Design and Real-Time Implementation of an Internal Model Speed Control for an Induction Motor

Haider A. F. Mohamed, S. S. Yang, M. Moghavvemi
Centre for Research in Applied Electronics (CRAE)
University of Malaya
50603 Kuala Lumpur, Malaysia
haider@um.edu.my

Abstract— **This paper carries out the design and real time implementation of an Internal Model Controller (IMC) to control the speed of an induction motor and precisely a squirrel cage type. The scheme in this paper is constructed using a model and a controller both being Artificial Neural Network (ANN) based. This ANN-based control scheme has been chosen because of its ability to handle the strong nonlinearities of the induction motor. The performance of the controller is tested by applying different types of input signals as well as load torque disturbances. Whether loaded or unloaded, the proposed internal model controller has proved to achieve high performance and accuracy.**

Keywords—**ANN, system identification, inverse model, induction motor speed control**

I. Introduction

Recently, there has been a considerable interest in the area of nonlinear black-box modelling and dynamic system controllers with structures based on neural networks [1-2]. Furthermore, from a dynamic control system and especially adaptive control techniques perspective neural network can deal with substantially greater range of uncertainty than can be tolerated by algorithms for other adaptive systems [3-4].

Adaptive control techniques have been developed for systems that must perform over large ranges of uncertainties due to large variations in parameter values, environmental conditions, and signal inputs. These adaptive techniques generally incorporate a second feedback loop, which is outside the first feedback loop. An example is the internal model control (IMC) in which there is emphasize on the role of system forward and inverse models [5]. IMC has been thoroughly examined and shown to yield robustness and stability [6]. Moreover, IMC extends readily to nonlinear control systems [7].

One way of improving the performance of nonlinear systems is to combine the powers of adaptive, nonlinear, and intelligent controllers using internal model control structures. Examples can be found in [8-11]. The key characteristic of this type of control strategy is having the inverse controller and the internal model. Using this internal model, the effect of uncertainties can be suppressed with the generated feedback signal.

One of the important nonlinear systems widely used by the industry is the induction motor, especially the squirrel-cage induction motor, which enjoys several inherent advantages like simplicity, ruggedness, low cost, reliability and compactness [12]. The main problem with this motor is that it's a highly coupled, nonlinear dynamic plant, and in addition, many of its parameters vary with time and operating condition [13].

The aim of this paper is to design robust induction motor speed controller that uses the internal model control structure based on neural network forward and inverse models. This will take care of the nonlinearities of the induction motor and adapt the system for parameter variations and perturbations.

II. Internal Model Control

In internal model control, the system model is placed in parallel with the real plant as shown in Fig. 1. Here, the nonlinear operators denoted by *P*, *M* and *C* are the plant, plant model, and controller respectively.

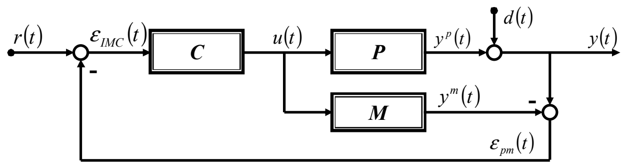

Figure 1. General block diagram of an IMC system.

The difference between the plant and model outputs ε_{pm} is used for feedback purposes and given by

$$\varepsilon_{pm}(t) = y(t) - y^m(t) \qquad (1)$$

$$y(t) = y^p(t) + d(t) \qquad (2)$$

where $d(t)$ is the disturbance.

This feedback signal is then processed by the controller subsystem in the forward path. Thus the real error ε_{IMC} which should be controlled is

$$\varepsilon_{IMC}(t) = r(t) - \varepsilon_{pm}(t) \qquad (3)$$

The properties of IMC dictate that this part of the controller should be related to the inverse model of the real plant. Given the forward and inverse network models, the realization of IMC using neural networks will be straightforward [14]. In this structure, the relationship between the inputs and outputs is:

$$y = \frac{PC}{1+C(P-M)} r + \frac{1-MC}{1+C(P-M)} \tag{4}$$

A. *IMC filter*

The discussion so far has considered only the idealized case of a perfect model, leading to perfect control. In practice, however, a perfect model can never be obtained. In addition, the infinite gain required by a perfect controller would lead to sensitivity problems under model uncertainty. The filter F is introduced to alleviate these problems. By suitable design, the filter can be selected to reduce the gain of the feedback system, thereby moving away from the ideal controller. Fig. 2 shows the IMC diagram with the filter. Here, the controller is $\hat{C}$:

$$\hat{C} = FC \tag{5}$$

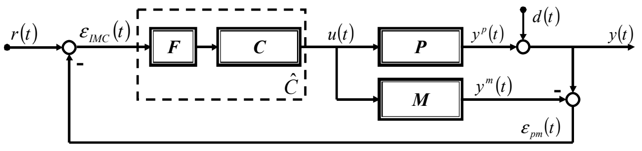

Figure 2. IMC with a filter block diagram.

