory Applicat.*, 1998, vol. 145, no. 2, pp. 177–188.
[20] P. Vas, *Vector Control of AC Machines*. Oxford, U.K.: Oxford Univ. Press, 1990.
[21] F. C. Chen and H. K. Khalil, "Adaptive control of a class of nonlinear discrete-time systems using neural networks," *IEEE Trans. Automat. Contr.*, vol. 40, pp. 791–801, 1995.
[22] M. A. Brdyś, Z. Ogonowski, and Z. Świder, "Predictive control of induction motor," School Electron. Elect. Eng., Univ. Birmingham, U.K., Tech. Rep., 1993.
[23] G. J. Kulawski and M. A. Brdyś, "Adaptive control of nonlinear plants using neural networks—Application to a flux control in AC drive system," in *Proc. Int. Conf. CONTROL'94*, Warwick, U.K., 1994, pp. 1472–1477.
[24] M. A. Brdyś, Z. Ogonowski, and Z. Świder, "Generalized predictive control of induction motor," in *Proc. 3rd European Contr. Conf.*, Rome, 1995, vol. 4, pp. 3253–3258.
[25] S. Sastry and M. Bodson, *Adaptive Control: Stability, Convergence and Robustness*. Englewood Cliffs, NJ: Prentice-Hall, 1989.
[26] M. M. Gupta, D. H. Rao, and P. N. Nikiforuk, "Dynamic neural network based inverse kinematic transformation of two- and three-linked robots," in *Proc. 12th IFAC World Congr.*, Sydney, Australia, 1993, vol. 3, pp. 289–296.
[27] P. S. Sastry, G. Santharam, and K. P. Unnikrishnan, "Memory neuron networks for identification and control of dynamical systems," *IEEE Trans. Neural Networks*, vol. 5, pp. 306–319, 1994.
[28] A. G. Parlos, K. T. Chong, and A. F. Atiya, "Application of the recurrent multilayer perceptron in modeling complex process dynamics," *IEEE Trans. Neural Networks*, vol. 5, pp. 255–266, 1994.
[29] J. A. K. Suykens, B. L. R. De Moor, and J. Vandewalle, "Nonlinear system identification using neural state-space models, applicable to robust control design," *Int. J. Contr.*, vol. 62, no. 1, pp. 129–152, 1995.
[30] J. A. K. Suykens, B. L. R. De Moor, and J. Vandewalle, "Stability criteria for neural control systems," in *Proc. 3rd European Contr. Conf.*, Rome, Italy, 1995, pp. 2766–2771.
[31] J. A. K. Suykens, J. Vandewalle, and B. L. R. De Moor, "Nonlinear H_∞ control for continuous-time recurrent neural networks," in *Proc. 4th European Contr. Conf.*, Brussels, Belgium, 1997.
[32] H. Verrelst, K. Van Acker, J. A. K. Suykens, B. Motmans, B. L. R. De Moor, and J. Vandewalle, "NL_q neural control theory: Case study for a ball and beam system," in *Proc. 4th European Contr. Conf.*, Brussels, 1997.
[33] K. I. Funahashi and Y. Nakamura, "Approximation of dynamical systems by continuous time recurrent networks," *Neural Networks*, vol. 6, pp. 801–806, 1993.
[34] L. Jin, P. N. Nikiforuk, and M. M. Gupta, "Approximation of discrete-time state-space trajectories using dynamic recurrent neural networks," *IEEE Trans. Automat. Contr.*, vol. 40, pp. 1266–1270, 1995.
[35] C. C. Ku and K. Y. Lee, "Diagonal recurrent neural networks for dynamic systems control," *IEEE Trans. Neural Networks*, vol. 6, pp. 144–156, 1995.
[36] L. Jin, P. N. Nikiforuk, and M. M. Gupta, "Model matching control of unknown nonlinear systems using recurrent neural networks," in *Proc. 12th IFAC World Congr.*, Sydney, Australia, 1993, vol. 1, pp. 337–344.
[37] L. Jin, P. N. Nikiforuk, and M. M. Gupta, "Adaptive control of discrete-time nonlinear systems using recurrent neural networks," *Proc. Inst. Elect. Eng. Contr. Theory Applicat.*, vol. 141, pp. 169–176, 1994.
[38] A. Delgado, C. Kambhapati, and K. Warwick, "Dynamic recurrent neural network for system identification and control," *Proc. Inst. Elect. Eng. Contr. Theory Applicat.*, vol. 142, pp. 307–314, 1995.
[39] K. S. Narendra and K. Parthasarathy, "Gradient methods for the optimization of dynamical systems containing neural networks," *IEEE Trans. Neural Networks*, vol. 2, pp. 252–262, 1991.
[40] M. A. Brdyś and T. Du, "Algorithms for joint state and parameter estimation in induction motor drive systems," in *Proc. Int. Conf. CONTROL'91*, Edinburgh, U.K., 1991, pp. 915–920.
[41] T. Du and M. A. Brdyś, "Implementations of extended Luenberger observers for joint state and parameter estimation of pwm induction motor drive," in *Proc. 5th European Conf. Power Electron. (IPE)*, Brighton, England, 1993.



Mietek A. Brdyś (M'90) received the M.Sc. degree in electronic engineering and the Ph.D. and the D.Sc. degrees in control systems from the Institute of Automatic Control at the Warsaw University of Technology, Poland, in 1970, 1974, and 1980, respectively.

From 1974 to 1983, he held the posts of Assistant Professor and Associate Professor at the Warsaw University of Technology. In 1992 he became Full Professor of Control Systems. Between 1978 and 1995, he held various visiting faculty positions at the University of Minnesota, City University, De Montfort University and University Polytecnic of Catalonia. Since January 1989, he has held the post of Senior Lecturer in the School of Electronic and Electrical Engineering at The University of Birmingham, U.K. He has served as the Consultant for the Honeywell Systems and Research Center in Minneapolis, GEC Marconi and Water Authorities in U.K., France, Germany, Spain, and Poland. His research is supported by the U.K. Research Council and industry and the European Commission. He is the author or coauthor of about 100 refereed papers and five books. His current research interests include intelligent control of nonlinear and uncertain systems, robust monitoring, and operational control with application to environmental systems.

Dr. Brdyś is a Chartered Engineer, a Member of the Institute of Electrical Engineers, a Fellow of IMA, and a member of IFAC Technical Committee on Large Scale Systems.



Grzegorz J. Kulawski was born in Poland in 1970. He received the B.Eng. degree in electronic and electrical engineering in 1993 and the Ph.D. degree in 1998, both from The University of Birmingham, U.K.

At present he is with Shell International Exploration and Production B.V., Research and Technical Services, Rijswijk, The Netherlands. His research interests include neural networks, nonlinear adaptive control, and modeling of industrial processes.