INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA

UMI®
800-521-0600

APPLICATION OF COMPUTATIONAL INTELLIGENCE TO HIGH PERFORMANCE ELECTRIC DRIVES

by

Amol S. Kulkarni

A dissertation submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

University of Washington

1999

Program Authorized to Offer Degree: Electrical Engineering

UMI Number: 9952851

UMI Microform 9952851

Copyright 2000 by Bell & Howell Information and Learning Company.
All rights reserved. This microform edition is protected against unauthorized copying under Title 17, United States Code.

Bell & Howell Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

Doctoral Dissertation

In presenting this thesis in partial fulfillment of the requirements for the Doctoral degree at the University of Washington, I agree that the Library shall make its copies freely available for inspection. I further agree that extensive copying of the dissertation is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. Copyright Law. Requests for copying or reproduction of this dissertation may be referred to Bell and Howell Information and Learning, 300 North Zeeb Road, P.O. Box 1346, Ann Arbor, MI 48106-1346, to whom the author has granted "the right to reproduce and sell (a) copies of the manuscript in microform and/or (b) printed copies of the manuscript made from microform."

Signature ASKulkarni

Date November 24, 1999

University of Washington
Graduate School

This is to certify that I have examined this copy of a doctoral dissertation by

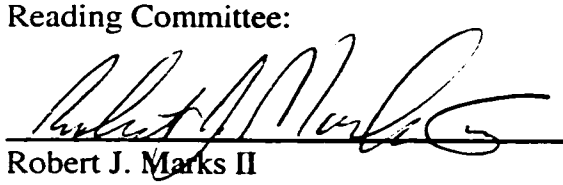
Amol S. Kulkarni

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Chair of Supervisory Committee:

Mohammed A. El-Sharkawi

Reading Committee:



Robert J. Marks II

Juris Vagners

Date: November 23, 1999

University of Washington

Abstract

APPLICATION OF COMPUTATIONAL INTELLIGENCE TO HIGH PERFORMANCE ELECTRIC DRIVES

by Amol S. Kulkarni

Chairperson of the Supervisory Committee

Professor Mohammed A. El-Sharkawi
Department of Electrical Engineering

This dissertation investigates the application of computational intelligence techniques to two open problems in high performance electromechanical drives. The objective is to develop non-linear state estimation methods and control systems for the problems under investigation.

The performance of speed sensorless control of induction motor drives currently suffers due to the absence of an accurate non-linear state estimator. An ideal estimator has to be robust to large motor parameters variations and should be able to account for non-linearities. Neural networks offer many advantages over mathematical models including the ability to learn from training examples and approximate any complex non-linear function. A new speed estimator for induction motor drives is presented here along with simulation results. Extended Kalman filters are widely used as non-linear state estimators when the model for a non-linear plant is known. A new adaptive, non-linear

state estimator is developed in this work which combines neural networks with an Extended Kalman filter. Simulation results for this adaptive estimator are presented. The developed state estimator does not specifically depend on any properties of induction motors and can be used with a class of non-linear systems.

The second problem investigated here is Precision Position Control using Elastic Links. At present, rigid transmission elements such as lead screws are used in applications requiring a positioning accuracy better than 50 microns. Elastic transmission elements such as belt drives can be used in such applications if the control system is improved to obtain better accuracy. The use of belt drives makes the plant non-linear and reduces the accuracy that can be obtained using linear control systems. A non-linear controller is required to obtain high accuracy control using such elements. In this work, a combination of non-linear feedback control using fuzzy logic and non-linear feed-forward control using evolutionary algorithm based system identification is developed. Experimental results show the efficacy of the control system in obtaining a positioning accuracy of 0.5 microns. The control system developed can be applied to other elastic systems without significant design changes. A self-tuning algorithm using evolutionary programming is developed which gives better results than a manually tuned controller.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to express my gratitude and deepest appreciation to Professor Mohamed A. El-Sharkawi for his support and guidance throughout my studies. His patience and understanding have helped me tremendously during this work. By giving me the freedom to decide the direction of my research while at the same time giving the required constructive suggestions, you have made this a very enjoyable experience for me. This thesis would not have been possible without your support and direction.

I would also like to offer a special thanks to Professor Robert J. Marks II for the encouragement, discussions and ideas that he shared with me during my studies. My friends and colleagues, Dr. N. Ravisekhar Raju and Dr. Craig Jensen helped make this period enjoyable. I am grateful for their advice and comments during discussions.

I thank the members of my supervisory committee including Professors Juris Vagners, Peter Lauritzen and Tony Woo for their review and helpful criticism of this thesis. I am also grateful for the support extended by David Skurnik, Kim Atherton and Gary Olson of Micro Encoder, Inc. as well as Keith Ritala of Washington Technology Center.

Finally, I would like to thank Pratibha for her understanding and patience during the last three years, Rahul for cheering me up and always keeping everyone happy with his energy, Prashant for his constant encouragement and Kaka and Mavshi for their support without which this thesis would not have been possible.

Chapter 1 Introduction

Electromechanical drives are used for the overwhelming majority of industrial actuation applications. Most such applications require the control of the position, speed or torque of the electric motor or the mechanical actuator. The combination of an electric motor, a power electronic controller, a mechanical transmission device and a feedback control system is commonly known as a servomechanism. Motion control can be defined as the application of high-performance servo drives to the control of torque, speed and/or position [1]. Major advances have been made in the development of high performance servomechanisms. However, the constant need for improved performance, along with a reduction in system and operational/control costs, results in many open problems which are the subject of active research. This dissertation investigates the application of computational intelligence methods to high performance electric drives by focussing on two key industrial problems: speed sensorless control of induction motor drives and precision position control of elastic actuators.

1.1 Induction Motor Drives

Perhaps the greatest progress in recent years among electric motor control systems is in the development of induction motor control. This can be attributed mainly to the invention of field-oriented control by Blaschke in 1972 [2], the rapid development of digital and power electronics in the last two decades and the development of direct torque control by Depenbrock [3] and Takahashi and Noguchi [4] in 1985. The induction motor is a very robust device suitable for use in harsh environments. To obtain precise speed control, a speed sensor, typically an optical shaft encoder, is normally used in induction motor drives. The presence of such a sensor brings several disadvantages from the standpoint of robustness, drive cost, reliability, machine size and noise immunity [5]. Elimination of the speed sensor or development of Speed Sensorless Control of Induction Motor Drives (SSCIMD) is thus an area of active academic and industrial research [6].

The induction motor is fundamentally a non-linear plant since the torque produced is a non-linear function of components of the rotor and stator currents. The values of these current components are also determined as non-linear functions of the rotor angle. Any significant error in the measurement or estimation of the motor speed drastically affects the response and consequently the speed control performance of any torque control system. A variety of non-linear state estimation methods have been proposed and used for SSCIMD in recent years. However, all the proposed solutions suffer from some limitations, leaving the problem of finding a satisfactory solution quite open. This is the first topic investigated in this work.

1.2 Precision Position Control using Elastic Transmissions

Many electromechanical servos require linear translation of load which requires the use of some transmission method to convert the rotary motion of the motor to linear motion. High precision positioning devices are widely used in a variety of applications such as machine tools, robotics, semiconductor inspection, printed circuit board populating, optical characterization, hard disc drives and bio-medical engineering. The positioning accuracy required for these applications varies from ± 100 μm to ± 0.1 μm. Rigid transmission elements such as acme lead screw or ball screw are commonly used for such applications. These components are manufactured to a high degree of accuracy giving highly repeatable and linear dynamic characteristics. However, this is an expensive option that is only suitable for short length actuation. An alternative is the use of elastic or flexible transmission elements, which are easier to manufacture at a much lower cost. The trade-off is the increase in the complexity of the plant due to the non-linearities associated with the elastic transmission. This reduces the accuracy that can be obtained using conventional linear control systems. Examples of elastic transmissions are timing belt or chain drives and plastic lead screws.

Most precision position control applications require trajectory tracking given a position track over time in addition to the final positioning. The range of motion required also

varies by application. Table 1 [7] shows the positioning accuracy and the range of travel required by different applications.

In addition to the reduced cost, elastic transmission systems also have two other advantages: a much longer travel range (more than 1 m) than rigid transmission systems and the ability to run at very high speeds (up to 2 m/s) which is desirable in many applications where force control is not required. On the other hand, lead screws are

Table 1 Required accuracy and range for precision positioning applications

Application	Positioning Accuracy	Travel Range
Optical characterization	1 – 10 μm	up to 100 mm
Hard disc drives	0.1 – 1 μm	up to 200 mm
Semiconductors	0.1 – 10 μm	up to 250 mm
Vision Systems	10 – 100 μm	more than 1 m
Bioengineering	1 – 10 μm	100 – 500 mm

unsuitable for high speed, long range applications due to buckling and due to their tendency to whip when rotated too fast [8]. It is, hence, desirable to develop control systems, so as to obtain high precision positioning using elastic drives with performance comparable to that obtained using rigid drives. This is the second topic investigated in this dissertation.

1.3 Computational Intelligence

The term computational intelligence (CI) generally refers to a group of techniques that use numerical data and computation to try to work like human intelligence. This is in contrast to artificial intelligence (AI) which refers to systems that operate on heuristically constructed rules based on human knowledge [9]. Examples of computational intelligence techniques include fuzzy logic [10, 11], artificial neural networks [12, 17], evolutionary computation and genetic algorithms [13-15], while artificial intelligence is generally concerned with expert systems. However, the boundary between AI and CI is not rigid, especially with respect to fuzzy logic which allows the use of expert knowledge

along with numerical data. This dissertation concentrates on the application of CI techniques including fuzzy logic to the problems described above.

Fuzzy logic is a means of representing systems and reason, using terms used in human language such as *low* or *very high*, in mathematical terms. Fuzzy logic is most widely used in control systems, the basic unit of which is a fuzzy inference engine. Fuzzy logic controllers have many advantages over conventional linear and non-linear controllers including:

- the ease with which expert knowledge can be incorporated in the rule base,
- they can be applied to plants which are difficult to model and whose dynamics change with time,
- they are more robust than model based controllers
- multiple objectives can be easily incorporated,
- they can be trained on available input-output data.

Fuzzy logic is thus a very powerful technique since it can combine available expert knowledge with numerical input-output data. In this sense, it combines the characteristics of CI and AI.

Neural networks have the ability to map non-linear, multiple-input multiple-output systems to any degree of accuracy and thus are referred to as universal approximators. Their main advantage is that they can be trained using numerical input-output data, which obviates the need for a physical model of any system. Although neural networks generally require a long time to train, once the training is completed, the mapped function is stored in the network and can be recalled very quickly. Another advantage is their excellent generalization capability which reduces the amount of training data required. The main disadvantage is the deterioration in performance with an increase in input or output dimensions, which is referred to as the "curse of dimensionality" [17].

Evolutionary computation uses the Darwinian theory of evolution to create a set of techniques that are most effectively used for parameter estimation of non-linear systems

[16]. These algorithms use multiple trial solutions, called chromosomes, which are refined iteratively by the application of special operations called mutation and crossover, to reach a goal defined by means of a cost function. Evolutionary algorithms can be used in complex problems where traditional search techniques cannot be used. Examples are discontinuous, non-linear systems where optimization methods dependent on gradient information cannot be applied.

1.4 Outline of the Dissertation

The main research aim of this dissertation is to solve two practical, open problems in electromechanical servo drives by applying computational intelligence techniques. Speed sensorless induction motor drives need an accurate non-linear state observer/estimator, whose performance is robust to motor parameter variations and which can consider effects such as magnetic saturation. Two new solutions are proposed here for this problem. The first is a simple, open loop speed estimator using an artificial neural network. The second is a new, non-linear state estimator which uses an artificial neural network based model of the plant as part of a traditional extended Kalman filter. The neural network weights are added to the model as augmented states to define an adaptive, non-linear state estimator.

The second problem focuses on using elastic transmission elements such as belt drives for high precision positioning applications requiring high accuracy and repeatability. The elasticity of the belt makes this plant highly non-linear and difficult to model accurately, thus making it suitable for application of fuzzy control. A fuzzy logic controller is developed using a combination of feedback and feedforward control. The developed control system is tested experimentally to verify its performance on a prototype system. The proposed methods are designed so that they are applicable to a class of non-linear systems similar to the problems being studied, thus increasing their application domain.

A self-tuning strategy is then developed for the controller, based on evolutionary programming.

The dissertation is organized as follows. Chapters 2 through 5 describe the work done with regard to speed sensorless control of induction motor drives, while Chapter 6 through 9 are concerned with precision position control of elastic drives. Conclusions are drawn in Chapter 10.

Induction motor control is introduced in Chapter 2 with emphasis on speed sensorless control. The two main types of torque control methods are outlined and the challenges in developing high performance speed sensorless drives are detailed. The desired characteristics of an ideal state estimator are also listed. The state of the art of speed sensorless control is reviewed in Chapter 3. Details of different methods proposed in the literature are given along with an analysis of the advantage and disadvantage of each method.

Chapter 4 describes the development of a simple, open loop speed estimator using neural networks. The mathematical basis for such an approach is outlined followed by details of the estimator including data generation and neural network training. Simulation results are given for this estimator and the advantages and shortcomings of such an approach are discussed. This is followed by the description of an adaptive state estimator using neural networks in Chapter 5. Detailed mathematical derivations are given for the developed algorithm used to simultaneously train the neural network and obtain motor state estimates. Simulation results obtained with the developed state estimator are presented including performance of the estimator in presence of parameter variations and magnetic saturation. Practical aspects in the use of the proposed estimator are outlined along with a discussion of its advantages and limitations. It is shown that the method is quite general and can be applied to a class of non-linear systems.

A review of different transmission elements for converting rotary motion to linear motion is presented in Chapter 6 including the characteristics of each method. Rigid and elastic elements are compared and the state of the art of precision position control is reviewed. Challenges present in the control of elastic transmission elements are identified. Chapter 7 describes two prototype positioning stages used as test setups for developing the control algorithms. A non-linear, state space model of the system is developed and the control problem is analyzed in detail, expanding on the challenges identified in Chapter 6. Based on the problem analysis, different control strategies are reviewed and the need for further development is identified.

Chapter 8 describes the design of a fuzzy feedback controller and the results obtained for the prototype stage. The need for a feedforward controller is identified and two design methods are proposed: one based on linear system identification and the other using evolutionary programming. Results with both the methods are presented and compared. Chapter 9 describes the self-tuning algorithm and the underlying assumptions, starting with the fundamentals of acceleration evolutionary programming. The modifications required for using the algorithm in an experimental application are detailed. Experimental results are presented to show the self-tuning and improvement in performance over a manually tuned controller.

Conclusions and ideas for future work are presented in Chapter 10.

Chapter 2 Induction Motor Drives

The induction motor is the workhorse of the industry accounting for more than 75% of electric drive applications [1]. The reasons for this dominance are the simple construction of the motor, resulting in a low manufacturing cost, high reliability and ruggedness and low maintenance requirement as compared to other motors. Unfortunately, the induction motor runs at a relatively fixed speed when fed from a Constant-Voltage, Constant-Frequency (CVCF) AC supply. With the availability of high power semiconductors, it has become possible to economically convert a three-phase CVCF alternating supply to a three-phase Variable-Frequency, Variable-Voltage (VVVF) supply enabling efficient wide range speed control of AC motors. This has allowed the use of induction motor drives in applications where speed control is required.

2.1 Induction Motor Characteristics

The induction motor consists of a stator on which a symmetrical 3-phase AC winding is wound. The majority of induction motors are squirrel cage type, which uses slanted bars in the rotor that are short-circuited. In this case there are only 3 external terminals. When a symmetrical three-phase AC voltage is applied to the stator windings of the motor, a sinusoidally distributed rotating magnetic field is produced in the air gap. This magnetic field rotates at the synchronous speed corresponding to the frequency of the applied voltage. The synchronous speed is given by:

$$N_s = \frac{120f}{P} \qquad (2.1)$$

where, N_s is in revolutions per minute, f is the frequency of the applied voltage in Hz and P is the number of poles. This rotating magnetic field induces an electromotive force (emf) in the rotor winding which in turn causes a circulating current in the short circuited rotor. The interaction of the rotor current with the air gap flux produces torque which by Lenz's law causes the rotor to rotate in the direction of the rotating flux. The difference

between the synchronous speed and the rotor speed is called slip speed, which depends on the load torque produced. Typical slip speed at rated torque is of the order of 2 to 8% of the synchronous speed, which implies that the induction motor is essentially a constant speed motor. The frequency of the induced rotor current is called the slip frequency and is given by the product of the percentage slip and the stator supply frequency. The induced rotor current also produces a rotating magnetic flux whose relative speed with respect to the *rotor* is equal to the slip speed. Thus, the flux rotates at synchronous speed with respect to the *stator* or at the same speed as the flux due to stator current. The interaction between these two fluxes produces the motor's torque.

Figure 2.1 shows the steady-state torque speed characteristics of induction motors. The area near the synchronous speed is the stable operating region since it is characterized by a decrease in speed as the torque increases. Typically the rated motor torque is of the order of one-half of the maximum torque.

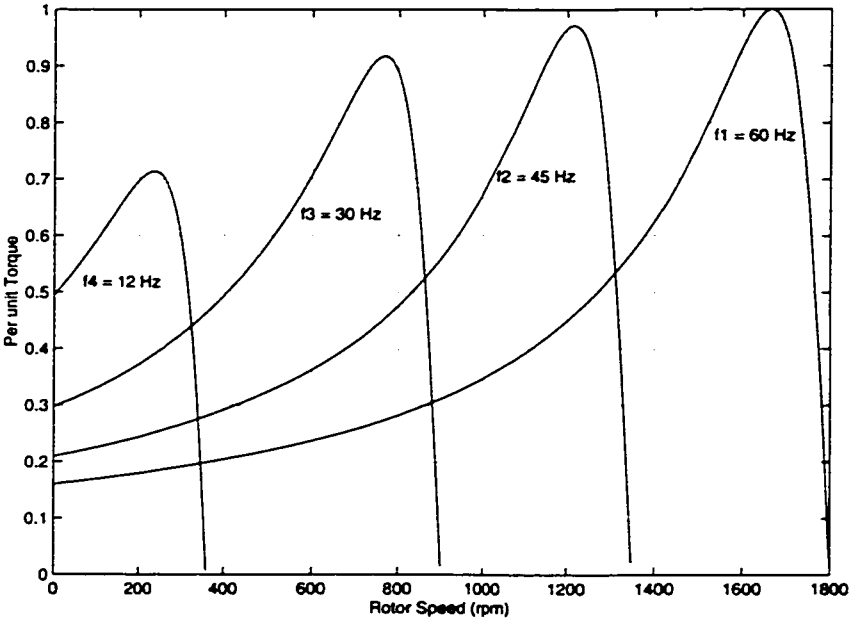


Figure 2.1 Induction motor steady state torque speed characteristics

The flux produced in the motor is proportional to the voltage applied to the stator minus the drop across the stator resistance, divided by the frequency of the stator voltage. To avoid magnetic saturation, the stator voltage to amplitude ratio is kept constant for variable frequency operation of the motor. The figure also shows the variation in the motor characteristics as the frequency of the applied stator voltage is changed. It can be seen that the synchronous speed of the motor is proportional to the frequency. For low speed operation, the stator resistance drop is high, so that a constant stator voltage results in a lower flux producing voltage. This cause a reduction in the flux and results in a lower maximum torque. Hence, to keep the air gap flux constant at low frequencies, the voltage amplitude is increased over the nominal V/f ratio. This is known as a V/f boost.

Variable speed operation can be obtained with the use of a VVVF voltage source, typically a voltage source inverter fed from a DC supply or a rectified AC supply. Using a VVVF supply, a constant torque load can be operated at any speed up to the rated synchronous speed of the motor. The stator voltage frequency can also be higher than the rated frequency, in which case, the motor runs at a speed greater than its rated synchronous speed. However, due to insulation considerations, the voltage amplitude cannot be increased (as the frequency is increased). This reduces the flux and hence the torque produced for speeds above the rated speed. This is called as the flux weakening or constant power region of operation, while operation below and up to the rated frequency is referred to as the constant torque region of operation.

2.2 Motor Model

Formulation of an induction motor model depends on the frame of the reference used. Two choices are a fixed frame of reference tied to the stator and a synchronously rotating frame of reference tied to the air gap flux. Figure 2.2 shows the stationary frame denoted by the direct (d) and quadrature (q) axes and the synchronously rotating frame denoted by axes x and y, in addition to a frame tied to the predicted position of the air gap flux. In

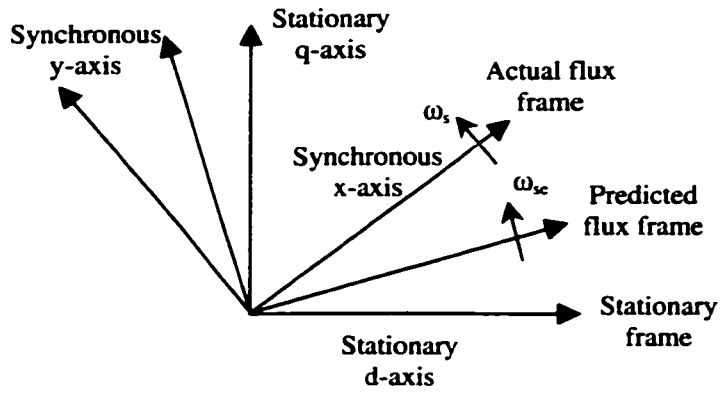


Figure 2.2 Reference frames used in induction motor control

the stationary d-q reference frame, the induction motor model in state space form is given by equations 2.2 through 2.7. This model uses stator currents i_{ds}, i_{qs} and rotor fluxes λ_{dr}, λ_{qr} as states and stator voltages v_{ds}, v_{qs} as inputs [18]. All quantities are resolved along the stationary d and q axes.

$$\dot{X}(t) = A(t) \cdot X(t) + B \cdot u(t) \tag{2.2}$$

where,

$$X(t) = \begin{bmatrix} i_{ds} & i_{qs} & \lambda_{dr} & \lambda_{qr} \end{bmatrix}^T \tag{2.3}$$

$$u(t) = \begin{bmatrix} v_{ds} & v_{qs} \end{bmatrix}^T \tag{2.4}$$

$$A(t) = \begin{bmatrix} a+b\rho & 0 & c\rho & c\omega_r(t) \\ 0 & a+b\rho & -c\omega_r(t) & c\rho \\ L_m\rho & 0 & -\rho & -\omega_r(t) \\ 0 & L_m\rho & \omega_r(t) & -\rho \end{bmatrix} \tag{2.5}$$

$$B(t) = \begin{bmatrix} \frac{1}{\sigma L_s} & 0 & 0 & 0 \\ 0 & \frac{1}{\sigma L_s} & 0 & 0 \end{bmatrix}^T \tag{2.6}$$

$$a = \frac{-R_s}{\sigma L_s}, \quad b = \frac{-L_m^2}{\sigma L_s L_r}, \quad c = \frac{L_m}{\sigma L_s L_r}, \quad \rho = \frac{R_r}{L_r}, \quad \sigma = 1 - \frac{L_m^2}{L_s L_r} \tag{2.7}$$

In the above equations, subscript *s* denotes stator quantities and *r* denotes rotor quantities. L_m is the mutual inductance, R is used for the resistance, L is used for the inductance of a particular winding using the proper subscript, and ω_r is the rotor speed. All the rotor quantities are *referred* to the stator which means they are scaled by the appropriate, equivalent turns ratio between the stator and rotor windings. The torque developed by the motor is given by:

$$T_e = \left(\frac{3}{2}\right)\left(\frac{P}{2}\right)\left(\frac{L_m}{L_r}\right)\left(i_{qs}\lambda_{dr} - i_{ds}\lambda_{qr}\right) \tag{2.8}$$

The torque equation as well as the state space dynamics show the non-linear nature of the plant. The developed torque is the plant output and is generally the controlled variable. The torque also controls the rotor speed, which appears in the state model equations.

2.3 Torque Control Methods

In a speed control application, it is desirable that the motor torque adapt quickly to any changes in load or to any disturbance. The desired torque response bandwidth for servo applications is typically of the order of 2 KHz. To obtain such wide bandwidth torque control of the induction motor, it is necessary that the air gap flux linkage space phasor or the rotor flux linkage space phasor be kept constant. This is required since the airgap flux changes slowly due to the large mutual inductance and the resulting large time constant of the magnetic circuit. However, it can be seen from the model equations that both the stator current components along the stationary d-q axes contribute to the rate of change of the rotor flux components. This can be contrasted with the DC motor where the flux and torque producing current components are decoupled and can be controlled independently. The two main torque control methods for induction motor drives aim at achieving this kind of decoupling between two derived stator current components, one which affects the rate of change of the rotor flux and another which changes the torque.

2.3.1 Field Oriented Control

Field-oriented control was invented by Blaschke in 1972 [2] and was refined for practical application by Leonhard in 1979 [19]. It uses a rotating frame of reference, where the x-axis is aligned with the rotor flux. This is shown as the synchronous x-y axes in Figure 2.2. Due to the choice of the axes, the rotor flux only has an x-axis component, so that λ_{xr} is the rotor flux and $\lambda_{yr} = 0$. Then, the developed torque is given by:

$$T_e = \left(\frac{3}{2}\right)\left(\frac{P}{2}\right)\left(\frac{L_m}{L_r}\right)\left(i_{ys}\lambda_r\right) \tag{2.9}$$

where i_{ys} is the torque producing component of the stator current and λ_r is the rotor flux which is aligned along the x-axis. The modified state model has only three states since $\lambda_{yr} = 0$ and is given by:

$$\dot{X}(t) = A(t)\cdot X(t) + B\cdot u(t) \tag{2.10}$$

where,

$$X(t) = \begin{bmatrix} i_{xs} & i_{ys} & \lambda_r \end{bmatrix}^T \tag{2.11}$$

$$u(t) = \begin{bmatrix} v_{xs} & v_{ys} \end{bmatrix}^T \tag{2.12}$$

$$A(t) = \begin{bmatrix} a+b\rho & \omega_s(t) & c\rho \\ -\omega_s(t) & a+b\rho & -c\omega_r(t) \\ L_m\rho & 0 & -\rho \end{bmatrix} \tag{2.13}$$

$$B(t) = \begin{bmatrix} \frac{1}{\sigma L_s} & 0 & 0 \\ 0 & \frac{1}{\sigma L_s} & 0 \end{bmatrix}^T \tag{2.14}$$

where, ω_s is the speed of the x-y reference frame with respect to the stator and the parameters are given by equation 2.7. When these equations are simplified, we obtain:

$$i_{ys} = \frac{(\omega_s - \omega_r)\lambda_r}{L_m\rho} \tag{2.15}$$

and,

$$\dot{\lambda}_r = L_m\rho\, i_{xs} - \rho\lambda_r \tag{2.16}$$

Equation 2.16 shows that the rotor flux is controlled only by the x-axis component of the stator current, while equation 2.15 shows that the torque is controlled by the y-axis

component of the stator current. Thus, if the position of the rotor flux space phasor can be obtained at every instant, the x-axis component of the stator current can be kept constant while the y-axis component can be changed based on the desired torque. This is the basis of field-oriented control.

Field-oriented control is implemented using a Current Regulated, Pulse Width Modulated (CR-PWM) voltage source inverter. Different current control methods are used which can be primarily classified as hysteresis current control methods and PWM methods. Field-oriented control depends strongly on the ability of the control system to accurately estimate the instantaneous flux position. It is extremely sensitive to variations in motor parameters, especially the rotor resistance at moderate to high speeds and the stator resistance and leakage inductance at low speeds [20]. Most practical implementations use separate algorithms for independently calculating the motor parameters so as to retain proper field-orientation [21].

2.3.2 **Direct Torque Control**

Direct Torque Control (DTC) developed independently by Depenbrock [3] and Takashi [4] also aims at keeping the air gap flux constant while controlling the motor torque. The main difference with respect to field-oriented control is that this method does not translate the desired torque and flux to desired values of stator current components. It uses flux and torque estimators/observers and based on the estimated flux and torque error, *directly* chooses the inverter switching states for the next time step of a discrete time controller, so as to reduce these errors. The basis for this is the fact that an inverter is a finite state machine and has a fixed number of possible switching states. In a three-phase inverter, there are six semiconductor switches and each phase can either be connected to the positive DC voltage or to the negative DC voltage at any given instant. This gives rise to eight possible switch combinations for the inverter and DTC selects the switch position for the next switching cycle based on the torque error. Any intermediate steps such as current control finally have to use one of the 8 inverter switching states to

achieve the desired effect. This method thus simplifies the torque controller to a great extent. It can also be applied to high power inverters which use Gate Turn-off Thyristors (GTO's) capable of switching at low frequencies up to 5 KHz.

The DTC technique has been refined into an industry leading product by ABB, Sweden [22]. DTC deals with the stationary or stator frame of reference and hence it is not as sensitive to parameter variations as field-oriented control. However, DTC also needs a detailed induction motor model that must be identified on-line to obtain all the states and information about the flux and torque.