The subsystem *F* is usually a linear filter which can be designed to introduce desirable robustness and tracking response to the closed-loop. The double lines used in the block diagram emphasize that the operators are nonlinear and that the usual block diagram manipulations do not hold. A second role of the filter is to project the signal ε_{IMC} into the appropriate input space of the controller. The structure given in Fig. 2 has shown to have good robustness against uncertainties [6, 15].

To design the IMC filter, the IMC performance and system type have to be discussed first;

B. *IMC performance*

The sensitivity function $\rho(s)$ which relates the external inputs r and d to the feedback error ε_{IMC} is given by

$$\frac{\varepsilon_{IMC}}{d-r} = \frac{y}{d} = \frac{1-MC}{1+C(P-M)} \overset{\Delta}{=} \rho(s) \tag{6}$$

The complimentary sensitivity function $\sigma(s)$, which determines the system robustness, is found by subtracting $\rho(s)$ from unity as follows

$$\frac{y}{r} = \frac{PC}{1+C(P-M)} \overset{\Delta}{=} \sigma(s) \tag{7}$$

When the model is exact (*P*=*M*), (6) and (7) reduce to

$$\rho(s) = 1 - MC \tag{8}$$

$$\sigma(s) = MC \tag{9}$$

Through the above IMC parameterization, the controller *C* is related to $\rho(s)$ and $\sigma(s)$ in a very simple linear manner which make the design of *C* easy. The effect of the classical controller on sensitivity function $\rho(s)$ and complimentary sensitivity function $\sigma(s)$ is more complex.

C. *Sensitivity and system types*

System types were defined in control system theory to classify the asymptotic closed–loop behavior [6]. Thus, for a system of type *m* the following should yield

$$\lim_{s\to 0} \frac{\rho(s)}{s^k} = 0 \, , \ 0 \le k < m \tag{10}$$

Using (4), this definition becomes:

$$\lim_{s\to 0} \frac{1-MC}{1+C(P-M)} \frac{1}{s^k} = 0 \, , \ 0 \le k < m \tag{11}$$

Condition (9) is satisfied if and only if $(1-MC)$ has *m* zeros at the origin which is the case if and only if:

$$\lim_{s\to 0} \frac{d^k}{ds^k} (1-MC) = 0 \, , \ 0 \le k < m \tag{12}$$

From the internal model control theory [8], the controller *C* is determined such that the integral squared error (ISE)

$$\left\| \varepsilon_{IMC} \right\|_2^2 = \int_0^\infty \varepsilon_{IMC}^2(t) dt \tag{13}$$

is minimized for a particular input *u*.

For the ISE to be bounded, the error has to vanish as $t \to \infty$. This implies that the controller *C* for a well-trained model has to generate a type 1 system. For the system to be type 1, it should fulfill the following condition

$$\lim_{s\to 0} MC = 1 \tag{14}$$

D. *Filter design*

For robustness *C* has to be augmented by a low pass filter *F*. In principle both the structure and the parameters of *F* should be determined such that an optimal compromise between performance and robustness is reached. To simplify the design task, the filter structure is fixed and small number of filter parameters (usually just one) is searched to obtain desired robustness characteristics. Here, it is logical to choose *F* such that the closed-loop system retains its asymptotic tracking properties. For systems of type *m*, *F* has to satisfy (10). Thus, the new condition is

$$\lim_{s\to 0}\frac{d^k}{ds^k}\left(1-MCF\right)=0\ ,\ 0\le k<m \tag{15}$$

In addition, one filter parameter with unity steady state gain of the following form is used:

$$F(s)=\left(\beta_{m-1}s^{m-1}+\cdots+\beta_1 s+1\right)\frac{1}{\left(\lambda s+1\right)^n} \tag{16}$$

where λ is the adjustable filter parameter, n is the order and selected large enough to make $\hat{C}$ proper and β_i is chosen to satisfy (15).

