

ON-LINE IMPLEMENTATION OF AN ADAPTIVE SPEED FILTER
AND ITS EXPERIMENTAL DEMONSTRATION

A Thesis

by

ANIS MOHAMED ABDUL

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of
MASTER OF SCIENCE

December 2001

Major Subject: Mechanical Engineering

ON-LINE IMPLEMENTATION OF AN ADAPTIVE SPEED FILTER
AND ITS EXPERIMENTAL DEMONSTRATION

A Thesis

by

ANIS MOHAMED ABDUL

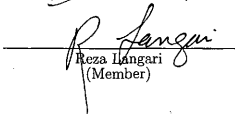
Submitted to Texas A&M University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

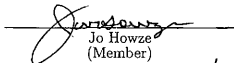
Approved as to style and content by:



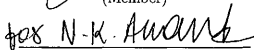
Alexander G. Parlos
(Chair of Committee)



Reza Langari
(Member)



Jo Howze
(Member)



John Weese
(Head of Department)

December 2001

Major Subject: Mechanical Engineering

ABSTRACT

On-line Implementation of an Adaptive Speed Filter
and Its Experimental Demonstration. (December 2001)

Anis Mohamed Abdul, B.S., Anna University, Chennai, India

Chair of Advisory Committee: Dr. Alexander G. Parlos

Sensorless speed estimation in induction machines is important for numerous applications like speed control and fault detection. Sensors are expensive and they are not reliable enough to be used in rugged industrial environments. In this work, a previously developed neural network speed filter is implemented for on-line induction motor speed estimation.

The speed filter is constructed using a combination of five neural networks. A neural networks framework developed in this work is used to construct the speed filter. The filter uses the three motor terminal voltages, the line currents, and the RMS of on-line current as inputs to estimate the speed. The data are preprocessed by a set of LabVIEW modules before they are sent to the neural networks. The preprocessed data are used by the neural networks to compute the induction motor speed. The output from the neural networks is then scaled to obtain the motor speed estimate.

The filter is implemented and tested using both off-line and on-line collected data. The filter is also tested with unbalanced power supply and faulty motors to study its generalization capability. The filter had an average estimation error between 0.1% to 0.3% for the data collected off-line. For the data obtained from on-line setup, the average estimation at steady state is 0.15%.

This research demonstrates the feasibility of using adaptive file-based software sensors instead of hardware sensors thereby significantly reducing implementation

costs and improving overall system robustness. The neural networks framework developed in this work adds flexibility and scalability in further improving the developed induction motor speed filter.

To My Parents

ACKNOWLEDGMENTS

I would like to express my sincere appreciation to Dr. Alexander G. Parlos for his guidance and support without which this work would not have been possible. I would like to thank my committee members Dr. Reza Langari and Dr. Jo W. Howze for their interest in this project. My thanks are due Dr. Raj M. Bharadwaj for his help throughout this work.

I would also like to extend my thanks to friends Vijaya Mallikarjun, Pierce and Aninda Bhattacharya for their support and help during the course of this project.

TABLE OF CONTENTS

CHAPTER		Page
I	INTRODUCTION TO INDUCTION SPEED ESTIMATION . . .	1
	A. Introduction	1
	B. An Overview of Estimation Problems	2
	1. Smoothing	2
	2. Filtering	3
	3. Prediction	3
	C. Introduction to Neural Networks	4
	D. Overview of Induction Motor Speed Estimation	5
	E. Objectives	6
	F. Proposed Approach	7
	G. Research Contribution	7
	H. Organization of the Thesis	8
II	OVERVIEW OF SPEED FILTER DESIGN	9
	A. Speed Filter Development	9
	1. Problem Statement	9
	2. Neural Networks Implementation	10
	3. Speed Filter Description	13
	4. Neural Network Training	14
	B. Chapter Summary	16
III	ON-LINE IMPLEMENTATION OF SPEED FILTER	17
	A. Neural Networks Architecture of the Speed Filter	17
	B. Neural Network Framework	19
	1. Neural Networks Implementation Framework De- velopment	20
	2. Control Script	20
	3. Neural Network Process Flow	23
	C. Components of the On-line Speed Filter	24
	1. Data Collection Module	24
	2. Data Preprocessing Module	26
	3. Speed Filter Module	30
	4. Data Postprocessing Module	30

CHAPTER		Page
	D. Chapter Summary	33
IV	EXPERIMENTAL DEMONSTRATION OF THE SPEED FILTER	34
	A. Speed Filter Tests with Off-Line Data Collection	34
	1. Experimental Setups Used in Off-Line Motor Data Collection and Speed Filter Tests	35
	a. Small Machine Setup	35
	b. Large Machine Setup	35
	2. Speed Filter Results with Off-Line Data Collection . .	36
	a. Small Machine Test Results	36
	B. Speed Filter with On-line Data Collection	49
	1. Experimental Setup Used in On-line Data Collec- tion and Speed Filter Tests	52
	2. Speed Filter Results with On-line Data Collection . .	54
	C. Chapter Summary	56
V	SUMMARY AND CONCLUSIONS	58
	A. Summary of Research	58
	B. Conclusion	59
	C. Recommendation of Future Research Work	60
	REFERENCES	61
	VITA	63

LIST OF TABLES

TABLE		Page
I	Speed Estimation Errors: Healthy Small Machines; Off-line Data . . .	50
II	Speed Estimation Errors: Faulty Small Machines; Off-line Data . . .	51
III	Speed Estimation Errors: Healthy Large Machines; Off-line Data . . .	51
IV	Speed Estimation Errors: Small Machines; Data from On-line Setup .	57

LIST OF FIGURES

FIGURE		Page
1	Types of State Estimation Problem.	3
2	Block Diagram of the State Filter.	10
3	Block Diagram of the Neural Network Speed Filter.	15
4	Neural Network Architectures of the Current Predictors.	18
5	Neural Network Architecture of the Speed Predictor.	19
6	Neural Network Architecture of the Speed Update.	20
7	Inputs and Outputs of Neural Network Framework Implementation of the Speed Filter.	21
8	Process Control Flow of the Neural Networks Framework.	25
9	Components of the Speed Filter.	26
10	Block Diagram of Preprocessing for Balanced Power Supply or Motor Stator.	28
11	Block Diagram of Preprocessing for Unbalanced Power Supply or Motor Stator.	29
12	Motor Load-based Filter Selection Scheme.	31
13	Speed Filter Module.	32
14	top: Speed Signal from Generator; middle: De-noised Speed Signal; bottom: Harmonic Speed Estimate	37
15	Speed Filter Response for 3 hp Healthy Motor; 0% – 70% Load Range.	38
16	Speed Filter Response for 3 hp Healthy Motor; 70% – 120% Load Range.	39

FIGURE		Page
17	Speed Filter Response for 3 hp Healthy Motor; 0% – 70% Load Range, with 0.5% Unbalanced Power Supply.	40
18	Speed Filter Response for 3 hp Healthy Motor; 70% – 120% Load Range, with 0.5% Unbalanced Power Supply.	41
19	Speed Filter Response for 3 hp Healthy Motor; 0% – 120% Load Range, with 0.5% Unbalanced Power Supply.	42
20	Speed Filter Response for 3 hp Healthy Motor; 0% – 70% Load Range, with 4.5% Unbalanced Power Supply.	43
21	Speed Filter Response for 3 hp Healthy Motor; 70% – 120% Load Range, with 5.4% Unbalanced Power Supply.	44
22	Speed Filter Response for 3 hp Motor with 3 Broken Rotor Bars; 0% – 70% Load Range.	45
23	Speed Filter Response for 3 hp Motor with 4 Broken Rotor Bars; 0% – 70% Load Range.	46
24	Speed Filter Response for Healthy 500 hp Motor; 100% – 50% Load Range.	47
25	Speed Filter Response for Healthy 800 hp Motor; 100% – 50% Load Range.	48
26	On-line Experimental Setup Used for On-line Speed Filter Tests. . .	53
27	Online Speed Filter Response for Healthy 3 hp Motor without Tuning; 30% Load.	55
28	Online Speed Filter Response for Healthy 3 hp Motor with Tuning; 30% Load.	56

CHAPTER I

INTRODUCTION TO INDUCTION SPEED ESTIMATION

A. Introduction

Induction motors represent some of the most widely used prime-movers in industry. In particular, squirrel cage induction motors are used to drive complex and vital components in power plants and process industries. Failure of such motors could result in unscheduled downtime, loss of productivity, causing heavy financial losses. Hence, there is a clear need to preempt such failures. Online monitoring and early detection of faults has become necessary to improve reliability and avoid catastrophic failures [3]. Most fault diagnosis schemes are based either on the inspection of the motor current spectrum and the detection of some speed dependent harmonics or the measurement of vibration levels [4, 5]. These schemes require an accurate knowledge of the motor speed for effective condition monitoring and fault detection. In the majority of industrial setups, like power plants and process industries, induction motors do not have any speed sensors. Further, the motors operate under both balanced and unbalanced power supply conditions.

Control of electrical motor drives is another application that requires the speed and/or position signal. Field-oriented or vector-controlled techniques have made possible the development of high dynamic performance induction motor drives. However, to obtain optimum dynamic performance, speed and/or position transducers are required. Speed sensors, like tachogenerator, encoders and Hall effect sensor, would increase the cost of induction motor drives [1]. Moreover, the failure probability of a sensor is generally higher than that of the motor. This reduces the intrinsic mechan-

ical robustness of the induction motor. To avoid the problems associated with the introduction of a speed sensor for induction motor control, sensorless speed control is fast emerging as a viable alternative.

Therefore, an effective sensorless speed estimation method is desirable not only for online condition monitoring of induction motors, but also for sensorless speed control applications. For the remainder of this thesis, a squirrel cage induction motor will be referred to simply as induction motor or induction machine.

B. An Overview of Estimation Problems

Estimation is the process of calculating the *state(s)* of a dynamic system using observations collected from the system, and a pre-specified or identified mathematical model of the system. The states of the system are nothing but variables that completely specify the behavior of the system.