Both these techniques give good torque control performance while compensating for the

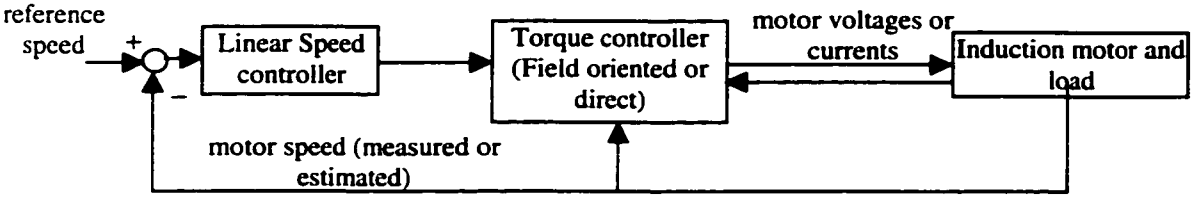


Figure 2.3 Block diagram for speed control using an inner torque control loop

non-linearities of the induction motor. Speed control is then accomplished by using a linear speed control loop which is usually a PID controller as shown in Figure 2.3. The torque controller generally requires the feedback of motor voltages and currents, which is shown as an arrow from the induction motor block to the torque controller block in the figure, as well as an estimate of motor speed.

2.4 Speed Sensorless Control

Development of the two torque control methods described in the previous section has helped induction motor drive technology reach a status of maturity with most drive manufacturers using these methods to deliver accurate servo systems. Both field-oriented control and direct torque control need information about the motor (or rotor) speed in order to perform torque control. The use of a speed sensor, such as a quadrature encoder,

considerably degrades the ruggedness of the motor-drive combination. Hence, development of Speed Sensorless Induction Motor (SSIM) control has been the topic of continuing research [6].

SSIM is often referred in technical literature simply as *sensorless control*, which is misleading since all SSIM control methods use sensors to measure other terms such as the motor voltage and current. These sensors can however be located on the three-phase inverter which drives the motor. In hazardous environments, which preclude the use of a speed sensor, the inverter is often located away from the motor. Thus, the use of current or voltage measurement sensors does not directly affect the ruggedness of the motor drive. The advantages of SSIM are:

- reduced cost,
- reduction in size of the drive machine, which is critical in many applications,
- increased reliability, especially in hazardous environments, and,
- elimination of the sensor cable, which improves noise immunity, since the sensor cable typically carries low current signals running close to the cable carrying the high frequency, motor currents.

2.4.1 **Challenges**

a. Model Characteristics and Parameter Variation

The induction motor model in the stationary reference frame (equations 2.2 through 2.7) is a non-linear model, where the rotor speed is a state as well as a model parameter. Thus, one of the main challenges in SSIM control is the need for a good, non-linear state estimator. In addition, the motor model includes parameters such as the rotor and stator resistance and the stator leakage inductance. These parameters vary widely over the domain of operation of the motor, as a function of motor speed, current and temperature. Specifically, in the low speed range of operation, the stator resistance and leakage reactance have higher influence over the stator voltage. Hence, any variation in these parameters has a greater effect when the motor speed is low. On the other hand, the rotor

resistance varies widely at high motor speeds and at higher values of load torque. The speed estimation method has to be able to compensate for changes in motor parameters.

b. Magnetic Effects

The motor model is well known and widely accepted as representing the actual plant. However, this model assumes that the motor is always operated in the linear region of the magnetic circuit. When the motor is controlled using a torque control method, due to flux estimation errors, the instantaneous current can be much higher than the rated current, causing localized saturation of the magnetic circuit. Inclusion of possible magnetic saturation in the plant model is not practicable due to the added complexity. The standard motor model also does not address other effects such as the use of a cross bar rotor. Normally the squirrel cage rotor winding is made of straight bars of aluminum or copper, which is then short circuited using an end ring. To reduce cogging torque, the rotor bars are skewed and such a rotor is called a cross-bar rotor. Skewing of rotor bars causes cross magnetic effects, which generally tend to reduce the rotor flux. This effectively reduces the mutual inductance of the magnetic circuit.

c. Simultaneous Flux and Speed Estimation

Both the induction motor torque control methods try to keep the airgap flux constant while changing the motor torque. This requires an estimate of the airgap or rotor flux which is generally obtained from the model equations using the measured speed. When the speed sensor is eliminated, the flux and the speed have to be estimated simultaneously. This is equivalent to simultaneous, non-linear, state and parameter estimation, since the speed can be considered as a parameter of the state model. The estimated motor speed is used directly in field-oriented control to calculate the position of the rotor flux. Any error in speed estimation results in detuning of the field-orientation, since the estimated position of the synchronously rotating frame of reference is no longer the same as the ideal frame of reference. This removes the decoupling of the flux and torque producing components, causing an increase in the torque control response time. Direct torque control is slightly more permissive in this regard, since it generally uses a

stationary frame of reference. However, it still depends on an accurate determination of speed and knowledge of motor parameters for obtaining decoupled torque control.

d. Need for fast update

Since the speed estimate is used for torque control, the estimation has to be updated at a fast rate to match the execution time of the torque control loop. For position control servos, the position control loop is generally executed with a sampling period of 1 ms (or a sampling frequency of 1 kHz). This requires a torque control bandwidth of at least 1 kHz, which in turn implies that the torque control sampling frequency has to be at least 4 kHz. Thus, any speed and flux estimation algorithm has to be executed within 250 μs along with the torque control calculations. This imposes constraints on the complexity of the estimation algorithms, although the continuing increase in microprocessor speeds and the simultaneous reduction in their cost, helps in reducing the importance of this factor.

2.4.2 Characteristics of an ideal state estimator

Based on the challenges outlined above, the desirable characteristics in an ideal state estimator for SSIM control are:

- A non-linear state estimator is required to be robust to variation in motor parameters, especially stator and rotor resistance and stator leakage reactance. Due to the wide range of variation of these parameters, an adaptive, non-linear state estimator is required.
- The estimation method should be independent of the torque control method used. State estimators dependent on a particular method are sensitive to errors in the torque control and cannot be universally applied.
- The estimator should be able to perform well in the presence of magnetic saturation and other magnetic effects.

- Since both speed and flux estimation is required, the need is for either a stand-alone speed estimator whose output can be used in a flux estimator, or a simultaneous speed and flux estimator.

Various SSIM control methods have been proposed in the literature and these are reviewed in detail in the next chapter.

Chapter 3 State-of-the-Art of Speed Sensorless Control

Researchers have proposed many methods for the implementation of speed sensorless induction motor control. This chapter presents a review of the published literature in this area. The advantages and limitations of each method are discussed considering the desired characteristics outlined in the previous chapter.

3.1 Theoretical Basis

Consider the state model of the induction motor as defined by equations 2.2 through 2.7, where the states are the two stator current components and the two rotor flux components in the stationary frame of reference. If the stator voltages and currents are known, the rotor flux components, λ_{dr} and λ_{qr} can be eliminated from the state space model, giving two equations for the rotor speed, ω_r, 3.1and 3.2.

$$\omega_r = \frac{\alpha \frac{di_{ds}}{dt} - L_r e_{ds} - R_r \int e_{ds} + R_r L_s i_{ds}}{L_r \int e_{qs} - \alpha i_{qs}} \tag{3.1}$$

$$\omega_r = \frac{-\alpha \frac{di_{qs}}{dt} + L_r e_{qs} + R_r \int e_{qs} - R_r L_s i_{qs}}{L_r \int e_{ds} - \alpha i_{ds}} \tag{3.2}$$

where, $$\alpha = \sigma L_s L_r,\ e_{ds} = v_{ds} - R_s i_{ds},\ e_{qs} = v_{qs} - R_s i_{qs} \tag{3.3}$$

Thus, if all machine parameters are known and the stationary d-q axis stator voltages and currents are measured, the speed can be estimated. This is possible even if one motor parameter is not known or is varying, since two equations are available to calculate the motor speed.

3.1.1 **Observability**

From the motor's state model, it can be seen that the rate of change of the stator currents is dependent on the rotor flux components, which appear as back emf's in the stator current equations 2.2 through 2.7. The same axis back emf component is due to the difference in the speed of the reference frame and that of the rotor, while the cross-axis back emf component is due to the rate of change of flux coupled through the mutual inductance [23]. This cross-coupling between the stator and the rotor disappears when DC voltages are applied to the stator. This is due to the fact that the two cross-coupling signals are equal and opposite, and hence are canceled. Thus, when DC voltages are applied to the stator, the rotor does not influence the stator quantities, even if the rotor is rotating. This renders the rotor flux and speed unobservable when the stator supply frequency is zero. It should be noted that when the motor is at standstill (zero speed) while producing torque, the stator voltage frequency is *not equal to zero*, but is equal to the slip frequency. The limitation on observability implies that the stator frequency must be raised to a certain minimum value for proper estimation of the rotor quantities. This is a basic limitation of speed sensorless control

The speed sensorless control methods proposed by researchers can be broadly classified into two categories:

- methods that use a particular physical characteristic of the motor such as rotor slot harmonics or those that need mechanical modifications to the motor, and,
- methods that are independent of the motor mechanical structure and can be applied to any induction motor.

The first group of methods has a limited application since any change in the design and manufacturing of the motor is undesirable. Hence, these methods are not considered further in this dissertation. The second group of methods can be further subdivided into three classes:

- open loop observers which assume either rotor or stator flux oriented torque control,

- model reference based methods, which use two methods to calculate the rotor flux components, with one of the methods being independent of rotor speed and the other method dependent on the speed. The error between the flux calculations from the two methods is used to adapt the estimate of the speed, which is a parameter of the adaptable model, and,
- full non-linear observers using techniques such as sliding mode control, extended Kalman filters, extended Luenberger observer and Lyapunov synthesis.

The following sections discuss methods falling under these classes in detail.

3.2 Open Loop Observers

Open loop observers presented in the literature depend on the use of field-oriented torque control. Ohtani et al describe a sensorless rotor flux orientation scheme based on the stator model [24]. Figure 3.1 shows the block diagram of the scheme where the rotor speed is estimated as a proportional-integrator function of the error between the desired

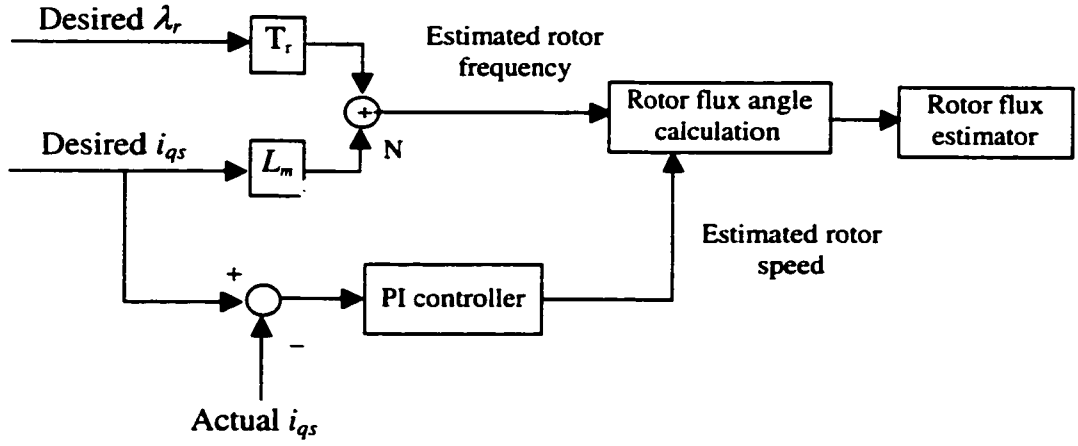


Figure 3.1 Block diagram of Open loop observer using rotor flux orientation

and actual stator current along the q-axis of the *synchronous frame of reference tied to the rotor flux*. The rotor frequency, which is also called the slip frequency, is calculated as a ratio of the desired rotor flux and q-axis stator current. The combination of these two estimates gives the position of the rotor flux, which is in turn used to estimate the rotor

flux. Desired quantities can be used to obtain the speed estimate since the method assumes that the controller is able to accurately predict the rotor flux position. This assumption, and the absence of any correction to the estimate calculations, makes this open loop scheme highly susceptible to parameter variations. The estimation has to be turned off at low speeds where the open integration used for obtaining the rotor flux position does not work. Thus, this method can only be used for moderately high speeds. Xu et al have proposed a similar scheme which assumes and implements stator flux orientation instead of rotor flux orientation [25]. That method is more susceptible to stator resistance changes due to the stator flux orientation.

3.2.1 **Neural network based method**

Based on equations 3.1 and 3.2 which relate the motor speed to stator voltages and currents and their derivatives and integrals, Mehrotra et al [26] use two neural networks to approximate the numerator and denominator of either of the two equations 3.1 and 3.2. The inputs to the network are motor voltages and currents and their integrals and derivatives. The speed is obtained by the ratio of the output from the numerator NN to that of the denominator NN after filtering the zero crossing points (where the numerator and denominator of equation 3.1 or 3.2 are zero) as shown in Figure 3.2.

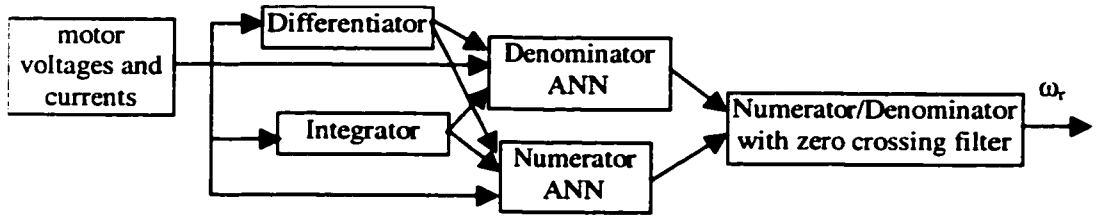


Figure 3.2 Use of two neural networks for speed estimation

This method attempts to use the excellent non-linear approximation capabilities of neural networks. However, the method needs external integration of motor voltages and currents, which drifts at low speeds. It is also very sensitive to motor parameter variations.

3.3 Model Reference Adaptive Methods

The model reference approach makes use of the redundancy of two machine models (or equations) of different structure, that estimate the same state variable using different input variables [27]. The state model of the induction motor given by equation 2.2 can be re-written as:

$$\begin{aligned} p\lambda_{dr} &= -\rho\lambda_{dr} - \omega_r\lambda_{qr} + L_m\rho i_{ds} \\ p\lambda_{qr} &= \omega_r\lambda_{dr} - \rho\lambda_{qr} + L_m\rho i_{qs} \end{aligned} \tag{3.4}$$

$$\begin{aligned} p\lambda_{dr} &= \frac{L_r}{L_m}\left[v_{ds} - (r_s + \sigma L_s p)i_{ds}\right] \\ p\lambda_{qr} &= \frac{L_r}{L_m}\left[v_{qs} - (r_s + \sigma L_s p)i_{qs}\right] \end{aligned} \tag{3.5}$$

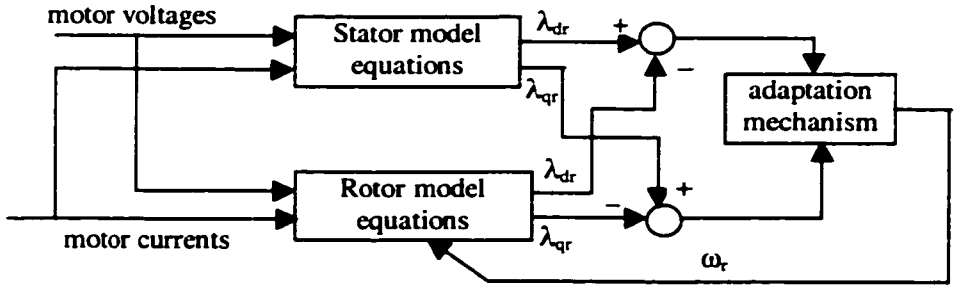


Figure 3.3 Block diagram of Model Reference Adaptive System

where, p is the differential operator. The other quantities are defined in equation 2.7. Equation 3.4 involves the rotor speed and hence is called the rotor model equation, while equation 3.5 is called the stator model equation, since it is independent of the rotor speed. These two differential equations can then be integrated to obtain estimates of the rotor flux vector. Schauder [27] was the first to propose a speed estimation scheme for induction motors using a model reference adaptive scheme shown in block diagram form in Figure 3.3, with the stator model being used as the reference model. This method uses

the error between the two flux estimates to adapt the estimated speed of the system. The adaptation method proposed was a simple PI controller. This system gives good speed estimates for speeds above 3% of the rated motor speed. However, both the stator and rotor model equations need integrators to obtain the flux estimates. Use of integrators is not possible due to problems with initial conditions, accuracy and drift. Hence, the integrators need to be replaced by low-pass filters, making the integration ineffective below the time constant of the low pass filter. Typically, a first order filter having a pole at 1 Hz is used, which renders this method unusable below that frequency. Even then, reversal of speed through zero is possible in a fast transient process. Low frequency operation for any significant length of time causes the speed estimate to go astray and the system looses its torque control capability. The method is also sensitive to variations in motor parameters, notably the stator resistance, since it directly affects the reference model (stator model) in Figure 3.3. Good speed estimation and control is obtained with this method for stator frequencies above 2 Hz.

3.3.1 **Modifications to eliminate need for open integrator**

A number of adaptations of the basic MRAS approach have been suggested which aim at removing the need for a pure integrator. Among them is a scheme proposed by Peng and Fukao [28]. This method uses an auxiliary quantity called counter electromotive force (emf), which is proportional to the time derivative of the rotor flux, instead of the actual rotor flux, in the rotor and stator model equations. The counter emf is defined as:

$$e_m = v_s - \left(R_s i_s + \sigma L_s \frac{di_s}{dt} \right) \tag{3.6}$$

Another equation for e_m, defines it in terms of the rotor speed and is given by:

$$e_m = L_m' \left(\omega_r \otimes i_m + \frac{i_s - i_m}{\tau_r} \right) \tag{3.7}$$

where,

$$i_m = i_s + \frac{L_r}{L_m}, \quad \tau_r = \frac{L_r}{R_r}, \quad L_m' = \frac{L_m^2}{L_r} \tag{3.8}$$

These two equations are used in the place of equations 3.4 and 3.5 in a structure similar to the one in Figure 3.3 to obtain the speed estimate. Robustness to stator resistance variation is achieved by using another auxiliary quantity, which is a cross product of the counter EMF, and the stator current vectors. However, simulation and experimental results given in the paper show that this method still gives poor performance at speeds below 1% of rated speed, which corresponds to a stator supply frequency of about 1.5 Hz. Another adaptation of the MRAS method is proposed by Tajima and Hori [29]. They show that the use of a PI controller in the adaptation mechanism leads to the presence of a zero and two poles and suggest a systematic method to choose the PI controller parameters so as obtain desired speed estimation dynamics. This method does not change the structure of the MRAS and hence continues to suffer from poor performance at low speed due to the presence of low pass filters that replace the integrators.

3.3.2 **Application of Neural networks**

Ben-Brahim et al [30] have proposed the use of a neural network to replace the rotor model in an MRAS system with the rotor speed as one of the *weights* in a feedforward network. The structure used is shown in Figure 3.4, where the adaptation mechanism is now replaced with back error propagation training of the feedforward neural network. The rotor model equation given by equation 3.4 is discretized using the backward difference method to obtain the following equation:

$$\vec{\lambda}_r(k) = W_1 X_1 + W_2 X_2 + W_3 X_3 \tag{3.9}$$

where,

$$W_1 = 1 - \frac{T}{T_r}, \quad W_2 = \omega_r T_r, \quad W_3 = \frac{L_m T}{T_r} \tag{3.10}$$

and X_1, X_2 and X_3 are space phasor quantities which are functions of delayed rotor flux phasor and stator current phasor. Here T_r is the rotor time constant and T is the sampling period of the estimator.

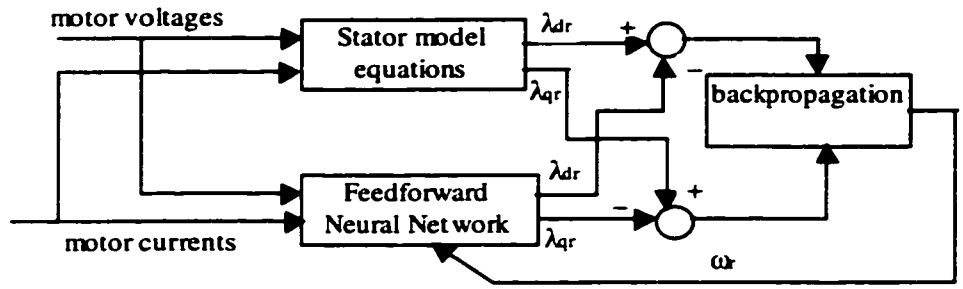


Figure 3.4 MRAS using neural network for speed estimation

A two layer neural network having X_1, X_2 and X_3 as the three inputs is used to approximate the above equation. The output of the neural network is the weighted linear sum of these three inputs, the weights being W_1, W_2 and W_3 respectively. The back-error propagation algorithm is then used to adjust the weights and after training, W_2 gives the rotor speed. It is obvious that although the authors describe this as an application of neural networks, it is basically an application of gradient descent parameter identification since all equations are linear. It is also evident from the structure of the equations that this method would be susceptible to rotor time constant variations. This is borne out by the results presented by the authors, where a 25% error in rotor resistance causes a 5% error in the speed estimate at a high speed of 175 rad/s. Results are not given at lower speeds. This method still requires an integrator or a low pass filter to solve the stator equations and hence suffers from the same limitations as other MRAS based methods.

3.4 Non-linear observers

Various standard techniques have been proposed for use in SSIM control. These include full order non-linear observers which use Lyapunov synthesis for implementation, extended Kalman filter and extended Luenberger observer, and sliding mode control. Each of these is summarized below in brief.

3.4.1 **Full order observers**

Consider the state model of the motor as given by equations 2.2 through 2.7. To implement any non-linear observer, the only error term which is readily available is the difference between the predicted and the measured stator currents. This error, multiplied by a complex speed dependent gain, is used to obtain corrective inputs to the stator and rotor model equations. Kubota et al [31] select the complex gain in such a way that the eigenvalues of the observer, $\lambda_{obs} = k\ \lambda_{machine}$, where $\lambda_{machine}$ represents the machine eigenvalues and k is a real constant greater than 1. Since the machine eigenvalues are a function of the mechanical speed, k has to be dynamically scaled so as to keep the observer dynamics faster than the machine dynamics. This scheme results in the following equations:

$$\tau_\sigma \frac{d\vec{i}_s}{dt} = -\vec{i}_s + \frac{L_m}{L_r \tau_r r_\sigma}(1 - j\omega\tau_r)\vec{\lambda}_r + \frac{1}{r_\sigma}\vec{v}_s - G_s(\hat{\omega})\left(\vec{i}_s - \hat{\vec{i}}_s\right) \quad (3.11)$$

$$\tau_r \frac{d\vec{\lambda}_r}{dt} = -\vec{\lambda}_r + j\hat{\omega}\tau\vec{\lambda}_{rr} + L_m\vec{i}_s - G_r(\hat{\omega})\left(\vec{i}_s - \hat{\vec{i}}_s\right) \quad (3.12)$$

where G_s and G_r are the gain factors which are functions of the estimated mechanical speed. The speed is estimated separately as the output of a PI controller, whose input is the cross-product of the estimated flux and the current error. The main limitation of this method is the constraint on the speed estimation dynamics imposed by the PI controller and a compounding of estimation errors due to the interdependence of the flux and speed estimators. An advantage is the relative robustness to parameter variations and Lyapunov synthesis is used for designing the gain factors so as to assure stability. The authors do not present any specific numerical data regarding the speed estimation performance obtained using this method.

3.4.2 **Observer based on Sliding Mode Control**

The effective gains of the error compensator used in the full order observer can be increased by using a sliding mode controller to tune the observer. This method is

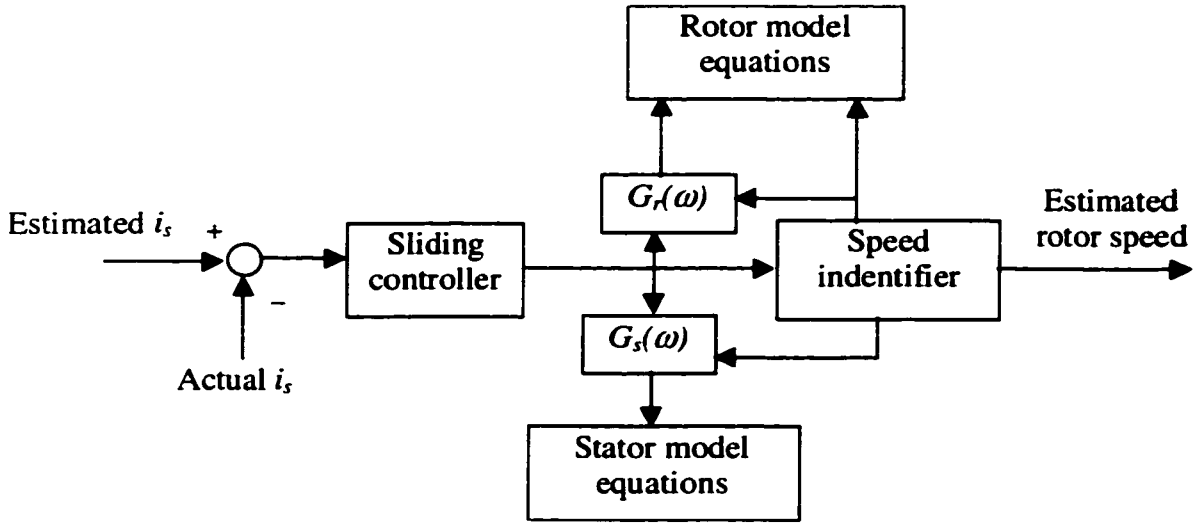


Figure 3.5 Block diagram for nonlinear observer using sliding mode control

proposed by Doki et al [32] as shown in Figure 3.5, where instead of the current error being fed to the speed dependent gains, a sliding surface is defined based on the current error. The output of the sliding controller is a switching signal which forces the current error to zero. The average value of the switched signal is used for calculating the speed. The authors use an H_∞ approach to obtain the poles of the observer in order to minimize the flux estimation error in the presence of parameter variations. Good speed control results are reported for speeds above 50 r/min.

3.4.3 **Extended Kalman filter and Luenberger observer**

Du and Brdys [33] have proposed the use of an extended Luenberger observer while Kim et al [34] use an extended Kalman filter for SSIM. Both techniques use the non-linear motor state space model which is then linearized around the operating point at every

sample. The methods use a joint state and parameter estimation approach by considering the motor speed as an additional state. This is done by augmenting the state space model with the speed. The rate of change of the motor speed is unknown and is set to zero for the purposes of the observer. This assumes that the observer sampling time is small as compared to the rate at which the speed changes. The main difference between the two approaches is that the Kalman filter uses a stochastic approach to approximate the difference between the model and the actual physics, while the Luenberger observer uses a deterministic approach.

Du et al [35] compare the two approaches in a different paper and contend that using the deterministic approach is better for induction motors, since the noise affecting the voltages and currents is actually the harmonics present due to pulse-width modulation (PWM) whose characteristics are well known. This argument fails when considering the fact that a Luenberger observer cannot approximate unmodeled dynamics that are unknown or not well defined. On the other hand, the Luenberger observer can be designed exactly by pole placement which avoids the trial and error associated with the process noise covariance matrix used in the Kalman filter design.

Both these approaches give good results with accurate speed estimation possible up to a very low motor speed. Another advantage is the simultaneous estimation of both speed and flux which avoids any cascading of estimation errors. The main disadvantage is the high parameter sensitivity which results from the fact that both these methods assume the motor model to be correct and do not have any ability to adapt for changes in motor parameters.

3.5 Summary

It can be seen from the above discussion that all the methods proposed in the literature have some advantages and limitations. The open loop observers have drift problems since there is no correction factor based on measured currents or other quantities. These are also predicated on the use of a particular torque control method. The model reference

based methods are less susceptible to motor parameter variations but need to replace an integrator with a low pass filter which places an inherent limit on the low speed performance. All non-linear observers use the error between the predicted and measured current to correct the speed and flux estimates. This category of observers is very sensitive to variations in motor parameters. One limitation common to all the methods is the inability to account for unmodeled effects such as magnetic saturation or the presence of cross-bar rotors.

In the next two chapters, two new observers are proposed for the SSIM control problem. After discussing the theory and implementation details, their performance is analyzed and compared with the methods discussed in this chapter.

Chapter 4 Open Loop Speed Estimator Using Neural Networks

The need for a better speed estimator for induction motors was outlined in the previous chapters based on the limitations of methods proposed till date. A new open loop speed estimation method based on neural networks is presented in this chapter. The chapter starts with an explanation of the theoretical basis for the proposed method. The implementation details and results obtained are then presented followed by a discussion of the advantages and limitations of this method.

Artificial Neural Networks (ANNs) constitute a powerful class of computational intelligence techniques. ANNs have been shown to be good nonlinear approximators [37]. This property can be used for duplicating the dynamic input-output relationship of nonlinear systems. A brief introduction to ANNs is given in the Appendix.

