Direct Neural Network Control via Inverse Modelling: Application on Induction Motors

Haider A. F. Almurib[1], Ahmad A. Mat Isa[2] and Hayder M.A.A. Al-Assadi[2]
[1]*Department of Electrical & Electronic Engineering, The University of Nottingham Malaysia Campus Semenyih, 43500*
[2]*Faculty of Mechanical Engineering, University Technology MARA (UiTM) Shah Alam, 40450*
Malaysia

1. Introduction

Applications of Artificial Neural Networks (ANNs) attract the attention of many scientists from all over the world. They have many advantages over traditional algorithmic methods. Some of these advantages are, but not limited to; ease of training and generalization, simplicity of their architecture, possibility of approximating nonlinear functions, insensitivity to the distortion of the network and inexact input data (Wlas et al., 2005). As for their applications to Induction Motors (IMs), several research articles have been published on system identification (Karanayil et al., 2003; Ma & Na, 2000; Toqeer & Bayindir, 2000; Sjöberg et al. 1995; Yabuta & Yamada, 1991), on control (Kulawski & Brys, 2000; Kung et al., 1995; Henneberger & Otto, 1995), on breakdown detection (Raison, 2000), and on estimation of their state variables (Simoes & Bose, 1995; Orłowska-Kowalska & Kowalski, 1996).
The strong identification capabilities of artificial neural networks can be extended and utilized to design simple yet good performance nonlinear controllers. This chapter contemplates this property of ANNs and illustrates the identification and control design processes in general and then for a given system as a case study.
To demonstrate its capabilities and performance, induction motors which are highly nonlinear systems are considered here. The induction machine, especially the squirrel-cage induction motor, enjoys several inherent advantages like simplicity, ruggedness, efficiency and low cost, reliability and compactness that makes it the preferred choice of the industry (Vas, 1990; Mehrotra et al., 1996; Wishart & Harley, 1995; Merabet et al., 2006; Sharma, 2007). On the other hand, advances in power switching devices and digital signal processors have significantly matured voltage-source inverters (VSIs) with the associated pulse width modulation (PWM) techniques to drive these machines (Ebrahim at el., 2010). However, IMs comprise a theoretically challenging problem in control, since they are nonlinear multivariable time-varying systems, highly coupled, nonlinear dynamic plants, and in addition, many of their parameters vary with time and operating condition (Mehrotra et al., 1996a; 1996b; Merabet et al., 2006).

2. System identification

This chapter will carry out the system identification of an induction motor using the artificial neural network and precisely the Back Propagation Algorithm. The procedure used to identify the system is as described in Fig.1.

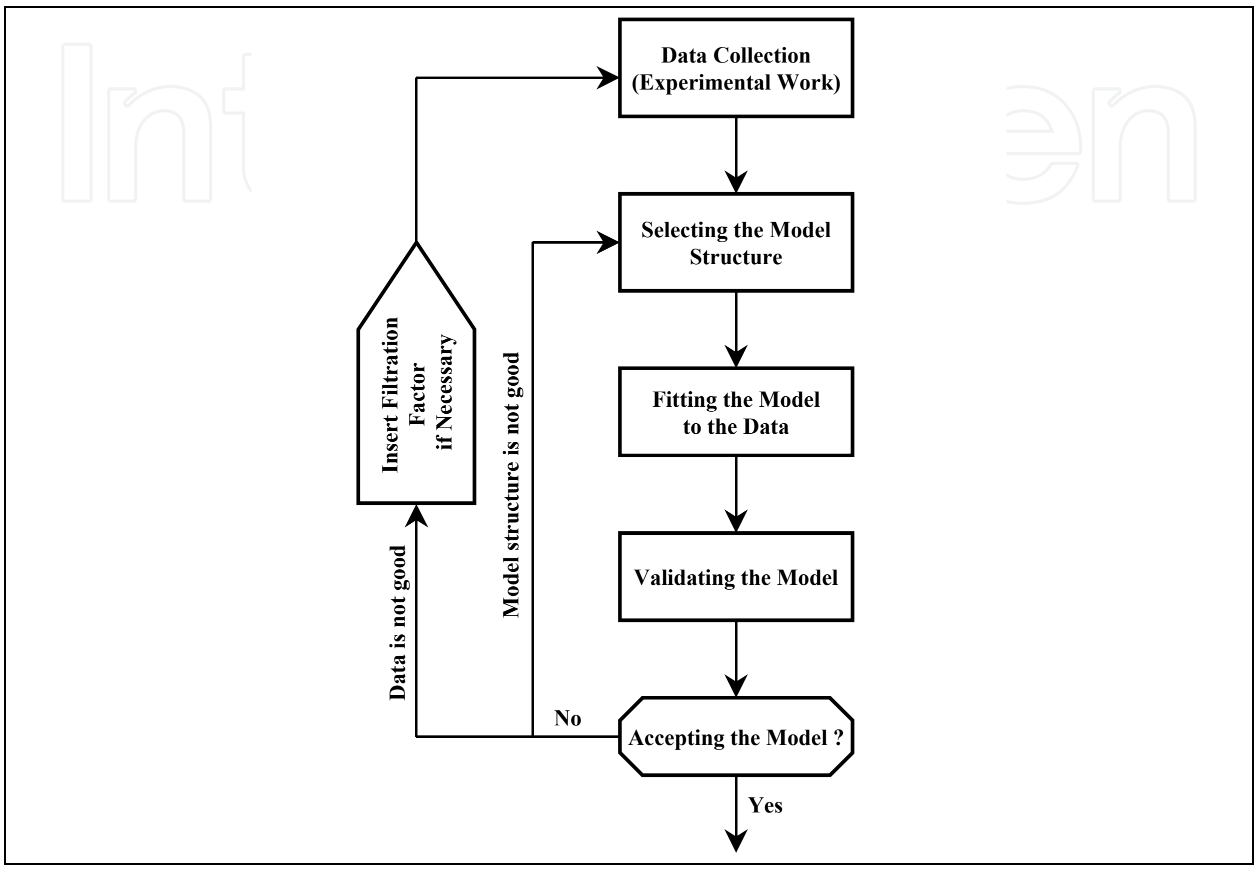

Data Collection (Experimental Work)
Selecting the Model Structure
Fitting the Model to the Data
Validating the Model
Accepting the Model ?
Yes
No
Insert Filtration Factor if Necessary
Data is not good
Model structure is not good

Fig. 1. System identification loop

Now, the system identification problem would be as follows: We have observed inputs, *u(t)*, and outputs, *y(t)*, from the plant under consideration (induction motor):

$$u^t = \left[u(1), u(2), \cdots, u(t)\right] \tag{1}$$

$$y^t = \left[y(1), y(2), \cdots, y(t)\right] \tag{2}$$

where u^t is the input signal to the plant (input to the frequency inverter) and y^t is the output signal (measured by the tacho-meter representing the motor's speed). We are looking for a relationship between past $\left[u^{t-1}, y^{t-1}\right]$ and future output, *y(t)*:

$$\hat{y}(t \mid \theta) = g\left[\varphi(t), \theta\right] \tag{3}$$

where $\hat{y}$ denotes the model output which approximates the actual output $y(t)$, g is a nonlinear mapping that represents the model, $\varphi(t)$ is the regression vector given by

$$\varphi(t) = \varphi\left(u^{t-1}, y^{t-1}\right) \tag{4}$$

and its components are referred to as regressors. Here, θ is a finite dimensional parameter vector, which is the weights of the network in our case (Bavarian, 1988; Ljung & Sjöberg, 1992; Sjöberg et al. 1995).
The objective in model fitting is to construct a suitable identification model (Fig. 2) which when subjected to the same input $u(t)$ to the plant, produces an output $\hat{y}(t)$ which approximates $y(t)$. However, in practice, it is not possible to obtain a perfect model. The solution then is to select θ in Eq. (3) so as to make the calculated values of $\hat{y}(t|\theta)$ fit to the measured outputs $y(t)$ as close as possible. The fit criterion will be based on the least square method given by

$$\min_{\theta} V_N\left[\theta,\varphi(t)\right] \tag{5}$$

where

$$V_N\left[\theta,\varphi(t)\right]=\frac{1}{N}\sum_{t=1}^{N}\left[y(t)-\hat{y}(t|\theta)\right]^2 \tag{6}$$

Hence, the error ε is given by

$$\varepsilon(t)=y(t)-\hat{y}(t|\theta) \tag{7}$$

This is illustrated in Fig. 2.