Estimation can be further classified into different categories. The notation used in this thesis for the state estimate, $\hat{x}(t|t)$, denotes its value at a discrete time t up to and including all the measurements till the discrete time instant t .

The categorization of the estimation problem is done based on the time instant for which a value of the state estimate, $\hat{x}(t|t)$, of the state, $x(t)$, is desired and the time instant until which the measurements, $y(t)$, are available and/or used [2]. The different categories are presented below.

1. Smoothing

Smoothing refers to the estimation of an unmeasurable or unmeasured variable of interest at time step (t), based on measurements up to and including time step ($t+\lambda$), $\lambda > 0$.

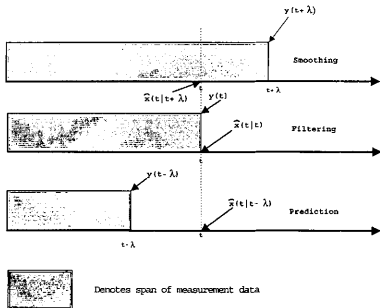


Fig. 1. Types of State Estimation Problem.

2. Filtering

Filtering refers to the estimation of an unmeasurable or unmeasured variable of interest at current time step t , based on measurements available up to and including the current time step t .

3. Prediction

Prediction refers to the estimation of a variable of interest at a future time step t_0 , based on the measurements available up to and including the current time step $(t - \lambda)$, $\lambda > 0$. Predictions that aim to estimate the value of a variable of interest at a time step $(t + 1)$ using measurements up to and including time step t are referred as *single-step-ahead* predictions. When the prediction aims to estimate the value of a variable at a time step $(t + 1)$ using measurements up to and including time step

$(t - p + 1), p > 0$ it is referred as *multi-step-ahead* prediction.

The aforementioned three types of state estimation problems are depicted in Figure 1.

C. Introduction to Neural Networks

Neural networks are an information-processing paradigm inspired by the way the densely interconnected, parallel structure of the mammalian brain processes information [14]. Neural networks are also referred to as connectionist architectures, parallel distributed processing and neuromorphic systems.

Neural network computing is composed of a large number of highly interconnected processing elements that are analogous to neurons and are tied together with weighted connections that are analogous to synapses that connect neurons. Learning typically occurs by example through training, or exposure to a truthed set of input/output data where the training algorithm iteratively adjusts the connection weights (synapses). These connection weights store the knowledge necessary to solve specific problems.

There are multitudes of different types of neural networks. Some of the more popular include the multilayer perceptron which is generally trained with the back-propagation of error algorithm, learning vector quantization, radial basis function, Hopfield, and Kohonen, to name a few. Some neural networks are classified as feed-forward while others as recurrent (i.e., implement feedback) depending on how data is processed through the network. Another way of classifying neural network types is by their method of learning (or training), as some neural networks employ supervised training while others are referred to as unsupervised or self-organizing. Supervised training is analogous to a student guided by an instructor. Unsupervised algorithms

essentially perform clustering of the data into similar groups based on the measured attributes or features serving as inputs to the algorithms. This is analogous to a student who derives the lesson totally on his or her own. Neural networks can be implemented in software or in specialized hardware [14].

It can be seen from the above text that neural networks are trained by adaptation using a cost criterion, and are believed to be good at interpolation and some extrapolation. This trait makes them a valuable tool for non-linear curve fitting [2].

D. Overview of Induction Motor Speed Estimation

Induction motor speed estimation is discussed in a variety of prior literature, with most addressing it from the motor control perspective. The methods discussed in the literature can be classified into two broad categories [1], namely those that estimate speed using an induction motor model reference strategy, and those that estimate speed by analyzing the harmonics of the stator current waveform. Some of the methods used in speed estimation are summarized below:

- Motor model-based speed estimation methods use a model of the induction motor whose speed is to be estimated. Three main types of algorithms have been proposed in the literature for induction motor speed estimation: extended estimators for rotor speed estimation [7, 8], linear regression approach [9] and model reference adaptive systems [10]. The limitation of the model-based speed estimation is its dependence on machine parameters. Thus, the knowledge of electrical and mechanical characteristics is needed. These parameters are not generally known or widely available. Moreover, many of the schemes discussed assume linear machine models and time-invariant parameters. This results in poor speed estimation.

- Speed estimation using harmonic analysis of the stator current is another technique discussed in the literature. This method relies on the detection of specific harmonics that are induced in the stator current due to the rotor slots. During the operation of an induction motor, the rotor-slot MMF harmonic will interact with the fundamental component of the air-gap flux because of the stator current. Several attempts at extracting the rotor slot harmonic for speed estimation have been reported in the literature [11]. The limitation of these methods is that the FFT-based signal processing used is computationally burdensome for on-line or real-time implementation. Also, for certain rotor-stator slot combinations, the speed harmonic may not be readily detectable.
- Neural networks-based speed estimation techniques have been proposed in [12, 13]. In [12], a neural networks-based speed estimator for vector controlled induction motors is discussed. Rotor speed estimation is done using an 8-16-1 neural network. The neural network is trained with sampled currents, reference voltages and a sampled speed signal. The main limitation with this approach is that the availability of a measured speed signal to train the network is assumed. However, this could be feasible if the manufacturers of motor/drive develop the speed estimators and embed them in the hardware [1].

E. Objectives

From the preceding sections, it can be seen that there is a strong motivation to use a neural networks-based speed filter for induction motor speed estimation. Bharadwaj [1] developed a neural network-based speed filter that has been used to estimate speed of induction motors. The filter developed in [1] was initially tested in an off-line environment, in which the input data is processed as a batch. The objective of

this work is to implement the filter developed in [1] in an experimental setup; i.e. on-line, and study its performance. Also, for implementing different neural network based entities, such as filters and predictors, it is desirable that a scalable neural networks framework is developed and used in this research.

The objectives of this work can be listed as follows:

1. Development of a neural networks framework using the C programming language that can be used to construct and implement complex estimators that use different combinations of feed forward and recurrent neural networks.
2. On-line implementation of the neural networks based speed filter developed in [1] using LabVIEW and the aforementioned neural networks framework.

F. Proposed Approach

For implementing the speed filter, we propose to develop a framework using the C programming language that can be integrated with a data acquisition and a data processing environment like LabVIEW. Once this framework is tested for accuracy, the speed filter developed in [1] will be constructed and implemented in an experimental setup. After integration with the data collection and data preprocessing modules, the system will be tested with the case studies used in [1]. Following a benchmarking of the speed filter with the off-line case studies, on-line testing will commence. Once the performance of the un-tuned filter is recorded, the filter networks will be tuned using the data collected from the experimental setup.

G. Research Contribution

The main research contribution of this work is in demonstrating the on-line implementation feasibility of a recently developed speed filter and in evaluating the efficacy

of the speed filter when implemented in a machine other than the one used to train it. This is an important step towards the goal of implementing similar speed filters in actual industrial setups. An additional contribution of this work is the development of a neural networks framework. The framework is designed to simplify the process of constructing estimators that use different combinations of feed forward or recurrent neural networks. The framework will provide a robust environment to build and test neural networks with minimal coding effort.

H. Organization of the Thesis

In Chapter II, the development of the speed filter by Bharadwaj is presented [1].

In Chapter III, details of the neural networks framework and the LabVIEW environment are presented. The procedure involved in implementing the speed filter is also presented in this chapter.

In Chapter IV, the experimental results obtained using the speed filter are presented. The results presented in this chapter include both online and off-line case studies.

A summary of this thesis, the conclusions of this research and directions for future work are presented in Chapter V

CHAPTER II

OVERVIEW OF SPEED FILTER DESIGN

This chapter summarizes the design of the speed filter developed in [1]. The following sections provides brief information regarding the design of the speed filter using neural networks, training of the neural networks, data collection, and data processing.

A. Speed Filter Development

In this section the state filtering problem is formulated and the neural network approach is presented.

1. Problem Statement

Consider the following representation in the discrete-time nonlinear state space form, also known as the *noise representation*,

$$\begin{aligned} \mathbf{x}(t+1) &= \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) + \mathbf{w}(t), \\ \mathbf{y}(t) &= \mathbf{h}(\mathbf{x}(t)) + \mathbf{v}(t), \end{aligned} \tag{2.1}$$

where $t = 1, 2, \dots$ is the discrete time instant, $\mathbf{y}(t)$ is the $n \times 1$ output vector of the nonlinear state-space model; $\mathbf{u}(t)$ is the $m \times 1$ input vector; $\mathbf{x}(t)$ is the state vector of the model; \mathbf{f} and \mathbf{h} are vector-valued unknown nonlinear functions; $\mathbf{w}(t)$ is the process noise; and $\mathbf{v}(t)$, is the measurement noise. It is assumed that $\mathbf{w}(t)$ and $\mathbf{v}(t)$ are independent processes.

The objective of the state filtering problem is to estimate, $\hat{\mathbf{x}}(t)$, for the state

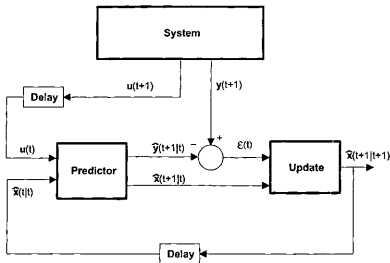


Fig. 2. Block Diagram of the State Filter.

variable $\mathbf{x}(t)$. In this case, the state to be filtered is the speed of the induction motor. The notation $\hat{\mathbf{x}}(t|t)$ is used to mean the state estimate at time t , following the update resulting from the measurements $\mathbf{u}(t)$ and $\mathbf{y}(t)$, at time t .

2. Neural Networks Implementation

The problem formulated in the preceding subsection is solved by employing neural networks to approximate the nonlinear functions $\mathbf{f}(\cdot)$ and $\mathbf{h}(\cdot)$. The inputs $\mathbf{u}(t)$, the outputs $\mathbf{y}(t)$ and the state $\mathbf{x}(t)$ are assumed available through measurements or computations. A block diagram depicting the structure is presented in Figure 2.