4.1 Theoretical basis for proposed method

Consider the state model of the induction motor given by equations 2.2 through 2.7 which are repeated here for convenience.

$$\dot{X}(t) = A(t) \cdot X(t) + B \cdot u(t) \tag{4.1}$$

where,

$$X(t) = \begin{bmatrix} i_{ds} & i_{qs} & \lambda_{dr} & \lambda_{qr} \end{bmatrix}^T \tag{4.2}$$

$$u(t) = \begin{bmatrix} v_{ds} & v_{qs} \end{bmatrix}^T \tag{4.3}$$

$$A(t) = \begin{bmatrix} a+b\rho & 0 & c\rho & c\omega_r(t) \\ 0 & a+b\rho & -c\omega_r(t) & c\rho \\ L_m\rho & 0 & -\rho & -\omega_r(t) \\ 0 & L_m\rho & \omega_r(t) & -\rho \end{bmatrix} \tag{4.4}$$

$$B(t)=\begin{bmatrix} \frac{1}{\sigma L_s} & 0 & 0 & 0 \\ 0 & \frac{1}{\sigma L_s} & 0 & 0 \end{bmatrix}^T \tag{4.5}$$

$$a=\frac{-R_s}{\sigma L_s},\ b=\frac{-L_m^2}{\sigma L_s L_r},\ c=\frac{L_m}{\sigma L_s L_r},\ \rho=\frac{R_r}{L_r},\ \sigma=1-\frac{L_m^2}{L_s L_r} \tag{4.6}$$

If the rotor flux quantities λ_{dr} and λ_{qr} are eliminated from the state model, equations 3.1 and 3.2 are obtained for the motor speed, which are also repeated here:

$$\omega_r=\frac{\alpha\frac{di_{ds}}{dt}-L_r e_{ds}-R_r\int e_{ds}+R_r L_s i_{ds}}{L_r\int e_{qs}-\alpha i_{qs}} \tag{4.7}$$

$$\omega_r=\frac{-\alpha\frac{di_{qs}}{dt}+L_r e_{qs}+R_r\int e_{qs}-R_r L_s i_{qs}}{L_r\int e_{ds}-\alpha i_{ds}} \tag{4.8}$$

where, $$\alpha=\sigma L_s L_r,\ e_{ds}=v_{ds}-R_s i_{ds},\ e_{qs}=v_{qs}-R_s i_{qs} \tag{4.9}$$

These equations show that given the motor model and the values for the model parameters, the motor speed can be obtained from stator voltage and current measurements. However, this estimation requires the integration of stator voltages, which is difficult and extremely parameter sensitive at low speeds. This limits the performance of many methods as seen in the previous chapter.

For an induction motor excited by a sinusoidal voltage supply, all the quantities in equations 4.7 and 4.8 are sinusoidal. It can also be shown that the numerator and denominator terms in the equations have the same phase so that the equations are indeterminate when the numerator (and denominator) goes through a zero crossing. Figure 4.1 shows a plot of the numerator and denominator of equation 4.7 for an induction motor fed from a sinusoidal supply. Although the numerator and denominator of equation 4.8 also have the same characteristic, the terms in equation 4.8 lag the terms in equation 4.7 by 90 electrical degrees. Thus, when one equation for the motor speed is close to indeterminate, the terms in the second equation are at their peak values, so that the motor speed can be obtained from *at least one of the two equations, at any given*

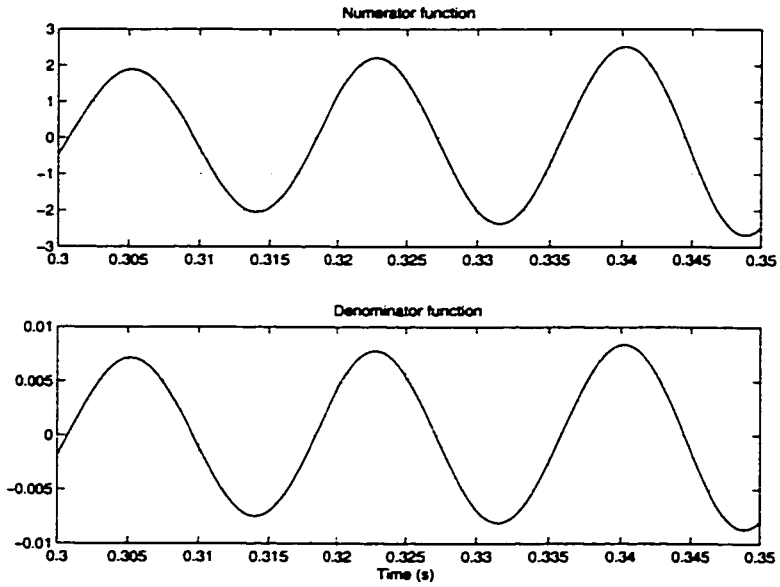


Figure 4.1 Numerator and denominator for induction motor speed equation

instant. This allows the use of a neural network to obtain the motor speed from stator voltage and current measurements, even though each equation when considered alone is *not square integrable.* If equations 4.7 and 4.8 are converted to discrete time form, the integrals and derivatives of stator voltage and current will have to be approximated by a difference equation which uses delayed sampled values of these quantities. The motor speed can then be written as a function of these delayed, sampled quantities:

$$\omega_r(k) = f\left(v_{ds}(k,\ldots), v_{qs}(k,\ldots), i_{ds}(k,\ldots), i_{qs}(k,\ldots)\right) \qquad (4.10)$$

where, f denotes the non-linear function which approximates either of the two continuous time equations and $(k,\ldots)$ denotes samples of quantities at the k^{th} sampling period and earlier samples. It should be noted that this discrete time equation includes the implicit division present in the continuous time equations. Based on the above discussion, a new speed estimator is proposed which uses an ANN. The salient points of this new speed estimator are as follows:

- An artificial neural network (ANN) is trained to estimate the motor speed given delayed stator voltage and current measurements

- The ANN is trained to approximate the physical relationship between the estimated and measured quantities instead of selecting one of the two equations.
- Although the speed equations include a division of two terms, no external division is performed to process the ANN output. The aim is that the ANN will perform the division implicitly and internally.
- The stator voltage less the resistive drop (e_{ds}, e_{qs}) are **not** explicitly integrated before input to the network. Again the aim is to train the network to approximate the integration and differentiation internally. This eliminates the fundamental limitation on low speed estimation due to the need for pure integration.

4.2 Implementation Details

To implement the speed estimator, the type and structure of the ANN have to be chosen. The data generated for training the ANN has to encompass the whole domain of operation of the motor. These and other implementation details of the proposed estimator are described below.

4.2.1 **Neural Network Structure**

Before deciding on the structure of the neural network, a choice has to be made between

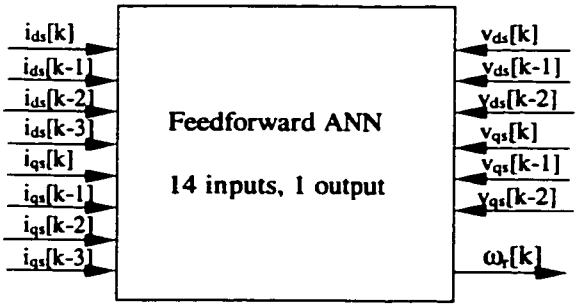


Figure 4.2 Inputs and outputs for the neural network

feedforward neural networks and recurrent networks. From equations 4.7 and 4.8 it can be seen that the speed is not dependent on any term derived from the speed, so that the discrete time equivalent equation for the speed at the present instant does not depend on any old samples of the speed. This means that the mapping to be learnt is static and a feedforward network is sufficient. The use of a recurrent network is impractical due to the associated disadvantages such as higher training time and possibility of unstable behavior.

The required inputs to the network are the stationary d and q-axis stator currents and the d and q-axis stator voltages, with sufficient number of delayed samples. The sole output from the network is the motor speed at the present instant. After some trial and error, it was concluded that 3 delayed samples of voltages and 4 samples of currents give sufficiently good results with negligible additional improvement due to addition of more delayed samples. Thus, the structure for the estimator has a feedforward NN with 14 inputs corresponding to 6 samples of voltages and 8 samples of currents. The inputs and outputs for the network are shown in a block diagram in Figure 4.2 where the index k denotes the present instant and k-i denotes a measurement delayed by i sampling intervals. It was observed by trial and error that the network gave optimum results using one hidden layer with 10 to 12 hidden neurons. Further increase in the number of hidden neurons does not improve the performance. Thus, the network has 14 inputs, 10 hidden neurons and one output and is fully interconnected between adjacent layers. Since the motor voltages, currents and speed are all bi-directional quantities going from positive rated to negative rated values, the *tan-sigmoid* (hyperbolic tan) function is used as the activation function for the hidden and output neurons.

4.2.2 **Data Normalization**

The data for training and testing the neural network is generated using simulation of the motor model in MATLAB and Simulink [86]. Since the output of a *tanh* function is limited to [-1.1], the data obtained from simulation also needs to be normalized. To

Table 2 Base values for normalization of motor quantities

Quantity	**Normalization base**
Stator voltages	Rated peak phase voltage
Stator currents	2*Rated peak line current
Speed	Rated speed

avoid saturation of the NN outputs, the data is actually normalized to [-0.9, 0.9]. Most industrial drives are designed to provide a maximum, instantaneous motor current equal to twice the rated current. For the stationary d-q frame of reference, the maximum current is equal to the peak value of the line-to-line current in the 3-phase (abc) reference frame. Hence, a value equal to twice the peak value of the 3-phase line current was used as the base current for normalization. Similarly, the peak value of the phase voltage was used for normalizing voltage measurements and the speed was normalized as a percentage of its rated value. This method corresponds well with the *per unit* system used in electric energy systems terminology for normalization. The normalization base values are tabulated in Table 2.

Since the output of the ANN is limited to [-1, 1], it is important that the normalization range be chosen carefully so that the data never exceeds the range of training data used.

4.2.3 **Generation of Training Data**

The data for training the speed estimator was generated by simulating the motor model given by equations 4.1 through 4.6 along with mechanical load dynamics, assuming a constant torque load, using fourth-order Runge-Kutta method. A 3 HP induction motor was used for the simulation, with the parameters given in Table 3 [18].

Table 3 Parameters and ratings of induction motor used for simulation studies

Machine rating			Torque	I_{abc}	R_s	L_s	L_m	L_r	R_r	J
HP	volt	rpm	N-m	A	Ω	mH	mH	mH	Ω	$kg.m^2$
3	220	1710	11.9	5.8	0.435	71.312	69.31	71.312	0.816	0.089

The data should span the whole hyperspace for the input and output variables, namely the motor voltages, currents and the speed so that the neural network is exposed to the whole domain of its operation. The frequency of the motor voltage is directly related to the motor speed and the amplitude is dictated by the maximum V/f ratio for the motor, so as to limit the maximum airgap flux created. The motor current is an indirect function of the airgap flux and the load torque on the motor. For obtaining the training and test data, the state model of the motor was simulated using open loop V/f control to obtain the desired range of speed, voltage and current variations. Figure 4.3 shows the block diagram used for the simulation. The s-function block shown in the figure is written in C and compiled as a MATLAB executable to improve the simulation speed.

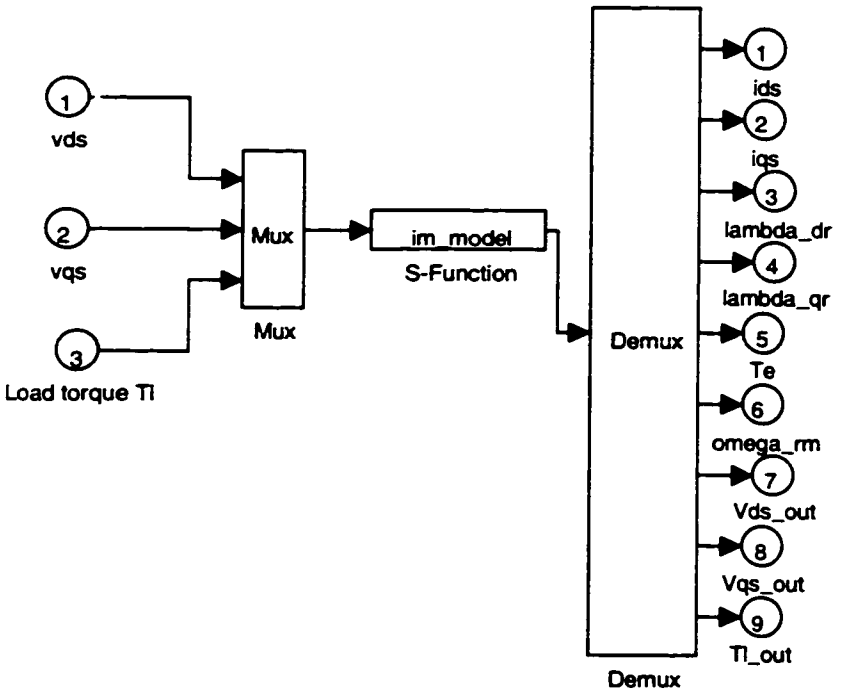


Figure 4.3 Simulink block diagram showing s-function

To generate training data, the frequency of the applied voltage was varied in steps from zero to rated frequency and back to zero in both positive as well as negative directions. This cycle was repeated for 5 different *constant* load torque values. A low acceleration rate was used to obtain more data at each speed and to limit the motor currents to twice the rated value. Simulink performs the integration of the differential equations using a

variable time step and interpolation was used to obtain the data at every 250 μs. The data set was then created using delayed samples of voltages and currents from the interpolated data set to obtain input-output data every 250 μs. This increases the size of the data set considerably, hence the data set is reduced by down sampling it 10 times. Thus, the training set consists of sets of 14 inputs and 1 output, sampled every 2.5 ms, although the delayed samples in *each* set correspond to a sampling rate of 250 μs each. The network was trained using back propagation with momentum and typically about 150 iterations were sufficient to reduce the RMS training and validation error to less than 1%.

4.2.4 **Testing Data**

Once the network has been trained, its performance is tested using a test data set. This test data must be different from the training data set so as to test the generalization capability of the neural network. While the training data set was generated using

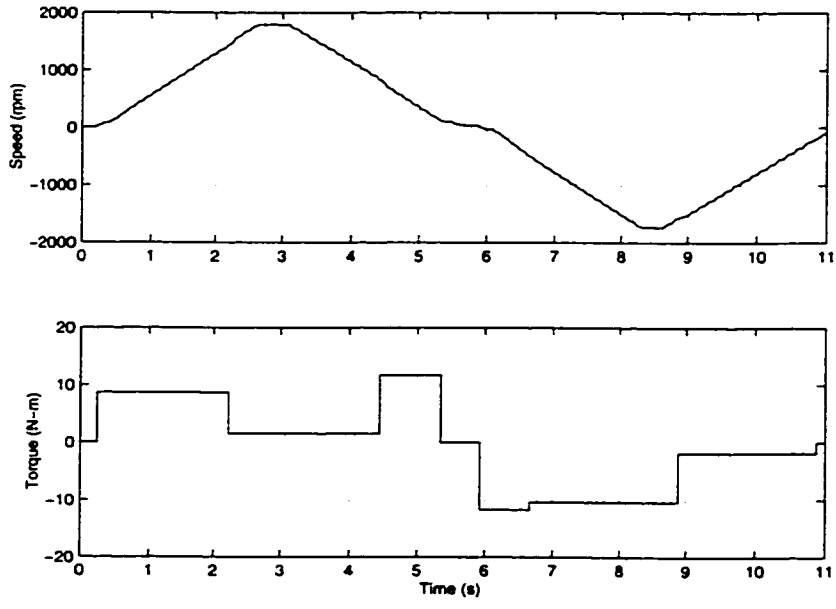


Figure 4.4 Variation of speed and torque in test data

different values of load torque keeping the load *constant* for a given range of frequency variation, this situation is seldom realized in practice where the load changes simultaneously with the motor speed (or supply voltage frequency). Hence, test data was generated by sweeping the applied voltage frequency from zero to positive and back to zero, with a similar cycle in the negative direction, while 5 *random* values of load torque were used over the speed cycle. Figure 4.4 shows the variation in speed and load torque for a typical test data set used. The motor speed rises from 0 rpm to 1800 rpm (rated speed) reduces back to zero and then goes to −1800 rpm and increases back to zero. The load torque takes 8 *random* values over this cycle of speed with step changes from one *random* value to the next.

4.3 Results

The output of the estimator for the test data of Figure 4.4 is shown in Figure 4.5. It can be seen that the estimator output tracks the actual motor speed very well, but is quite

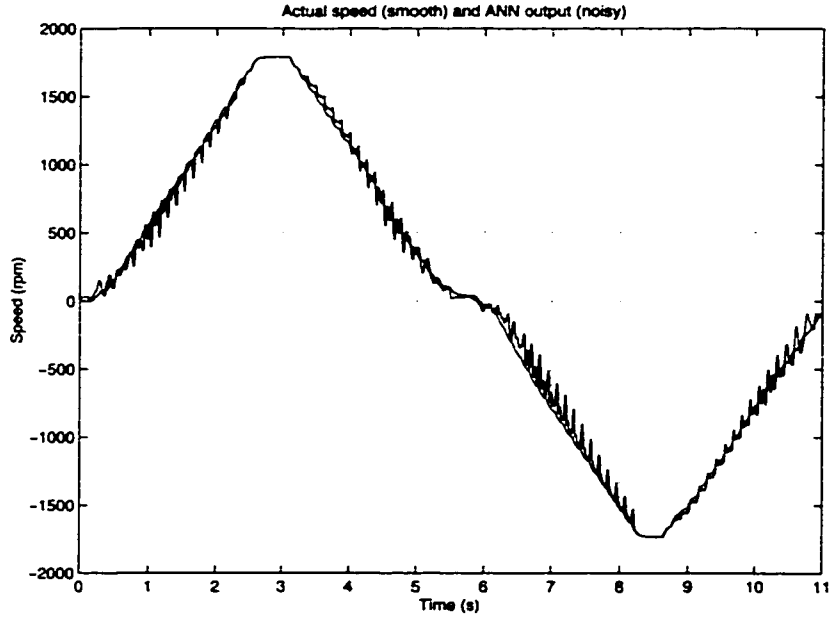


Figure 4.5 Results of open loop estimator for test data

noisy. This noise in the output of the network is an artifact of a higher learning rate during the training phase and the presence of measurement noise in the input test data. A noisy neural network output is generally due to an overfitting of the data. However, when the learning rate was reduced, the RMS training and validation errors for the ANN were higher, with a higher noise rejection. The noise present in the output can be easily filtered, while an overall RMS error cannot be easily compensated. Hence the learning rate was kept high as a tradeoff and a simple moving average filter was applied to the output of the ANN to get the final estimated speed.

The filtered output for the same test data is shown in Figure 4.6 along with the actual motor speed. The RMS error over the speed cycle after filtering is of the order of 1.5% (1.86% for the data set shown). There is almost no effect of the varying load torque on

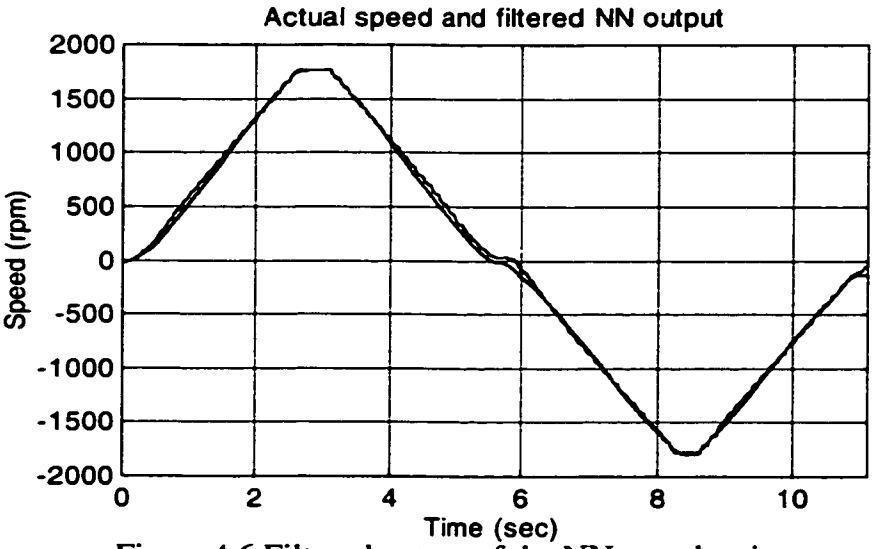


Figure 4.6 Filtered output of the NN speed estimator

the accuracy of speed estimation. This shows that this new method is able to successfully perform the necessary function approximation to obtain a correct speed value without any need for complicated preprocessing of data, such as integration of the stator voltage. From the results, it can also be seen that the low speed performance of the estimator is excellent with error less than 2 % of the actual speed. This is quite in contrast to the performance of other speed estimation methods analyzed in the previous chapter.

4.4 Parameter Variations

The rotor resistance of an induction motor varies widely as a function of motor current, load and temperature. It is desirable that the neural network speed estimator perform equally well in the presence of significant variation in rotor resistance. Since there are two equations for the rotor speed (equations 4.7 and 4.8), it is possible to eliminate the rotor resistance to obtain a single equation for speed, given by equation 4.11.

$$\omega_r = \frac{m_{qs}\left(\int e_{ds} - L_s i_{ds}\right) - m_{ds}\left(\int e_{qs} - L_s i_{qs}\right)}{n_{ds}\left(\int e_{ds} - L_s i_{ds}\right) + n_{qs}\left(\int e_{qs} - L_s i_{qs}\right)} \tag{4.11}$$

where, $$m_{xs} = L_r e_{xs} - \alpha \frac{di_{xs}}{dt}, n_{xs} = \int m_{xs} \quad \text{with } x = d \text{ or } q \tag{4.12}$$

Equation 4.11 is numerically ill conditioned, similar to equation 4.7 and 4.8, but now there is only one equation for estimating the speed. However, this equation forms a

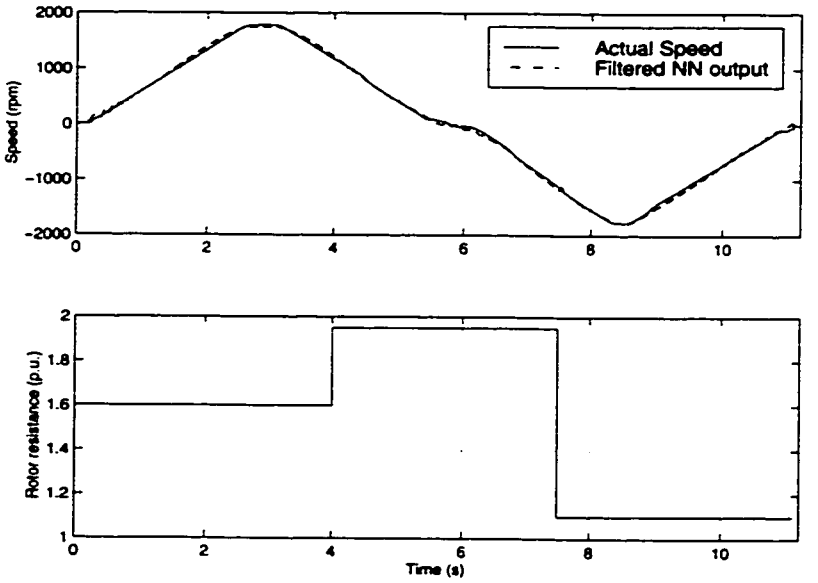


Figure 4.7 Robustness of proposed estimator to variations in rotor resistance

theoretical basis for obtaining a method robust to **rotor resistance variations**. Additionally, the singularities in the equation occur only if the stator voltage is a pure sinusoid and there are no saturation effects. In reality the motor is fed from an inverter,

whose output contains high frequency harmonics in addition to the fundamental frequency voltage. This coupled with any non-linear effects eliminates the singular points making the equation continuous. To test this theory, the motor was simulated with an inverter voltage input. In addition, some noise was added to the applied voltage to simulate the effect of sensor noise. Training data was collected for the same speed and torque changes as before. This data was augmented by duplicating the same tests twice: with 150% rotor resistance and 200% rotor resistance (as compared to the nominal value). Values in excess of 100% of the nominal were chosen since the rotor resistance generally increases with motor current and temperature.

Figure 4.7 shows the results for the tests with variation in rotor resistance. The test data was generated using 3 *random* values of rotor resistance combined with speed and torque variation. The ratio of the rotor resistance to its nominal value is also shown in the figure. The RMS error for this test was 2.09% and the overall range of RMS errors for different tests was of the order of 1.75%, which is very close to the figure obtained without rotor resistance variation. This proves the excellent performance of this estimator even in presence of motor parameter variations. This research and the obtained results were published in [38].

4.5 Discussion

Based on the results obtained from the estimator, the advantages and limitations of this method can be enumerated. The method has many advantages:

- The NN based estimator is able to learn the relationship between the stator quantities and speed, to a high degree of accuracy without being affected by the singularities in the individual equations relating the speed to the stator quantities. This validates the following propositions:
  - A particular equation does not have to be selected for training the network.

  - The numerator and denominator of the speed equations do not have to be estimated separately by different neural networks. This is a major advantage over NN based speed estimation methods proposed till date.
  - Explicit integration and differentiation of stator quantities does not have to be carried out before feeding the data to the estimator. This removes the *inherent limitation* on the low speed performance, which affects other methods. This is proven by the excellent low speed performance obtained.
- Since the estimator learns the relationship between stator quantities and the motor speed based on input-output data, any non-linearities which are not represented by the plant model equations can also be captured by this estimator. This includes non-linearities such as magnetic saturation and presence of cross-bar rotors.
- Complicated pre-processing of data is not required before it is fed to the estimator. This improves the execution speed after the neural network has been trained and makes it an attractive proposition for real-time applications.
- The performance is robust to drastic, instantaneous variations in motor load torque.
- The method developed is robust to variations in the rotor resistance. It should be noted that the method is tested for instantaneous step changes in rotor resistance, which is the worst case test. In reality the rotor resistance varies slowly over a long period of time.
- No assumption is made regarding the *speed/torque control method* used. Thus, this speed estimator can be applied equally easily with either field-oriented control or direct torque control.

The main limitation of this estimator is that it only provides estimates of the motor speed. Rotor flux estimates required for torque control have to be obtained separately which can cause accumulation of errors. Rotor flux estimation cannot be performed using this method since the collection of training data will then require the use of rotor flux sensors. Another disadvantage is that there is no direct error correction available. If the motor

model changes considerably over time with changes in multiple parameters, the performance of the estimator may deteriorate.

Considering these limitations a new closed loop state estimator is proposed which provides simultaneous estimates of motor flux and speed and has error correction abilities. This method is described in detail in the next chapter.

Chapter 5 Adaptive State Estimator Using Neural Networks

The previous chapter detailed the development of an open loop, neural network based, induction motor speed estimator. The results obtained from that development suggest a need for a closed loop estimator which can adapt to changes in the plant model, while retaining the advantages of the developed open loop estimator. The main advantage of using neural networks is their ability to train using input-output data and approximate any non-linear function. This includes non-linearities in the motor dynamics, such as magnetic saturation, which are difficult to model. In addition, the use of neural networks removes the need for any explicit integration of measured voltages and currents which removes any *inherent limitation* on the low speed estimation ability. Another advantage is the good robustness to variation in the motor parameters. The main limitation of the open loop estimator is its inability to *simultaneously* estimate motor speed and flux. Neural networks are also limited in that they do not have a systematic method of filtering the inevitable measurement noise, which can cause their output to be very noisy as seen in the results of Chapter 4.

On the other hand, it can be seen from Chapter 3, that non-linear state estimators gave the best performance among methods proposed in the literature. Among these, the Extended Kalman Filter (EKF) based joint state and parameter estimator performs very well at all speeds. The EKF uses a systematic approach when dealing with measurement and process noise and gives a state estimator whose outputs are the rotor flux and motor speed. However, the dependence of the EKF on the exact plant model makes the method highly susceptible to variations in motor parameters. The advantages and limitations of EKF and neural networks are complimentary which suggests the use of a combination of the two techniques for the development of the closed loop estimator.

5.1 Combination of NN with EKF

Based on the above discussion, a neural network based, adaptive, induction motor state estimator using EKF is developed. The EKF normally uses a mathematical model for the induction motor. This model has d and q-axis stator current and rotor flux components as the states. The motor speed is estimated simultaneously by augmenting the state vector with the speed and setting the time derivative of the speed to zero, as explained in section 3.4.3. The motor speed is then an adjustable parameter of the model which is updated by the EKF.

If the motor model used for the EKF is augmented or replaced by a neural network, a generalized non-linear plant model is obtained, which can capture any unmodeled dynamics. If the NN weights are added to the state vector, the EKF can then update these weights in addition to updating the speed estimate. This approach has the potential of combining the advantages of both the EKF and neural networks. The EKF systematically handles measurement noise and any process noise while the neural network functions as a non-linear approximator. The structure is inherently adaptive, since the NN weights are updated by the EKF. Although this approach is developed here for the induction motor problem, it can be applied to any *observable*, non-linear state estimation problem. Two structures are possible for the estimator based on the contribution of the neural network and these are discussed in the next section.

5.2 Structure of the Estimator

From the non-linear, state space model of the induction motor, the states are the d and q-axis stator current components and rotor flux components. For SSIM control, the motor speed is considered as an additional state and is added to the state vector. Thus, the augmented state vector is:

$$\left[i_{ds}\ i_{qs}\ \lambda_{dr}\ \lambda_{qr}\ \omega_r\right]^T \tag{5.1}$$

The inputs to the motor are the d and q-axis stator voltage components, v_{ds} and v_{qs}. Thus, the plant has 2 inputs, 5 states with 2 measured outputs being the 2 stator current components. If a neural network is to be used, either to augment the known plant dynamics or as a complete plant model, it should have the same structure as the discrete time equivalent motor model given by:

$$X(k+1) = f(X(k),\ U(k)) \tag{5.2}$$

where,

$$X(k) = \left[i_{ds}(k)\, i_{qs}(k)\, \lambda_{dr}(k)\, \lambda_{qr}(k)\, \omega_r(k)\right]^T \tag{5.3}$$

Thus, the neural network should have 7 inputs and 5 outputs, with the number of hidden neurons and layers being selectable. Two topologies are possible for such an estimator depending on the role of the neural network. One option is to use the NN as the complete motor model and the second option is to use the NN in conjunction with the known mathematical model of the motor.