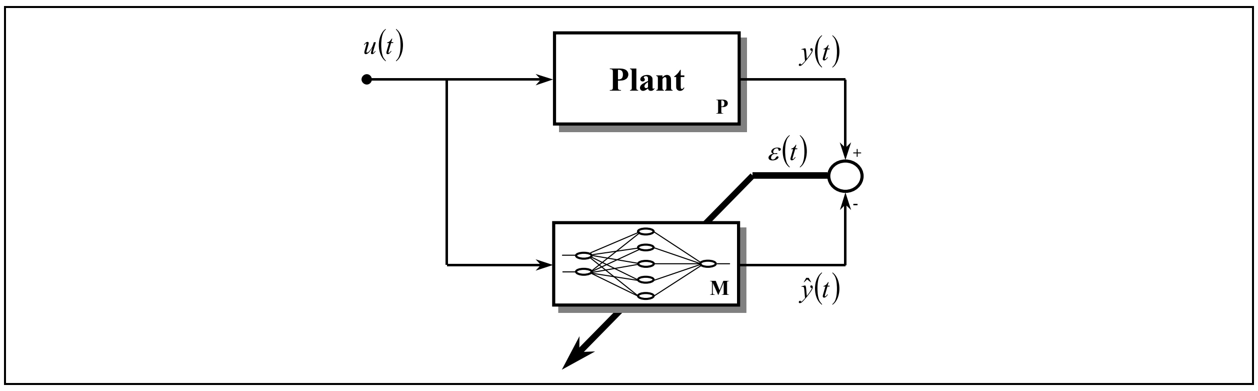


Fig. 2. Forward plant modelling

3. Artificial Neural Networks

Strong non-linearities and model uncertainty still pose a major problem for control engineering. Adaptive control techniques can provide solutions in some situations however in the presence of strongly non-linear behaviour of the system traditional adaptive control algorithms do not yield satisfactory performance. Their inherent limitations lie in the linearization based approach. A linear model being a good approximation of the non-linear plant for a given operation point cannot catch up with a fast change of the state of the plant and poor performance is observed until new local linear approximation is built.
Artificial neural networks offer the advantage of performance improvement through learning using parallel and distributed processing. These networks are implemented using massive connections among processing units with variable strengths, and they are attractive for applications in system identification and control.

3.1 The network architecture

Figure 3 shows a typical two-layer artificial neural network. It consists of two layers of simple processing units (termed neurons).

The outputs computed by unit j of the hidden-layer and unit k of the output-layer are given by:

$$x_j = f_h\left(H_j\right) \qquad j = 1,\ 2,\ ...,\ h \tag{8}$$

$$y_k = f_o\left(I_k\right) \qquad k = 1,\ 2,\ ...,\ m \tag{9}$$

respectively, where f_h and f_o are the bounded and differentiable activation functions. Thus, the output unit k will result in the following:

$$y_k = f\left[\sum_j w_{kj} f\left(\sum_i v_{ji} u_i\right)\right] \tag{10}$$

where y_k here is the vector representing the network output.

It has been formally shown (Lippman, 1987; Fukuda & Shibata, 1992) that Artificial Neural Networks with at least one hidden layer with a sufficient number of neurons are able to approximate a wide class continuous non-linear functions to within an arbitrarily small error margin.

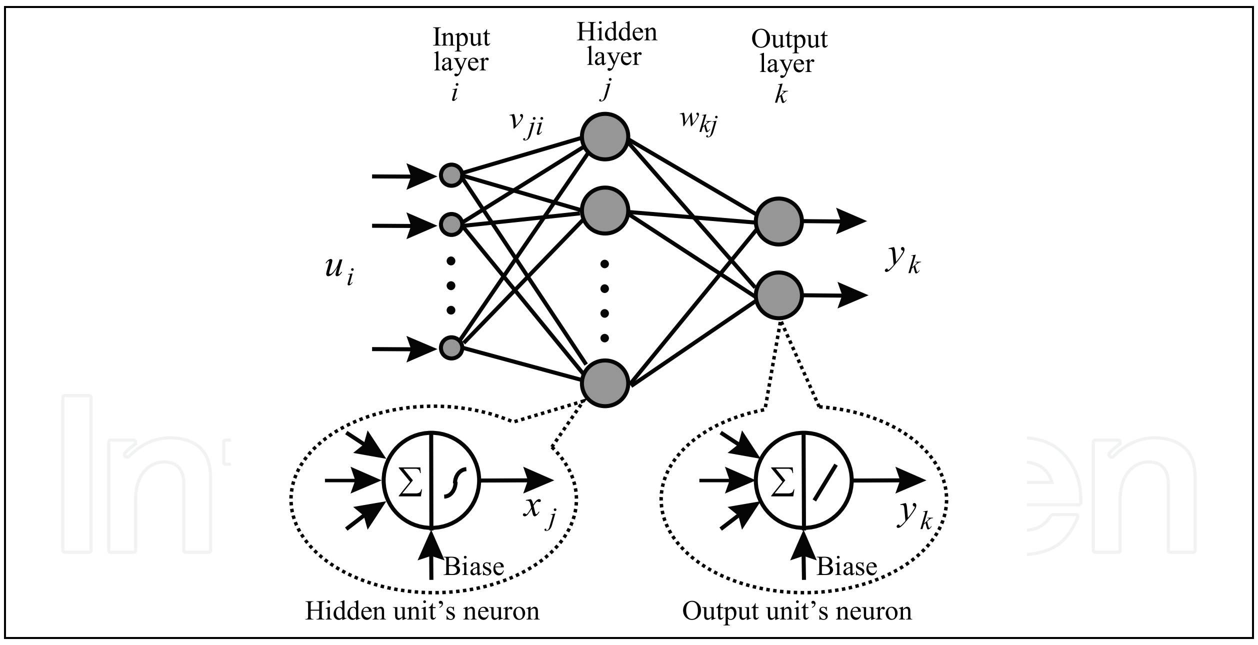


Fig. 3. A two layer artificial neural network

3.2 The training agorithm

In developing a training algorithm for this network, we want a method that specifies how to reduce the total system error for all patterns through an adjustment of the weights. This chapter uses the Back-Propagation training algorithm which is an iterative gradient algorithm designed to minimize the mean square error between the actual output of a feed-forward network and the desired output (Lippman, 1987; Weber et al., 1991; Fukuda & Shibata, 1992).