The nonlinear equations (2.1) can be rewritten in the *innovations* form as shown below:

$$\hat{\mathbf{x}}(t+1|t) = \mathbf{f}_{innov}(\hat{\mathbf{x}}(t|t-1), \mathbf{u}(t), \mathcal{E}(t)) \quad (2.2)$$

$$\hat{\mathbf{y}}(t+1|t) = \mathbf{h}_{innov}(\hat{\mathbf{x}}(t|t-1), \mathbf{u}(t), \mathcal{E}(t)), \quad (2.3)$$

where $\mathbf{f}_{innov}(\cdot)$ and $\mathbf{h}_{innov}(\cdot)$ are nonlinear functions related to $\mathbf{f}(\cdot)$ and $\mathbf{h}(\cdot)$.

The ‘‘innovations’’ function, $\mathcal{E}(t)$ is defined as $\mathcal{Y}(t) - \hat{\mathcal{Y}}(t|t-1)$, where $\mathcal{Y}(t)$ and $\mathcal{Y}(t|t-1)$ are $n_y \times 1$ vectors and are defined as follows:

$$\mathcal{Y}(t) \equiv [\mathbf{y}(t), \mathbf{y}(t-1), \dots, \mathbf{y}(t-n_y)], \quad (2.4)$$

$$\hat{\mathcal{Y}}(t|t-1) \equiv [\hat{\mathbf{y}}(t|t-1), \hat{\mathbf{y}}(t-1|t-2), \dots, \hat{\mathbf{y}}(t-n_y|t-n_y-1)]. \quad (2.5)$$

The aforementioned equations 2.5 can be written in a *prediction-update* form. In the prediction-update form of the state filter, the prediction step obtains a future estimate of the state $\hat{\mathbf{x}}(t+1|t)$ using measurements up to and including time step t . In the update step, the predicted state value, $\hat{\mathbf{x}}(t+1|t)$, is updated to account for the stochastic and/or modelling inaccuracies present in the prediction step. The *innovations* term in the filter account for the stochastic effects and system modelling uncertainties unpredictable in the prediction step. Since the updated state value already has been compensated by the innovations term, use of both $\mathcal{Y}(t)$ and $\mathcal{Y}(t|t-1)$ in obtaining the state/output prediction is not necessary. In this filter formulation, the most recent value of $\mathcal{Y}(t)$ is used instead of $\mathcal{Y}(t|t-1)$. The nonlinear functions $\mathbf{f}_{innov}(\cdot)$ and $\mathbf{h}_{innov}(\cdot)$ of the predictor can be approximated using neural networks as discussed further in [1].

The prediction and the update step equations using neural networks as approximators of the non linearities are non linear functions is presented below.

Step 1 - Prediction Step:

The state and output predictor values are obtained using the following equations:

$$\hat{\mathbf{x}}_{NN}(t+1|t) = \mathbf{f}_{NN}(\hat{\mathbf{x}}_{NN}(t|t), \mathbf{u}(t), \hat{\mathcal{Y}}_{NN}(t|t-1)),$$

(2.6)

$$\hat{\mathbf{y}}_{NN}(t+1|t) = \mathbf{h}_{NN}(\hat{\mathbf{x}}_{NN}(t|t), \mathbf{u}(t), \hat{\mathcal{Y}}_{NN}(t|t-1)),$$

where $\hat{\mathbf{x}}_{NN}(t+1|t)$ and $\hat{\mathbf{y}}_{NN}(t+1|t)$ are the neural state and output predictions, and where $\hat{\mathcal{Y}}_{NN}(t|t-1)$ is the vector containing the present and past output predictor responses.

Step 2 - Update Step:

The state prediction is updated using

$$\hat{\mathbf{x}}_{NN}(t+1|t+1) = \mathbf{K}_{NN}(\hat{\mathbf{x}}_{NN}(t+1|t), \mathcal{Y}(t+1), \mathcal{E}(t+1)), \quad (2.7)$$

where the vectors $\mathcal{Y}(\cdot)$, $\hat{\mathcal{Y}}_{NN}(t+1|t)$, $\mathcal{E}(\cdot)$ are defined as

$$\mathcal{Y}(t+1) \equiv [\mathbf{y}(t+1), \mathbf{y}(t), \dots, \mathbf{y}(t-n_y+1)]^T, \quad (2.8)$$

$$\hat{\mathcal{Y}}_{NN}(t+1|t) \equiv [\hat{\mathbf{y}}_{NN}(t+1|t), \dots, \hat{\mathbf{y}}_{NN}(t-n_y+1|t-n_y)]^T, \quad (2.9)$$

$$\mathcal{E}(t+1) = \mathcal{Y}(t+1) - \hat{\mathcal{Y}}_{NN}(t+1|t), \quad (2.10)$$

$$\mathcal{E}(t+1) \equiv [\epsilon(t+1), \epsilon(t), \dots, \epsilon(t-n_c+1)]^T, \quad (2.11)$$

and where ϵ is a $n_e \times 1$ vector and is defined as,

$$\epsilon(t+1) \equiv \mathbf{y}(t+1) - \hat{\mathbf{y}}_{NN}(t+1|t), \quad (2.12)$$

is the innovations term as defined in the standard Kalman Filter. The nonlinear functions represented by $\mathbf{f}_{NN}(\cdot)$, $\mathbf{h}_{NN}(\cdot)$, and $\mathbf{K}_{NN}(\cdot)$ are the neural networks ap-

proximations in the filter equations [1].

3. Speed Filter Description

The above formulation of the state filtering problem and its solution is applied to the problem of induction motor speed filtering.

Information about the induction motor such as number of pole pairs, p ; slip at rated load, $f_{s(RATED)}$; no load slip, f_{nl} ; are used in the filter development. The measurements include the motor currents and voltages. Rotor slot harmonic analysis is used to extract the target speed estimate for the neural network training [11].

The output predictor, the second of equations (2.6), consists of three feed forward multi-layered perceptron (FMLP) neural networks that predict the three motor currents $\hat{I}_a(t+1|t)$, $\hat{I}_b(t+1|t)$ and $\hat{I}_c(t+1|t)$, respectively. The state predictor, the first of equations (2.6) is modelled by a single FMLP neural network that predicts the induction motor speed $\hat{\omega}_{NN}(t+1|t)$.

The predictor equations (2.6) can be re-written with the currents and speed substituted as shown below:

$$\begin{aligned}\hat{\omega}_{NN}(t+1|t) &= \mathbf{f}_{NN}(\hat{\omega}_{NN}(t|t), \mathbf{u}(t), \hat{\mathbf{I}}_{NN}(t|t-1)), \\ \hat{\mathbf{I}}_{NN}(t+1|t) &= \mathbf{h}_{NN}(\hat{\omega}_{NN}(t|t), \mathbf{u}(t), \hat{\mathbf{I}}_{NN}(t|t-1)),\end{aligned}\tag{2.13}$$

where $\hat{\mathbf{I}}_{NN}(t|t-1)$ is the vector containing the history of the motor current predictions.

The update network contained in equation (2.7) is modelled by a FMLP neural network. This network updates the predicted speed $\hat{\omega}_{NN}(t+1|t)$ by taking into account the stochastic and/or modelling inaccuracies. The update equation (2.7) is re-written with the motor speed substituted as

$$\hat{\omega}_{NN}(t+1|t+1) = \kappa_{NN}(\hat{\omega}_{NN}(t+1|t), \hat{\mathbf{I}}_{NN}(t+1|t), \mathbf{I}_{RMS}(t+1), \mathcal{E}(t+1)), \quad (2.14)$$

where κ_{NN} is the filter gain, and the vectors are defined as

$$\mathbf{I}(t+1) \equiv [\mathbf{I}(t+1), \mathbf{I}(t), \dots, \mathbf{I}(t-n_y+1)]^T, \quad (2.15)$$

$$\mathcal{E}(t+1) \equiv [\epsilon(t+1), \epsilon(t), \dots, \epsilon(t-n_e+1)]^T, \quad (2.16)$$

and where,

$$\epsilon(t+1) \equiv \mathbf{I}(t+1) - \hat{\mathbf{I}}_{NN}(t+1|t), \quad (2.17)$$

is the innovations term. It should be noted that the predictor networks are dynamic FMLPs, whereas the update network is a static FMLP. A block diagram of the neural network based speed filter is shown in Figure 3.

The training schemes used to train the neural networks are briefly discussed in the following subsection.

4. Neural Network Training

In the preceding subsection, the neural network solution for the motor speed filtering problem was presented. The neural networks need to be trained before they can be implemented in the filter.

Training data set development is one of the most crucial step in neural network training as the accuracy of the state filter will depend on the accuracy of the values used to train it. Data collected from experiments was used to create the training set. The training set is constructed carefully so as to be a representative of the filter

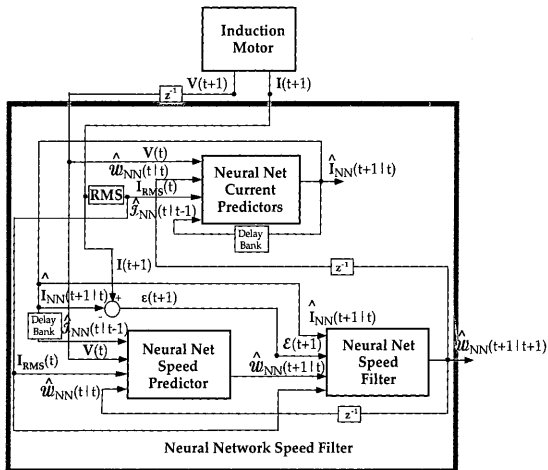


Fig. 3. Block Diagram of the Neural Network Speed Filter.

operating range. The evaluation (or cross-validation) data set is collected in a similar manner to the training data set.

The neural networks need to be trained with some target speed signal. In this study, the actual induction motor speed signal was not used for training because of the assumption that the induction motors in industrial setups do not have speed sensor. The speed signal used for training the neural networks in this study is obtained using rotor slot harmonic (RSH) analysis of the induction motor current. More information about the accuracy is discussed in chapter IV.