5.2.1 **Plant model augmented by a Neural Network**

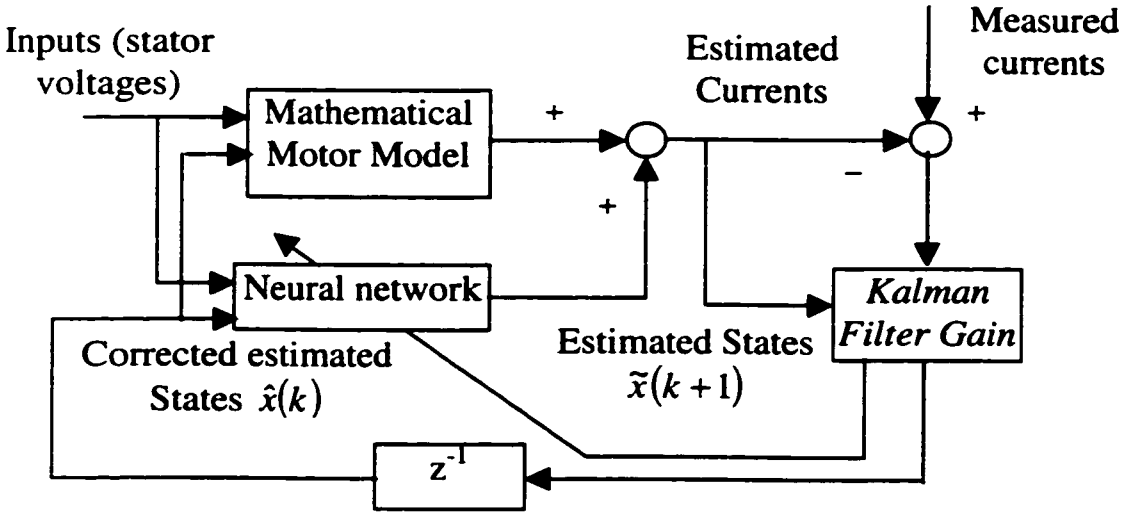


Figure 5.1 Block diagram of adaptive EKF using a plant model augmented by a NN

Figure 5.1 shows the block diagram of the adaptive estimator where the NN is used to augment the discrete time motor model. The output of the NN is the estimated state vector (5 motor states at the next sampling instant) which is added to the estimated state

vector as obtained from the mathematical, discrete time motor model, to get the final estimated states. Of these states, the motor currents are measured, and the error between the measured and estimated currents is fed to the extended Kalman filter algorithm, which is shown by the *Kalman Filter Gain* block in the figure. The outputs from this block are the corrected motor states at the next sampling time and the updated NN weights. The corrected states are fed back to the NN and the motor model, after passing through a delay block, while the NN weights are directly updated. The function of the NN is to approximate the error between the motor and its model.

With this structure, all NN weights are initialized to zero, which implies that the motor model is assumed to be correct at initialization. Thus, if the motor parameters can be obtained, the NN does not have to be trained offline. The NN weights are then updated as part of the EKF depending on the error in the estimation of the measured states. Another advantage of this method is that the information of the available motor model is effectively utilized. The disadvantage is the need to evaluate the motor model as well as the NN at every step. The EKF also has to calculate the Jacobian matrix for the motor model as well as the NN at every step, which imposes a higher computational burden.

5.2.2 **NN as complete plant model**

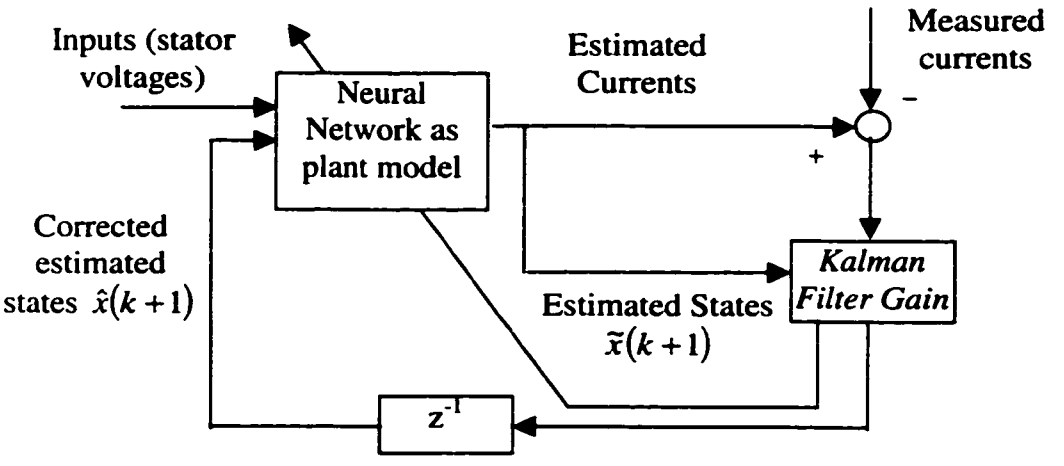


Figure 5.2 Adaptive EKF using NN as complete plant model

Figure 5.2 shows the block diagram of the second possible structure for the adaptive estimator. Here, the NN is used as a complete discrete time plant state model.
The inputs to the NN are the plant inputs and the corrected states at the previous instant and the outputs are the plant states at the next instant. Since the NN outputs are used as states, it is not possible to start with zero weights. The NN has to be trained offline using data obtained from simulating a motor model. It should be noted that experimental data cannot be used to train the network since the rotor flux components cannot be measured. The advantage of this structure is the elimination of the motor model, which reduces the computational burden. However, if the NN is not trained over the entire operational hyperspace, the estimator can become unstable when the NN is fed with inputs that are not used during the training phase.

5.3 Mathematical basis

The adaptive EKF based estimator structures proposed above are based on the theory of joint state and parameter estimation using Kalman filters [35, 39, 40]. Joint state and parameter estimation is implemented by augmenting the state vector with the parameters to be estimated and setting the time derivative of that parameter to zero. For SSIM without motor parameter adaptation, the augmented state vector has 5 quantities: the stator current components, the rotor flux components and the motor speed. Then, the motor model can be written in discrete time form as:

$$\begin{aligned} X_{k+1} &= f_k(X_k, U_k) + \eta_k \\ Z_k &= h(X_k) + \nu_k \end{aligned} \tag{5.4}$$

where, X_k is the motor state vector at time k (including motor speed), U_k is the plant input vector at time k (motor voltages), η_k is the additive process noise, ν_k is the additive measurement noise, $f(\bullet)$ represents the non-linear motor state dynamics including unmodeled effects, and $h(\bullet)$ is the output coupling function for the motor (outputs are motor currents).

If a neural network is used either to augment or replace the motor model, the state dynamics of new motor model is given by:

$$\begin{aligned} \tilde{X}_{k+1} &= \hat{f}\left(\hat{X}_k, U_k, \hat{W}_k\right) \\ \tilde{Z}_k &= h\left(\tilde{X}_{k+1}\right) \end{aligned} \tag{5.5}$$

where, w_k are the weights of the NN, $\tilde{X}_{k+1}$ is the estimated state vector, $\hat{X}_k$ is the estimated state vector after correction by the Kalman filter, U_k is the input vector to the plant, $\hat{W}_k$ is the NN weight vector, $\tilde{Z}_k$ is the estimated plant output vector and $\hat{f}(\bullet)$ approximates the actual plant dynamics $f(\bullet)$.

If the NN weights have to be adjusted based on the error between the measured outputs Z_k and the estimated outputs $\tilde{Z}_k$, it is necessary to include the NN weights as additional states in the EKF algorithm. Then the augmented state model is:

$$\begin{bmatrix} \tilde{X}_{k+1} \\ \tilde{W}_{k+1} \end{bmatrix} = \begin{bmatrix} \hat{f}\left(\hat{X}_k, u_k, \hat{W}_k\right) \\ \hat{W}_k \end{bmatrix} \tag{5.6}$$

Thus, the EKF estimates the motor states as well as the weights of the NN which are assumed to have the dynamics:

$$W_{k+1} = W_k \tag{5.7}$$

with the residual being,

$$f_k(\bullet) - \hat{f}_k(\bullet) \tag{5.8}$$

Assuming observability, as:

$$h_k(X_k, U_k) - h_k\left(\tilde{X}_k, U_k, \tilde{W}_k\right) \rightarrow 0 \ \forall \ k, \tag{5.9}$$

$$\begin{aligned} &\hat{X}_k \rightarrow X_k, \quad \text{and} \\ &f_k(\bullet) - \hat{f}_k(\bullet) \rightarrow 0 \end{aligned} \tag{5.10}$$

The induction motor plant is observable for all non-zero frequency stator voltages as shown in section 3.1.1 of Chapter 3. This is the basis of the proposed adaptive state estimator using neural networks. The equations for applying the algorithm are now derived starting from the equations for a standard EKF as applied to the induction motor plant.

The discrete time model of the induction motor with the motor speed as an augmented state is given by [41]:

$$
\begin{aligned}
i_{ds}(k+1) &= [(a+b\rho)T_s+1]i_{ds}(k)+T_s c\rho\lambda_{dr}(k)+T_s c\omega_r(k)\lambda_{qr}(k)+\frac{T_s}{\sigma L_s}v_{ds}(k)\\
i_{qs}(k+1) &= [(a+b\rho)T_s+1]i_{qs}(k)-T_s c\omega_r(k)\lambda_{dr}(k)+T_s c\rho\lambda_{qr}(k)+\frac{T_s}{\sigma L_s}v_{qs}(k)\\
\lambda_{dr}(k+1) &= L_m\rho T_s i_{ds}(k)+(1-\rho T_s)\lambda_{dr}(k)-T_s\omega_r(k)\lambda_{qr}(k)\\
\lambda_{qr}(k+1) &= L_m\rho T_s i_{qs}(k)+T_s\omega_r(k)\lambda_{dr}(k)-(1-\rho T_s)\lambda_{qr}(k)\\
\omega_r(k+1) &= \omega_r(k)
\end{aligned}
\qquad (5.11)
$$

where, T_s is the sampling period for the discrete-time system.

This can be written in a compact form as equations 5.2 and 5.3. The measurable outputs are the stator current components, $i_{ds}(k)$ and $i_{qs}(k)$, so that the output equation is:

$$Z_k = HX_k \qquad (5.12)$$

where,
$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix} \qquad (5.13)$$

and,
$$Z_k = [i_{ds}(k) \quad i_{qs}(k)]^T \qquad (5.14)$$

For a non-linear, discrete time system given by equation 5.5, the Extended Kalman filter is given by:

$$
\begin{aligned}
\tilde{X}_{k+1} &= \hat{f}(\hat{X}_k, U_k)\\
\tilde{P}_{k+1} &= F_k\hat{P}_k F_k^T + Q_k
\end{aligned}
\qquad (5.15)
$$

where,
$$F_k = \left.\frac{\partial f(\bullet)}{\partial X}\right|_{\hat{X}_k, U_k} \qquad (5.16)$$

Equation 5.15 is known as the time update equation of the EKF. The tilde("~") operator is used to denote the estimated states obtained by updating the previous corrected state through the estimated dynamics. The estimated states are then corrected via the measurement update to obtain the final (corrected) estimates which are denoted by the hat ("^") operator.

The measurement update is given by:

$$\begin{aligned}\hat{X}_k &= \hat{X}_k + K_k\left(Z_k - H\tilde{X}_k\right)\\ \hat{P}_k &= [I - K_k H]\,\tilde{P}_k\,[I - K_k H]^T + K_k R_k K_k^T\\ K_k &= \tilde{P}_k H^T\left[H\tilde{P}_k H^T + R_k\right]^{-1}\end{aligned} \tag{5.17}$$

For the induction motor plant model, the Jacobian, F_k is given by:

$$F_k = \begin{bmatrix} 1+(a+b\rho)T_s & 0 & c\rho T_s & c\hat{\omega}_r(k)T_s & c\hat{\lambda}_{qr}(k)T_s \\ 0 & 1+(a+b\rho)T_s & -c\hat{\omega}_r(k)T_s & c\rho T_s & -c\hat{\lambda}_{dr}(k)T_s \\ L_m\rho T_s & 0 & (1-\rho T_s) & -T_s\hat{\omega}_r(k) & -T_s\hat{\lambda}_{qr}(k) \\ 0 & L_m\rho T_s & T_s\hat{\omega}_r(k) & (1-\rho T_s) & T_s\hat{\lambda}_{dr}(k) \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.18}$$

When the plant model is augmented with a neural network as shown in Figure 5.1, the dynamics of the augmented model are given by:

$$X_{k+1} = \hat{f}_k(X_k, U_k) + g_k(X_k, U_k, W_k) \tag{5.19}$$

where, W_k is an ideal NN weight vector, which is unknown, and g_k is the function represented by the neural network. The NN output with the ideal weight vector, when combined with the known plant modes gives an exact match with the actual plant dynamics. According to the theory of joint state and parameter estimation, the network weights are added as new states for estimation using the EKF, to give the new time update of the state equations as:

$$\begin{bmatrix}\tilde{X}_{k+1}\\ \tilde{W}_{k+1}\end{bmatrix} = \begin{bmatrix}\hat{f}\left(\hat{X}_k, U_k\right) + g_k\left(X_k, U_k, \hat{W}_k\right)\\ \hat{W}_k\end{bmatrix} \tag{5.20}$$

where, the time update does not change the neural network weights.

The covariance update is given by:

$$\tilde{P}_{k+1} = \left(L_k + \begin{bmatrix}M_k\\ 0\end{bmatrix}\right)\ \hat{P}_k\ \left(L_k + \begin{bmatrix}M_k\\ 0\end{bmatrix}\right)^T + Q_k \tag{5.21}$$

where,
$$L_k = \begin{bmatrix}F_k & 0\\ 0 & I\end{bmatrix},\quad M_k = \frac{\partial g_k}{\partial \hat{X}_{new}} \quad and \quad \hat{X}_{new} = \begin{bmatrix}\hat{X}_k\\ \hat{W}_k\end{bmatrix} \tag{5.22}$$

The "0" terms in these equations are matrices having appropriate dimensions and having all terms as zero while I represents a unit matrix with the appropriate dimensions. The dimensions can be obtained based on the dimensions of the state vector and the NN weight vector. If the plant has n states and the NN has l weights, the covariance matrix has $(n+l)$ rows and columns, g_k has n equations, so that M_k has n rows and $(n+l)$ columns. Thus, the zero matrices in the covariance update equation have l rows and $(n+l)$ columns. The zero matrices in the top row of L_k have n rows and l columns, while that in the bottom row has l rows and n columns. The identity matrix is has dimension l.

The measurement update equations can be easily obtained once the output coupling matrix is suitably modified. Since none of the added states (NN weights) are measured, the modified output coupling matrix is given by:

$$H_{new} = [H_{mxn} \quad 0_{mxl}] \tag{5.23}$$

where, H is the original output coupling matrix and 0 is a zero matrix. The dimensions of each matrix is also included in the equation as a subscript. The measurement update equations are:

$$\begin{aligned} \begin{bmatrix} \hat{X}_k \\ \hat{W}_k \end{bmatrix} &= \begin{bmatrix} \tilde{X}_k \\ \tilde{W}_k \end{bmatrix} + K_k \left(Z_k - H_{new} \begin{bmatrix} \tilde{X}_k \\ \tilde{W}_k \end{bmatrix} \right) \\ \hat{P}_k &= [I - K_k H_{new}] \tilde{P}_k [I - K_k H_{new}]^T + K_k R_k K_k^T \\ K_k &= \tilde{P}_k H_{new}^T \left[H_{new} \tilde{P}_k H_{new}^T + R_k \right]^{-1} \end{aligned} \tag{5.24}$$

Since H_{new} has a zero matrix, these measurement update equations can be updated to obtain:

$$\begin{aligned} \begin{bmatrix} \hat{X}_k \\ \hat{W}_k \end{bmatrix} &= \begin{bmatrix} \tilde{X}_k \\ \tilde{W}_k \end{bmatrix} + K_k \left(Z_k - H \tilde{X}_k \right) \\ \hat{P}_k &= [I - K_k H_{new}] \tilde{P}_k [I - K_k H_{new}]^T + K_k R_k K_k^T \\ K_k &= \begin{bmatrix} \tilde{P}_{knn} H^T \\ \tilde{P}_{kln} H^T \end{bmatrix} \left[H \tilde{P}_{knn} H^T + R_k \right]^{-1} \end{aligned} \tag{5.25}$$

where,

$$\tilde{P}_k = \begin{bmatrix} \tilde{P}_{knn} & \tilde{P}_{knl} \\ \tilde{P}_{kln} & \tilde{P}_{kll} \end{bmatrix} \tag{5.26}$$

It can be seen that the simplified equations have a much lower computational cost, since only an n dimensional matrix has to be inverted, instead of an $(n+l)$. This is the same matrix that has to be inverted for the standard EKF (i.e. when the NN is not used). Thus, if the plant has 5 states and the NN has 25 weights, only a 5 x 5 matrix has to be inverted. This considerably alleviates one of the main disadvantages of this proposed adaptive estimator.

The time update of the covariance matrix requires the Jacobian matrix of the neural network outputs with respect to the plant states (which also constitute the NN inputs) and NN weights. This Jacobian matrix M_k is given by:

$$M_k = \frac{\partial g_k}{\partial \begin{bmatrix} \hat{X}_k \\ \hat{W}_k \end{bmatrix}} = \begin{bmatrix} \frac{\partial g_1}{\partial x_1} & \frac{\partial g_1}{\partial x_2} & \cdots & \frac{\partial g_1}{\partial x_n} & \frac{\partial g_1}{\partial w_1} & \frac{\partial g_1}{\partial w_2} & \cdots & \frac{\partial g_1}{\partial w_l} \\ \vdots & \vdots & \cdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ \frac{\partial g_n}{\partial x_1} & \frac{\partial g_n}{\partial x_2} & \cdots & \frac{\partial g_n}{\partial x_n} & \frac{\partial g_n}{\partial w_1} & \frac{\partial g_n}{\partial w_2} & \cdots & \frac{\partial g_n}{\partial w_l} \end{bmatrix} \qquad (5.27)$$

The partial derivatives of NN outputs with respect to each weight can be obtained using the chain rule. The main difference between this Jacobian and the backpropagation training algorithm used for feedforward networks is the use of the partial derivatives of the network outputs with respect to the network inputs in addition to the partial derivatives with respect to the NN weights.

5.3.1 Comparison with Recurrent Networks

If the proposed estimator is viewed from the neural network point of view, it basically implements a recurrent neural network along with an EKF based training algorithm. EKF based recurrent network training has been proposed by many researchers [42, 43] and has been compared with other recurrent network training algorithms with respect to computational complexity [44, 45]. Most researchers have concluded that the EKF algorithm is useful for training recurrent networks especially when noise is present in the data used for training the networks. The complexity of the EKF is also found to be

similar to other recurrent network training methods such as Real-time recurrent learning (RTRL) and Backpropagation through time (BPTT).

However, all the published research shows work for training a neural network when **all its outputs are measured**. Thus all the work in references [42-45] deals with training of recurrent networks to approximate a dynamic system and **does not deal with non-linear state estimation**, where some of the NN outputs are not measured and have to be estimated.

5.3.2 **Data Normalization**

Whenever a neural network is used in any application, the input data to the network and the desired outputs must be normalized. The same applies here and there are two ways of working around this limitation. One is to normalize all the dynamic equations so that all quantities are normalized. This may require changes in the parameters used of the plant model, which is augmented by the neural network. The other method is to normalize the data before it is fed to the NN and re-scale the NN outputs back to physical quantities. This involves changes to the calculation of the Jacobian for the NN outputs, M_k. Both approaches were tried in this work and the second method has been used due to its relative simplicity.

5.4 Results

The performance of the adaptive estimator was tested for different conditions and the results obtained are described below. The performance is compared with that of a standard EKF.

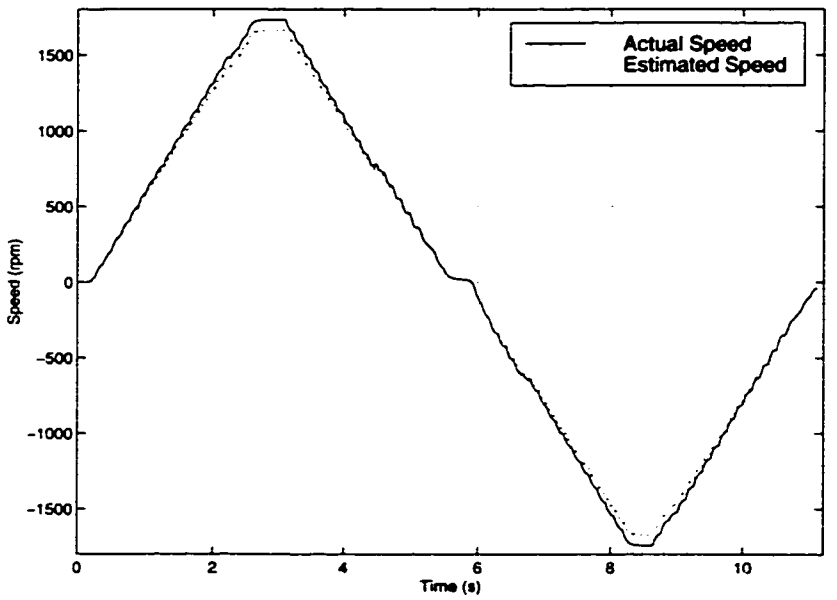


Figure 5.3 Actual and Estimated Motor Speed using standard EKF

5.4.1 **Fixed motor parameters**

The proposed algorithm was evaluated by simulating the induction motor at different speeds and load torque's and concurrently simulating the estimator on-line. As a first step, the standard EKF algorithm was used without any NN augmentation. This serves as a basis for further comparison with the proposed adaptive estimator. The motor parameters used were the same as that used for the open loop estimator in Chapter 4 given in Table 3. For proper comparison with the open loop estimator results, the actual motor speed was varied in the same range.

Figure 5.3 shows the actual motor speed and the speed as estimated by a standard EKF (not augmented with a neural network). The actual motor data is generated using constant motor parameters, with a variation in the load torque as shown in Figure 5.4. The RMS error in the speed estimation is 2.09 % which is similar to the error obtained using the open loop, NN based speed estimator described in the previous chapter. This testing shows the efficacy of the open loop estimator which needs much less computation

than the standard EKF. Hence it is much faster in terms of CPU time. It should be noted, however, that the EKF performs simultaneous flux and speed estimation, while the open loop estimator only estimates motor speed.

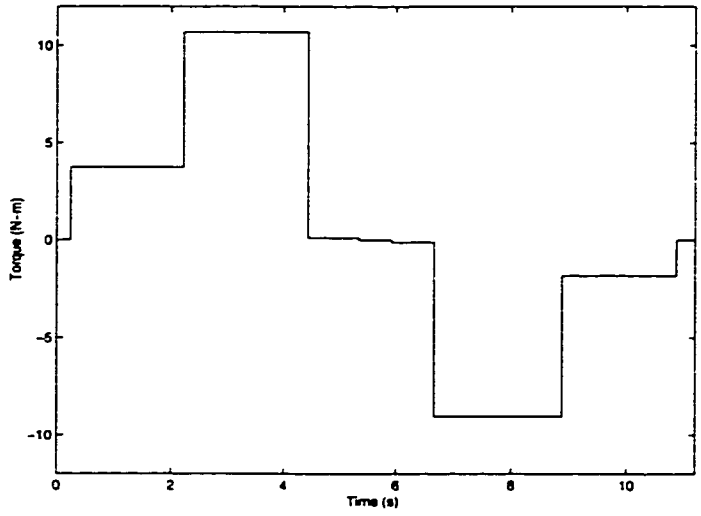


Figure 5.4 Variation of load torque for testing EKF

The flux estimation performance of the standard EKF is shown in Figure 5.5.

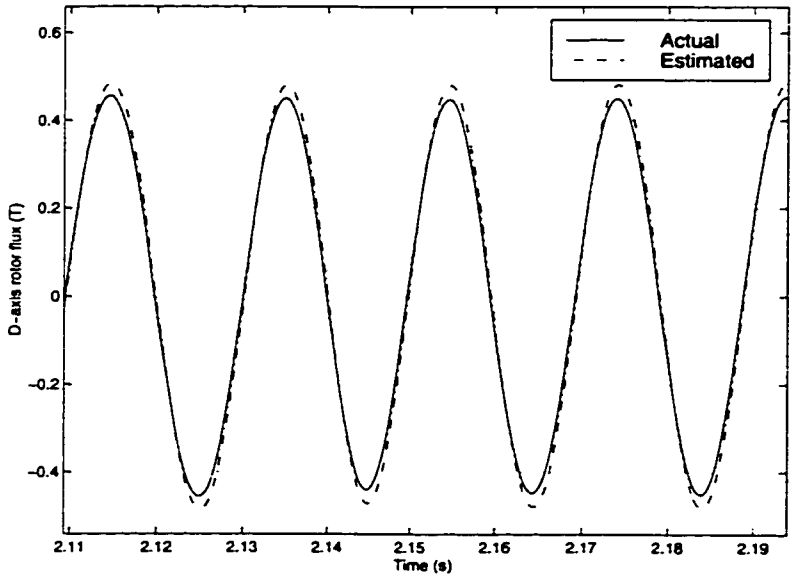


Figure 5.5 Flux estimation results using standard EKF

Since the flux varies sinusoidally at a frequency close to 60 Hz for a motor speed of about 1800 rpm, a plot of the actual and estimated flux for the full time period (from time=0 to time=11.2 seconds) is very cluttered. Hence, the flux performance for a few cycles is shown. It can be seen that the EKF overestimates the rotor flux and the maximum error is about 0.04 Tesla when the actual flux has an amplitude of about 0.5 Tesla. This gives a flux estimation error of about 8%. Figure 5.6 shows the flux

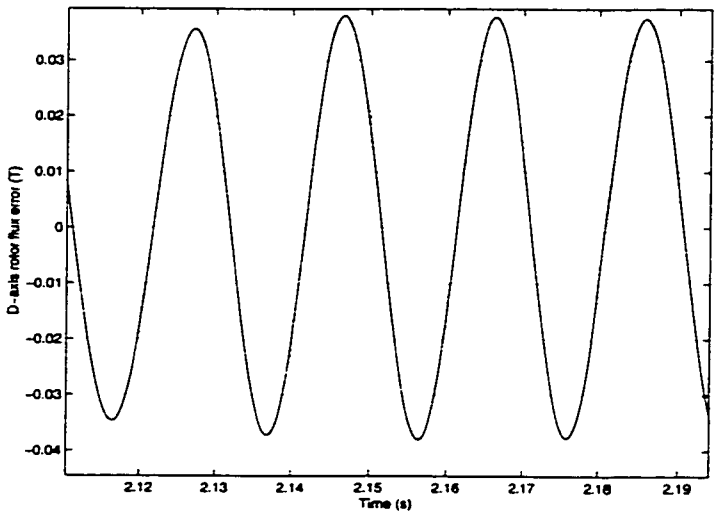


Figure 5.6 Flux estimation error using standard EKF

estimation error for the same time axis as the flux estimation performance plot in Figure 5.5. The flux and speed estimation errors are a function of the process and noise covariance matrices Q_k and R_k used in the EKF calculations. The process covariance matrix has to be adjusted by trial and error.

To validate the equations derived for the adaptive estimator, it was also tested for the same data with no variation in the motor parameters. Since the parameters are not varied during this testing, it was expected that the results with the NN augmented, adaptive state estimator would be the same as that with the standard EKF or better. Figure 5.7 shows the speed estimation performance obtained using the adaptive estimator. The RMS error

is about 0.4944% which is much better than that obtained using the standard EKF. In fact, it is difficult to discern any speed estimation error from Figure 5.7.