The simplest filter of type 1 that has the form of (16) and satisfies (15) is

$$F(s)=\frac{1}{\left(\lambda s+1\right)^n} \tag{17}$$

III. ARTIFICIAL NEURAL NETWORKS

ANNs offer the advantage of performance improvement through learning using parallel and distributed processing.

A. Network architecture

It has been formally shown by [16] that artificial neural networks with at least one hidden layer and sufficient number of neurons are able to approximate a wide class of continuous nonlinear functions within an arbitrarily small error margin. Fig. 3 shows a typical two-layer artificial neural network.

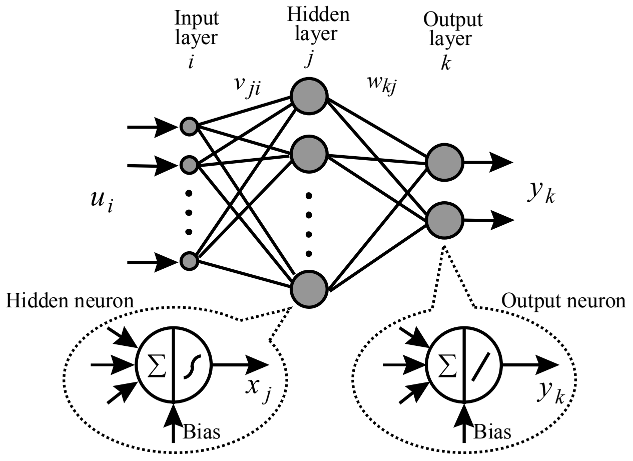

Figure 3. A two layer artificial neural network.

Defining the following terms

$$H_j=\sum_i v_{ji}\,u_i\ ,\ j=1,2,\dots,h \tag{18}$$

$$I_k=\sum_j w_{kj}\,x_j\ ,\ k=1,2,\dots,m \tag{19}$$

where H_j is the combined or net input to hidden-layer unit *j*, while I_k is the net input to unit *k* of the output-layer.

Outputs computed by unit *j* of the hidden-layer and unit *k* of the output-layer are given by:

$$x_j=f\left(H_j\right),\ j=1,2,\dots,h \tag{20}$$

$$y_k=f\left(I_k\right),\ k=1,2,\dots,m \tag{21}$$

respectively, where *f* is an arbitrary, bounded, differentiable function (the activation function). Therefore, unit *k* output y_k will have the following expression:

$$y_k=f\left(I_k\right)=f\left[\sum_j w_{kj}f\left(\sum_i v_{ji}u_i\right)\right] \tag{22}$$

B. Training algorithm

Based on the iterative gradient algorithm method, the backpropagation training algorithm is designed to minimize the mean square error between the output of the feedforward network and the desired output [16-17]. In this method, errors are propagated backwards, layer by layer, with weights correction being made to the subsequent layer in an iterative manner, Fig. 4. The process is repeated a number of times for each pattern in the training set until the error criterion is reached. For minimization, we calculate the predicted error. First, every iteration step *s*, the equivalent error δ_k of neuron *k* in the output layer is given by:

$$\delta_k(s)=\varepsilon_k(s)=y_k(s)-\hat{y}_k(s) \tag{23}$$

Then, the equivalent hidden layer error δ_j of neuron *j* is:

$$\delta_j(s)=\frac{df\left[H_j(s)\right]}{dH_j(s)}\sum_k \delta_k(s)w_{kj} \tag{24}$$

Then, weights connecting the hidden and output layers are adjusted according to:

$$\begin{aligned} w_{kj}(s)&=w_{kj}(s-1)+\Delta w_{kj}(s)\\ \Delta w_{kj}(s)&=\eta\delta_k(s)x_j(s)+\beta\Delta w_{kj}(s-1)\end{aligned} \tag{25}$$

where η and β are the learning rate and the momentum parameters respectively. While the weights connecting the input and hidden layer are corrected based on

$$\begin{aligned} v_{ji}(s)&=v_{ji}(s-1)+\Delta v_{ji}(s)\\ \Delta v_{ji}(s)&=\eta\delta_j(s)u_i(s)+\beta\Delta v_{ji}(s-1)\end{aligned} \tag{26}$$

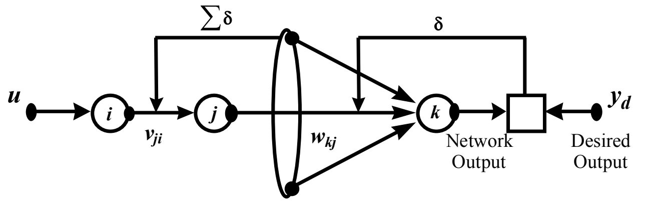

Figure 4. Backpropagation algorithm.