The back-propagation training is carried out as follows: the hidden layer weights are adjusted using the errors from the subsequent layer. Thus, the errors computed at the output layer are used to adjust the weights between the last hidden layer and the output layer. Likewise, an error value computed from the last hidden layer output is used to adjust the weights in the next to the last hidden layer and so on until the weight connections to the first hidden layer are adjusted. In this way, errors are propagated backwards layer by layer with corrections being made to the corresponding layer weights in an iterative manner. The process is repeated a number of times for each pattern in the training set until the criterion minimization is reached. This is illustrated in Fig. 4. Therefore, we first calculate the predicted error at each time step s (we refer to s here to introduce the discrete time factor). Then, an equivalent error is calculated for each neuron in the network. For example the equivalent error δ_k of the neuron k in the output layer is given by (taking into account that the derivative of the output layer's activation function is unity because it is a linear activation function):

$$\delta_k(s) = \varepsilon_k(s) = y_k(s) - \hat{y}_k(s) \tag{11}$$

The equivalent error δ_j of neuron j in the hidden layer is given by:

$$\delta_j(s) = \frac{df\left(H_j(s)\right)}{dH_j(s)} \sum_k \delta_k(s) w_{kj} \tag{12}$$

Weights connecting the hidden and output layers are adjusted according to:

$$\begin{aligned} w_{kj}(s) &= w_{kj}(s-1) + \Delta w_{kj}(s) \\ \Delta w_{kj}(s) &= \alpha \delta_k(s) x_j(s) + \beta \Delta w_{kj}(s-1) \end{aligned} \tag{13}$$

where: α and β are the learning rate and the momentum parameters respectively.
Weights connecting the input and hidden layer are adjusted according to:

$$\begin{aligned} v_{ji}(s) &= v_{ji}(s-1) + \Delta v_{ji}(s) \\ \Delta v_{ji}(s) &= \alpha \delta_j(s) u_i(s) + \beta \Delta v_{ji}(s-1) \end{aligned} \tag{14}$$

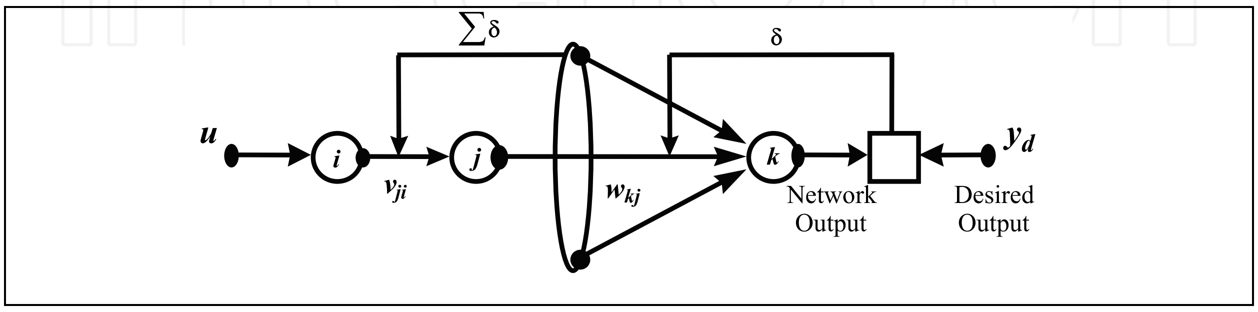


Fig. 4. Back-propagation algorithm

In summary, the training algorithm is as follow: the output layer error is calculated first using Eq. (11) and then backpropagated through the network using Eq. (12) to calculate the equivalent errors of the hidden neurons. The network weights are then adjusted using Eq. (13) and Eq. (14).

3.3 Model validation

In order to check if the identified model agrees with the real process behavior, model validation is necessary. This is imperative as to taken into account the limitations of any identification method and its final goal of model application. This includes a check to determine if the priori assumptions of the identification method used are true and to compare the input-output behaviour of the model and the plant (Ljung & Guo, 1997).

To validate the model, a new input will be applied to the model under validation tests. The new outputs will be compared with the real time outputs and validation statistics is calculated. These statistics will decide whether the model is valid or not.

To carry out the validation task, we use the following statistics for the model residuals:

The maximal absolute value of the residuals

$$M_N^{\varepsilon} = \max_{1 \le t \le N} \left| \varepsilon(t) \right| \tag{15}$$

Mean, Variance and Mean Square of the residuals

$$m_N^{\varepsilon} = \frac{1}{N} \sum_{t=1}^{N} \varepsilon(t) \tag{16}$$

$$V_N^{\varepsilon} = \frac{1}{N} \sum_{t=1}^{N} \left[\varepsilon(t) - m_N^{\varepsilon} \right]^2 \tag{17}$$

$$S_N^{\varepsilon} = \frac{1}{N} \sum_{t=1}^{N} \varepsilon(t)^2 = \left(m_N^{\varepsilon} \right)^2 + V_N^{\varepsilon} \tag{18}$$

In particular we stress that the model errors must be separated from any disturbances that can occur in the modelling. As this can correlates the model residuals and the past inputs. This plays a crucial role. Thus, it is very useful to consider two sources of model residuals or model errors ε. The first error originates from the input $u(t)$ while the other one originates from the identified model itself. If these two sources of error are additive and the one that originates from the input is linear, we can write

$$\varepsilon(t) = \Delta(q) u(t) + v(t) \tag{19}$$

Equation (19) is referred to as the separation of the model residuals and the disturbances. Here, *v(t)* would not change, if we changed the input *u(t)*. To check the part of the residuals that might originate from the input, the following statistics are frequently used:

If past inputs are $\phi(t) = \left[u(t), u(t-1), \cdots, u(t-M+1) \right]^T$ and $R_N = \frac{1}{N} \sum_{t=1}^{N} \phi(t) \phi(t)^T$, then the scalar measure of the correlation between past inputs $\phi(t)$ and the residuals $\varepsilon(t)$ is given by:

$$\xi_N^M = r_{\varepsilon u}^T R_N^{-1} r_{\varepsilon u} \tag{20}$$

where $r_{\varepsilon u} = \left[r_{\varepsilon u}(0), \cdots, r_{\varepsilon u}(M-1) \right]^T$

with $r_{\varepsilon u}(\tau) = \dfrac{1}{\sqrt{N}} \sum_{t=1}^{N} \varepsilon(t) u(t-\tau)$.

The obtained model should pass the validation tests of a given data set. Then we can say that our model is unfalsified. Here, we shall examine our model when the validation test is based on some of the statistics given previously in Eqs. (15-20).

Let us first assume that the model validation criterion be a positive constant $\mu > 0$ for the maximal absolute value of the residuals M_N^{ε} stated in Eq. (15)

$$g\left[\varphi(t), \theta\right] \text{ is not validated iff } M_N^{\varepsilon}(\theta) \le \mu \tag{21}$$

The problem of determining which models satisfy the inequality of Eq. (21) is the same problem that deals with set membership identification (Ninness & Goodwin, 1994). Typically this set is quite complicated and it is customary to outerbound it either by an ellipsoid or a hypercube. Therefore, it is agreed that a reasonable candidate model for the true dynamics should make the sample correlation between residuals $\varepsilon(t,\theta) = y(t) - \hat{y}(t \mid \theta)$ and past inputs $u(t-1), \cdots, u(t-m)$ small within certain criterion. One possible validation criterion is to require this correlation to be small in comparison with the Mean Square of the Model Residuals S_N^{ε} stated in Eq. (18). This is given by:

$$g\left[\varphi(t), \theta\right] \text{ is not validated iff } \xi_N^M(\theta) \le \gamma S_N^{\varepsilon}(\theta) \tag{22}$$

where γ is a subjective threshold that will be selected according to the application.

4. The neurocontroller

Conceptually, the most fundamental neural network based controllers are probably those using the inverse of the plant as the controller. The simplest concept is called direct inverse control, which is used in this chapter. Before considering the actual control system, an inverse model must be trained. There are tow ways of training the model; generalized training and the specialized training. This chapter uses the generalized training method. Figure 5 shows the off-line diagram of the inverse plant modelling.