The networks were trained separately before they are coupled with each other. The predictor networks are trained before being coupled to the speed update network. Of the predictor network bank, the line current predictors are trained before the speed predictor network. Finally, the speed update network is trained.

The network thus trained is applied to the setup from which the training data is gathered. When the filter needs to be implemented in other induction motors, some amount of tuning of the weight files is necessary to improve the accuracy of the filter.

B. Chapter Summary

In this chapter, the details of the neural network induction motor speed filter developed in [1] were presented. The filter developed was primarily tested for off-line conditions where the data collection and speed filtering are done separately. The filter will now be re-written so that it can be implemented on-line where data collection and data processing is done simultaneously. In the next chapter the details of the developed neural network framework and LabVIEW development and integration details are presented.

CHAPTER III

ON-LINE IMPLEMENTATION OF SPEED FILTER

This chapter deals with the on-line development and integration of the neural networks based speed filter discussed in the preceding chapter. This chapter is divided into four sections. In the next section, the filter architecture is discussed. In section B the issues involved in the development of the neural networks framework are described. In section C the components of the neural network speed filter are presented. In the last section, a summary of the chapter is presented.

A. Neural Networks Architecture of the Speed Filter

The neural networks based filter presented in chapter II is made up of five neural networks that are interconnected. Three 8-5-1 dynamic FMLP neural networks are used to predict the three line currents, I_a , I_b and I_c . The inputs to the predictors are the three line voltage $[V_A(t), V_B(t), V_C(t)]$, latest available speed estimate $\hat{\omega}_{NN}(t|t)$, the root-mean-square (RMS) of line current $I_{A,RMS}(t)$ and the delayed current predictions $[\hat{I}_{NN,A}(t|t-1), \hat{I}_{NN,B}(t|t-1), \hat{I}_{NN,C}(t|t-1)]$. The outputs of the predictor are $\hat{I}_{NN,A}(t+1|t)$, $\hat{I}_{NN,B}(t+1|t)$ and $\hat{I}_{NN,C}(t+1|t)$. The structure of the predictor bank is shown in Figure 4.

The speed predictor is characterized by a 9-7-1 static FMLP network. The speed predictor uses the three sampled line voltages $[V_A(t), V_B(t), V_C(t)]$, the three current predictions $[\hat{I}_{NN,A}(t|t-1), \hat{I}_{NN,B}(t|t-1), \hat{I}_{NN,C}(t|t-1)]$ and the latest available filtered speed $\hat{\omega}_{NN}(t|t)$ to predict the motor speed $\hat{\omega}_{NN}(t+1|t)$. Figure 5 depicts the speed predictor.

The speed update network uses the innovations $[\epsilon_A(t+1), \epsilon_B(t+1), \epsilon_C(t+1)]$ from the current predictions, the line current predictions $[\hat{I}_{NN,A}(t+1|t), \hat{I}_{NN,B}(t+1|t), \hat{I}_{NN,C}(t+1|t)]$

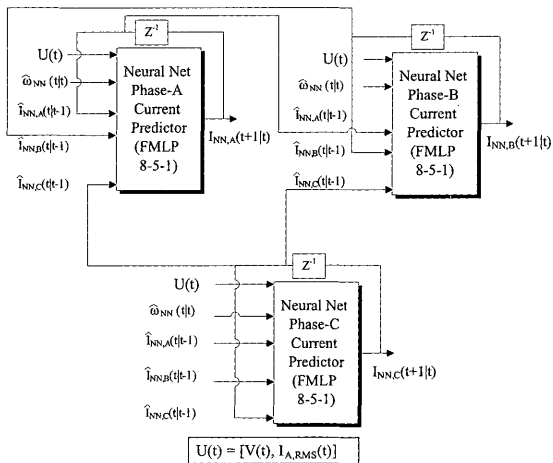
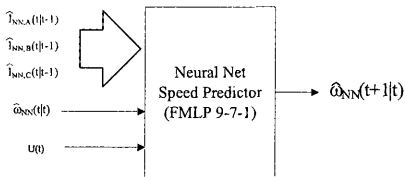


Fig. 4. Neural Network Architectures of the Current Predictors.



$$U(t) = [V(t), I_{A,RMS(0)}]$$

Fig. 5. Neural Network Architecture of the Speed Predictor.

$1|t)$, $\hat{I}_{NN,C}(t+1|t)$], the RMS value of the line currents and the predicted speed $\hat{\omega}_{NN}(t+1|t)$ from the speed predictor network to generate the filtered speed estimate, $\hat{\omega}_{NN}(t+1|t+1)$. The speed update network is characterized by a static 8-16-1 FMLP network. The speed update network is shown in Figure 6.

B. Neural Network Framework

As described in the previous section, the speed filter is composed of five couple neural networks of different architectures. Also, the structure and/or the architecture of the networks can change in future implementations. Hence a generic neural networks framework that can be used to construct such entities is needed. The following subsection discusses the development of such neural networks framework.

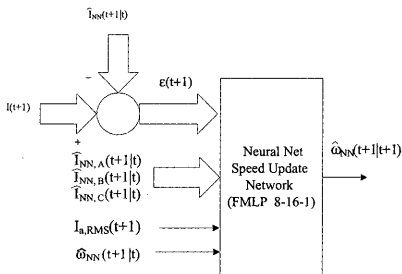


Fig. 6. Neural Network Architecture of the Speed Update.

1. Neural Networks Implementation Framework Development

The implementation framework is developed using the C programming language. The framework is used for constructing different entities like filters and predictors, that use different combinations of neural networks. The inputs to the framework are the neural network data, such as the architecture of the networks to be constructed and the weights and biases for each of the networks. The inputs and outputs of the neural networks framework implementation of the speed filter are shown in Figure 7. A control script is used to specify the architecture of the entity to be constructed. The details of this control script are presented in the following subsection.

2. Control Script

The control script contains the following information:

1. Number of networks

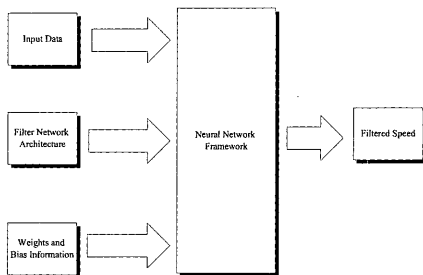


Fig. 7. Inputs and Outputs of Neural Network Framework Implementation of the Speed Filter.

2. Number of input channels
3. Input data mode
4. Mode of operation of each network
5. Number of layers per network
6. Number of nodes per layer per network
7. Weight and Bias filename per network
8. Input layer mapping for each network

Each of the input specified in the script is explained below:

Number of networks:

This field specifies the total number of networks that are used to construct an entity. In the case of the speed filter, this value is equal to five.

Number of input channels:

This quantity specifies the number of external inputs sent to the filter. This quantity does not refer to the number of inputs to each of the networks in the filter, but corresponds to the total number of external input signals sent to the filter. Depending on the input data mode, this corresponds to either the number of channels in the data acquisition system or to the number of columns in the input data file. In the case of the speed filter, this is value equal to eight.

Input data mode:

This field specifies the mode of data input. The data to the network can come from a file as in the case of an off-line run, or from a data acquisition system as in the case of an on-line execution.

Mode of operation:

This field is specified for each of the network. A network can execute either in static (FMLP) mode or in recurrent (RMLP) mode. A network said to be in RMLP mode when there are *cross-talk* between the nodes in the hidden layer. The value of 0 and 1 correspond to FMLP and RMLP respectively. Since this quantity is specified for each of the networks, an entity can contain different combination of networks operating in either one of these modes.

Number of layers:

This field is valued for each of the networks. The number of layers of a network is specified in this field.

Number of nodes per layer:

This field is specified for each of the networks. The number of nodes per layer is specified in this field. The number of nodes in the output layer is always set equal to

one.

Weight and bias filename:

The weight and bias file information is specified in this field. One file per network is specified.

Input layer mapping:

The program requires precise information regarding the mapping of input data point to the input layer nodes of each of the network. This set of inputs contains the information regarding the assignment of data to the input layer for each of the neural network. This field is equal to the number of input nodes per network. The information provided by this field specifies the source of data (i.e., measurement or predictions), delay value of data and/or any arithmetic operation that needs to be performed between different data (i.e., an arithmetic operation such as addition or subtraction can be performed between two data samples and the result of that operation can be specified as one of the inputs).

3. Neural Network Process Flow

The control script provides all the information necessary to construct and execute different combinations of neural networks. The control script is read, and based on the information regarding the number of layers per network and number of nodes per network, appropriate amount memory for weights and biases is allocated. The next step is to load the data points into a bi-directional link-list structure. It can be seen that there is no need to specify the number of input data samples in the control script. The link-list structure is necessary because in the control script no information is specified regarding the number of data points to be processed. Once all the data are loaded, the processing through the network is initiated. For every data point selected, all the networks are processed sequentially. The input layer of a network

selected for processing is assigned with appropriate data values based on the mapping information provided in the control script. The next step is to perform an FMLP or an RMLP forward pass on the network based on the specifications provided in the control script. The above routine is performed for all of the data points provided. In special cases, when the mapping of the input layer specifies unavailable data (i.e. delayed values of an output), the input node is assigned a zero and the processing is continued till the appropriate data become available. The flow chart for the execution described is presented in Figure 8.

C. Components of the On-line Speed Filter

This section describes the components of the on-line speed filter. The data collected from the induction machine consist of speed signal from speed sensor, three line voltages, and the three line currents. A series of steps, namely data preprocessing, speed filtering and data postprocessing are performed on the data collected from the induction motor to estimate the speed. Figure 9 shows the components of the speed filter. Each of the step is discussed in detail in the following subsections.