C. Generalized inverse model

Conceptually, the simplest approach to obtain the inverse model of the system is the generalized inverse modeling method shown in Fig. 5. Here, a synthetic training signal (the plant input) is introduced to the system. The plant output is then used as input to the network. The network output is compared with the training signal (the system input) and this error is used to train the network. This structure will clearly force the network to represent the inverse of the plant. The resulting inverse model M will be controller C in Fig. 2.

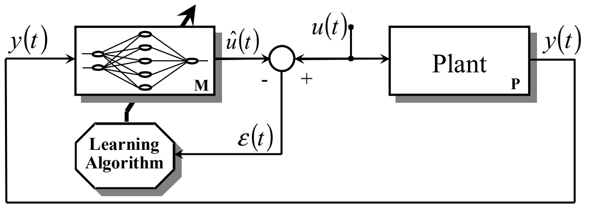

Figure 5. Generalized inverse plant modeling.

Commonly, the first stage in this method is to collect a data set Z^N that covers the overall plant behavior. Here Z^N is

$$Z^N = \left[u^N, y^N\right] \tag{27}$$

where

$$u^N = [u(1), u(2), \cdots, u(N)] \tag{28}$$

$$y^N = [y(1), y(2), \cdots, y(N)] \tag{29}$$

The objective with inverse plant modeling is to formulate a controller, such that the overall controller/plant architecture has a unity transfer function, i.e. if the plant can be described as

$$\hat{y}(t \mid \theta) = g[\varphi(t), \theta] \tag{30}$$

where $\hat{y}$ denotes the model output and g is a non-linear function parameterized by θ which is a finite dimensional parameter vector (the weights of the network in our case) and $\varphi(t)$ is the regressors vector.

The inverse network is trained as the inverse of the plant model, i.e.

$$\hat{u}(t \mid \theta) = g^{-1}[\varphi(t), \theta] \tag{31}$$

where $\hat{u}$ is the output of the inverse model and also will be the control signal.

However, modeling errors perturb the transfer function away from unity. Therefore, $\hat{g}^{-1}[\varphi(t), \theta]$ is used instead of $g^{-1}[\varphi(t), \theta]$. Thus

$$\hat{u}(t \mid \theta) = \hat{g}^{-1}[\varphi(t), \theta] \tag{32}$$

To obtain the inverse model in the generalized training method, a network is trained off-line to minimize the criterion:

$$W_N\left(\theta, Z^N\right) = \frac{1}{N}\sum_{t=1}^{N}[u(t) - \hat{u}(t \mid \theta)]^2 \tag{33}$$

Once the modeling of the inverse plant is carried out, the model is applied as the controller for the system by inserting the desired output (the reference) instead of the system output.

IV. IMC DESIGN FOR INDUCTION MOTORS

The induction motor being used in this paper is a three phase squirrel-cage (380v, 50Hz, 4 -pole, 0.1 kW) and Y-connected. The input to the plant is a voltage signal that will be converted through a voltage-frequency converter to give the desired speed of the motor. The rotor speed is read by a tachometer and presented as a voltage signal.

A. Data collection and presentation

The input data set is chosen randomly in the range from 0 to 1390 rpm to fully excite the whole speed range which allows the network to recognize the system's behavior. The input-output data set is shown in Fig. 6. The data set will be divided into two sets; one for training and the other for validation.

$$Z^N = [u(1), \cdots, u(N); y(1), \cdots, y(N)] \tag{34}$$

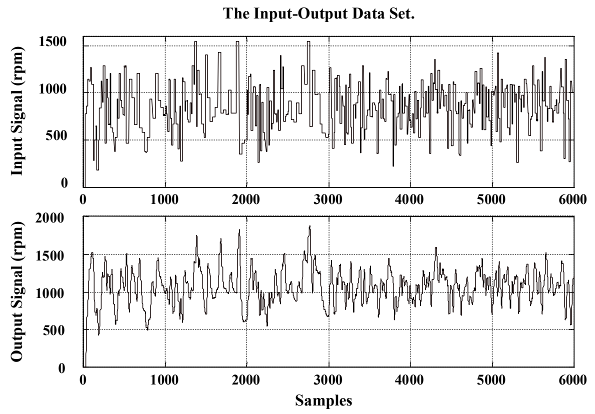

Figure 6. Input-output data set.