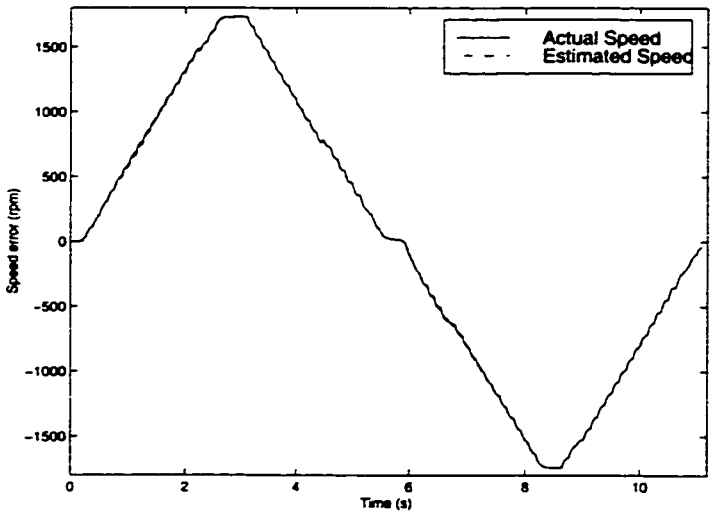


Figure 5.7 Speed estimation performance of the adaptive estimator with no parameter variation

To better compare the performance of the two estimators, the speed estimation errors are

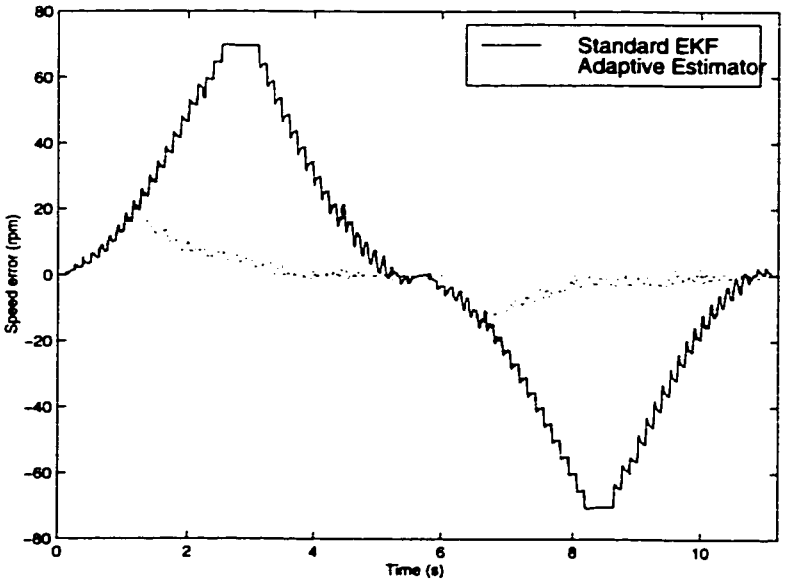


Figure 5.8 Speed estimation error using standard EKF and adaptive estimator

plotted on the same graph in Figure 5.8. It can be seen that the estimation error for the adaptive estimator is initially the same as that for the standard EKF. This corresponds to the time period where the NN weights are almost zero so that the contribution of the NN to the motor dynamics is negligible. As the state estimation error increases, the NN weights change and the NN output adds to the motor model output to reduce the estimation error. Figure 5.9 shows the estimation error for the d-axis rotor flux, λ_{dr} using the adaptive estimator. The maximum flux estimation error is 0.0184 T, which is about half that obtained using the standard EKF. Although the output of the adaptive estimator is noisy, the noise has very small amplitude, and hence it does not affect the flux estimation performance. In addition to the better state estimation, the performance of the

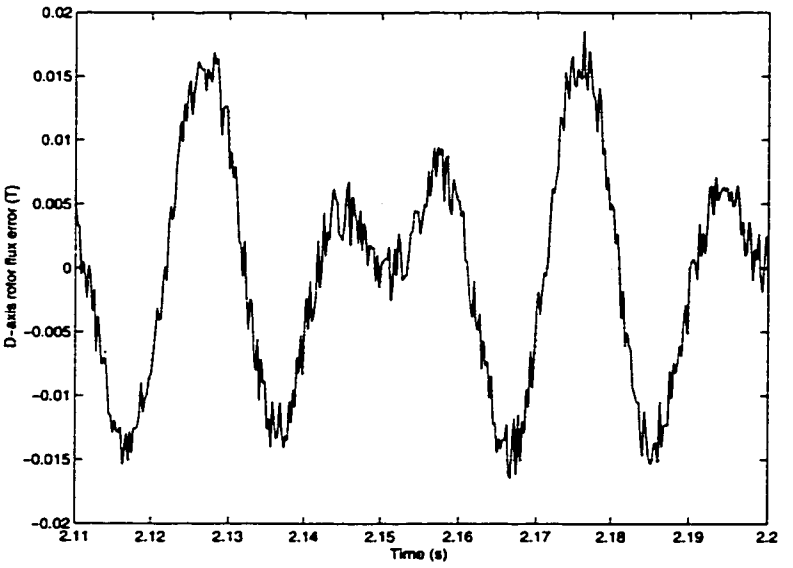


Figure 5.9 D-axis rotor flux estimation error using adaptive estimator

adaptive estimator is also robust to the choice of the process noise covariance matrix Q_k in the EKF calculations. The performance of the standard EKF is quite sensitive to the proper choice of this covariance matrix.

5.4.2 **Parameter Variation**

As the next step, the performance of the standard EKF and the adaptive estimator was compared with data obtained by varying the rotor resistance. The motor model in the standard and the adaptive EKF used a fixed value of the rotor resistance which was set to the rated value, while the motor model used varying rotor resistance to generate the actual data.

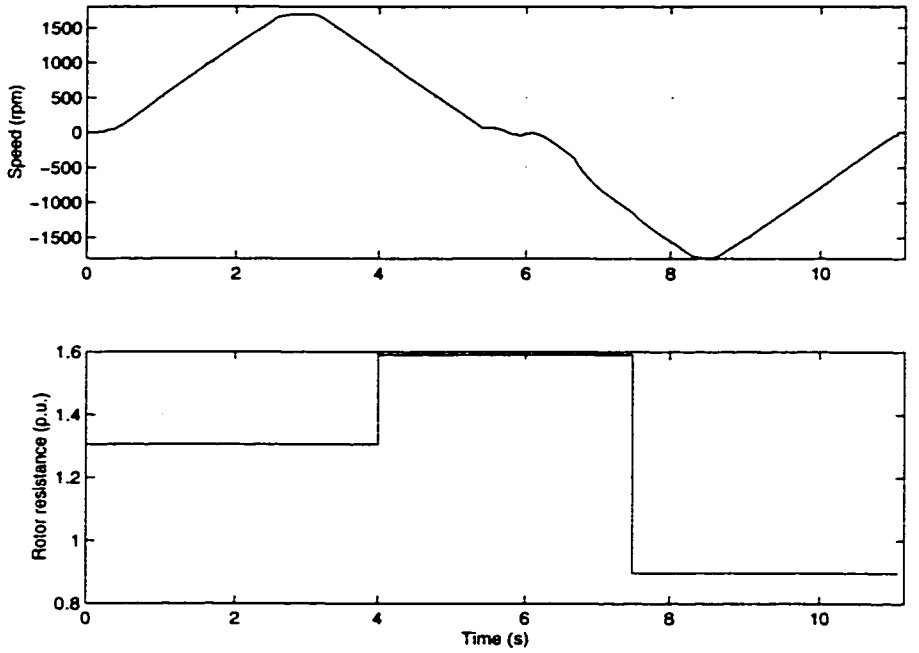


Figure 5.10 Speed and rotor resistance variation used for testing estimator performance

Figure 5.10 shows the motor speed and rotor resistance variation in the test data set used to evaluate the performance of the EKF and the adaptive estimator in presence of parameter variation. In practical applications, the rotor resistance varies slowly as the motor temperature increases. However, the rotor time constant can change rapidly due to high current and changes in the rotor leakage inductance due to magnetic saturation. Hence, the rotor resistance is varied in 3 random steps in the test data while keeping the rotor leakage inductance constant, thus simulating worst case changes in the rotor time

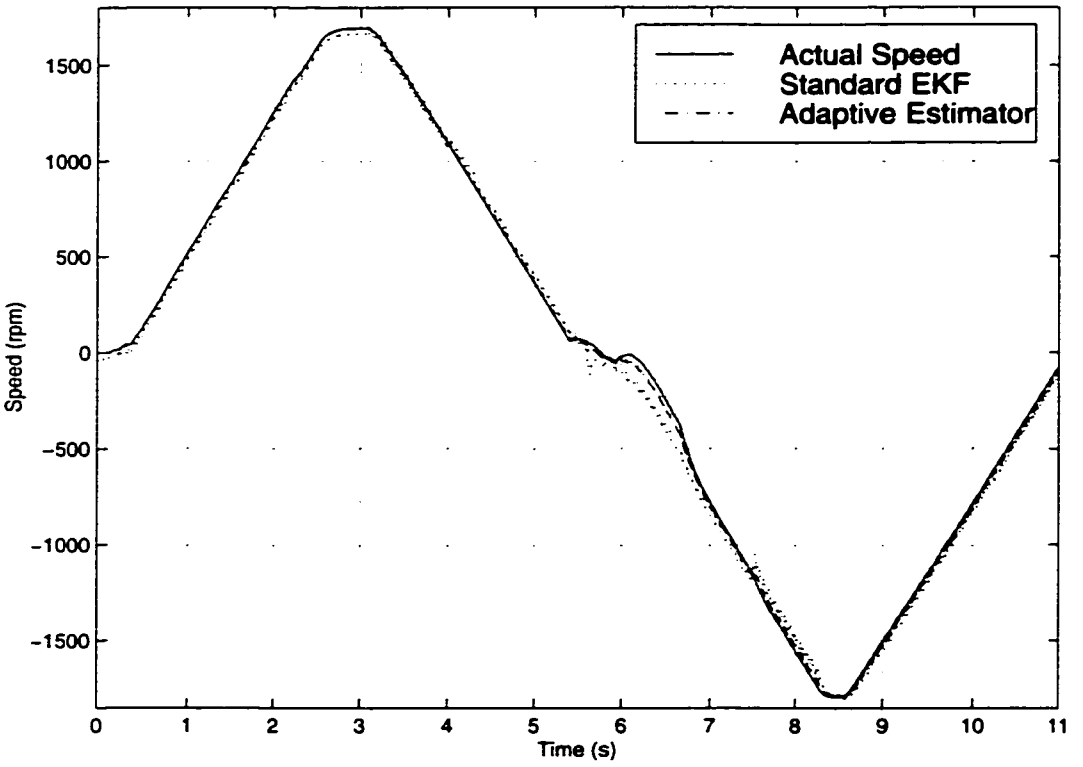


Figure 5.11 Comparison of speed estimation performance in the presence of parameter variation

constant. Figure 5.11 plots the actual motor speed along with the estimated speed for the standard EKF and the adaptive estimator. It can be seen that the adaptive estimator has a much better performance. The RMS speed estimation error is 3.3966 % for the standard EKF, while the error is 1.0857 % for the adaptive estimator. The standard EKF deteriorates in performance as the rotor resistance varies away from its nominal value of 1 per unit. On the other hand, the adaptive speed estimator adapts to the changes in rotor resistance and is able to track the motor speed very well, with an RMS error of the same order as that obtained for no motor parameter variation.

To compare the two methods in a better way, the speed estimation errors for the two methods are plotted in Figure 5.12. Although the estimation errors for both methods follow similar curves, the error obtained with the adaptive estimator is much lower. The maximum error obtained with the standard EKF is about 170 r/min which is almost 10 %

of the rated motor speed, while that with the adaptive estimator is only 50 r/min which is 2.78 % of the rated motor speed.

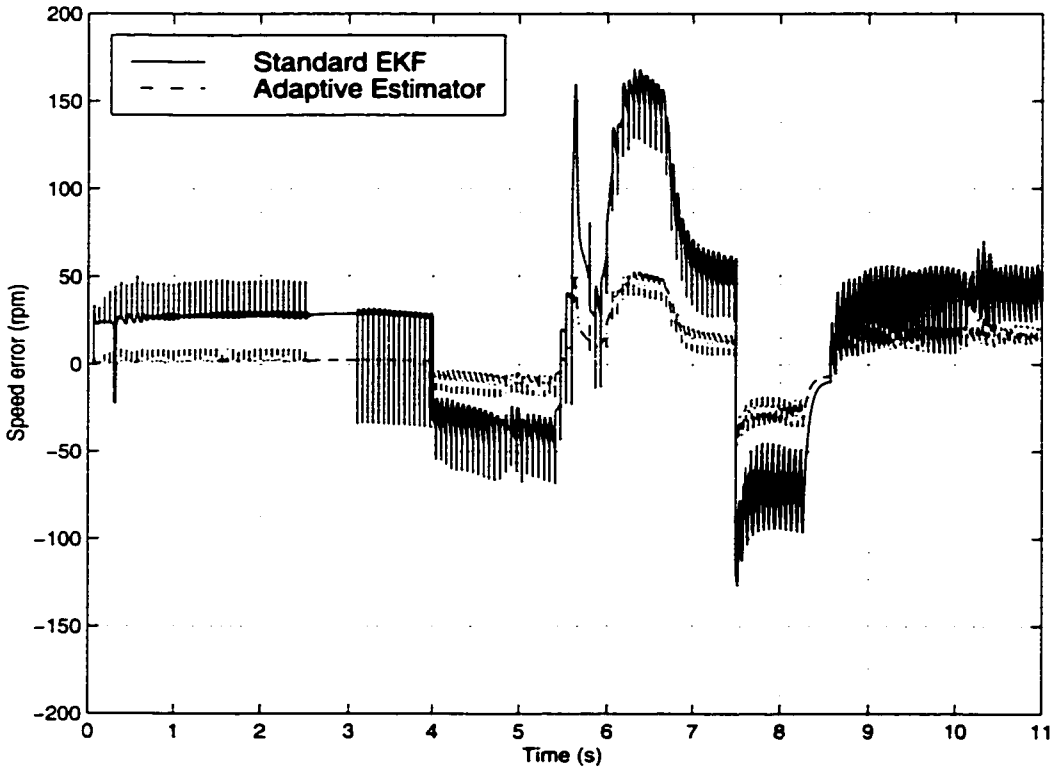


Figure 5.12 Speed estimation error for standard EKF and the adaptive estimator

The flux estimation errors also follow the same pattern with the maximum flux error in the d-axis rotor flux λ_{dr} being 0.074 T for the standard EKF, which is reduced to 0.032 T when the adaptive estimator is used. These results prove the efficacy of the adaptive state estimator.

5.4.3 **Results with NN as complete plant model**

All the above results for the adaptive estimator are obtained using the augmented plant model, which uses the mathematical motor model in conjunction with the NN. The other possible topology uses a neural network as a complete plant model. This method was

tried by training the NN offline, using model simulation data and then using this trained NN in the adaptive EKF algorithm according to the block diagram of Figure 5.2. It was observed that the speed estimation was adequate in most areas of operation. However, the estimator has a tendency to become unstable with the estimated quantities either increasing to very high values or reducing to zero. The topology is also susceptible to changes in the parameters of the motor model which renders it impractical. Further investigation and development may lead to a feasible solution for this topology. However, this approach was not pursued further due to the excellent results obtained using the augmented model topology.

5.5 Discussion

Based on the foregoing theoretical development and results, the advantages and limitations of the adaptive estimator are discussed here. The method has many advantages which address most of the limitations of the open loop speed estimator. These include:

- The developed method yields a general purpose, non-linear state estimator which can be applied to any observable, non-linear plant with parameter variation.
- Results obtained for the induction motor plant prove the excellent performance and the adaptation ability, especially when contrasted with a standard, fixed model EKF.
- This method is robust to changes in plant parameters as well as to the selection of the process noise covariance matrix. This eliminates the two main disadvantages of the extended Kalman filter, while retaining all its advantages.
- Simultaneous motor speed and rotor flux estimation gives a full-fledged state estimator which can be directly used for speed sensorless control of induction motor drives.
- The estimation algorithm does not depend on the choice of any motor torque control method, and is hence universally applicable.

- The use of neural networks helps in capturing the non-linear dynamics that cannot be easily modeled.

The only disadvantage of the developed method is the high computational burden imposed due to the large matrix multiplication operations involved. From software profiling, it was seen that the adaptive estimator adds an additional computing time of 40% over that required for the standard EKF. Given the excellent performance obtained, this does not seem to be a heavy burden. It should be noted that the adaptive estimator needs the same matrix inversion operation (same dimensions) as the standard EKF while adding matrix multiplication operations. As digital signal processors get better optimized for matrix operations, this disadvantage should be further mitigated.

This concludes the description of the work done with regard to speed sensorless induction motor drives. The next 4 chapters concentrate on precision position control of elastic link drives which is the second topic of focus of this dissertation.

Chapter 6 Elastic Link Motion Control

High precision positioning servo control is used in many applications such as robotics, machine tools, semiconductors and biomedical engineering. In fact high performance precision position control is the backbone of many of these industries. Hydraulic, pneumatic or electrical actuators are used for such applications. Electrical motor drives are the preferred actuators for most such applications especially in the low to moderate power range, due to the ease of control and the range of output power available. Most positioning servo applications need linear position control while electric motor drives typically provide rotary motion. This necessitates the use of some mechanical transmission device for converting the rotary motion to linear motion. At present, rigid transmission elements such as lead screws or ball screws are used for this purpose, resulting in high accuracy systems, while elastic transmission elements such as a timing belt are used for lower end systems [7]. The lower accuracy associated with elastic transmissions is due to their non-linear dynamics and the associated performance penalty when conventional, linear control systems are used. The work done here focuses on studying the feasibility of using non-linear control systems with elastic links to obtain high precision positioning.

The next section defines and describes the terms used for measuring the performance of positioning servo systems, followed by a discussion of the three main transmission methods along with their characteristics and applications. Sensors and actuators for positioning servos are discussed in brief, followed by detailed objectives of this research.

6.1 Precision Motion Control

The performance of precision positioning servo systems is generally measured by their positioning accuracy and repeatability. Additional performance measures include the maximum speed and acceleration for which the tracking error is below a desired limit and

the settling time. Accuracy (or absolute accuracy) is defined as the error between the desired position and the actual position. In this sense, the terms accuracy or positioning error are used interchangeably. Repeatability is defined as the ability of a positioning system to return to a particular location from a given starting point, when approaching the

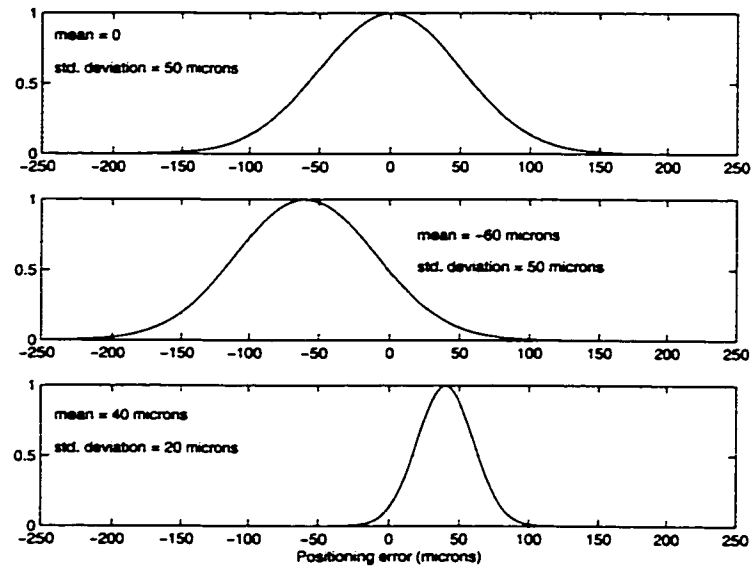


Figure 6.1 Accuracy and repeatability for positioning applications

desired location from the same direction. For single-axis positioning systems, bi-directional accuracy is defined as the ability of a system to reach a particular location when approaching it from either direction. Uni-directional and bi-directional repeatability are the primary specifications for most linear positioning applications [46]. Figure 6.1 shows three sample plots of distribution functions of positioning error, with the y-axis showing a normalized density. The first plot shows has the positioning error to have a mean of 0 and a standard deviation of 50 μm (or microns), which means that the accuracy is very good (zero error). But the positioning error can vary between –100 μm to +100 μm, if 2 standard deviations are considered. In this case, the repeatability is said to be ± 100 μm, which is poor. The second plot shows an accuracy of –60 μm while the repeatability is the same as for the first plot. The third plot shows an accuracy of

accuracy of 40 μm which is worse than the first case. However, the repeatability is ± 40 μm, which is much better than the first two cases. Although good accuracy and repeatability are desired, repeatability is more important since any fixed error can be easily compensated.

Three transmission methods are normally used for converting the rotary motion obtained at the motor shaft to linear motion for the stage. Each of these has particular characteristics which are discussed below [47].

6.1.1 **Acme Lead Screw**

The acme lead screw is one of the most commonly used transmission method for low

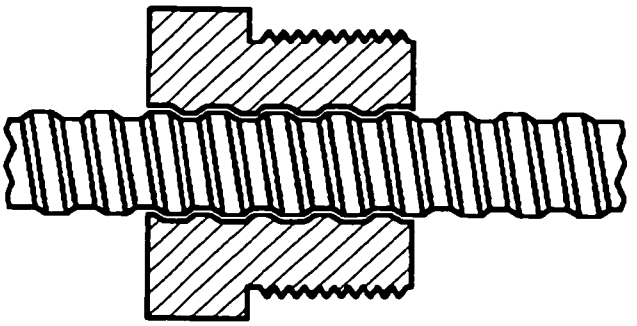


Figure 6.2 Acme lead screw

speed, high precision and high thrust applications. Figure 6.2 shows a cross-sectional view of typically lead screw. It uses a solid brass or steel nut that slides along the threads of a screw. Since there are no rolling elements between the nut and the screw, lead screws are characterized by high values of friction. This results in a transmission efficiency of only 45-50% and the need for rapid dissipation of the heat generated due to friction limits the use of lead screws for low duty cycle applications. A great benefit of the acme lead screw is its ability to hold a vertical load even when the motor power is turned off. All screw driven systems (including ball screw systems described next) have a critical speed limit determined by the first natural frequency of the mechanical

structure. At the critical speed, the screw starts to resonate and the phenomenon is also known as whipping. This critical speed is reduced as the length of the screw increases or as the diameter decreases. Hence, lead screw systems are limited to low travel length applications. Lead screws are manufactured to very close tolerances so that high precision control with submicron accuracy can be obtained using simple, linear controllers.

6.1.2 **Ball screw**

The ball screw removes many of the limitations associated with the acme lead screw.

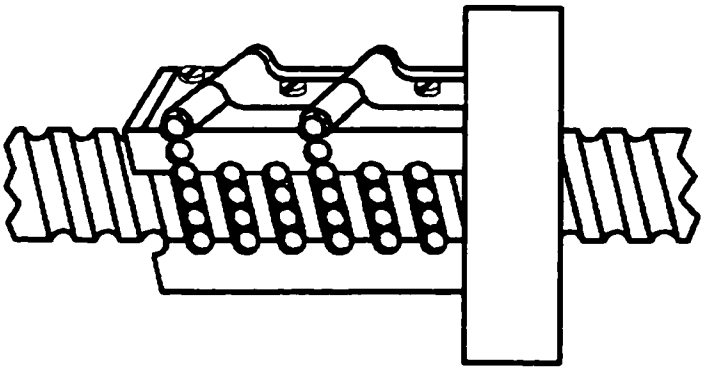


Figure 6.3 Cross-sectional view of ball screw

Figure 6.3 shows a cross-sectional view of the ball screw. It uses a ball nut which moves over a threaded screw that uses one or more circuits of recirculating steel balls rolling between the nut and ballscrew threads. This results in low friction and a high transmission efficiency close to 90 %. Due to this high efficiency, ball screw systems are used in a majority of linear positioning applications. Low cost rolled ball screws are used for low accuracy actuators, while high accuracy, high cost ground ball screws are used for high precision positioning applications. The presence of the threaded screw retains the critical speed limitation of the acme lead screw. In addition, the operational speed is also limited by the maximum recirculation speed of the ball nut circuit. Due to these

characteristics, ball screw systems are used for high accuracy, moderate speed applications that require a high duty cycle and moderately high travel lengths.

6.1.3 **Timing Belt with Pulley**

Belt drive systems generally use a steel reinforced, polyurethane timing belt along with 2 pulleys to convert rotary motion to linear motion. Due to belt elasticity, the system is non-linear which results in much lower accuracy, repeatability and dynamic performance

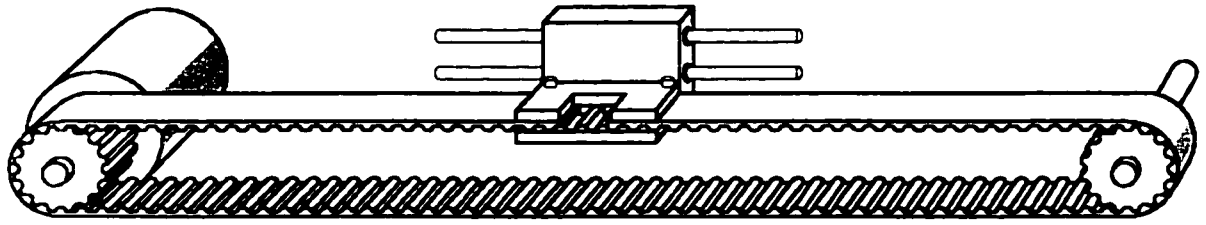


Figure 6.4 Belt drive system

than rigid systems when used with conventional controllers. Figure 6.4 shows a typical belt drive system consisting of a timing belt, 2 pulleys, an actuator motor and a positioning stage, which is supported on a linear bearing. Timing belts generally provide more linear motion for the same motor rotation as compared to screw driven systems. This also results in a reduction in positioning accuracy. Belt drive systems have high efficiency with few moving parts and do not have the critical speed limits associated with screw driven systems. They can also be used for stages with large travel lengths in contrast to screw driven systems. Their thrust capability is smaller than screw driven systems and they do not have load hold-off capability with motor power turned off. These characteristics make belt drives ideal for high speed, high duty cycle, low precision and low thrust applications that require long travel lengths.

Other important characteristics of transmission drive systems are the amount of backlash and wear involved in the use of each system. Backlash in belt drive systems is found in the reduction gear used to couple the motor to the pulley as well as at the belt-pulley interface. An increase in belt elasticity can dynamically change the amount of backlash

Table 4 Comparison of rigid and elastic transmission methods

Characteristic	Acme Screw	Ball Screw	Belt Drive
Repeatability using 0.5 μm linear encoder	0.5 μm	0.5 μm	200 μm
Maximum speed	0.2 m/s	0.4 m/s	2 m/s
Mechanical Efficiency	50 %	85-90 %	90 %
Operating Noise	Low	High	Moderate
Friction	High	Moderate	Low
Maximum Travel Length	500 mm	500 mm	1500 mm
Cost	High	Higher	Low
Life (Mechanical Wear)	Low (High)	High (Low)	High (Low)
Shock Load Capacity	High	Moderate	Very low
Dynamic Characteristic	Linear	Linear	Non-linear

experienced. Aged belts can be easily replaced at low cost, so that aging is not very important in belt drives. Backlash can also increase with age due to wear and tear. For lead screws, backlash is mainly seen between the screw and the nut which increases with age as the nut and screw surfaces deteriorate, while the backlash associated with ball screw systems remains substantially constant despite aging. Table 4 shows typical values for most of the characteristics of the three transmission methods from which the range of application of each method can be easily visualized.

6.2 Sensors and Actuators

Apart from the transmission method, the choice of the motor driving the linear positioning device and the sensors measuring the motor and stage position are important factors that determine the positioning performance. Three types of permanent magnet motors are commonly used for high precision positioning applications: DC motors, stepper motors and brushless DC motors. DC motors have the advantages of low cost and good reliability. However, the presence of brushes increases the maintenance requirements, while the commutator limits the maximum motor speed before the current arcs over adjacent commutator segments. Since the armature (rotor) winding carries the torque producing current, all heat is generated on the rotor. This must travel across the air gap and through the stator so as to be dissipated. This long thermal path results in a motor that is less efficient motor and hence larger in size than the brushless DC motor. The motor inertia is also higher than brushless DC motors. The typical torque control bandwidth available with DC motors controlled by an H-bridge power electronic circuit is around 2.5 KHz.

Stepper motors use variation in reluctance and discrete voltage pulses to move in small steps. Typical *full-step* sizes are 1.8° or 0.9° rotation of the motor shaft. This can be reduced by using microstepping, resulting in minimum incremental motion of as low as 0.0072°. Stepper motors are generally used in low power, open loop applications due to the low dynamic response and limits on the slew rate of pulses that can be applied to the motor. The advantages are low cost and maintenance as well as high torque at low speeds. These motors are generally used in lower cost, low accuracy applications using open loop control and requiring short, rapid motion with high acceleration.

Brushless servo motors are constructed with the windings on the stator and permanent magnets on the rotor. Most of the heat is generated on the stator and can be easily dissipated. This results in a compact motor as compared to the DC motor. Low inertia and absence of the commutator-brush assembly results in higher speed and acceleration capabilities. The stator windings carry alternating current with a frequency proportional

to motor speed. To obtain optimum torque, the stator current has to be controlled based on the motor shaft position, so as to keep it orthogonal to the flux produced by the rotor magnets. This is generally done by sensing motor position using either a rotary encoder

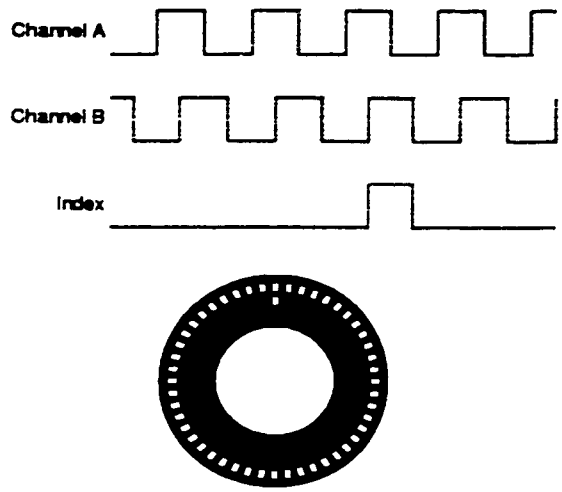


Figure 6.5 Quadrature rotary encoder

or a hall effect sensor. A coarse hall effect sensor can be used since the switching requires knowledge of the motor shaft position to the nearest 30° interval. Alternating current is applied to the stator winding using a 3 phase inverter, which converts a DC voltage source to a variable frequency, variable voltage, 3 phase AC source. The motor position sensor and 3 phase inverter add complexity and cost to the drive electronics. The total cost of the motor and drive electronics package is higher for brushless DC motors as compared to DC motors. Hence, brushless DC motors are used for higher cost, high accuracy systems which require high speeds and acceleration. The torque control bandwidth for a brushless DC motor can be as high as 4 KHz with a PWM inverter switching at 20 KHz.