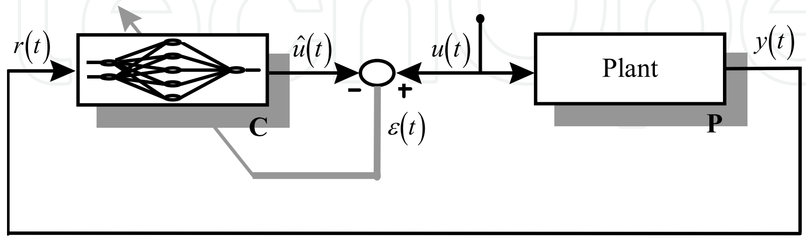


Fig. 5. Inverse plant modelling

Given the input-output data set which will be referred to as Z^N over the period of time $1 \le t \le N$

$$Z^N = \left\{ u(1), y(1), \ldots, u(N), y(N) \right\} \tag{23}$$

where *u(t)* is the input signal and $y(t)$ is the output signal, the system identification task is basically to obtain the model $\hat{y}(t\,|\,\theta)$ that represent our plant;

$$\hat{y}(t\,|\,\theta) = g\left(\theta, Z^N\right) \tag{24}$$

where $\hat{y}$ denotes the model output and g is some non-linear function parameterized by θ which is the finite dimensional parameter vector, the weights of the network in our case (Ljung & Sjöberg 1992; Ljung, 1995; Sjöberg, 1995).
The objective with inverse plant modelling 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 in Eq. (24), a network is trained as the inverse of the process:

$$\hat{u}(t\,|\,\theta) = g^{-1}\left(\theta, Z^N\right) \tag{25}$$

However, modelling errors perturb the transfer function away from unity. Therefore, $\hat{g}^{-1}\left(\theta, Z^N\right)$ will be used instead of $g^{-1}\left(\theta, Z^N\right)$.
To obtain the inverse model in the generalized training method, a network is trained off-line to minimize the following criterion instead:

$$W_N\left(\theta, Z^N\right) = \frac{1}{N}\sum_{t=1}^{N}\left(u(t) - \hat{u}(t\,|\,\theta)\right)^2 \tag{26}$$

In other words, our aim is to reduce the error ε where:

$$\varepsilon(t) = u(t) - \hat{u}(t\,|\,\theta) \tag{27}$$

Once we carry out that, the inverse model is subsequently applied as the controller for the system by inserting the desired output (the reference) instead of the system output. This is illustrated in Fig. 6.

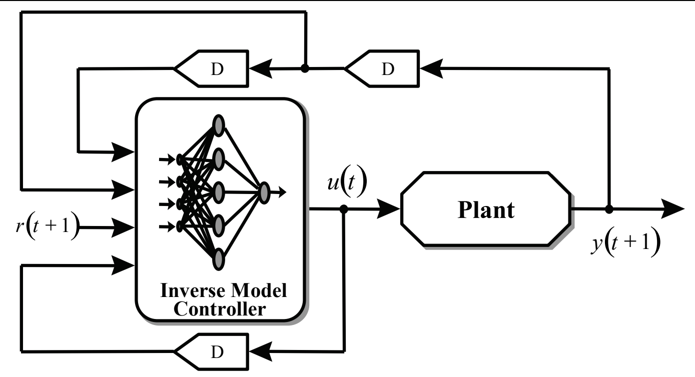


Fig. 6. Direct inverse control

5. Simulation and results

The first step is to collect training data from the real plant, which is a three phase squirrel-cage induction motor with the following ratings: 380V, 50Hz, 4-pole, 0.1kW, 1390rpm, and is

Y-connected. That was carried out by using a data acquisition card to interface the induction motor and the inverter and its inputs and outputs to the computer. A voltage signal is to be sent to the frequency inverter which changes the three phase lines frequency into a new signal with different frequency to drive the induction machine speed. That was the input signal. The output signal is taken from a tachometer connected directly to the rotor shaft and back to the interfacing data acquisition card as the speed signal. Figure 7 shows the overall experimental system setup.

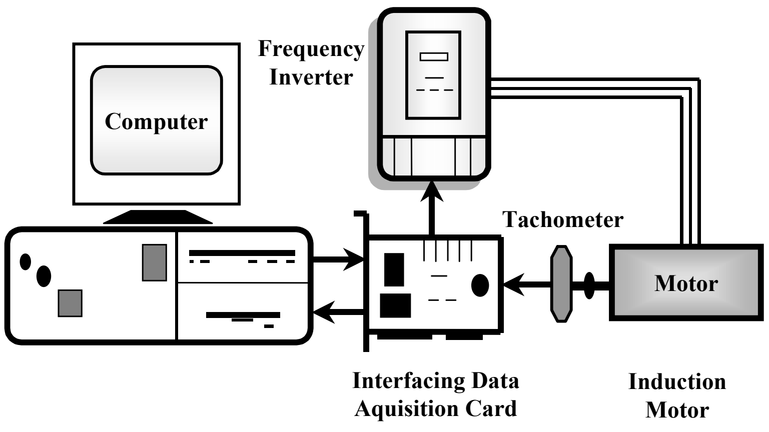


Fig. 7. The experimental work

5.1 Results of system identification

The input data set is designed to be a PRBS signal chosen randomly, both in amplitude and frequency, to fully excite the whole speed range which allows the network to recognize the overall system's behaviour. In addition, the sampling time is made to be 40 times smaller than the settling time of the system to obtain more accurate model and avoid aliasing problems. The input-output data set is shown in Fig. 8. The data set will be divided into two sets; a network training set and a model validation set.

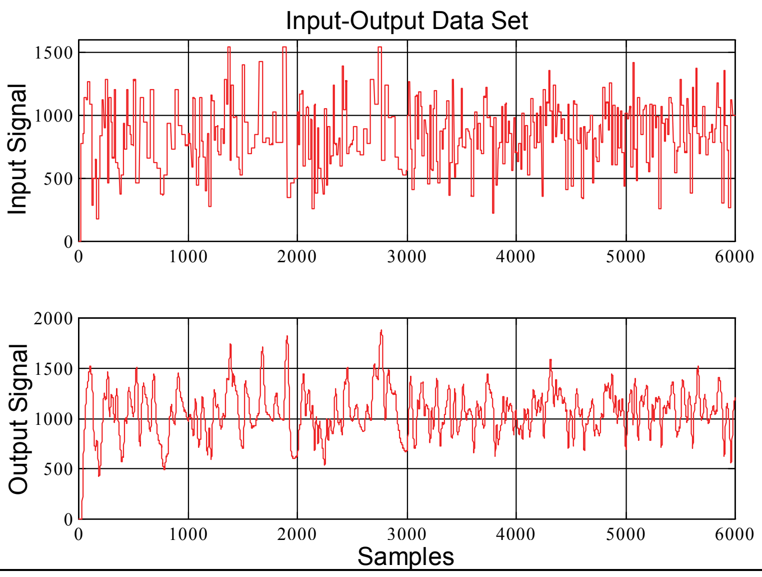


Fig. 8. The input-output data set

Since the system is a single-input single-output nonlinear system, this work uses a second order NARX model. This means that the regressor vector is as follows:

$$\varphi(t) = \left[y(t-1), y(t-2), u(t-1), u(t-2)\right] \tag{28}$$

The network structure is a two-layer hyperbolic tangent sigmoidal feed-forward architecture (one hidden layer with a tanh activation function and one output layer with a linear activation function). The weights for both hidden layer and output layers 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 a global minima rather than a local minima (Patterson, 1996).
Too many hidden neurons can cause the over-fitting, while too few neurons cause the under-fitting (Patterson, 1996). Moreover, a big network (many neurons) causes the training process to become very slow. The training showed good results when a five hidden neurons is used and 3000 samples are used as a training set. During each back propagation iteration the Sum of Squared Errors (SSE) are computed and compared to an error criteria α , i.e.

$$SSE = \sum_{i=1}^{N} \left[y(t) - \hat{y}(t)\right]^2 < \alpha \tag{29}$$

The SSE decreased gradually during the training process until it is within the criteria threshold after approximately 370 iterations. To test whether the network can produce the same output as the plant or not, and considering the over-fitting problem, the output $\hat{y}(t \mid \theta)$ of the model will be compared with the plant output $y(t)$ to calculate the residuals $\varepsilon(t) = y(t) - \hat{y}(t \mid \theta)$. The results of applying both training and validation data sets are shown in Table 1.