B. Forward plant model

The work of this paper is a continuation of a previous work published in [18]. Also, the data set used to train the inverse model is the same.

C. *Inverse plant model*

The system is a single-input single-output (SISO) system. To obtain the inverse plant model in the generalized method, the output *y(t)* is fed to the training algorithm as the input and *u(t)* as the output. The regressors are chosen so that the inverse model will be of a second order. This is done by choosing two past inputs and two past outputs. The regressor vector is as follows

$$\varphi(t) = [y(t-1), y(t-2), u(t-1), u(t-2)] \tag{35}$$

The network structure is a two-layer hyperbolic tangent sigmoidal feedforward architecture (one hidden layer with a *tanh* activation function and one output layer with a linear activation function).

D. *Training the Network*

The weights for both the hidden layer and the output layer are initially randomized around the values of -0.5 and +0.5 before the training. This is useful so that the training would fall in global minima rather than local minima [19]. The training showed good results using five hidden neurons and 3000 samples as a training set.

The aim now is to obtain $\hat{g}^{-1}[\varphi(t), \theta]$ that represents the inverse model and produces the control action $\hat{u}(t \mid \theta)$,

$$\hat{u}(t \mid \theta) = \hat{g}^{-1}[\varphi(t), \theta] \tag{36}$$

During the back propagation iteration, the Sum of Squared Errors (SSEs) are calculated as follows:

$$SSE = \sum_{i=1}^{N} [u(t) - \hat{u}(t)]^2 < \alpha \tag{37}$$

where α is the criterion threshold.

The results of the inverse plant model training algorithm are shown in Fig. 7 where α was chosen as 1. The network architecture contains one hidden layer with six neurons and a hyperbolic tangent (*tanh*) activation function and one output layer with a linear activation function.

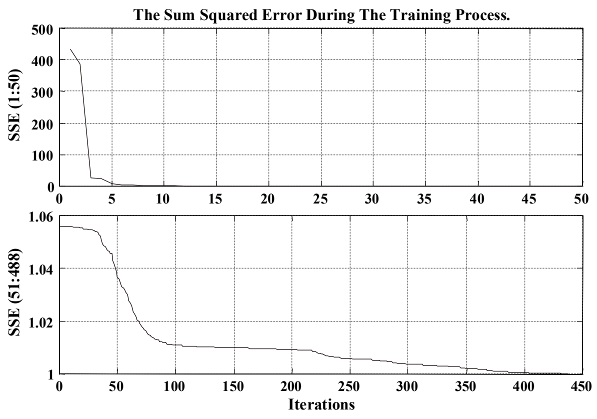

Figure 7. Sum of squared errors during training process.

V. EXPERIMENTAL RESULTS

To implement the IMC scheme, the controller is programmed using the C++ programming language. The control action in the program is performed during the interrupt service routine. This is done at each sampling time.

The following sections describe the step, load disturbance, and tracking responses of the real time experimentation:

A. *Step response and disturbance rejection performance*

A unit step reference signal representing 1390 rpm is fed to the controller while a load torque step signal of 2 N.m. (full load) is applied to the shaft during the period of 4 to 8 seconds. The results are shown in Fig. 8.

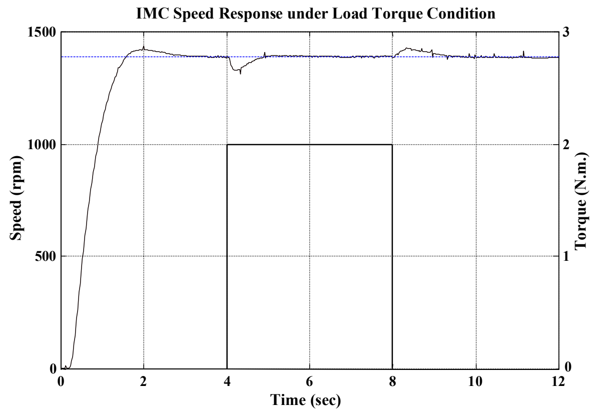

Figure 8. Unit step speed response under IMC scheme.