1. Data Collection Module

Data collection is performed in seven channels by a LabVIEW data acquisition system. The input data consists of three line voltages, three line currents and speed signal sampled at 3840 Hz. It should be noted that the speed signal is not used in processing at any time by the speed filter. The speed sensor signal is collected only for comparison of the speed estimate of the filter. The filter cut-off is set at the Nyquist frequency to prevent signal aliasing. The sampled data are buffered and are made available to the filter modules. The sampled measurements are referred to as the *Raw Data*. The

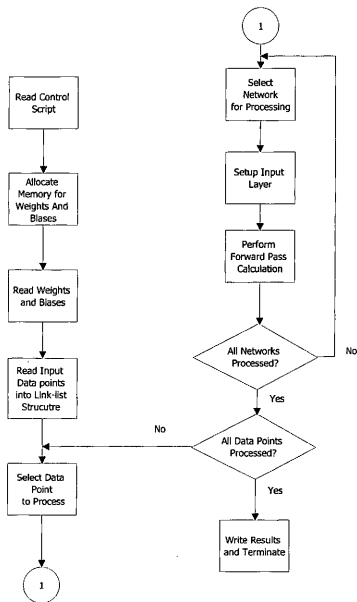


Fig. 8. Process Control Flow of the Neural Networks Framework.

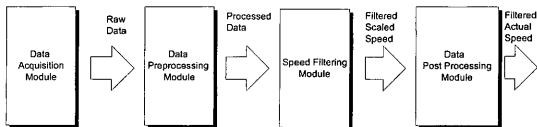


Fig. 9. Components of the Speed Filter.

size of the data scanned varies and it can be controlled by the user depending on the desired speed of operation of the speed filter.

2. Data Preprocessing Module

Raw data from the data acquisition system must to be preprocessed before it can be used by the neural networks. The preprocessing consists of three steps namely, RMS calculation, cropping and scaling. This series of data preprocessing is described below.

RMS Calculation:

The neural network speed filter requires the RMS of the line current as one of the inputs. This information is calculated from the raw line currents. A moving window RMS calculation is performed on the input currents. The window size varies from 2 cycles to 6 cycles depending on the desired speed of filter operation. There is a trade-off between filter accuracy and speed ,as the RMS window is varied. It should be noted that an even number of cycles is used for the RMS calculation. The moving distance of the window is set at 1, but this value can be changed by the user

as desired. If the input power supply is balanced, the RMS is calculated from one of the three line currents selected arbitrarily. But, if the input power supply is not balanced, then RMS calculation is performed on all the three line currents and the average of the three is used. Even in the case of a balanced supply, the average RMS should be used because of the imbalance due to faulty motors.

Cropping and merging of the data:

The data length of the resulting line current RMS calculations is less than the length of the other data signals by an amount equal to the window size used for the RMS calculations. Thus, the rest of the data must to be cropped to the size of the RMS array size before they can be merged. The cropping module removes equal amount of data in the leading and the trailing ends of the currents, voltages and speed data. The RMS value is inserted as the fifth column after the speed and the three voltages in the data pool. This operation is performed using a LabVIEW program.

Scaling of the data:

The neural networks used in the speed filter require the input data to be in the range +0.5 and -0.5. Data beyond the limit of ± 0.5 will cause the filter networks to saturate. Hence, the cropped and merged data is scaled based on some scaling transformation and associated parameters. The scaling parameters are selected based the data set present in the network training. In the current implementation, the scaling developed in [1] is used.

The preprocessing steps for balanced power supply is shown in Figure 10. Figure 11 shows the preprocessing for unbalanced power supply and unbalanced motor stator.

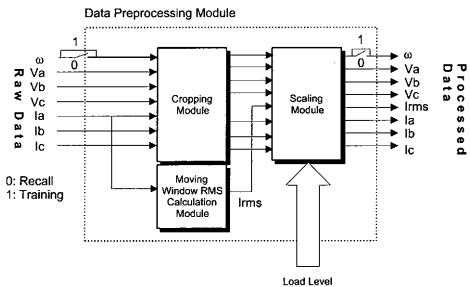


Fig. 10. Block Diagram of Preprocessing for Balanced Power Supply or Motor Stator

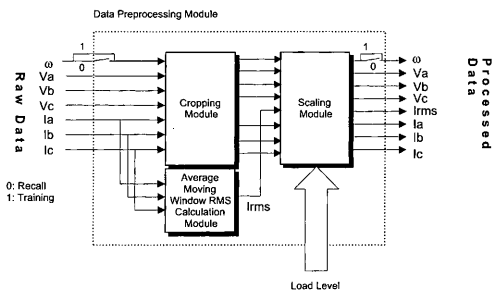


Fig. 11. Block Diagram of Preprocessing for Unbalanced Power Supply or Motor Stator.

3. Speed Filter Module

The processed data is passed to the speed filter for calculating the speed information. The neural network framework is used to construct the speed filter. In this implementation two filters, one for 0% – 70% and another 70% – 120% of full load, are used to estimate the speed based on the motor load level. The RMS value of the line current is used to determine the motor loading level. In order to use the appropriate filter parameters during load changes, the RMS value of the current is monitored. The load level detected is used to select the appropriate filter based on the load level detected. Since the RMS is monitored continuously, the filter is able to estimate speed during load variations. The load-based filter selection scheme is presented in Figure 12.

The neural networks framework is compiled as a dynamic link library (DLL) that can be called by a special LabVIEW routine known as the *call library function*. This function passes the data from the preprocessing module to the DLL and retrieves the filtered speed results from the DLL. The speed filter module is presented in Figure 13.

4. Data Postprocessing Module

The speed estimated by the speed filter module does not reflect the actual speed of the motor. The speed values needs to be un-scaled to indicate the actual speed of the motor. The parameters used for scaling the input data are used in the un-scaling process. The un-scaled speed is presented in a waveform chart. The data can also be stored in a file or passed to any other application that requires speed information.

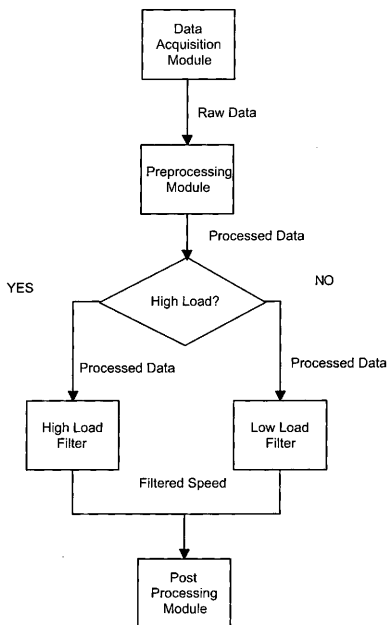


Fig. 12. Motor Load-based Filter Selection Scheme.

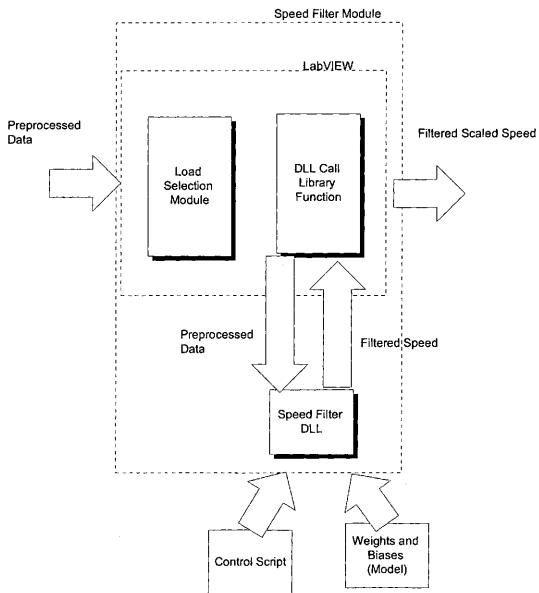


Fig. 13. Speed Filter Module.

D. Chapter Summary

In this chapter, the development of the neural networks framework as presented. Then the implementation of the speed filter using the developed neural network frameworks and the LabVIEW environment was discussed. In the next chapter, the off-line and the on-line case studies performed for filter validation and the associated speed estimation results are presented.

CHAPTER IV

EXPERIMENTAL DEMONSTRATION OF THE SPEED FILTER

This chapter presents the response of the speed filter to different sets of pre-recorded data and to data from the experimental setup used especially in this research. This chapter is divided into two categories, namely, off-line and on-line collection. The case studies with off-line data results are presented in the first section. In section B, the implementation of the speed filter with on-line data collection and the associated results are presented. The filter implemented used in both off-line and online data sets is the same. Only the source of data is changed. The last section summarizes the results of the speed filter experimental demonstrations. One of the application of the speed filter is in the area of induction motor fault detection. Most of the fault detection strategies require a motor speed estimate. Once the performance of the speed filter under faulty motor conditions is studied, it can also be used in fault detection schemes. Hence, the speed filter is tested with data collected from both healthy and faulty motors.

A. Speed Filter Tests with Off-Line Data Collection

The neural networks based speed filter is tested with data collected from different test beds. This was done to study the filter performance under different operating conditions. Off-line data is available for small machines (3 hp) and large machines (500 and 800 hp). As mentioned earlier, the off-line data is collected from both healthy and faulty motors so that the filter performance can be studied. The following subsection presents the experimental setup used in the off-line data collection from small and large machines.

1. Experimental Setups Used in Off-Line Motor Data Collection and Speed Filter Tests

a. Small Machine Setup

The small machine setup consists of a 3- ϕ , 4 pole, 3 hp induction motor powered by supply mains operating at 60 Hz. The motor is connected to two DC generators in tandem. The first DC generator is used to load the induction motor. The load on the motor is changed by varying the armature resistance of the DC generator. The second DC generator is used to measure the speed signal which is used for comparisons with the speed filter results. An 8-channel LabVIEW data acquisition system is used to record the three line voltages, the three line currents and the speed signal. All of the signals are sampled at 3840 Hz.

b. Large Machine Setup

The large machine data are available for a 500 hp and a 800 hp machine. Experiments were conducted at *Public Service Electric and Gas Motor Repair Facility*, Sewaren, New Jersey. The 500 hp, 3- ϕ , 6 pole and the 800 hp, 3- ϕ , 8 pole inductions motors are run directly from mains. The motor was connected to a dynamometer which was used to load the induction motor. A 13-channel IOtech data acquisition system was used to record the three line voltages, the three line currents, encoder speed signal, and the four vibration signals. The signals were sampled at 40 KHz sampling frequency. The current and voltage signals are then downsampled to 3840 Hz. The vibration signals are not used by the filter for estimating the induction motor speed.