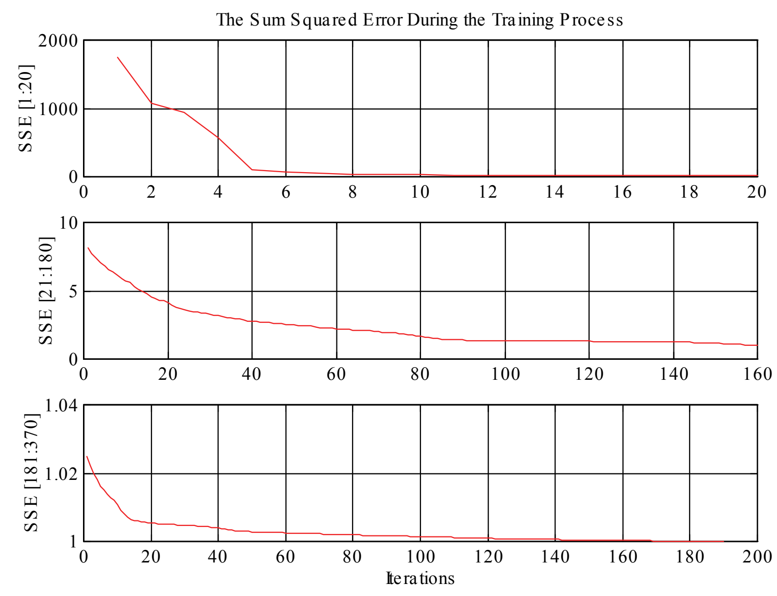


Fig. 9. The Sum Squared Error during the training process

Residual Statistics	Training data set	Validation data set
Mean Square S_N^{ε}	$2.869 \cdot 10^{-4}$	$2.936 \cdot 10^{-4}$
Maximal Absolute Value M_N^{ε}	2.9692%	3.0286%

Table 1. Residual Analysis

From the table we can see that there are only small differences in the residual statistics between the training data set and the validation data set. Thus the inequality of Eq. (21) is satisfied. However, one should check the correlation ξ_N^M between the residuals $\varepsilon(t,\theta)$ and past inputs $u(t-1),\cdots,u(t-m)$ because the residual statistics are not enough to judge the quality of the network model. This is done by constructing the past input vector and then calculating the correlation function.

The correlation results are shown in Fig. 9, where it can be seen that the auto-correlation of the residuals lies within the 99% confidence limits which gives a strong indication that the model is acceptable. Furthermore, we can see that the cross correlation between the past inputs and the residuals lies between the 99% confidence limits also.

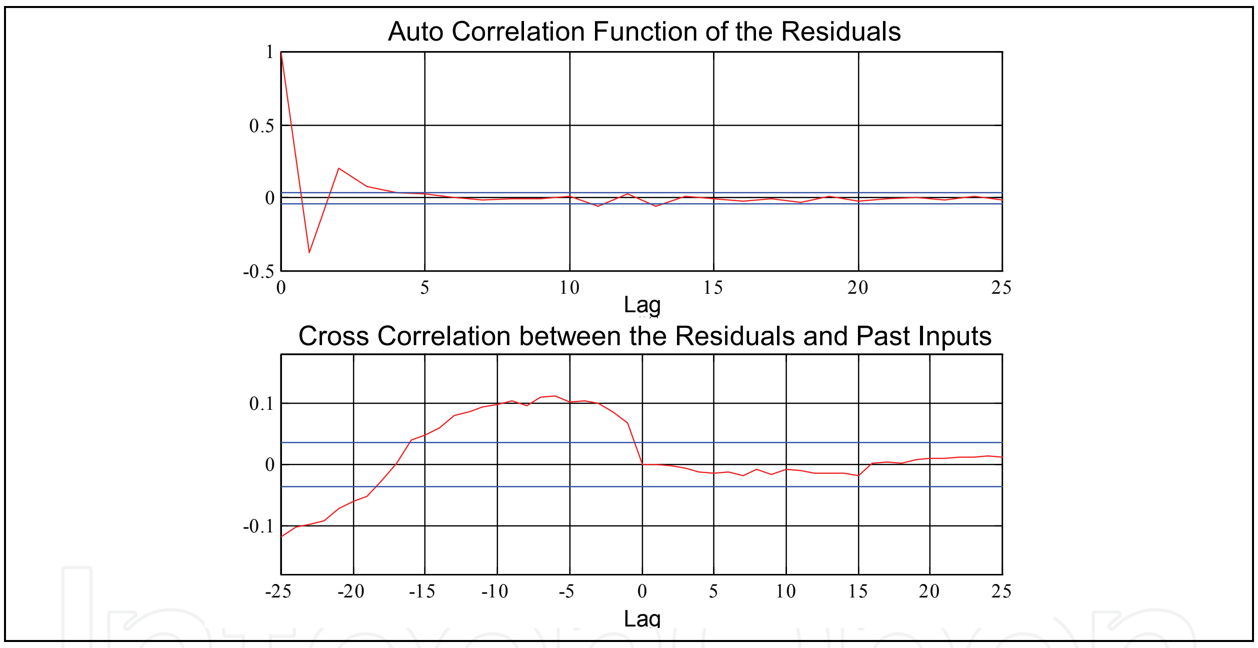


Fig. 9. Correlation Analysis of the Validation Data Set

5.2 Results of inverse training and control

As mentioned earlier, it is clear that the plant is a single-input single-output (SISO) system. First the regressors are chosen based on inspiration from linear system identification. The model order was chosen as a second order which gave us good results. Clearly, the input vector to the network contains two past plant outputs and two past plant inputs.

$$Z^N = \left[y(t-1), y(t-2), u(t-1), u(t-2)\right] \tag{30}$$

The network structure is a two layer hyperbolic tangent sigmoidal feed-forward architecture (one hidden layer with a tanh activation function and one output layer with a linear activation function). The network weights are initially randomised around the values -0.5 and +0.5 before the training.

The back-propagation training showed good results when using a network structure with two layer feed forward architecture neuron and 3000 samples as a training set. The network architecture contains one hidden layer with a hyperbolic tangent (tanh) activation function and one output layer with a linear activation function. The hidden layer consists of six hidden neurons while the output layer consists of one neuron. The results of the inverse plant model training algorithm is shown in Fig. 10 where α is chosen as 1.