It can be seen from the figure that speed of the induction motor followed the reference signal with an acceptable steady state error equals to 0.2878%. The transient response shows a maximum overshoot of 1430rpm. In addition, the internal model controller could recover from the disturbance caused by the applied load torque and the induction motor speed followed the reference signal after a short time.

B. *Speed tracking performance*

To check the controller's performance over different types of speed reference signals, sine, ramp, and square wave signals are fed to the system and the results are recorded in Figs. 9, 10, and 11 respectively.

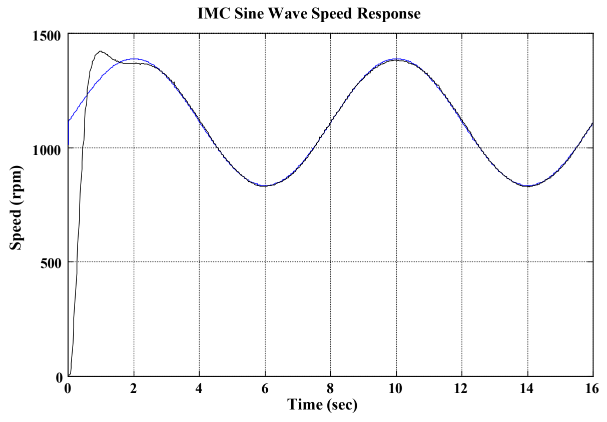

Figure 9. Speed response to a sine-wave reference signal.

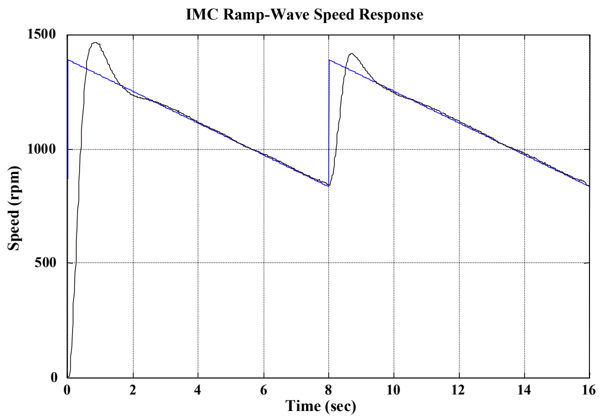

Figure 10. Speed response to a ramp wave reference signal.

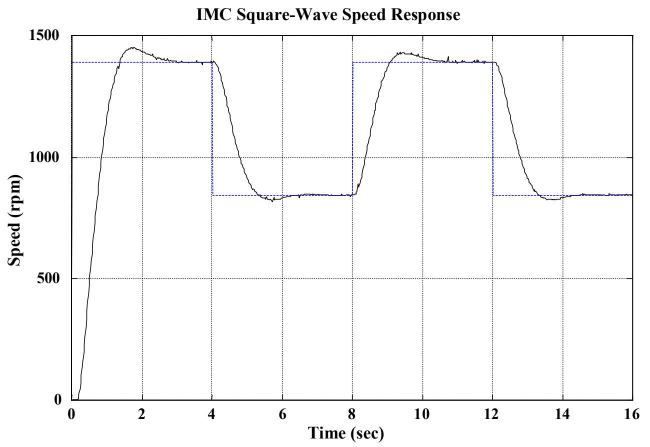

Figure 11. Speed response to a square wave reference signal.

From these reference tracking plots, Table I is constructed to show the maximum and the minimum errors recorded for each of the three reference signals.

TABLE I. STEADY STATE ERROR ANALYSIS

Reference Signal	**Steady State Error**	
	Minimum Error	***Maximum Error***
Sine	-0.59%	+0.57%
Ramp	-0.53%	+0.14%
Square	-0.06%	+0.06%

From Table I, small steady state errors can be noticed for all three responses. These errors are considered to be acceptable since the maximum and minimum errors are around ±0.6% of the reference signal. The overshoot also shows acceptable values where the maximum overshoot is 3% for the square waveform signal.