2. Speed Filter Results with Off-Line Data Collection

This subsection presents the results of the neural network based speed filter operating on the off-line data collected from small and large machines.

The speed filter structure that is implemented is described in detail in the preceding chapter. The filter components are implemented in LabVIEW. The raw data collected from the induction motor are downsampled to 960 Hz and the average RMS of the line current is calculated. The data is then scaled between -0.5 to 0.5 using the appropriate scaling parameters and is passed to the speed filter. The speed filter, which is a dynamic link library (DLL) developed in the C programming language, accepts data from the LabVIEW subsystem and returns the filtered speed back to the LabVIEW subsystem. Separate weight and bias files are used to estimate the the speed for the low load and high load region. A parameter passed from LabVIEW to the DLL indicates the appropriate weight and bias files to be used during speed estimation. The LabVIEW program is equipped with controls to change the filter parameters, such as the size of data used for processing, and the RMS window size. For the small machine tests, two sets of weight files, one for low motor load ($0\% - 70\%$) and another for high motor load ($70\% - 120\%$) are used. One set of weight and bias file 500 hp and another for the 800 hp machines, respectively.

a. Small Machine Test Results

All the results presented are compared with the rotor slot harmonic speed estimate. This estimate is obtained based on some speed dependent harmonics in the current signal and is considered as a good estimate of the speed. Another reason for using the estimate is that the neural networks are trained using the harmonic speed estimate. This makes further tuning of networks possible without the need of a speed sensor.

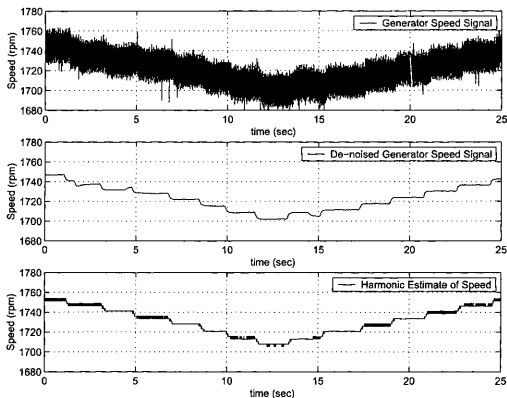


Fig. 14. top: Speed Signal from Generator; middle: De-noised Speed Signal; bottom: Harmonic Speed Estimate.

Figure 14 shows the harmonic estimate for small machine speed signal. It can be seen that the estimate is within 5 – 10 rpm of the actual speed. It should be noted that the rotor slot harmonic speed estimate can be done only in off-line conditions because of computation delays.

The filter estimation errors are calculated for each of the case study presented in this work. The average and maximum error percentage is presented at the end of each section. The estimation error is calculated as,

$$Error = \frac{(\omega_{RSH} - \hat{\omega}_{NN})}{\omega_{RSH}} \quad (4.1)$$

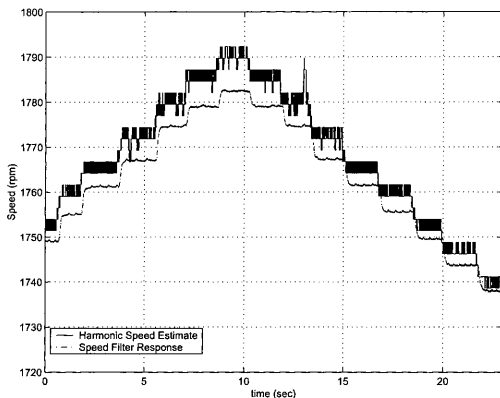


Fig. 15. Speed Filter Response for 3 hp Healthy Motor; 0% – 70% Load Range.

where $\hat{\omega}_{NN}$ and ω_{RSH} correspond to filter speed estimate and harmonic speed estimate, respectively. The response of the speed filter for a healthy motor operating under low load (0% – 70%) and high load (70% – 120%) conditions is presented in figures 15 and 16.

In industrial setups, the 3- ϕ power supply is not always balanced. Due to the supply imbalance, the speed of the machine will be reduced. The filter was also tested with data collected from motors with input supply imbalance. Figures 17 and 18 show the filter response in the low load and high load range under 0.5% unbalanced power supply condition.

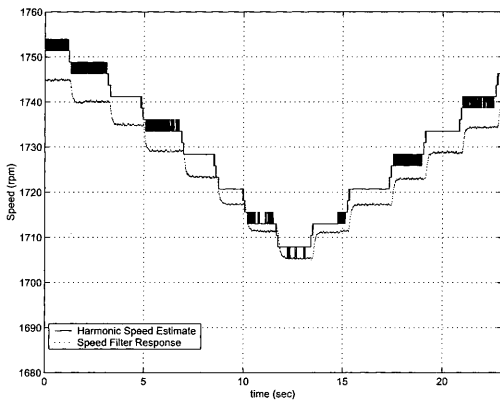


Fig. 16. Speed Filter Response for 3 hp Healthy Motor; 70% – 120% Load Range.

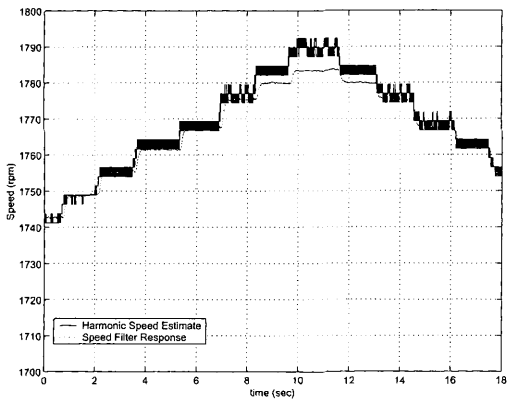


Fig. 17. Speed Filter Response for 3 hp Healthy Motor; 0% - 70% Load Range, with 0.5% Unbalanced Power Supply.

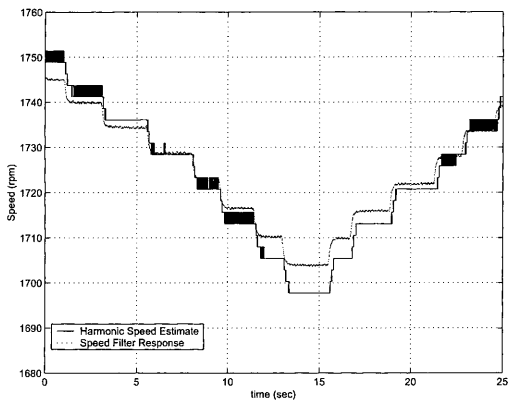


Fig. 18. Speed Filter Response for 3 hp Healthy Motor; 70%–120% Load Range, with 0.5% Unbalanced Power Supply.

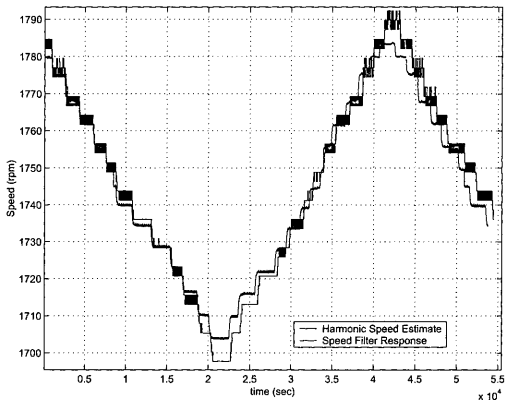


Fig. 19. Speed Filter Response for 3 hp Healthy Motor; 0% – 120% Load Range, with 0.5% Unbalanced Power Supply.

The response of the speed filter when the load changes from the low load region to the high load region is presented in Figure 19. The RMS of the motor current is monitored and appropriate filter parameters are used depending on the motor load level detected.

Figures 20 and 21 show the filter response to 4.5% unbalanced supply in the 0% – 70% load range, and 5.4% unbalanced supply in the 70% – 120% load range, respectively. It can be seen that the filter performance does not degrade significantly.

The filter was also tested with signals collected from motors with broken rotor

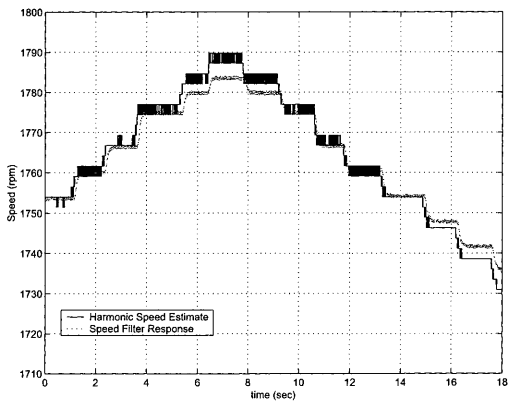


Fig. 20. Speed Filter Response for 3 hp Healthy Motor; 0% – 70% Load Range, with 4.5% Unbalanced Power Supply.

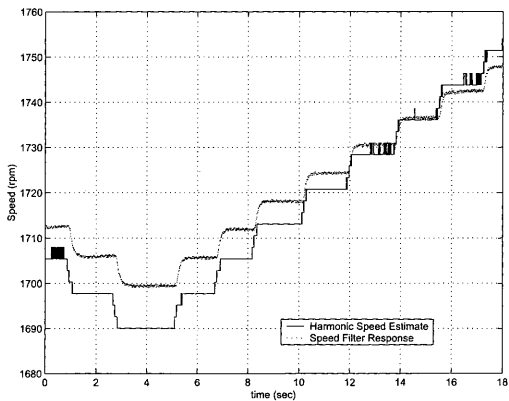


Fig. 21. Speed Filter Response for 3 hp Healthy Motor; 70%–120% Load Range, with 5.4% Unbalanced Power Supply.