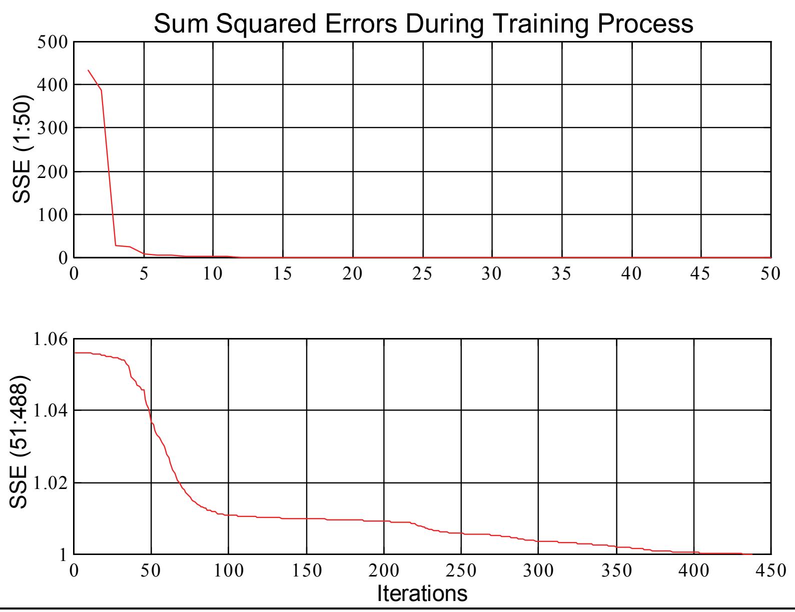


Fig. 10. SSE of inverse plant modelling

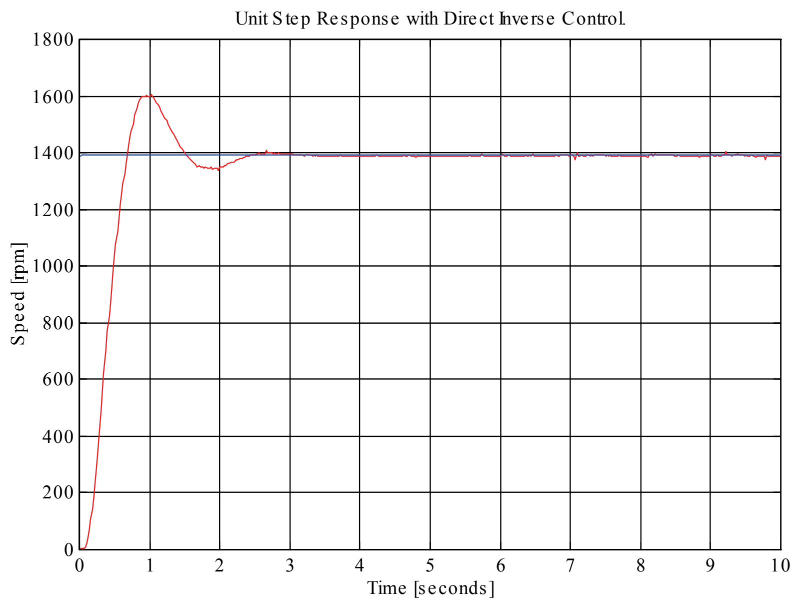


Fig. 11. Speed error due to a step reference signal

The final step after obtaining the inverse model is to implement the controller. The same setup of Fig. 7 is used to control the speed of the motor. First, to check the controller performance, a step input signal with the value of 1390rpm is fed to the system. The resulting response and error between the reference signal and the measured output speed

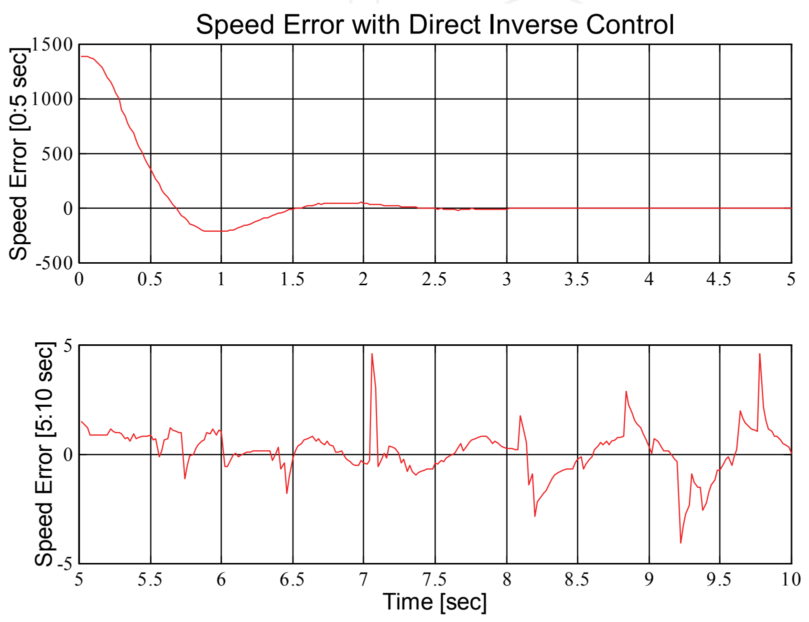


Fig. 12. Speed error due to a step reference signal

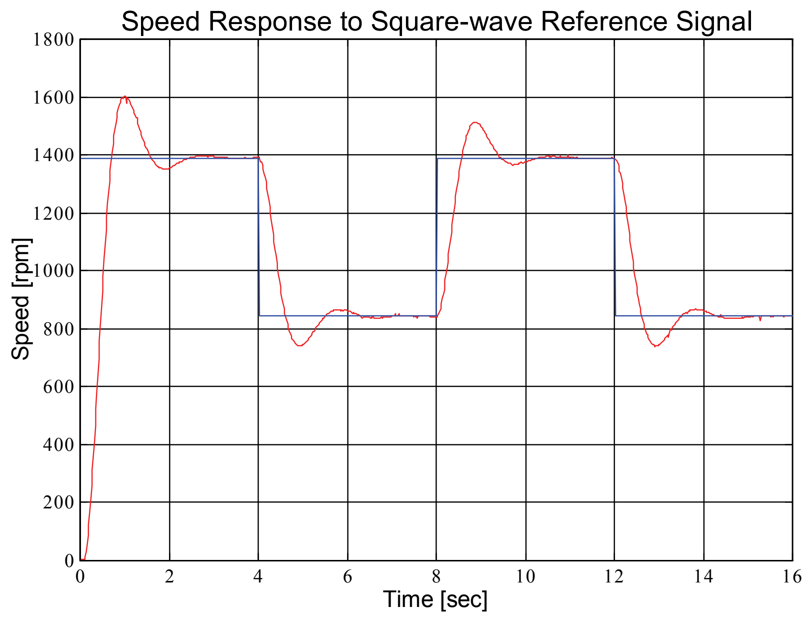


Fig. 13. System response to a square wave reference signal

are illustrated in Figs. 11 and 12 respectively. It can be seen from Fig. 12 that the speed of the induction motor followed the reference signal with an acceptable steady state error equals to 0.2878%. The results of Figs. 11 and 12 also show a maximum overshoot of less than 13%.
To investigate the tracking capabilities of the system, different reference signals were fed to the controller and its performance is examined. The following real time tests will explore