To implement high accuracy closed loop position control, either the motor position or the linear position of a stage have to be sensed and fed back to the controller. Optical rotary encoders are used for sensing angular position of the motor shaft. These sensors produce

two square wave signals that are 90° out of phase (or in quadrature). By counting the number of pulses or pulse edges, the change in angular position can be determined, but these encoders do not have any means of sensing the absolute angular position. Hence, they are called incremental quadrature encoders. Figure 6.5 shows an example of the encoder disc used and typical signal waveforms. Also included is an index pulse generated once every motor rotation. This pulse is used to correct the absolute position obtained in software from the incremental signals. The number of pulses per revolution determines the resolution of an encoder when used with a quadrature decoder, which is a circuit that captures each edge of both the encoder square wave signals and determines the change in angular position.

High precision positioning systems require the use of a linear encoder in addition to or in place of a motor shaft encoder, so as to correct positioning errors *downstream* of the motor. Optical linear encoders use the same principle as rotary encoders but use a linear optical scale instead of the rotary disc, and the typical quadrature resolution is 0.5 μm. Linear encoders are available with lengths extending to more than 1.5 m, although their cost increases rapidly with the length. When lead screw or ball screw driven systems are used with linear encoders, the absolute accuracy obtained is only limited by the resolution of the encoder, since the system is linear except for the presence of non-linear friction. Continuing research in the development of other encoding technologies has produced lower cost, capacitive and inductive linear encoders. These encoders have better dynamic characteristics, giving a higher operating speed than optical encoders, and also have the ability to encode and transmit absolute position information [48]. It is expected that further research will enable the use of such encoders in industrial products.

6.2.1 **Collocated and Non-Collocated Control**

Based on the location of the actuator (motor) and the sensor (encoder), two types of controllers are defined. Collocated control refers to a system which uses feedback of motor position so that the actuator and sensor are located at the same place (or

collocated). On the other hand, if a control system uses feedback of the load position from the linear encoder to control a motor located at one end of a linear travel stage, it is called non-collocated control. These terms are mainly used in aeronautics when dealing with flexible space structures and their control [49]. Currently high precision, rigid transmission systems use two encoders: a rotary encoder to measure the motor position and a linear encoder to sense the load position. A linear controller is then used using motor position feedback, and the load position information is used to compensate for errors downstream of the actuator. This two encoders approach adds additional cost which can be avoided if non-collocated control is used.

6.3 Research Objectives

From the foregoing description of transmission methods, it can be seen that rigid link drives with load position feedback are used for precision position control, while elastic link drives such as belt drives are limited to low accuracy applications. The inability to obtain high accuracy positioning using belt drives is mainly due to the deficiencies in the linear control algorithms used today, which are unable to satisfactorily compensate the system non-linearities. There are many applications which need positioning accuracy of 1 to 10 μm for longer travel lengths and higher speeds than those using lead screws or ball screws. If suitable control algorithms can be developed for belt drive systems to achieve positioning accuracy in the 1-10 μm range, these systems can be used for many applications at significant cost savings. The non-linearities affecting belt drive systems are the belt elasticity, friction at the linear bearing and backlash between the belt and the driven pulley. All these processes are complex and cannot be modeled easily. In addition, variation in belt properties and tension can cause drastic changes in system dynamics, which makes the application of any model-based control difficult. In light of this, the objectives of this research are:

- Study the characteristics of belt drive systems and identify the challenges present in controlling them.

- Develop a model independent non-linear controller that can achieve positioning accuracy similar to the screw driven systems.
- The developed controller should preferably use only load position information obtained from a linear encoder, thus implementing non-collocated control.
- The controller to be developed should be able to self-tune itself so as to adapt to changes in system dynamics
- Layout a systematic design approach that can be applied to any belt drive system and that can be used for development of practical products.

Table 5 Control objectives for belt drive system development

Parameter	**Specification**
Travel Range	Up to 1m
Positioning repeatability	±10μm
Positioning accuracy	±10μm
RMS track following error	±50μm
Maximum Speed	1 m/s
Acceleration	5 m/s^2

It is desirable to quantify the objectives in terms of desirable positioning accuracy and repeatability. These are tabulated in Table 5, where the most important specifications are the high repeatability and accuracy desired. Also important are the travel length and maximum speed at which the tracking error is to be achieved. For development and testing of different control algorithms, two test setups were built. The next chapter describes these setups including the hardware and software platform. A general-purpose mathematical model for belt drive system is derived. The challenges present in controlling such systems are identified, followed by an analysis of different control strategies.

Chapter 7 Test Setup and Model

Two test setups were built to study the efficacy of different control algorithms for belt drive systems. The first setup was a single axis stage that was used to establish the feasibility of using belt drives for precision positioning. To extend the results obtained from that feasibility study and to alleviate some of the problems associated with faulty bearings, a two axis X-Y setup was built. All the work on the X-Y setup, described here, has been done on the X-axis and hence the model derived here applies equally to both stages.

Figure 7.1 shows a simplified view of the single-axis test setup. The motor shaft is

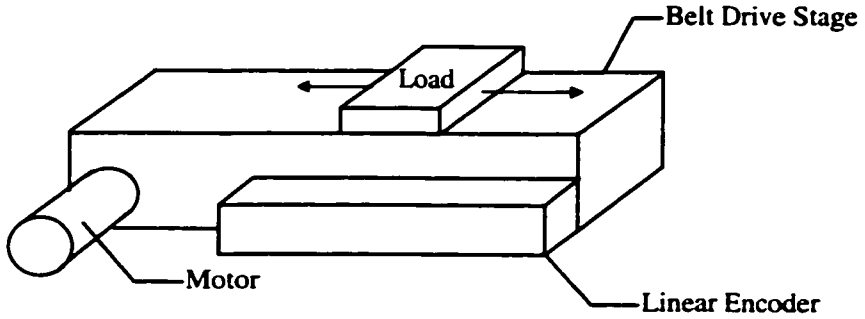


Figure 7.1 Single-axis test setup

connected to the driven pulley through a reduction gear box. The driven pulley drives the belt that moves the load plate. The load plate is supported by a linear bearing over the length of the stage and its position is sensed by an optical linear encoder. The scale of the linear encoder is attached to the stage and its read head moves along with the load plate. The linear encoder is manufactured by Micro Encoder, Inc. and has a resolution of 0.5 μm when used with a quadrature decoder. The motor used is a brushless DC motor manufactured by Aerotech, Inc. and is rated at 0.8 HP, 4000 rpm with a maximum continuous torque rating of 1.94 N-m. The motor torque is controller by a 3-phase servo amplifier, current controlled, PWM inverter. The belt drive stage, also manufactured by

Aerotech, uses a steel reinforced, polyurethane timing belt and the stage has a maximum travel length of 1.074 m.

Figure 7.2 shows a schematic top view of the X-Y setup. It has two brush DC motors, one for each axis. The X-axis has two belts driven by the same motor: one directly and

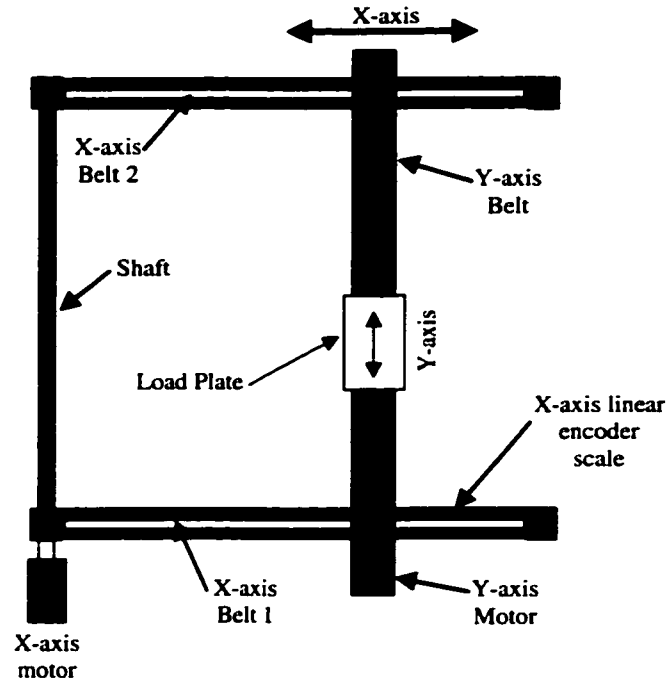


Figure 7.2 Top view of X-Y table

the second through the shaft. When the X-axis motor rotates, the vertical beam moves on two linear bearings located on the X-axis belt housings. The Y-axis has one belt and the rotation of the Y-axis motor moves the load plate as shown in the figure. The Y-axis motor is located below the main beam and hence is hidden in the figure. Both axes have linear encoders having a resolution of 0.5 μm when used with a quadrature decoder. These encoders are manufactured by Renishaw, Inc. and the stage is manufactured by Industrial Devices Corporation. The DC brush motors are controlled by H-bridge, servo amplifiers, PWM, DC-DC converters. Both motors and the servo amplifiers are made by Cleveland Motion Controls, Inc. The X-axis motor has a rating of 1 HP at 4500 r/min while the Y-axis motor is rated at 0.4 HP at 4500 r/min.

7.1 Control Setup

The control system is implemented on a Pentium-II based PC running Windows NT using a real-time kernel called RTX made by VenturCom Inc. The kernel sits between the hardware and the Windows NT OS enabling it to provide soft real-time performance. The software is developed in C and C++, which enables it to be easily ported to any hardware platform. The hardware platform and the software development are described in detail in next sections.

7.1.1 **Hardware Components and Signal Flow**

Figure 7.3 shows a simple signal flow block diagram of the test setup. The controller, running on a PC, gets the stage position y from the linear encoder and a desired position

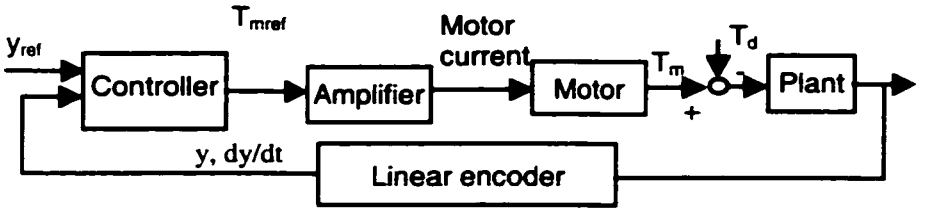


Figure 7.3 Block diagram of test setup

y_{ref}. It uses these to produce a motor torque reference command T_{mref}, that is fed to the servo amplifier. The servo amplifier controls the motor torque by controlling the motor current(s). The linear encoder is interfaced with the PC using an ISA bus encoder interface plug-in card made by Comark Corp., which provides quadrature decoding. This board requires a minimum encoder pulse width of 1 μs for reliable counting, which corresponds to a maximum stage speed of 1 m/s. The controller relies on feedback from the linear encoder to determine the stage position. If the stage speed goes above 1 m/s, the encoder interface board will miss counts, resulting in an increase in the positioning error which results in a higher torque reference command from the encoder. This effect gets cascaded, resulting in a runaway condition, which should be avoided at all costs.

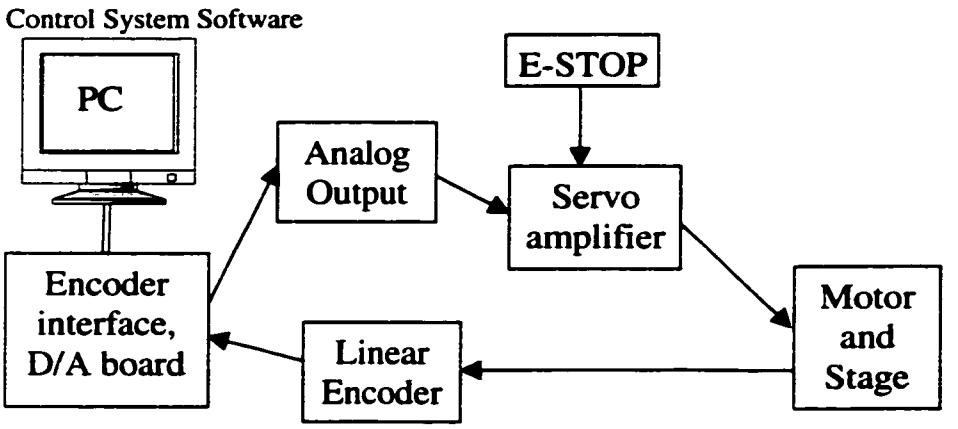


Figure 7.4 Block diagram of control hardware

Thus, the stage is operated at speeds well below 1 m/s and the desired position command, y_{ref} is rate limited to increase at a maximum speed of about 0.75 m/s. The servo amplifier requires a differential analog input as the torque reference command. The encoder interface board also has on-board 12-bit D/A converters and one of them is used to provide this analog command output. An emergency stop switch is used to disable the servo amplifier to provide manual protection to the stage. Figure 7.4 shows a block diagram of the hardware with arrows showing the direction of signal flow.

7.1.2 **Software Setup**

The control software loop needs to run in real time and execute at a fixed frequency of at least 1 kHz with low latencies, so as to update the torque reference. This can only be achieved by using a real-time operating system. Initially, the MS-DOS operating system was used to allow for hardware timer interrupts to be serviced with low latencies. However, the user interface for the control software left much to be desired since DOS graphics libraries had to be used. DOS also has other serious limitations: it is not re-entrant, it does not support multiple processes or threads and DOS programs have a

640 KB memory limits when working in the real memory mode. These limitations made it difficult to test different control algorithms.

To overcome these limitations, a real-time kernel for Windows NT known as Real Time Extension (RTX) for NT, made by VenturCom Inc. is used as the software platform [50]. This kernel acts as a bridge between the hardware and Windows NT. It runs a real-time

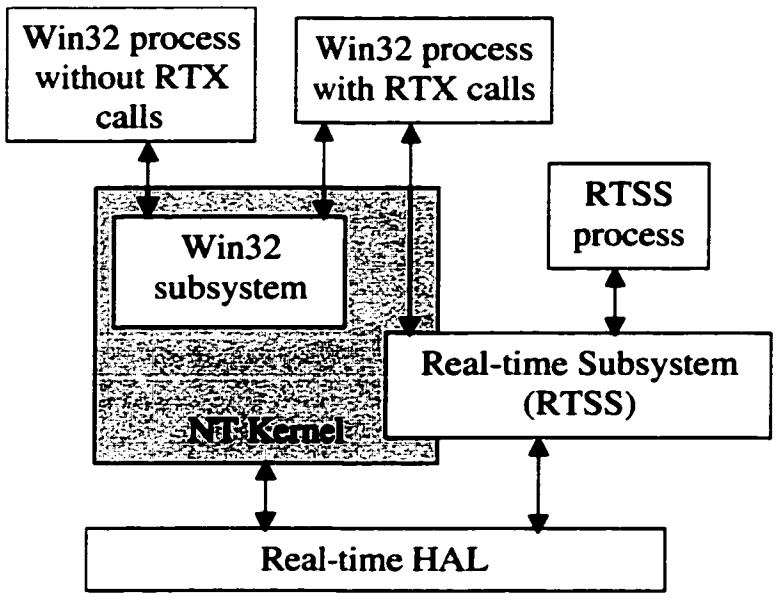


Figure 7.5 RTX architecture and information flow

subsystem (RTSS) which has its own real-time process scheduler that supports multithreaded real-time processes with 256 priority levels. The windows architecture does not guarantee any real-time servicing of process. RTX achieves this by replacing the Windows NT Hardware Abstraction Layer (HAL) with a real-time HAL. All RTSS processes are executed first and control is passed to Win32 processes only when all active RTSS processes are blocking or waiting. RTSS binaries interact with the desktop using a text based interface which is piped through Windows NT.

The real power of the kernel lies in the fact that Win32 processes can be run simultaneously with RTSS processes and the two types of processes can communicate with each other via real-time Inter-Process Communication (IPC) objects which can be manipulated either by RTSS or Win32 processes. These objects, which include semaphores, events, mutexes and shared memory, use a common namespace that is

shared between the Win32 and the RTSS subsystem. RTSS processes have a rich Application Programming Interface (API) which includes support for most of the functions in Microsoft C compiler's standard C run-time library. In addition, Win32 processes can call functions from the RTX API. This allows for using the RTSS process to perform real-time tasks while the Win32 process can handle the user interface, exchanging information with the RTSS process. The architecture and interaction between RTSS and Win32 is shown in Figure 7.5. The use of a Windows based real-time software platform enables rapid prototyping and testing of different control algorithms and strategies.

7.2 System Model

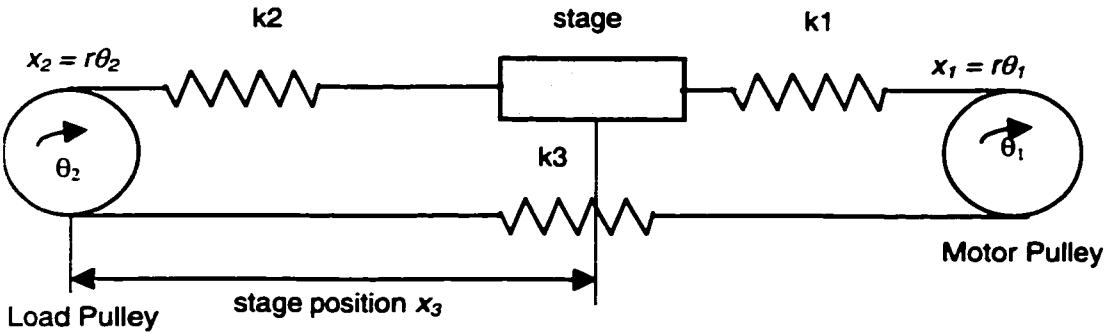


Figure 7.6 Model of the belt drive system

In order to gain some insight into the characteristics and functionality of belt drive systems, a state model is developed. Figure 7.6 shows a simple diagram used for developing the model [51]. The position of the driving (motor) pulley is shown as angle θ_1 and the linear position at the pulley circumference is denoted by x_1. Similarly the position of the load pulley is shown as angle θ_2 and its linear circumferential position is denoted by x_2. The stage position is denoted by x_3 and the elasticity in the three sections of the belt is shown using springs $k1 - k3$. This model can be further analyzed using the free body diagrams at both pulleys and at the load as shown in Figure 7.7. The tension in

each section of the belt is denoted by F_{bi} where $i = 1,2,3$ for the sections denoted by spring constants k1, k2 and k3 in Figure 7.6.

At the motor end, we have:

$$r\,T_m + (F_{b3} - F_{b1})\,r^2 = I_1\,d^2x_1/dt^2 + F_m(dx_1/dt) \qquad (7.1)$$

where, T_m is the motor torque; I_1 is the moment of inertia of the motor, coupling shaft and the pulley; F_m is the friction at the motor which is a function of the motor speed; r is the

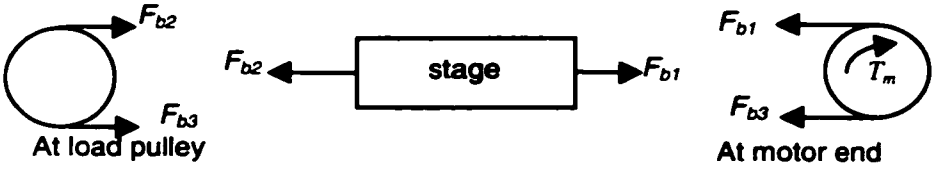


Figure 7.7 Free body diagrams

radius of the pulley and F_{b1} and F_{b3} are the tensions in the slack and tight side of the belt.

At the stage:

$$F_{b1} - F_{b2} - F_s(dx_3/dt) = m\,d^2x_3/dt^2 \qquad (7.2)$$

where m is the load, mass, inertia etc. on the stage and F_s is the stage friction as a function of stage speed.

At the end pulley, there is no external force and if the friction is neglected, we get:

$$I_2\,d^2x_2/dt^2 = r^2\,(F_{b2} - F_{b3}) \qquad (7.3)$$

where I_2 is the inertia of the load pulley.

The tension in each section of the belt is given by:

$$F_{b1} = \frac{AE}{\left(\frac{L}{2} - x_3\right)}(x_1 - x_3) \qquad (7.4)$$

$$F_{b2} = \frac{AE}{\left(\frac{L}{2} + x_3\right)}(x_3 - x_2) \qquad (7.5)$$

$$F_{b3} = \frac{AE}{(L + x_1 - x_2)}(x_2 - x_1) \tag{7.6}$$

where L is the length of the belt between the two pulleys, A is the cross sectional area of steel in the belt and E is the Young's coefficient for steel [51]. These equations are obtained by assuming that the belt spring constant is inversely proportional to the *instantaneous* supported length of the belt (from the stage to either pulley or between two pulleys) and that there is no additional dynamics involved. Also the stretch in a belt section is assumed to be equal to the difference between the motion at either end of the particular section i.e. effectively the belt is assumed to be three unconnected sections with the *unstretched* length of each section varying with stage position. Actually the dynamics can be much more complicated since the belt stretch can carry over the pulley causing the tension in the tight side to reduce and the tension in the slack side to increase. Also the interaction between the belt and the pulley teeth, which may include some sliding is not modeled. There may be additional dynamics due to the belt in the form of a damping force that is proportional to the stage speed. However, it is difficult to model such a force since its dependence on the position of the stage is not known.

The above equations can be grouped in state space form to obtain:

$$\begin{aligned}
\dot{x}_1 &= x_4 \\
\dot{x}_2 &= x_5 \\
\dot{x}_3 &= x_6 \\
\dot{x}_4 &= \frac{r}{I_1}[(F_{b3} - F_{b1})r + T_m - F_m(\dot{x}_1)] \\
\dot{x}_5 &= \frac{r^2}{I_2}(F_{b2} - F_{b3}) \\
\dot{x}_6 &= \frac{F_{b1} - F_{b2} - F_s(\dot{x}_3)}{m}
\end{aligned} \tag{7.7}$$

where F_{b1}, F_{b2}, F_{b3} are the tensions in different sections of the belt as given by equations 7.4, 7.5 and 7.6 and $F_m(\bullet)$ and $F_s(\bullet)$ are functions representing frictional forces at the motor and the stage, respectively.

This is a sixth order model which has the position and speed of the two pulleys as well as the position and speed of the stage as its states. The output of the plant is the stage position (x_3). In comparison, a model of a rigid system using a lead screw has only two states: the stage position and speed. Due to the elasticity of the belt, a linear, second order model has expanded to a non-linear, sixth order model.

7.2.1 **Friction Model**

The stage and motor friction varies as a function of stage velocity. The general form of the variation is shown in Figure 7.8 [51].

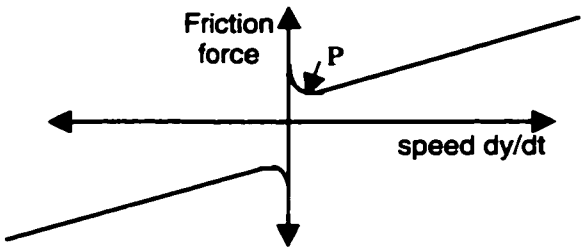


Figure 7.8 Variation of stage friction with speed

Three components of the frictional force can be readily identified. These are:

- an exponential decaying part due to static friction or stiction known as the Stribeck effect [52] where the frictional force reduces as the speed increases till the point P, in Figure 7.8
- a constant frictional force due to Coulomb friction, and
- a viscous force which is proportional to the speed.

A proper model for low speed operation also includes stick-slip effects where some hysteresis is observed. A detailed analysis of friction models is given in [52], while [53] deals with modeling of the effects of backlash. Backlash is more important for systems having a gear reducer between the motor and the belt pulley. The X-Y test setup uses a direct drive motor so that backlash exists only between the pulley and belt teeth. A seven

parameter friction model [52] is widely accepted as fairly representative of the actual friction dynamics and is reproduced here:

$$F_f = F_v\dot{x} + \left(F_C + (F_{ST} - F_C)\left(\frac{1}{1+\left(\frac{\dot{x}(t-\tau_l)}{\dot{x}_s} \right)} \right) \right) \mathrm{sgn}(\dot{x}) \qquad (7.8)$$

where, F_C is the Coulomb friction constant, F_v is the viscous friction constant, F_{ST} is the static friction which is modeled separately, $\dot{x}_s$ is the velocity of the Stribeck friction, which represents the point P in Figure 7.8, τ_l is a time delay between a change in speed and a corresponding change in friction at low speeds. Sgn() is the signum function which evaluates to 1 if its argument is positive, -1 if it is negative and zero if it is zero. The static frictional force can be modeled as a constant number. However, it is observed that the stiction increases with the increase in the time that the two bodies (between which the friction is modeled) are at rest relative to each other. This time is known as the dwell time and is denoted by t_w and the rising static friction with dwell time is given by:

$$F_{ST} = F_{ST,\infty} - (F_{ST,\infty} - F_C)e^{-\gamma t_w^{\zeta}} \qquad (7.9)$$

where $F_{ST,\infty}$ is the stiction for a very high dwell time and γ and ζ are empirical constants that define the variation of stiction with dwell time.

Equations 7.8 and 7.9 define the seven-parameter friction model. In practice, the frictional force is seen to be different in different directions of motion, so that one set of model parameters has to be used for each direction of motion giving a total of 14 parameters for modeling the friction at each point where it is present.

7.2.2 **Simplified Model**

Apart from the uncertainty in the friction terms, the state model has the position and velocity of the load pulley as two of its states. It is not possible or desirable to use sensors to obtain these states. These could however, be estimated using state estimation

if all the other states are available and the model has been accurately identified, which is a problem in itself. Hence, it is desirable to obtain a simplified system model that does not use the load pulley position and velocity as its states.

This is done by making the following assumptions:

- the inertia of the load pulley is assumed to be negligible, and
- the load pulley is assumed to be frictionless

With these assumptions, the tension in the belt on both sides of load pulley has to be equal, since any inequality will give infinite acceleration of the zero-inertia load pulley. This gives $F_{b2} = F_{b3}$ in Figure 7.7 and results in a simplified state model given by equation 7.10. It should be noted that the definitions of F_{b2} and F_{b3} are now changed.

$$\begin{aligned} \dot{x}_1 &= x_4 \\ \dot{x}_3 &= x_6 \\ \dot{x}_4 &= \frac{r}{I_1}\left[(F_{b2} - F_{b1})r + T_m - F_m(\dot{x}_1)\right] \\ \dot{x}_6 &= \frac{F_{b1} - F_{b2} - F_s(\dot{x}_3)}{m} \end{aligned} \tag{7.10}$$

The assumption also implies that the belt tension in the slack side is determined by the length of the belt from the motor pulley to the stage around the load pulley. Thus, F_{b2} and F_{b3} are now given by:

$$F_{b2} = F_{b3} = \frac{AE}{\left(\frac{3L}{2} + x_3\right)}(x_3 - x_1) \tag{7.11}$$

It can be seen that the simplified state model has four states instead of six and does not depend on information about the load pulley.

7.3 Control Problem

It can be seen from the developed model that the belt drive system is a non-linear plant due to the presence of belt elasticity and friction. If model based linear control is to be applied, the non-linear model has to be linearized around some operating point. Since the belt elasticity depends on the length between the stage and either pulley, the stage position affects the non-linearity. Hence, the only way that the model can be linearized is around a given stage position. This is not very helpful because a fixed stage position would render the belt drive stage useless or in other words, there is no suitable equilibrium point around which the system can be linearized. Thus, application of a model based, linear controller is ruled out for this system.

7.3.1 **Model based non-linear control**

The suitability of a model based, full state feedback, non-linear controller can be readily analyzed. The system is *feedback linearizable* if full state information is available [54] since it can be written in the form:

$$\begin{aligned} \dot{x} &= f(x) + g(x)u \\ y &= h(x) \end{aligned} \tag{7.12}$$

This requires the use of a high-resolution shaft encoder at the motor pulley. The required resolution of such an encoder can be easily derived from the specific elasticity of the belt and the torque rating of the motor. For the X-Y setup, the belt used is of type AT5 with has a specific stiffness of 47950 lb/in or 8.3969 N/micron of belt stretch. This force is 10% of the maximum force that can be applied by the motor. Thus, any practical model based controller will need a motor shaft sensor with a resolution of at least 1 μm along the belt. This corresponds to a resolution of 38000 lines per revolution, while standard high accuracy rotary encoders have a resolution of 4000 lines per revolution. This solution is hence impractical and infeasible. A better alternative would be the use of state estimators to obtain the angular position and speed of the motor pulley. This requires the

plant to be observable, which is not the case. Whenever the stage is stationary and a force is applied by the motor, there is a duration where the belt stretches to produce the tension required to move the stage. In this period, the motor pulley states are unobservable. Similarly, the presence of friction at the motor pulley renders the model unobservable.

Given the above difficulties, both linear and non-linear, model-based approaches are not practicable for this system.

7.3.2 **High gain PD or PID control**

Many researchers have used high gain PD control for precision positioning of rigid link systems [57]. Since rigid link systems can be effectively represented by a second order linear equation, PD control is equivalent to linear full state feedback control. Lim *et al* [55] use a PD controller with a disturbance observer. The disturbance observer includes a low pass filter, which effectively adds integral control to the PD full state feedback control. The authors note that this method works well without a system model but it is valid only for linear systems. In addition, it is well known that PD or PID control is not robust to changes in plant parameters or operating conditions [56].