VI. CONCLUSION

In this paper, nonlinear black-box inverse modeling of an induction motor is carried out using the back propagation training algorithm. Half of the experimentally collected data was employed for ANN training and the other half was used for model validation. The obtained ANN inverse model cascaded with a filter is then used together with an ANN forward model from a previous work to construct an internal model control structure. The real time implementation of the neural network internal model control scheme has been presented and its performance has been tested over different types of reference signals and applied load torques. The controller tracked the given reference signals and overcame the applied disturbance.

REFERENCES

[1] B.Karanayil, M. F. Rahman and C. Grantham, “Implementation of an on-line resistance estimation using artificial neural networks for vector controlled induction motor drive”, IECON '03 29th Annual Conf. of the IEEE Industrial Electronics Society, Vol. 2, pp. 1703-1708, 2003.

[2] R. S. Toqeer and N. S. Bayindir, “Neurocontroller for induction motors”, ICM 2000. Proc. 12th Int. Conf. on Microelectronics, pp. 227-230, 2000

[3] L. Ljung and J. Sjoberg, “A system identification perspective on neural nets”, Technical Repor, No. LiTH-ISY-R-1373 at www.control.isy.liu.se, 1992.

[4] B. Bavarian, “Introduction to neural networks for intelligent control”, IEEE Control Systems Magazine, Vol. 8(2). pp. 3-7, 1988.

[5] C. E. Garcia and M. Morari, “Internal model control: a unifying review and some new results”, Ind. Engineering Chemistry Process Des. Dev.,Vol. 21, pp. 308-323, 1982.

[6] M. Morari and E. Zafiriou, “Robust process control”, Prentice Hall, Englewood Cliffs, NJ, 1989.

[7] C. G.Economou, M. Morari and B. O. Palsson, “Internal model control: extension to nonlinear systems”, Ind. Engineering Chemistry Process Des. Dev., Vol. 25, pp. 403-411, 1986.

[8] Han-Xiong Li and Hua Deng, “An approximate internal model-based neural control for unknown nonlinear discrete processes”, IEEE Trans. Neural Networks, Vol. 17(3), pp. 659-670, 2006.

[9] S. Bel Hadj Ali, A. El Abed-Abdelkrim, and M. Benrejeb, “An internal model control strategy using artificial neural networks for a class of nonlinear systems”, IEEE Int. Conf. Systems, Man and Cybernetics, Vol. 5, pp. 4-7, 2002.

[10] I. Rivals and L. Personnaz, “Nonlinear internal model control using neural networks: Application to processes with delay and design issues”, IEEE Trans. Neural Networks, Vol. 11(1), pp. 80-90, 2000.

[11] A. Fink and O. Nelles, “Nonlinear internal model control based on local linear neural networks”, IEEE Int. Conf. Systems, Man and Cybernetics, Vol. 1, pp. 117-122, 2001.

[12] P. Mehrotra, J. E. Quaicoe and R. Venkatesan, “Development of an Artificial Neural Network Based Induction Motor Speed Estimator”, PESC '96 IEEE Power Electronics Specialists, Vol. 1, pp. 682-688, 1996.

[13] P. Mehrotra, J. E. Quaicoe and R. Venkatesan, “Induction motor speed estimation using artificial neural networks”, IEEE Canadian Conf. Electrical and Computer Engineering, Vol. 2, pp. 607-610, 1996.

[14] K. J. Hunt and D. Sbarbaro, “Neural networks for nonlinear internal model control”, Proc. IEE Pt. D., Vol. 138, pp. 431-438, 1991.

[15] G. Lightbody and G. W. Irwin, “Nonlinear control structures based on embedded neural system models”, IEEE Trans. Neural Networks, Vol. 8(3), pp. 553-567, 1997.

[16] T. Fukuda and T. Shibata, “Theory and application of neural networks for industrial control systems”, IEEE Trans. on Ind. Electronics, Vol. 39(6), pp. 472-489, 1992.

[17] M. Weber, P. B. Crilly and W. E. Blass, “Adaptive noise filtering using an error-backpropagation neural network”, IEEE Trans. on Instrum. Meas., Vol. 40(5), pp. 820-825, 1991.

[18] H. A. F. Mohamed and S. Yaacob, “Direct inverse speed control of induction motor using artificial neural networks”, MS2006 Proc. Int. Conf. Modeling and Simulation, pp. 125-130, 2006.

[19] D. W. Patterson, “Artificial neural networks: theory and applications”, Prentice Hall, Simon and Schuster (Asia) Pte. Ltd., Singapore, 1996.