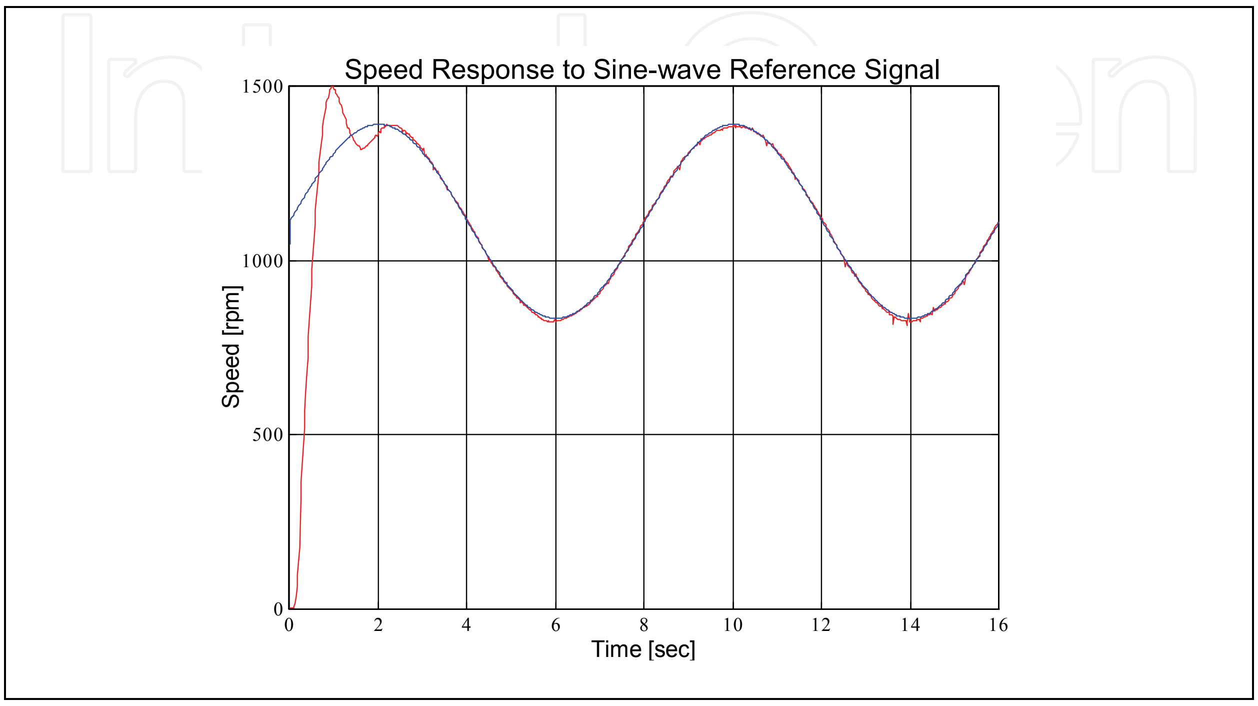


Fig. 14. System response to a sine wave reference signal

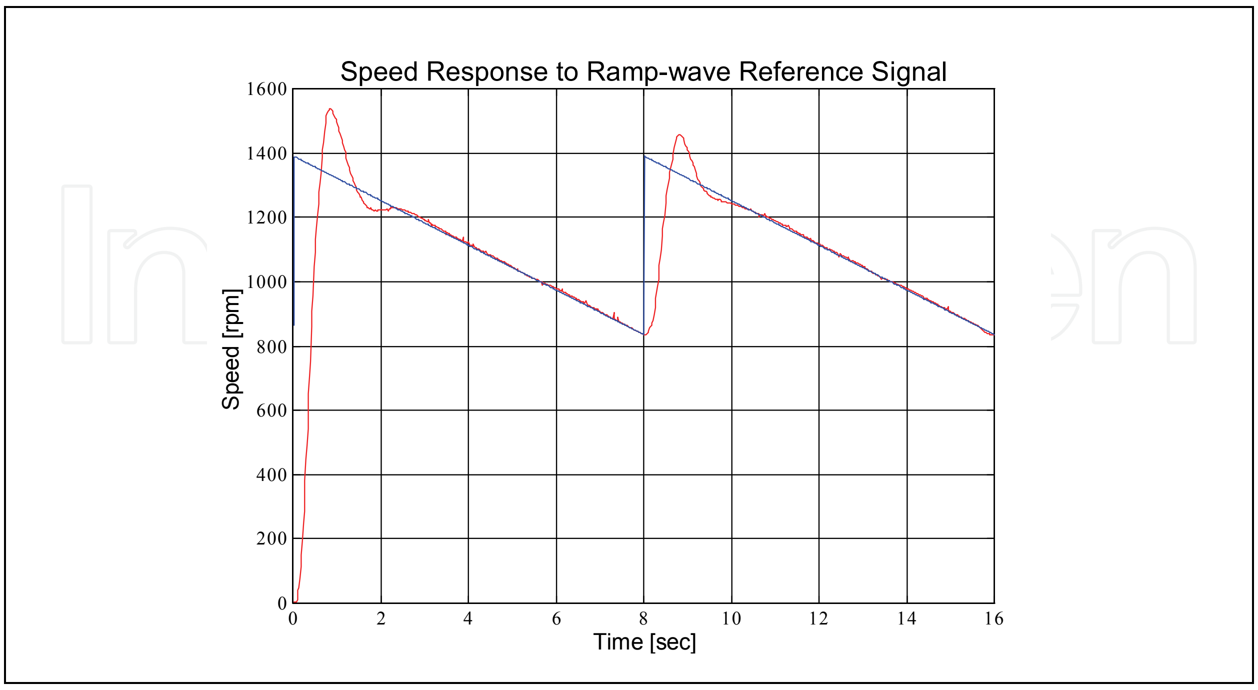


Fig. 15. System response to a saw-tooth wave reference signal

the response to three different types of speed reference signals; square wave (Fig. 13), sine wave (Fig. 14), and saw-tooth wave (Fig. 15) reference signals. In addition, the steady state errors are recorded in Table 2.

Reference Signal	Min. Error	Max. Error
Square wave	–0.31%	+0.56%
Sine wave	–0.43%	+1.00%
Saw-tooth wave	–0.65%	+0.29%

Table 2. Steady state errors analysis for different reference signal types

The previous figures suggest that the direct inverse model control scheme can track changes in the reference signal while maintaining good performance.
Next, to test the system under disturbances in the form of load torque conditions, a step reference signal representing 1390 rpm is fed to the system while a load torque step signal of 2 N.m (which is the full load) is applied to the shaft during the period of 4 to 8 seconds. The results are shown in Fig. 16. It can be seen from the figure that the direct inverse controller could recover the disturbance caused by the applied load torque. The induction motor speed followed the reference signal in a short time.

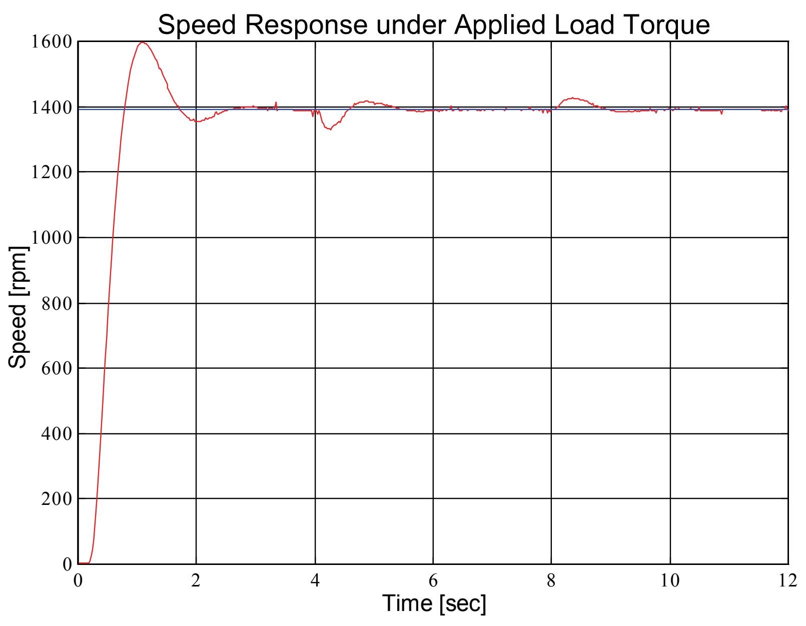


Fig. 16. Speed response under load torque condition

6. Conclusion

In this chapter, the nonlinear black box modelling for 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. Applying the validation tests, the network model could pass the residual tests and the cross correlation tests resulting into a simple yet a highly accurate model of the induction motor. The same method was then used to model the inverse model of the system. The real time implementation for the direct inverse neural network based control scheme has been presented and its performance has been tested over different types of reference signals and applied load torque. The controller tracked the given reference speed signals and overcame the applied load torque disturbance demonstrating the strong capabilities of artificial neural networks in nonlinear control applications.