Very little work has been reported in published literature on high precision control of belt drive systems. Li and Rehani [58] use a high gain PD control scheme with friction compensation for control of a belt drive stage. The authors achieve tracking errors of about 300 μm which does not compare with the range of accuracy obtained by the rigid link systems. In addition, the authors report their results based on a motor shaft encoder, effectively neglecting any belt elasticity effects, which renders their results practically meaningless.

7.3.3 **Fuzzy PD or PID control**

Compared to the experiences reported by researchers using high gain, linear PD or PID control, fuzzy control offers a better alternative. Fuzzy controllers are non-linear controllers designed based on heuristic knowledge of the plant characteristics [59]. Other advantages include:

- they do not need model information and hence are robust to plant parameter variations and noise [60]
- expert knowledge of the plant can be incorporated in the rule base.
- it is easy to tune fuzzy controllers and many techniques exist for developing adaptive fuzzy controllers [61].

Thus, a fuzzy controller is a tunable, model independent, non-linear control which seems to be an ideal candidate for a belt drive system.

7.3.4 **Other non-linear controllers**

Sliding mode control [62] is also an attractive method for non-linear control which does not require exact model information. It has been successfully applied to many non-linear problems such as brush and brushless DC motor control [63-64]. However, sliding mode control exhibits chattering, when controlling the system around the sliding surface. This problem is generally solved by using a boundary layer region near the sliding surface. In this region the control action is proportional to distance from the sliding surface, instead of switching between positive and negative maximums. Such a sliding mode controller is equivalent to a fuzzy controller [65] for second order systems. In fact, sliding mode control principles can be used for the design of a fuzzy control based on this similarity.

The above analysis shows the suitability of using fuzzy control for belt drive systems and this approach is pursued further. The next chapter describes the development of the control algorithm and the experimental results obtained.

Chapter 8 Control System

The development of a non-linear controller for belt drive systems is described in detail in this chapter. The controller uses two computational intelligence techniques: fuzzy logic and evolutionary programming. This chapter starts with the description of a standard test used for comparing performance of different controllers. This is followed by the design of a fuzzy feedback controller which includes a fuzzy PD controller, a fuzzy PI controller and a fuzzy friction and elasticity compensator. The results obtained using the feedback controller are discussed and the need for a feedforward controller is established. The results of the application of two techniques for the design of a feedforward controller are presented. The chapter ends with a summary of the results and enumerates the steps to be taken for designing a control system for any belt drive system.

A brief introduction to fuzzy control and evolutionary programming is given in the Appendix.

8.1 Testing and Track Generation

In order to test the performance of any control system to be developed, a standard set of tests are necessary based on the control objectives enumerated in Table 5. The desired control objectives are focused on low steady state positioning error with a high degree of repeatability. Also important are the maximum travel speed achieved and the RMS tracking error for a desired position track. To test all these factors, a step-like position

Table 5 Control objectives for belt drive system development

Parameter	Specification
Travel Range	Up to 1m
Positioning repeatability	±10μm
Positioning accuracy	±10μm
RMS track following error	±50μm
Maximum Speed	1 m/s
Acceleration	5 m/s^2

track is used with the following variables:

- desired change in position which is called the step size (mm)
- the maximum speed allowable for the position track (m/s), and
- the maximum acceleration allowed for the position track (m/s^2)

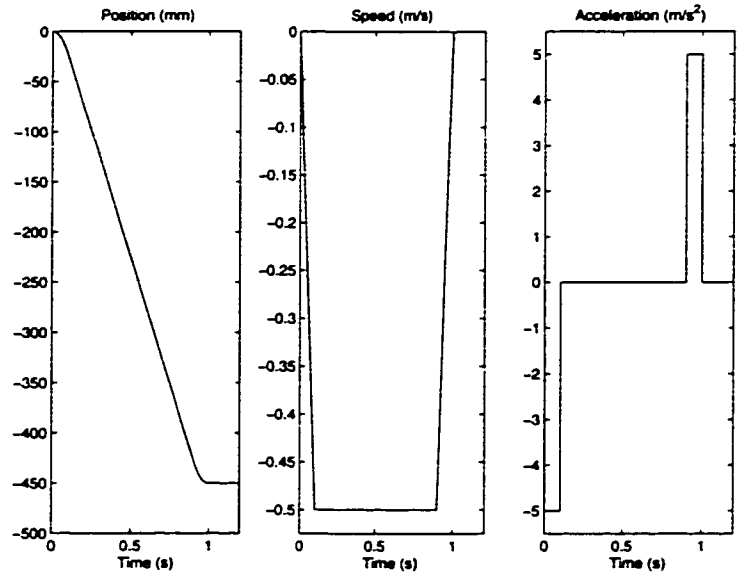


Figure 8.1 Step-like position track with speed and acceleration limits

Given these three variables, a smooth position track can be generated which is limited in speed and acceleration. Such a track provides a reasonable basis for testing the position tracking performance of any controller. Various step sizes ranging from 250 mm to 600 mm are used and the track is limited to different speeds and accelerations, ranging from 0.1 m/s to 0.75 m/s and 1 m/s^2 to 5 m/s^2. Testing over all different combinations of step size, the speed and acceleration give a good indication of the performance of the controller. Tests are performed for motion in both directions so as to test the bi-directional performance and repeatability of the controller. Figure 8.1 shows a track with a step of –450 mm, a maximum speed of 0.5 m/s and a maximum acceleration of 5 m/s^2. Although the control objective has a desired maximum speed of 1 m/s, due to the limits imposed by the quadrature decoder PC plug-in card, the stage has to be run well below

1 m/s. Due to this limitation and for safety, the track generator algorithm is set to limit position tracks to a maximum speed of 0.75 m/s.

8.2 Design of Fuzzy PD controller

Consider the model of the belt drive system developed in the previous chapter. If the belt elasticity and stage friction are neglected, the model reduces to a linear, second order system with the stage position and speed as the states. This simple rigid body dynamics can be represented by:

$$m_e \ddot{y} + B\dot{y} = F_m \tag{8.1}$$

where, m_e is an equivalent system mass, B is a damping term and F_m is the force applied by the motor.

The output of this system is the position y, which implies that the plant has a pole at the origin. For such a plant, the optimal controller is a linear PD controller. When the belt elasticity and stage friction are considered, a fuzzy PD controller gives a non-linear controller with characteristics similar to a variable gain PD controller. This was chosen as a first attempt at developing the controller. The fuzzy PD controller has two inputs:

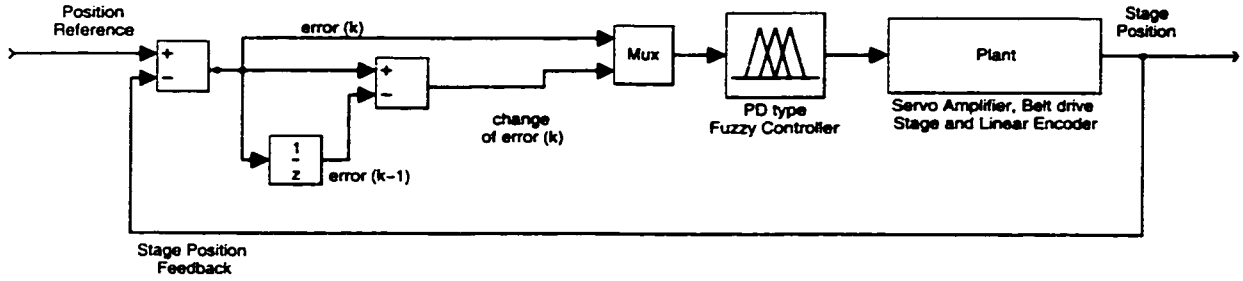


Figure 8.2 Block diagram of PD-type fuzzy controller

position tracking error, $y_{ref} - y$ and the change of this error over a sampling period. The sole output is a motor torque reference, u_{PD}, that is fed to the servo amplifier. Both inputs, error and change or error, as well as the output u_{PD} are assigned seven fuzzy sets each, namely: *LN, MN, SN, ZE, SP, MP, LP* where *L* stands for *Large*, *M* for *Medium*, *S* for *Small*, *ZE* for *Zero*, *N* for *Negative* and *P* for *Positive*. All the variables are limited to

a normalized active range of [-6, 6]. Actual error and change of error are multiplied by scaling factors called *error gain* and *change of error gain*, to obtain the normalized error and change of error which are then fuzzified. Figure 8.2 shows a block diagram of the controller.

8.2.1 **Membership functions and Rule base**

Different functions such as triangular, trapezoidal and gaussian have been suggested for defining the membership functions for fuzzy sets. Use of a gaussian function results in a smooth membership function that is useful when adapting the membership function by using gradient based methods. However, such a choice also increases the computational burden imposed by the fuzzy controller. Hence, triangular and trapezoidal membership functions are used. Trapezoidal memberships functions are used for all fuzzy sets except for *ZE*, where a triangular membership function is used. This seems to give better results than the triangular membership functions for all sets. All the three fuzzy variables (2 inputs and 1 output) use the same membership functions which are plotted in Figure 8.3. The centroids of the fuzzy set membership functions are located at [-5.5, -3, -1.5, 0, 1.5, 3, 5.5]. The membership functions are symmetric around zero and

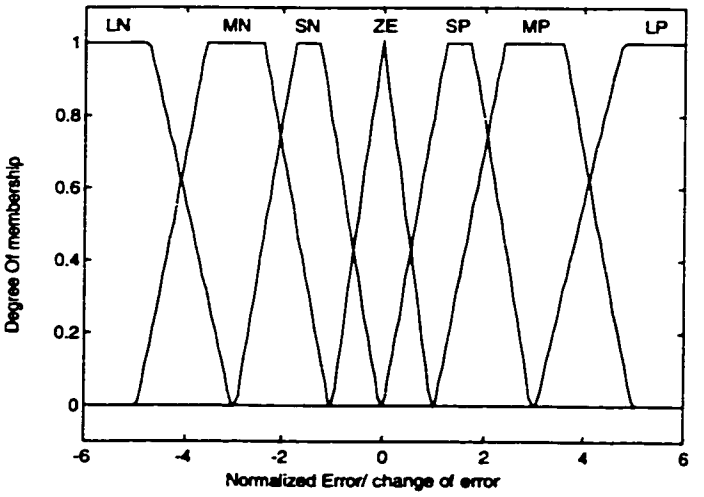


Figure 8.3 Membership functions for error, change of error and control for PD type of fuzzy controller

are designed to be narrow near zero and increase in area and width away from zero in either direction. This gives more resolution around zero where small changes in error and change of error require different response, while the resolution is coarse away from zero where the required control action is intuitive.

The rule base used for the controller is given in Table 6. The design of the rule base is intuitive – if error is LN and change of error is ZE, the error is large negative and it is not

Table 6 Rule base for the PD type fuzzy controller

		Change of Error						
		LN	MN	SN	ZE	SP	MP	LP
	LN	LN	LN	LN	LN	LN	LN	MN
E	MN	LN	LN	LN	MN	SN	SN	ZE
R	SN	MN	SN	SN	SN	ZE	ZE	SP
R	ZE	SN	ZE	ZE	ZE	ZE	ZE	SP
O	SP	SN	ZE	ZE	SP	SP	SP	MP
R	MP	ZE	SP	SP	MP	LP	LP	LP
	LP	MP	LP	LP	LP	LP	LP	LP

changing very fast. Hence, a large negative control signal (torque reference) is required to reduce the error. Using actual crisp numbers, if the reference position is 10 and the actual position is 15, then the error is large negative (in a crisp sense) and if the error is not changing, then a large negative torque is applied to reduce the actual position to 10. Similarly, if the reference position is –10 and the actual position is -5, with the change of error being close to zero, a large negative torque is necessary to reduce the actual position. On the other hand, if error is MN and change of error is LP, then the error is medium negative but is reducing very fast (change of error is in the correct direction to reduce the error), so that the controller should output a zero torque command. The rule base is tuned manually by trials and errors by observing the response of the system and then modifying the rules that are activated in the data being observed. An easy way of performing such a tuning is by observing the state trajectory of the error on a phase plane i.e. by plotting the normalized error vs. normalized change of error.

The output of the rule base is defuzzified using a Center-of-Area defuzzifier as described in the appendix. Since the areas of the membership functions are different, both the centroids and the areas have to be used in the defuzzification. The defuzzified output is a normalized u_{PD} which is then multiplied by the control gain (scaling factor). The control gain is set such that the maximum output from the controller is equal to the maximum

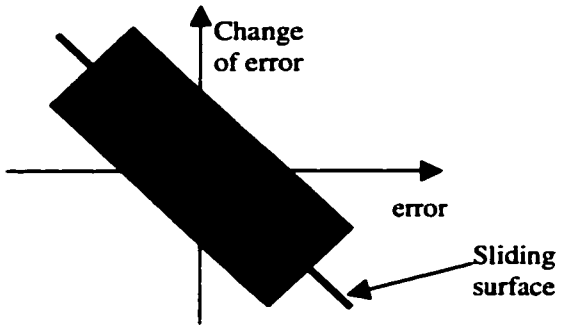


Figure 8.4 Fuzzzy Sliding Mode Control

range of the D/A converter. The maximum output of the controller is the centroid of LP or LN which is ±5.5, while the maximum digital value for the 12-bit D/A converter is 2048, giving the control gain as 372.36. The use of this value ensures that the whole range of the output is mapped to the rule base. This scaling method makes the fuzzy controller equivalent to a sliding mode controller with a band around the sliding surface where the controller output is not equal to positive or negative maximum. Such a controller has been referred to as a *fuzzy sliding mode controller* [68] and it is said to have a fuzzy sliding surface as shown in Figure 8.4. The fuzzy inference engine defines the output of the controller in the rectangular box around the sliding surface. Any operating point that lies outside the rectangular box results in a maximum positive or negative output. The controller output is positive above the sliding surface and is negative below the sliding surface. If the fuzzy rule base is observed, an imaginary surface can be drawn connecting the rules whose output is ZE. This implicitly defines a sliding surface. In fact, using a fuzzy controller, the sliding surface that is defined is non-linear which gives additional design and tuning flexibility [75].

8.2.2 **Results using fuzzy PD control**

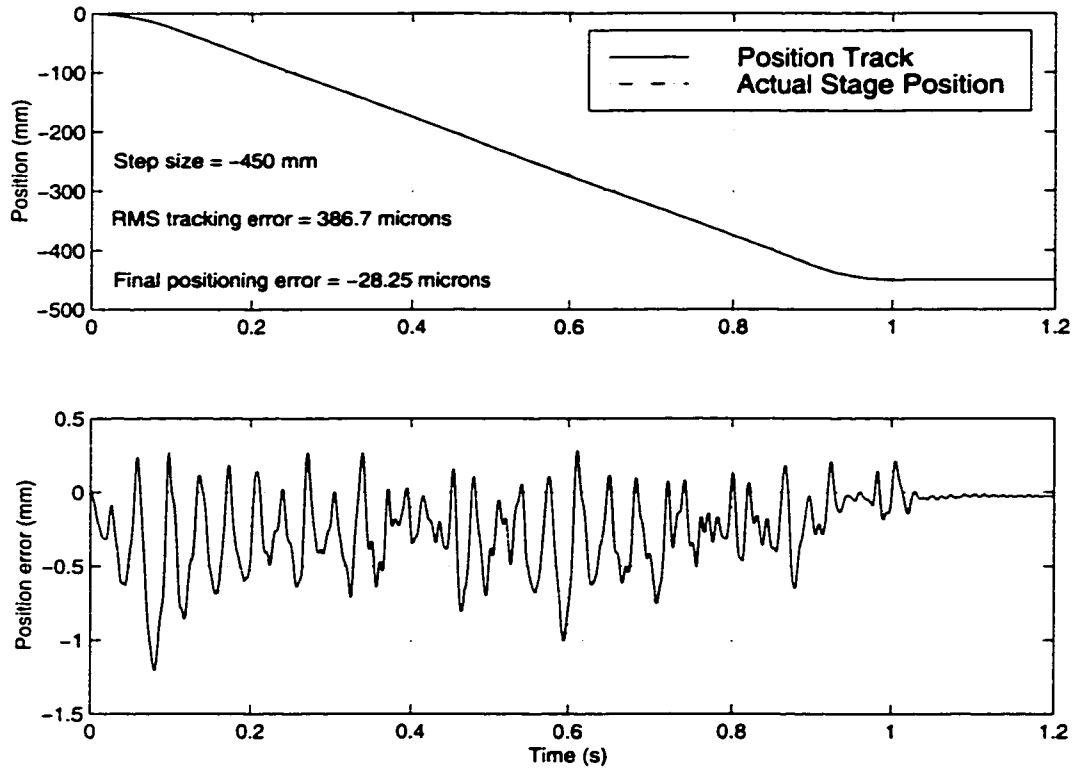


Figure 8.5 Results using fuzzy PD controller on single-axis test setup

Figure 8.5 shows the results obtained using the PD type fuzzy controller on the single axis test setup. The top plot shows the desired position track and the actual position superimposed on each. The difference in the two curves is barely discernible since the error is small as compared to the step size (displacement) which is –450 mm. The track is limited to a maximum speed of –0.5 m/s and a maximum acceleration of 5 m/s^2. The second plot shows the position tracking error in mm. The RMS tracking error for this particular test was 386.7 μm (or 0.386 mm) with a peak tracking error of about 1.2 mm. The final positioning error is –28.25 microns. For a range of tests carried out by varying the step size, maximum speed and maximum acceleration, the RMS tracking error varies between 300 and 400 μm, while the final positioning (or steady state) error varies between 25 and 90 μm. Although the tracking and steady state errors are high, these results establish the feasibility of using belt drives for high precision control. It was

observed that the single-axis stage had a very high bearing friction as well as a belt with a low specific elasticity (Force per micron of stretch in the belt). These two factors caused a high opposing force against any initial motion of the stage, i.e. the belt elasticity implicitly increased the static friction effect and caused high final positioning errors.

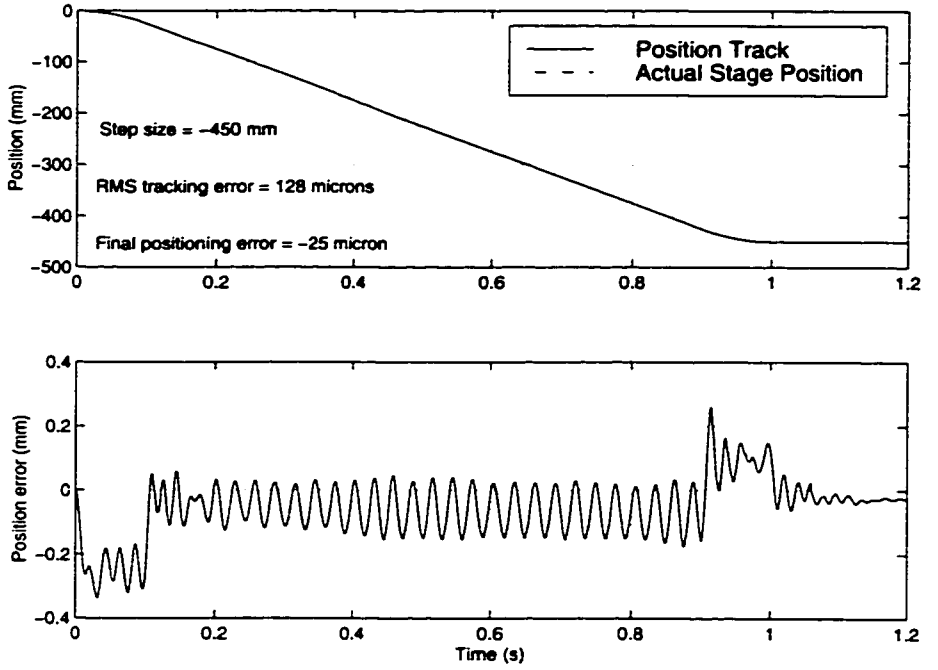


Figure 8.6 Results using PD type fuzzy controller on X-Y setup

Figure 8.6 shows the results using the same controller for the X-Y setup. The X-Y setup is much more smoother in terms of the linear bearing friction. In addition, the belt used for the X-Y setup has a much higher specific elasticity. These factors help the performance of the controller giving an RMS error ranging from 100 μm to 225 μm and a steady state error ranging from 20 μm to 80 μm. It can be seen that the peak tracking error reduces from 1.2 mm for the single axis stage to about 0.4 mm for the X-Y setup.

To obtain high precision control, it is necessary to overcome these static friction and belt elasticity effects. Hence, a friction and elasticity compensator is implemented.

8.3 Friction and Elasticity Compensation

Considerable amount of work has been reported on the modeling of friction and development of friction and backlash compensation schemes. Armstrong *et al* present an excellent survey of friction modeling and techniques for friction compensation in [52]. Friction effects are present in rigid link systems also. However, the presence of the belt adds the following challenges:

- dependence of the dead zone introduced by belt elasticity on the direction of travel,
- variation of this dead zone depending on the position of the stage, and,
- an overall increase in friction due to use of pre-loading to reduce the transmission backlash.

Friction compensation schemes proposed in the literature and reviewed in [52] can be classified into model based and non-model based methods. Model independent schemes include:

- high gain PD control – this basically serves to increase stiffness and eliminate stick-slip,
- integral control – this removes the steady state error associated with *stiff* control but can introduce hunting and limit cycles, and,
- impulse control – typically these methods use small torque pulses to achieve accuracy close to the resolution available from the encoder.

Model based methods include:

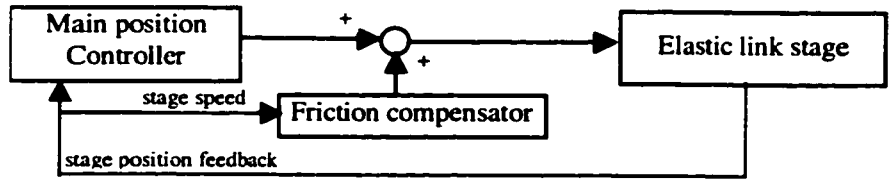


Figure 8.7 Block diagram of a model based friction compensator

- fixed compensation – this is based on the addition of a torque command to that generated by the main position controller. The additional torque command is generally a function of the velocity of the load being controlled as shown in Figure 8.7. The major difficulty in performing more specific model based friction compensation is the difficulty in obtaining a good friction model.
- adaptive friction compensation – the main problem is obtaining good estimates of the friction model coefficients and assuring stability of the adaptive system [72]. Methods include model reference adaptive control techniques usually assuming a first or second order reference model and using an adaptation law based either on a Lyapunov function or Popov's hyperstability criterion [69].

Considerable amount of work has been reported on the use of novel, computational intelligence methods for friction compensation. Notable among these are the use of neural networks [73] and fuzzy neural networks [71]. These methods are basically a way of using non-linear friction compensation, using a fuzzy friction compensator (also called as a pre-compensator by some researchers [57]). The main modification to Figure 8.7 is the addition of the output of the main position controller as another input to the fuzzy friction compensator. It has been shown that fuzzy friction compensation works as good as, if not better than, other model based friction compensation methods.

The fuzzy friction compensation method presented in [70] uses an acceleration-based method instead of a velocity based method. If a certain torque reference command is applied by the controller to the servo amplifier, it is expected that after a certain delay, the stage will accelerate and the acceleration will depend on the applied torque reference command. If for a given torque reference, the stage does not accelerate, this indicates the presence of an opposing static friction force which is greater than the applied force. Similarly, if the stage accelerates but the acceleration is slower than expected, this indicates the presence of viscous friction. For both these cases an appropriate additional term is added to the torque reference. Thus, the inputs to the friction compensator are the torque reference applied to the motor delayed by some sampling period and the acceleration of the stage. The output is an additional torque reference that is added to the

output of the fuzzy PD controller. This approach is inherently model independent. This method basically implements a *fuzzy disturbance observer and compensator* and can be equally useful for elasticity compensation. Hence, this method was selected for use with appropriate modifications to the rule base, membership functions and scaling factors.

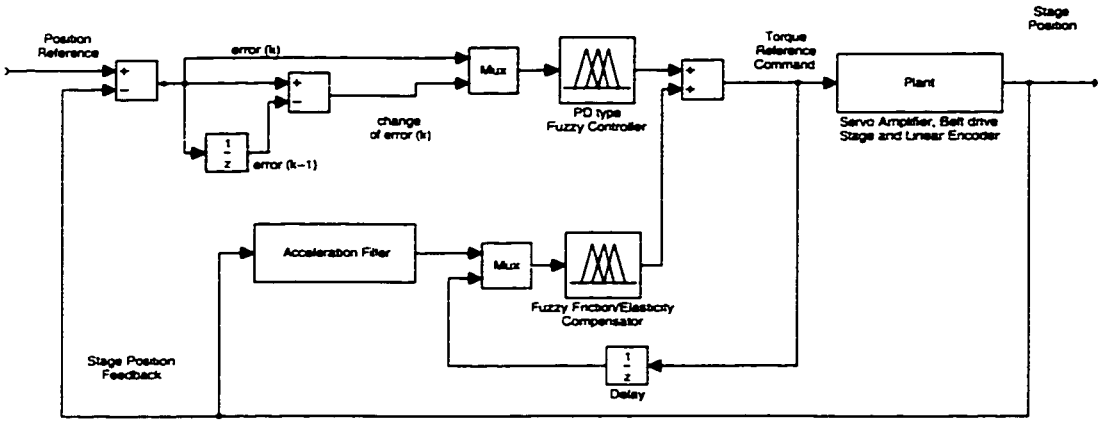
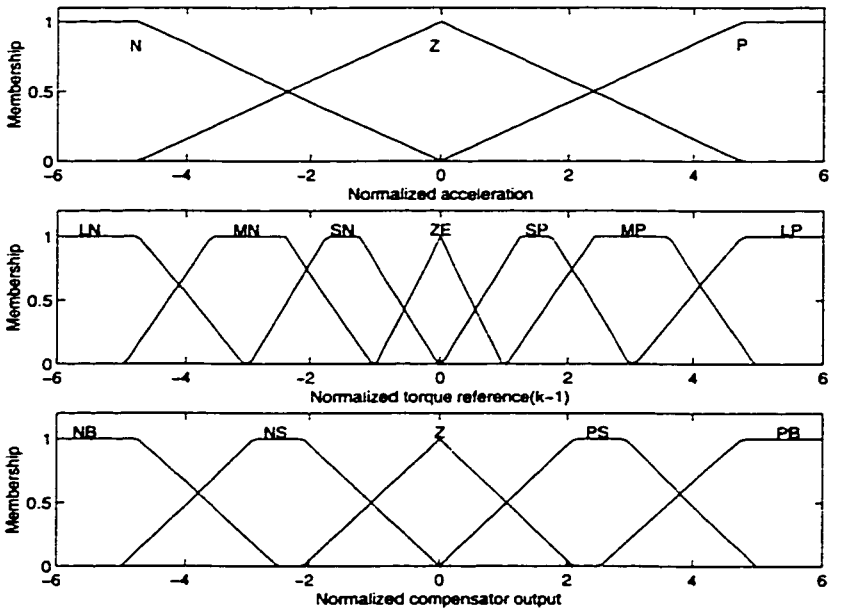


Figure 8.8 Block diagram of controller with fuzzy compensator

Figure 8.8 shows the block diagram of the controller with the fuzzy compensator



P

Figure 8.9 Membership functions used for the friction compensator

included. To obtain good controller resolution, seven fuzzy sets are used for the torque reference command. For the acceleration input only 3 sets are sufficient while the output of the compensator has five fuzzy sets. The membership functions used are a combination of trapezoidal and triangular and these are shown in Figure 8.10 for the inputs and output. The membership functions and the scaling factors used for normalizing the inputs and outputs to/from the compensator were adjusted based on the performance of the controller especially with regards to the final positioning error.

Table 7 Rule base for the fuzzy friction and elasticity compensator

		Delayed Torque Reference						
		LN	MN	SN	ZE	SP	MP	LP
A	N	NB	NS	NS	Z	Z	Z	PS
C	Z	NB	NS	NS	Z	PS	PS	PB
C	P	NS	Z	Z	Z	PS	PS	PB
N								

The rule base used for the compensator is very simple and intuitive and is given in Table 7, where ACCN in the first column stands for the acceleration.