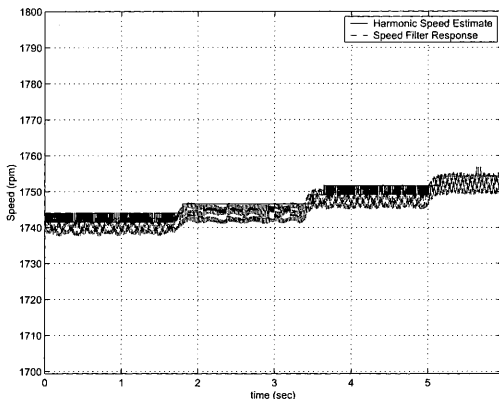


Fig. 22. Speed Filter Response for 3 hp Motor with 3 Broken Rotor Bars; 0% – 70% Load Range.

bars. Figures 22 and 23 show the filter response with three and four broken rotor bars, respectively. It should be noted that the speed filtering is done with the filter parameters developed for a healthy motor. Due to this reason, the filter performance somewhat suffers.

The filter performance for the large machines is presented in Figures 24 and 25. Figure 24 shows the filter response for the 500 hp motor and Figure 25 shows the filter response for the 800 hp motor. It can be seen that the filter performs reasonably well in both the cases. All the cases presented in this study used the RMS value of the current calculated using 4 cycles of data.

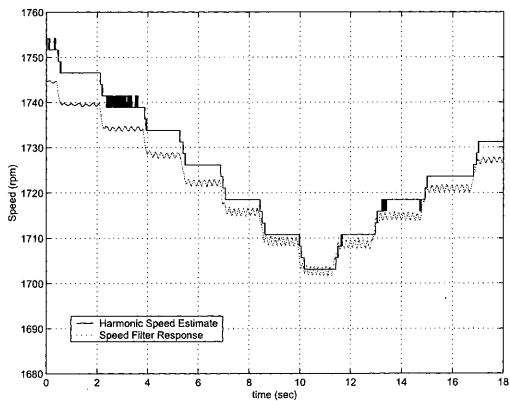


Fig. 23. Speed Filter Response for 3 hp Motor with 4 Broken Rotor Bars; 0% – 70% Load Range.

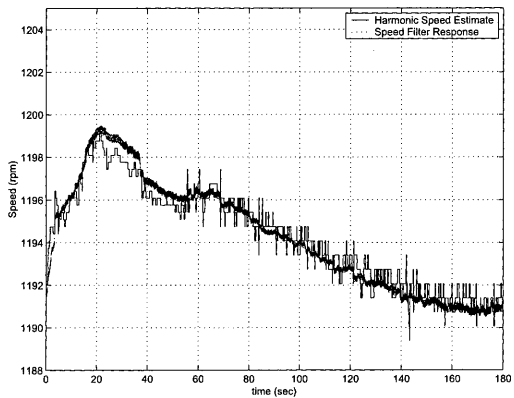


Fig. 24. Speed Filter Response for Healthy 500 hp Motor; 100% – 50% Load Range.

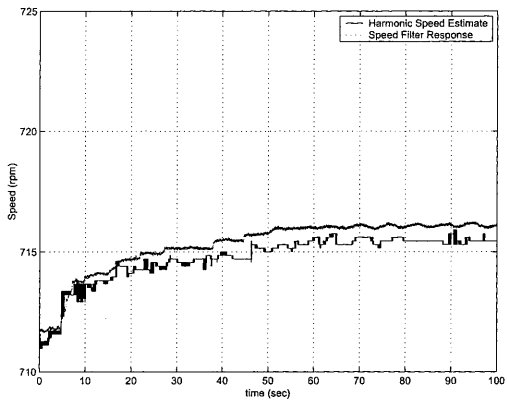


Fig. 25. Speed Filter Response for Healthy 800 hp Motor; 100% – 50% Load Range.

The maximum and average estimation errors for the above case studies with healthy small machines are presented in Table I. The average and maximum estimation errors for the case studies with faulty small machines are presented in Table II. Table III

B. Speed Filter with On-line Data Collection

One of the main objectives of this work is to implement the neural networks based filter in an online environment. In the preceding section, the results presented are from data collected at different times from different setups. In order to use the filter in on-line fault detection schemes, the filter should act on real-time data generating speed estimates at real-time or close to real-time. This would eliminate the need for expensive speed sensors, that will render the fault detection schemes financially unattractive. Moreover, the speed sensors are less robust and are more prone to failures than the induction motor itself. This would reduce the overall robustness of the induction motor setup. Lastly, most industrial setups do not have speed sensors as a standard component and it will be easier to use this filter as an alternative with minimal cost increase. The above mentioned reasons are the motivation for this work.

The test results with off-line data collection prove the efficacy of the filter in estimating the speed from the current and voltage signals. In the case studies with off-line data collection off-line implementation, the speed of execution of the filter is not a major factor as the program acts on the data as a batch, and the results are generated once all the data are processed. In an on-line setup, the filter acts on a small window of data and the estimates of the speed are generated as soon as the data window is processed. Ideally, the window should be just one data point for each of the three line currents and the three line voltages. But, the RMS calculation requires

Table 1. Speed Estimation Errors: Healthy Small Machines; Off-line Data

Case Study	Load Level	Average Estimation Error (%)	Average Estimation Error (rpm)	Peak Estimation Error (rpm)	Peak Estimation Error (%)
Healthy Small m/c	Low (0%-70%)	0.26	4.65	14.79	0.83
Healthy Small m/c	High (70%-120%)	0.29	5.08	13.73	0.78
Healthy Small m/c; 0.5% Unbal.	Low (0%-70%)	0.13	2.23	10.09	0.56
Healthy Small m/c; 0.5% Unbal.	High (70%-120%)	0.16	2.67	10.30	0.60
Healthy Small m/c; 4.5% Unbal.	Low (0%-70%)	0.15	2.56	10.41	0.58
Healthy Small m/c; 5.4% Unbal.	High (70%-120%)	0.30	5.12	14.71	0.87

Table II. Speed Estimation Errors: Faulty Small Machines; Off-line Data

Case Study	Load Level	Average Estimation Error (%)	Average Estimation Error (rpm)	Peak Estimation Error (rpm)	Peak Estimation Error (%)
Three Broken Rotor Bars; Small m/c	Low (0%-70%)	0.15	2.68	7.50	0.15
Four Broken Rotor Bars; Small m/c	Low (0%-70%)	0.57	3.79	10.09	0.22

Table III. Speed Estimation Errors: Healthy Large Machines; Off-line Data

Case Study	Load Level	Average Estimation Error (%)	Average Estimation Error (rpm)	Peak Estimation Error (rpm)	Peak Estimation Error (%)
Healthy Large m/c (500 hp)	Mix (50%-100%)	0.03	0.41	2.21	0.18
Healthy Large m/c (800 hp)	Mix (50%-100%)	0.09	0.56	11.30	1.59

a minimum of one cycle and hence the minimum window that can be processed is equal to one cycle length. The following subsection presents the experimental setup used in the on-line data collection and speed filter tests.

1. Experimental Setup Used in On-line Data Collection and Speed Filter Tests

This experimental setup consists of a 3- ϕ , 3 hp, 2 pole induction motor running of a 3- ϕ supply mains operating at 60 Hz. The motor is connected to a torque meter that includes an optical encoder to detect the motor speed. The load of motor include two rotor disks, a gear box and a centrifugal pump. The rotor disks are perfectly aligned and the gear box is used to increase or decrease the speed depending on the specific requirements. The centrifugal pump is connected to a variable area valve and a variable height water reservoir. By changing the valve position from 25% open to 100% open and by changing the head applied to the pump by positioning the reservoir at different heights, the load applied to the induction motor can be varied. In this setup however, the effect of the valve position and head changes on the motor load are negligible compared to the static load from the rotor disks and the gear box. This limitation prevented the filter from being tested online under varying load conditions. The encoder speed is used in determining the speed filter accuracy. The motor line currents and phase voltage are measured current transformer (CTs) and potential transformers (PTs), respectively. A 16-channel LabVIEW data acquisition system is used to record data from the experimental setup. The data are sampled at 3840 Hz. The sampled data are sent to the LabVIEW subsystem residing on a Intel Pentium III based computer. A schematic diagram of the experimental setup used for online filter testing is shown in Figure 26.

The LabVIEW subsystem provides various means to access the collected data. In this work, the data was buffered and then read by the LabVIEW filter framework

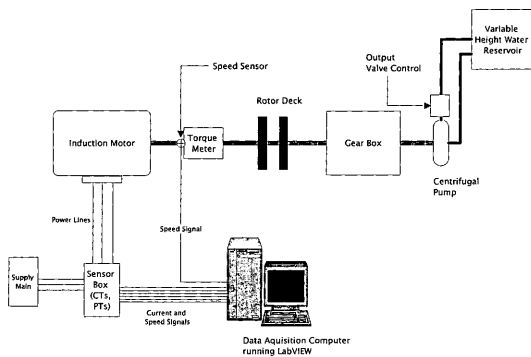


Fig. 26. On-line Experimental Setup Used for On-line Speed Filter Tests.

program. The amount of data to be scanned and the sampling frequency can be controlled by the user. Different amounts of data were scanned to test the operating speed of the speed filter. The results of this on-line implementation are presented in the following paragraphs.

The filter parameters used in used for this implementation are the same as those used in the tests with off-line data collection. The filter parameters are later tuned with data collected from this experimental setup to improve filter performance.

2. Speed Filter Results with On-line Data Collection

Due to the previously described limitations in changing the motor load level, the motor was operated at a constant load level. The data from the motor are used for tuning the off-line filter weights. The training requires a harmonic speed estimate as a target. The harmonic speed estimate was obtained from a FFT-based program that detects some speed dependent harmonics in the motor current. Since the data does not contain any transients, the training set was compiled using constant motor load data collected at different times. Tuning was performed for 100 iterations to obtain the updated speed filter parameters. The results presented here include contain both un-tuned and tuned speed filter tests. The estimation errors for the tuned and un-tuned speed filter are presented in Table IV.