7. References

Bavarian B. (1988). Introduction to Neural Networks for Intelligent Control, *IEEE Control Systems Magazine*, Vol. 8, No. 2, pp. 3-7.

Ebrahim Osama S., Mohamed A. Badr, Ali S. Elgendy, and Praveen K. Jain,(2010). ANN-Based Optimal Energy Control of Induction Motor Drive in Pumping Applications. *IEEE Transactions On Energy Conversion*, Vol. 25, No. 3, (Sept. 2010), pp. 652-660.

Fukuda T. & Shibata T. (1992). Theory and Application of Neural Networks for Industrial Control Systems, *IEEE Trans. on Industrial Electronics*, Vol. 39, No. 6, pp. 472-489.

Henneberger G. and B. Otto (1995). Neural network application to the control of electrical drives. *Proceeding of Confrence of Power Electronics Intelligent Motion*, pp. 103–123, Nuremberg, Germany.

Karanayil B.; Rahman M. F. & Grantham C. (2003). Implementation of an on-line resistance estimation using artificial neural networks for vector controlled induction motor drive, Proceeding of *IECON '03 29th Annual Conference of the IEEE Industrial Electronics Society*, Vol. 2, pp. 1703-1708.

Kulawski G. and Mietek A. Brdyś (2000). Stable adaptive control with recurrent networks. *Automatica A Journal of IFAC, the International Federation of Automatic Control*, vol. 36, No. 1, (Jan. 2000), pp. 5–22.

Kung Y. S., C. M. Liaw, and M. S. Ouyang (1995). Adaptive speed control for induction motor drive using neural network. *IEEE Transactions on Industrial Electronics*, vol. 42, no. 1, (Feb. 1995), pp. 9–16.

Lippman R. P. (1987). An Introduction to Computing with Neural Nets, *IEEE ASSP Magazine*, Vol. 4, pp. 4-22.

Ljung L. & Guo L. (1997). Classical model validation for control design purposes, *Mathematical Modelling of Systems*, 3, 27–42.

Ljung L. & Sjöberg J. (1992). A System Identification Perspective on Neural Nets, Technical Report, Report No. LiTH-ISY-R-1373, At the location: www.control.isy.liu.se, May 27.

Ljung L. (1995). System Identification, Technical Report, Report No. LiTH-ISY-R-1763, At the location: www.control.isy.liu.se.

Ma X. & Na Z. (2000). Neural network speed identification scheme for speed sensor-less DTC induction motor drive system, *PIEMC 2000 Proceeding. 3rd Int. Conference on Power Electronics and Motion Control*, Vol. 3, pp. 1242-1245.

Mehrotra P. ; Quaicoe J. E. & Venkatesan R. (1996a). Development of an Artificial Neural Network Based Induction Motor Speed Estimator, *PESC '96 IEEE Power Electronics Specialists Conference*, Vol. 1, pp. 682-688.

Mehrotra P.; Quaicoe J. E. & Venkatesan R. (1996b). Induction Motor Speed Estimation Using Artificial Neural Networks, *IEEE Canadian Conference on Electrical and Computer Engineering*, Vol. 2, pp. 607-610,

Merabet Adel, Mohand Ouhrouche and Rung-Tien Bui (2006). Neural Generalized Predictive Controller for Induction Motor, *International Journal of Theoretical and Applied Computer Sciences*, Vol. 1, 1 (2006), pp. 83–100.

Mohamed H. A. F.; Yaacob S. & Taib M. N. (1997). Induction Motor Identification Using Artificial Neural Networks, *APEC 97 Electric Energy Conference,* pp. 217-221, 29-30th Sep..

Ninness B. & Goodwin G. C. (1994). Estimation of Model Quality, *10th IFAC Symposium Proceeding. on System Identification*, 1, pp. 25-44.

Orłowska-Kowalska T. and C. T. Kowalski (1996). Neural network based flux observer for the induction motor drive. *Proceedingceding of International Confrence of Power Electronics Motion Control,* pp. 187–191, Budapest, Hungary, 1996.

Patterson D. W. (1996). Artificial Neural Networks: Theory and Applications, Simon and Schuster (Asia) Pte. Ltd., Singapore: Prentice Hall.

Raison B., F. Francois, G. Rostaing, and J. Rogon (2000). Induction drive monitoring by neural networks. *Proceeding of IEEE International Conference of Industrial Electronics,Control Instrumentation*, pp. 859–863, Nagoya, Japan, 2000.

Sharma A. K., R. A. Gupta, Laxmi Srivastava (2007). Performance of Ann Based Indirect Vector Control Induction Motor Drive, *Journal of Theoretical and Applied Information Technology*, Vol. 3, No. 3, (2007), pp 50-57.

Simoes M. G. and B. K. Bose (1995).Neural network based estimation of feedback signals for a vector controlled induction motor drive. *IEEE Transactions on Industry Applications.,* vol. 31, no. 3, (May/Jun. 1995), pp. 620–629.

Sjöberg J.; Zhang Q., Ljung L., Benveniste A., Deylon B., Glorennec P. Y., Hjalmarsson H., & Juditsky A. (1995). Nonlinear Black-Box Models In System Identification: A Unified Overview, *Automatica*, Vol. 31, No. 12, pp. 1691-1724.

Toqeer R. S. & Bayindir N. S. (2000). Neurocontroller for induction motors, ICM 2000. Proceeding. *12th Int. Conference on Microelectronics*, pp. 227-230.

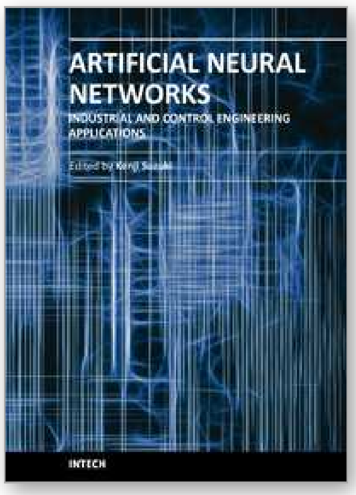
Vas P. (1990). Vector Control of AC Machines, Clarendon Press, Oxford.

Weber M.; Crilly P. B. & Blass W. E. (1991). Adaptive Noise Filtering Using an Error-Backpropagation Neural Network, *IEEE Trans. Instrum. Meas.*, Vol. 40(5), pp. 820-825.

Wishart M. T. & Harley R. G. (1995). Identification And Control Of Induction Machines Using Artificial Neural Networks, *IEEE Transactions on Industry Applications*, Vol. 31(3), pp. 612-619.

Wlas M.; Krzeminski, Z.; Guzinski, J.; Abu-Rub, H.; Toliyat, H.A. (2005). Artificial-Neural-Network-Based Sensorless Nonlinear Control of Induction Motors, *IEEE Transection of Energy conversion*, Vol. 20, 3 (Sept 2005), pp. 520-528.

Yabuta T. & Yamada T. (1991). Learning Control Using Neural Networks. *Proceeding of the 1991 IEEE International Conference on Robotics and Automation*, Vol. 1, pp. 740-745.



Artificial Neural Networks - Industrial and Control Engineering Applications
Edited by Prof. Kenji Suzuki

ISBN 978-953-307-220-3
Hard cover, 478 pages
Publisher InTech
Published online 04, April, 2011
Published in print edition April, 2011