Figure 27 shows the response of the un-tuned speed filter. The RMS window used is 4 cycles, and 3 seconds of data is scanned per iteration.

The filter performance after tuning presented in Figure 28. The filter operating speed can be changed by changing the number of data points scanned at every iteration. It is found that for a data window of 0.05 sec, 0.09 seconds are needed to filter the speed on an average. It can be seen that the filter does not operate in real-time, but this result is good enough to be used in fault detection schemes, since faults do

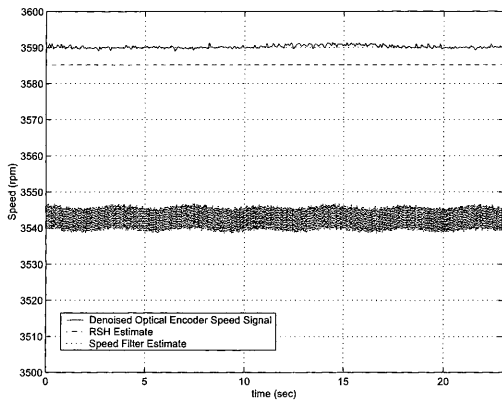


Fig. 27. Online Speed Filter Response for Healthy 3 hp Motor without Tuning; 30% Load.

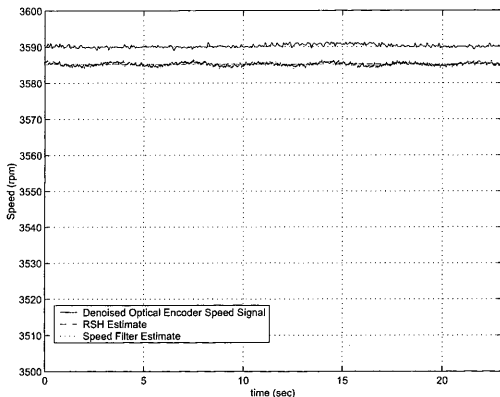


Fig. 28. Online Speed Filter Response for Healthy 3 hp Motor with Tuning; 30% Load.

not change drastically with time. The errors presented in the Table IV show that the filter can be tuned effectively with less effort and the performance of the filter can be improved significantly.

C. Chapter Summary

In this chapter, the speed filter implementation and testing using both for the off-line and on-line data collection is presented. The speed filter response under unbalanced power supply conditions and faulty motor scenarios is also presented. In the case of the faulty motors with broken rotor bars, it is seen that the speed filter performance

Table IV. Speed Estimation Errors: Small Machines; Data from On-line Setup

Case Study	Load Level	Average Estimation Error (%)	Average Estimation Error (rpm)	Peak Estimation Error (%)	Peak Estimation Error (rpm)
Healthy Small m/c; Before Tuning	Low (0%-70%)	1.18	42.53	1.30	46.76
Healthy Small m/c; After Tuning	Low (0%-70%)	0.01	0.36	0.03	1.29

does not deteriorate drastically. The on-line implementation details, the speed filter result and the estimation errors under constant load conditions is also presented in this chapter.

CHAPTER V

SUMMARY AND CONCLUSIONS

In this chapter, the research presented in earlier chapters is summarized. The neural networks framework development, the speed filter components and the filter implementation details is revisited. Finally, research contribution and suggestions for future research are presented.

A. Summary of Research

The objective of this work was to develop a generic neural network framework and to implement the neural networks based speed filter in an online environment. The framework was developed so that new entities using different combinations of neural networks could be easily constructed. Different components were developed to process the raw data from the induction motor setup before it can be used by the neural networks.

In chapter I, the motivation and background information for this work is presented. A brief note on current techniques for induction motor speed estimation and neural networks are also presented. Finally, the objective and the procedure of this work is clearly delineated.

In chapter II, the details of the neural networks based filter and the theory behind it are presented. A brief note on the training of neural networks is also presented in this chapter.

In chapter III, the neural networks architecture of the speed filter is presented. The filter is made up of five FMLP neural networks that are coupled to generate the speed estimate. Three networks predict the three motor currents. One network acts as the speed predictor and predicts the speed of the motor, and another network

updates the predicted speed based on latest available measurements. The details of the neural network framework design and development are also presented in this chapter. It can be seen that the framework is developed so that the design of the filter can be modified without much coding effort. All of the filter neural network parameters can be changed using the control script. The different components of the online speed filter are then discussed.

In chapter IV, the speed filter implementation details using off-line and on-line data collection are presented. The experimental setup used in off-line data collection and the response of the speed filter in many case studies are also presented. The filter was tested with data collected from faulty machines and unbalanced supply. It can be seen that the filter performance is reasonable in the presence of unbalanced power supply, but there is a notable performance loss when faulty machines are used. The filter's on-line implementation is discussed and the filter results with on-line data collection are presented. Due to limitations in the experimental setup used for on-line data collection, on-line filter response during load changes could not be studied. The filter results before and after tuning are also presented. The filter was tuned with less than 200 training iterations. This highlights the good generalization capabilities of the filter.

B. Conclusion

The objectives of this work are to develop a generic neural networks framework and to implement a previously developed speed filter in an on-line environment. Both the objectives are met in this study. The neural networks framework is capable of constructing new entities using different combinations of neural networks. There is no restriction on the number of neural networks, the number of layers, nodes per layer

(except the last layer should always be one), coupling between networks and input layer mapping. The implemented filter was able to generate speed estimates in the on-line conditions. The processing speed of the filter is not in real-time due to need to calculate the RMS of the line current and due to speed limitations in hardware. But, the accomplished processing speed is sufficient for the filter to be used in motor fault detection schemes, as these algorithms are not very sensitive on real time speed estimates. Implementation of the speed filter in a digital signal processor (DSP) environment should speed-up processing, but the inherent limitation imposed by the calculation on the RMS current remains.

C. Recommendation of Future Research Work

This research shows the feasibility of implementing neural networks based speed filters on-line. Such filters can be used instead of a speed sensor in limited cases. Some of the possible topics for future work are:

1. The speed filter processing speed can be improved by using specialized hardware to generate the RMS of the line current instead of calculating it in the program. More work can be done in this direction to allow operation of the filter in real-time.
2. Integration of an adaptation mechanism in speed filter to take care of model drift is another direction to follow.
3. Improving the training methods would help in better estimates of the speed. This will have a positive impact on the overall performance of the filter.

REFERENCES

- [1] R. M. Bharadwaj, "Adaptive State Filtering with Neural Networks for Sensorless Induction Motor Speed Estimation," Ph.D. dissertation, Texas A&M University, College Station, Texas, Dec 2000.
- [2] S. K. Menon, "Adaptive Filtering in Complex Process Systems Using Recurrent Neural Networks," Ph.D. Dissertation, Texas A&M University, College Station, Texas, 1996
- [3] D. J. T. Siyambalapitiya and P. G. McLaren, "Reliability Improvement and Economic Benefits of Online Monitoring Systems for Large Induction Machines," in *IEEE Transactions on Industry Applications*, vol. 26, no. 6, pp. 1018-1025, 1990.
- [4] J. R. Cameron, W. T. Thompson, and A. B. Dow, "Vibration and Current Monitoring for Detecting Airgap Eccentricity in Large Induction Motors," in *IEE Proceedings, Part-B*, vol. 133, no. 3, pp. 155-163, 1986.
- [5] P. Vas, *Parameter Estimation, Condition Monitoring, and Diagnosis of Electrical Machines*, Clarendon Press, Oxford, 1993.
- [6] J. Penman and A. Stravrou, "Broken Rotor Bars: Their Effect on Transient Performance of Induction Machines," in *IEE Proc.-Electron. Power Appl.*, vol. 143, no. 6, pp. 449-457, 1996.
- [7] E. von Westerholt, M. P.-David, and B. de Fornel, "Extended State Estimation of Nonlinear Modeled Induction Machines," in *Proc. of Power Electron. Spec. Conf.*, 1992, pp. 271-278.

- [8] Y.-R. Kim, S.-K. Sul, and M.-H. Park, "Speed Sensorless Vector Control of Induction Motor Using Extended Kalman Filter," in *IEEE Transactions on Industry Applications*, vol. 30, no. 5, pp. 1125–1233, September/October 1994.
- [9] M. V.-Reyes and G. C. Verghese, "Subset Selection in Identification, and Application to Speed and Parameter Estimation for Induction Machines," in *Proc. of the Fourth IEEE Conf. Appl.*, 1995, pp. 991–997.
- [10] M. Sumner, B. Conroy, and T. Alexander, "Evaluation of Encoderless Vector Control Techniques for Induction Motor Drives," in *Proc. of the IEEE Symp. of Ind. Electron.*, July 1995, pp. 315–320.
- [11] K. D. Hurst and T. G. Habetler, "Sensorless Speed Measurement using Current Harmonic Spectral Estimation in Induction Machine Drives," in *IEEE Transactions on Power Electronics*, vol. 11, no. 1, pp. 66–73, 1996.
- [12] P. Vas, *Artificial-Intelligence-Based Electrical Machines and Drives: Application of Fuzzy, Neural, Fuzzy-Neural, and Genetic Algorithm Based Techniques*, Oxford University Press, Oxford, 1999.
- [13] A. G. Parlos, S. K. Menon, and A. F. Atiya, "Adaptive State Filtering in Complex Systems using Recurrent Neural Networks," submitted to *IEEE Transactions on Neural Networks*, 2000.
- [14] Anonymous, "What is an Artificial Neural Network?," in <http://www.emsl.pnl.gov:2080/proj/neuron/neural/what.html>, 1997; Date of Access: August 17, 2001.

VITA

Anis Mohamed Abdul was born in Madurai, Tamilnadu, India. He received his B.S. degree in mechanical engineering from College of Engineering, Guindy, Anna University, in 1997. He worked as an Assistant Systems Analyst at Tata Consultancy Services, Madras, India from August 1997 to July 1999. In August 1999, he joined the master's degree program in Mechanical Engineering at Texas A&M University.

The typist for this thesis was Anis Mohamed Abdul.