If the acceleration is calculated by using the backward difference approximation twice on the sampled stage position, the output is very noisy and the use of such a noisy term in the friction/elasticity compensator is impractical. Hence, the acceleration filter estimates the stage acceleration from the stage position samples using a Kalman filter, where the dynamics of the system are given by:

$$\frac{d}{dt}\begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix}$$
$$y = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}\begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix} \qquad (8.2)$$

This basically uses a linear system with the stage position, velocity and acceleration as

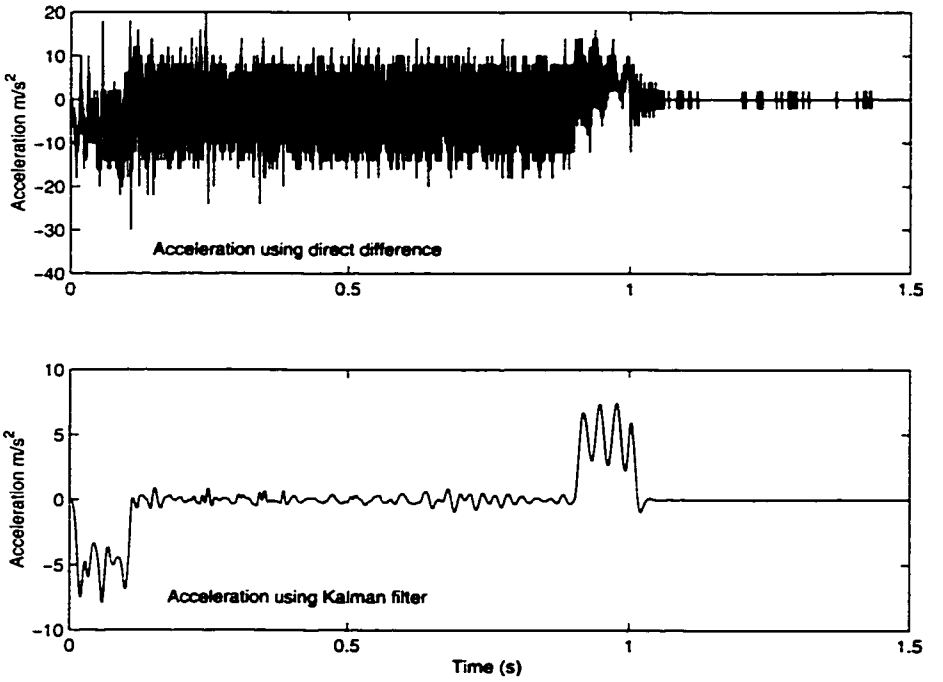


Figure 8.10 Comparison of acceleration calculation using direct difference and Kalman filtering

the 3 states. The acceleration is to be estimated, hence it is added to the state matrix to create the augmented state matrix. The rate of change of acceleration which is not known is set to zero. The measured output y is the stage position x. Figure 8.10 compares the output of the Kalman filter to the acceleration obtained using direct difference. The process noise covariance matrix used in the Kalman filter calculations is adjusted so to give the minimum delay in the filtered acceleration while providing maximum noise rejection.

8.3.1 **Results**

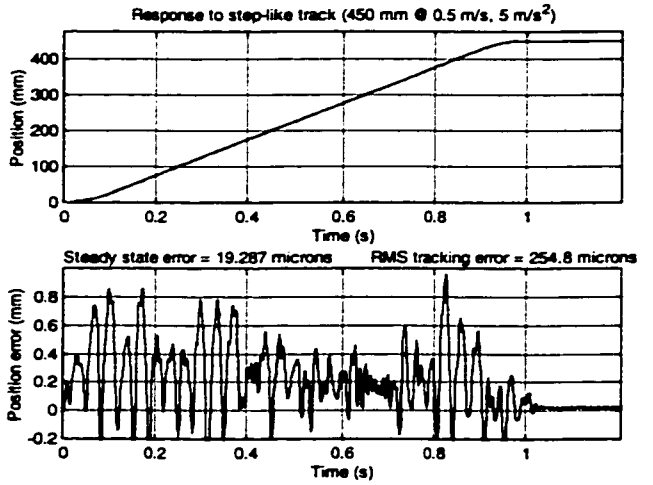


Figure 8.11 Results using friction and elasticity compensator for the single-axis stage

Figure 8.11 shows the results obtained using the compensator in combination with the fuzzy PD controller for the single-axis stage. The step size is 450 mm and the track is limited to a speed of 0.5 m/s and an acceleration of 5 m/s^2. The final positioning error obtained is between 5 and 30 μm, while the RMS tracking error is between 225 and 300 μm, for multiple tests at different speeds and for different step sizes. The effect of the compensator is seen in the reduced final positioning error and greater repeatability. Without the compensator, the error varies between 25 and 90 μm, with a mean of 50 μm and a standard deviation of 20 μm. With the compensator the mean error is about 17 μm and the standard deviation is about 4 μm.

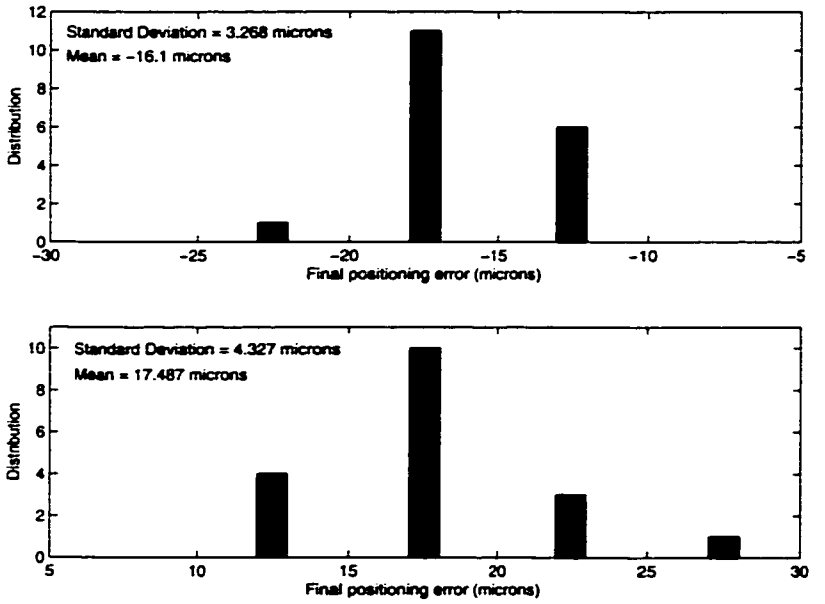


Figure 8.12 Histogram of final positioning errors for single-axis test setup

Figure 8.12 shows bar graphs of the distribution of final positioning error for different step sizes with different track speed and acceleration limits. A total 36 tests were conducted, 18 in each direction. The mean and standard deviation of the error which represent the accuracy and repeatability are also shown on each graph. The results show the excellent accuracy and repeatability achieved despite using a belt with very low stiffness (and hence low cost) and a linear bearing which has high friction.

8.4 Fuzzy Integral Control

The performance of the fuzzy controller was compared to that of a standard, linear PID controller. For the single-axis test setup where the belt elasticity and friction are high, the linear PID controller causes large limit cycle oscillations around the final goal position. The RMS tracking error with a linear PD controller was of the order of 3 mm, which is about 10 times that achieved with the fuzzy controller. However, the X-Y setup has a stiffer belt and the bearing friction is also low. Since the non-linearities are not severe, a

PI type fuzzy controller was added to the fuzzy PD controller to replace the fuzzy compensator. This effectively implements a fuzzy PID controller. However, this arrangement is better than using a 3 input, single output fuzzy controller to implement a fuzzy PID controller. This is because the number of rules for a 3 input, 1 output fuzzy controller with 7 fuzzy sets for each input, are 7^3 or 343, while the number of rules when two different controllers are used is 49+49 or 198. Another advantage of this method is that instead of integrating the error before feeding it to the fuzzy PI controller, the output of the fuzzy controller can be integrated, which has been reported to give better results [74].

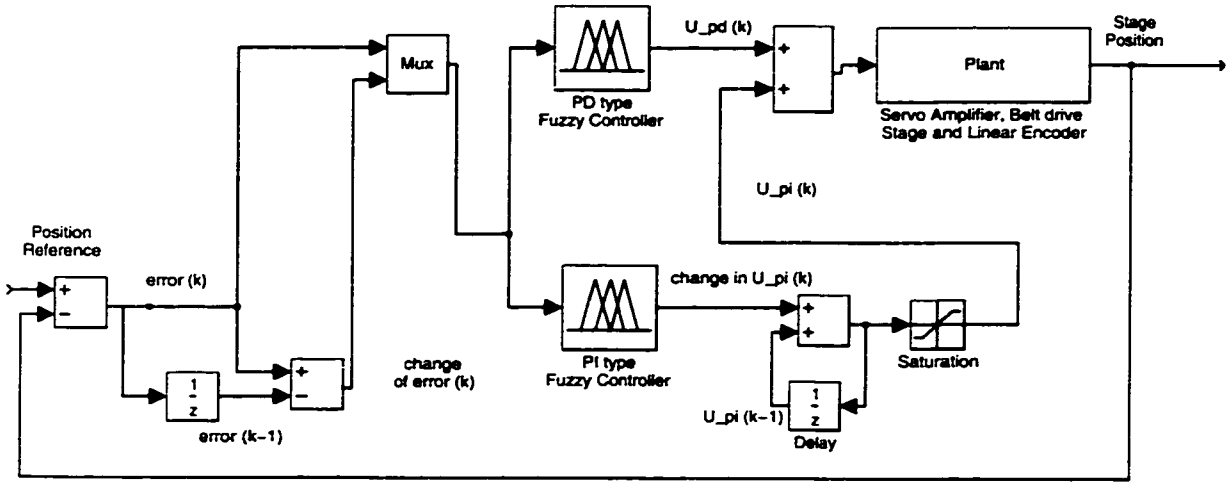


Figure 8.13 Block diagram of controller using fuzzy PD and PI type control

The inputs to the PI type fuzzy controller are the position error and the change of position error. The output is a change in control, Δu_{PI}, which is added to the summed fuzzy PI controller output at the previous instant, $u_{PI}(k-1)$ to obtain the summed output at current instant:

$$u_{PI}(k) = u_{PI}(k-1) + \Delta u_{PI} \qquad (8.3)$$

The output of the fuzzy PI controller u_{PI} is added to that of the fuzzy PD controller, u_{PD} to get the torque reference command fed to the servo amplifier. Figure 8.13 shows the block diagram of the combined fuzzy PID controller. The fuzzy PI controller output is limited to a fixed, tunable value, as shown by the saturation block in the figure. This implements anti-windup of the controller. It should be noted that by the very nature of

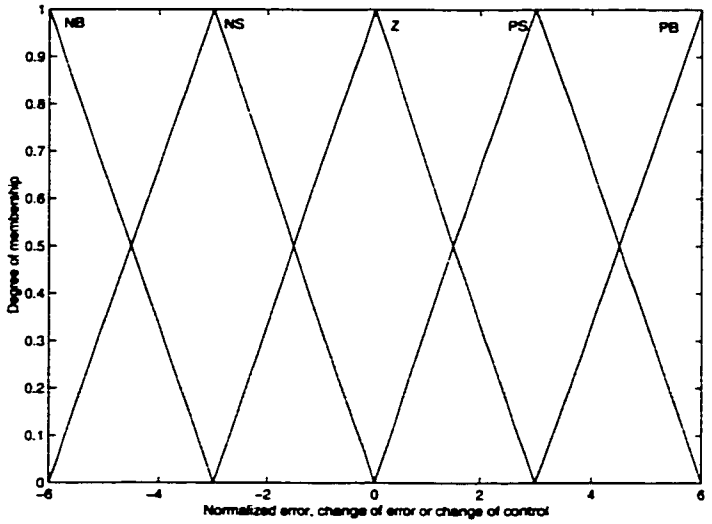


Figure 8.14 Membership functions used by the PI type fuzzy controller

the fuzzy control, the maximum change in control, Δu_{PI}, is limited. Five fuzzy sets are used for error, change of error and change of control having symmetric, equal area, triangular membership functions as shown in Figure 8.14. The linguistic names of the fuzzy sets are *NB (negative big), NS (negative small), Z (zero), PS (positive small)* and *PB (positive big).* These names are deliberately kept different from those used for the PD type fuzzy controller for ease of programming. All the variables are limited to a normalized active range of [-6, 6]. Actual error and change of error are multiplied by scaling factors to obtain the normalized error and change of error which are then fuzzified. The centroids of the fuzzy set membership functions are located at [-6, -3, 0, 3, 6].

8.4.1 **Rule Base**

A simple symmetric rule base used for this controller is shown in Table 8. The logic used in designing the rule base is similar to that for the PD type fuzzy controller. The output from the rule base is the fuzzy change of control which is defuzzified using a center-of-area defuzzifier. Here, the defuzzification operation is simple since all the membership

Table 8 Rule base for the PI type fuzzy controller

		Change of Error				
		NB	NS	Z	PS	PB
ERRO	NB	NB	NB	NS	NS	Z
	NS	NB	NS	NS	Z	PS
	Z	NS	NS	Z	PS	PS
	PS	NS	Z	PS	PS	PB
	PB	Z	PS	PS	PB	PB

functions have the same area. The crisp (defuzzified) change of control is then multiplied by the change of control gain (scaling factor).

8.4.2 **Results**

The combined fuzzy PD and fuzzy PI control was applied to the X-Y setup and a series of tests were conducted to obtain the accuracy and repeatability. Figure 8.15 shows a typical plot of the results obtained. The final positioning error for the plot in the figure is

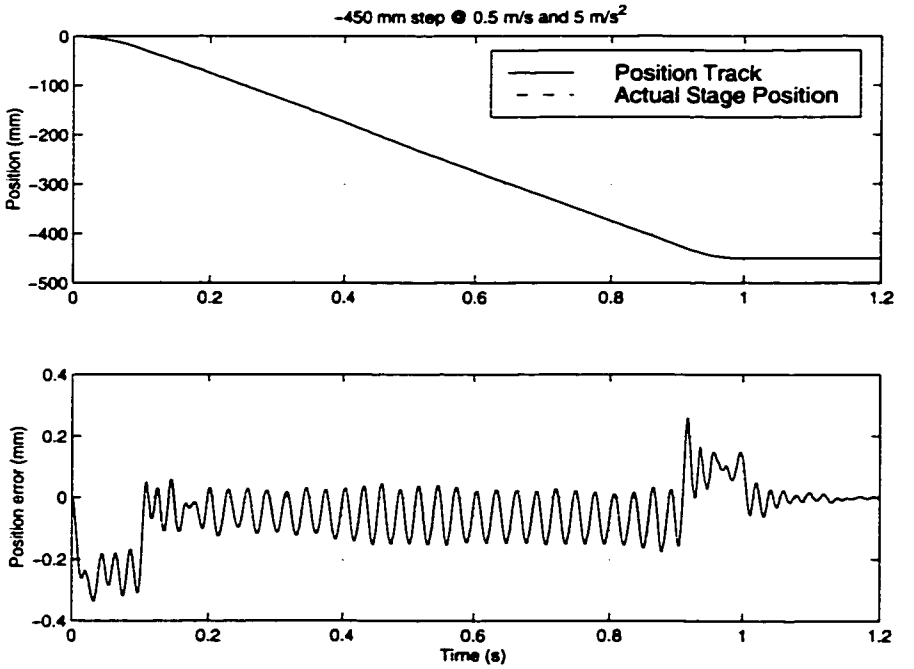


Figure 8.15 Results obtained using fuzzy PID control for the X-Y table

zero while the RMS tracking error remains unchanged (as compared to using only the fuzzy PD controller) at about 130 μm. The tracking error does not change because the scaling used for error, change of error and change of control for the fuzzy PI controller, as well as the limit imposed on u_{PI}, ensures that the integral control acts only when the stage is moving very slowly (at the beginning and end of the step-like track). The final positioning error varies between -0.5 μm and 0.5 μm, which means that the accuracy and repeatability is only limited by the resolution of the encoder. It is worth mentioning that this repeatability is obtained for motion in both directions and hence, is a *bi-directional repeatability*. These results are **similar to those that have been reported with lead screw or ball screw based systems**. Thus, high precision positioning can be obtained with belt drive systems using fuzzy control.

Although the accuracy and repeatability obtained are very good, the tracking error is high for both the single-axis setup and the X-Y setup. This is because the fuzzy PD controller has zero output for zero error. When such a point is reached while the position (and the position track) are changing, the stage again starts lagging behind the position track. This is the cause of some of the oscillations in the tracking error. One approach to reducing the tracking error is to increase the region of operation of the fuzzy PI controller to cover the whole range. Then, after the integrator stabilizes, the fuzzy PI controller output constitutes the average value required for the tracking while the fuzzy PD controller output provides robustness and sample-to-sample error compensation. However, it is seen that the limit on u_{PI} has to be increased to a high value for the integral control to provide a major portion of the torque reference during tracking. This causes the system to exhibit oscillations around the target position resulting in a large settling time, which is not acceptable.

To reduce the tracking errors while keeping good final positioning performance, a feedforward controller is required, which can provide the required average torque reference command during tracking. The design of this controller is described next.

8.5 Feedforward control

It can be easily shown that a feedforward term is required to obtain perfect tracking for any linear system [62]. Feedforward terms are recommended when full state feedback control is used for tracking control applications [76]. For linear systems, the feedforward controller is generally the stable part of the plant inverse. Many methods have been proposed for the design of feedforward controllers for linear systems, all of which try to bring the closed loop transfer function close to unity. These include:

- the zero phase error tracking method which includes cancelable plant zeros in the denominator of the feedforward transfer function and approximates the uncancelable zeros with their reciprocal in the numerator [78],
- designs based on identification of the low frequency modes of the system and inverting the transfer function containing these modes [77], and,
- methods using extended pole placement that use the transfer function of a non-causal Finite Impulse Response filter as the reference model for pole placement [79].

For precision position control using rigid links, velocity and acceleration feedforward of the position track are used along with PID control. However, the application of feedforward control to non-linear systems is not straightforward unless a system model is available which can be inverted, although acceleration feedforward is used as part of sliding mode control. Luh and Rizzoni have proposed a genetic algorithm based nonlinear system identification method that uses an invertible, Non-linear, Auto Regressive, Moving Average, eXogenous (NARMAX) model [80]. The terms used in the NARMAX model are not fixed but are obtained by the system identification technique. This approach is limited by the fact that the NARMAX model essentially has to be linear in the plant model so as to be invertible, which basically implies that it is useful only for a feedback linearizable system.

Three different approaches are adopted here for the design of the feedforward controller, and these are discussed below. The approaches used here do not try to obtain an inverse for the complete system model including the belt elasticity effects, since such an attempt

would defeat the purpose of developing the model independent feedback controller. The idea behind development of the feedforward controller is to obtain a reasonably good model of the second order system that would be obtained if the belt elasticity is neglected. This would enable the feedforward controller to provide the baseline torque reference command while the fuzzy feedback controller can continue to compensate for the belt elasticity and other non-linear effects. It also allows a reduction in the gain of the feedback controller, thus reducing the control effort.

8.5.1 **Linear system identification approach**

As a first step, linear system identification tools were used to obtain an approximate, linear model of the system. Input-output data of the system was collected by applying different inputs so as to provide persistent excitation as well as to cover the range of all variables. The MATLAB System Identification toolbox was used to obtain a linear system model [81] relating the stage speed with the torque reference command. This model is then inverted to obtain the feedforward controller. A 4^{th} order ARX model gives the best fit to the data, although this fit was not very good due to the limitations of using a linear model. The model works well as a few steps ahead predictor but fails when used to simulate the system. The feedforward controller obtained was then used in conjunction with the feedback controller already developed. The results obtained did not give any noticeable improvement in the tracking control performance due to the poor match between the model and the plant. It was observed that if data for a particular operating speed was used for identifying the plant model, the system identification works well for that particular speed, but when such a model was applied for other speeds there was an average error between the model output and the plant data This suggested the presence of a term for the Coulomb friction effect that switches with stage speed and can be written as:

$$F_c \, \mathrm{sgn}(\dot{x}) \tag{8.4}$$

where *sgn* is the signum function and F_c is a constant. This Coulomb friction term is non-linear and hence cannot be directly included while using MATLAB's System Identification Toolbox. Hence, the constant F_c was adjusted by trial and error manually, in conjunction with the linear system identification. For the single-axis setup, the model obtained with this term, is given by:

$$\dot{x}(k)-1.226\,\dot{x}(k-1)-0.249\,\dot{x}(k-2)+0.416\,\dot{x}(k-3)+0.0612\,\dot{x}(k-4)+385\,\mathrm{sgn}(\dot{x}(k))$$
$$=2.423e-5\,(\,u(k-1)-0.214\,u(k-2)-0.0832\,u(k-3)-0.521\,u(k-4))$$

..... (8.5)

where, u is the torque reference command fed to the servo amplifier. This equation is rearrangement with *u(k-1)* on the right hand side to obtain the feedforward controller:

$$u(k)=a_1\dot{x}_d(k+1)+\cdots+a_5\dot{x}_d(k-3)+385\,\mathrm{sgn}(\dot{x}_d)+b_1u(k-1)+\cdots+b_3u(k-3) \qquad (8.6)$$

For the X-Y setup, the linear system identification gave better results when stage position was used instead of stage velocity. Also, a coulomb friction term was not required. The resultant system model for the X-Y setup is given by:

$$x(k+1)-1.998x(k)+0.9981x(k-1)=1.277e-5\,u(k)-1.091e-5\,u(k-1) \qquad (8.7)$$

and the feedforward controller is again obtained by rearranging the terms. Both the feedforward controllers are non-causal which does not pose any additional constraints

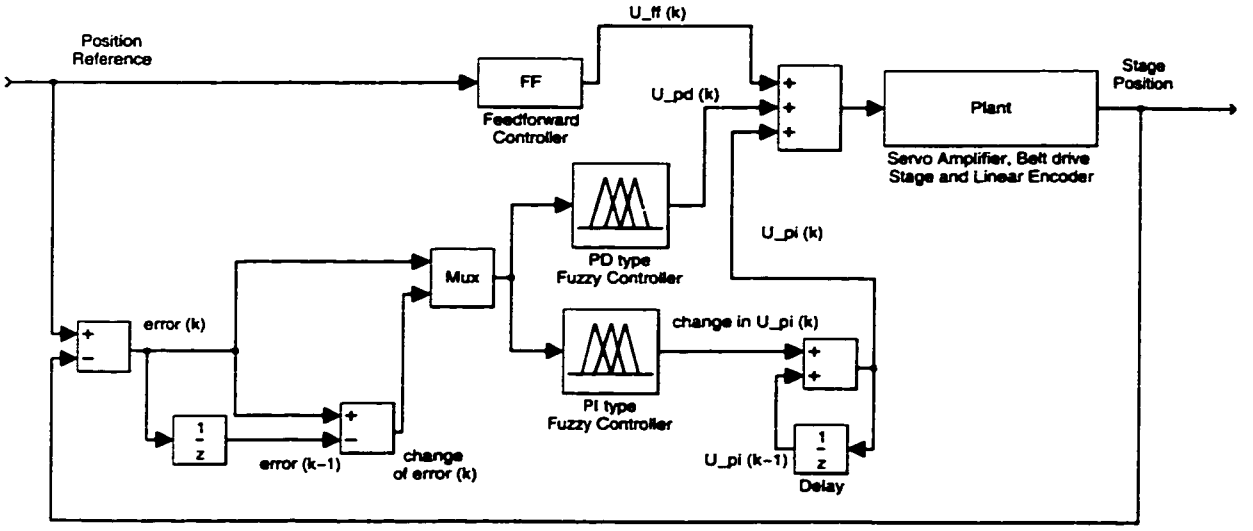


Figure 8.16 Block diagram of control system showing the three controllers

since it requires one step ahead knowledge of the position track. In positioning systems, typically the position track is generated on-line using some interpolation methods, so that the desired position information is available at least a few samples in advance.

These feedforward controllers are used along with the developed feedback controller for both test setups. Figure 8.16 shows a block diagram of the complete control system for the X-Y setup, while the PI type fuzzy controller block is replaced by the fuzzy friction compensator for the single-axis setup. Figure 8.17 shows the results obtained using this complete controller for the single-axis setup. It can be seen that the tracking error is considerably reduced while the final positioning error is not affected.

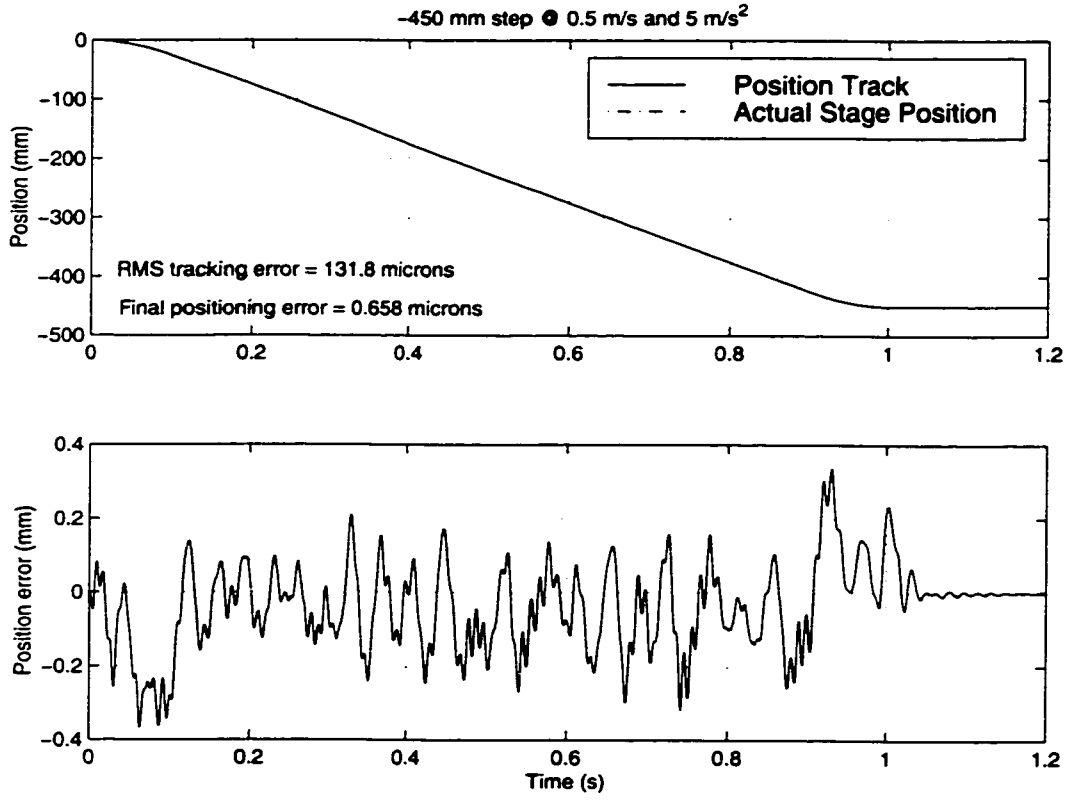


Figure 8.17 Results for the single-axis test setup using complete control system

The RMS tracking error for the single-axis setup varies from 110 μm to 165 μm, which is a reduction of more than 50% as compared to the results obtained without the

feedforward controller. The final positioning error remains below 25 μm, with excellent repeatability.

8.5.2 **Non-linear system identification using Evolutionary programming**

To further improve the feedforward controller, it is desirable that other effects due to friction are identified. This is done by attempting to identify the 7 parameters of the friction model given by equation 7.8 and is repeated here:

$$F_f = F_v\dot{x} + \left(F_C + (F_{ST} - F_C)\left(\frac{1}{1+\left(\frac{\dot{x}(t-\tau_l)}{\dot{x}_s} \right)} \right) \right) \mathrm{sgn}(\dot{x}) \qquad (8.8)$$

where, F_C is the Coulomb friction constant, F_v is the viscous friction constant, F_{ST} is the static friction, $\dot{x}_s$ is the velocity of the Stribeck friction and τ_l is the time constant representing the hysteresis effect. This model is inherently non-linear. It is also discontinuous, due to the presence of the sgn() term in the model and also since the static frictional force depends on the force applied to the stage. These characteristics make this a suitable problem for applying evolutionary programming based system identification. To apply evolutionary programming, the following system model was assumed:

$$m_e\ddot{x} + F_s(\dot{x}) = F_m \qquad (8.9)$$

where, $F_s(\bullet)$ represents the frictional force given by the 7 parameter model, m_e is an equivalent mass, F_m is the force equivalent to the reference torque command. This is precisely the model of a system using a rigid transmission element such as a lead screw or a ball screw. Since the frictional forces in the two directions of motion can be different, different frictional parameters are chosen for the two directions of motion. Thus, the total number of parameters to be estimated are 15 (including the equivalent mass). The method used has the following features:

1. Representative open or closed loop data sets are collected from the setup.

2. Given a set of parameters, the assumed system model is simulated for the torque reference commands included in the data set and the simulation output is compared to the actual data collected.
3. The RMS error between the simulated stage position and actual stage position gives a function which is used as a cost index to be minimized by the evolutionary program (EP).
4. The EP produces sets of parameters as part of its iterative process and then obtains the cost for each chromosome using Steps 3 and 4.

Figure 8.18 shows a block diagram of this process, where a set of parameters produced by the EP are used to simulate the model for the given data. The EP algorithm is written

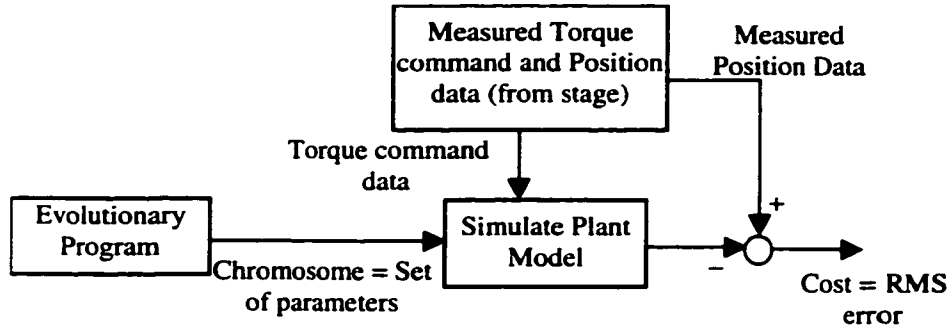


Figure 8.18 Identification of friction parameters using Evolutionary Programming

in C++, while the model is simulated using Simulink and MATLAB. The MATLAB API has "Engine" routines which enable other programs to interface with it and use its capabilities. This is done by exchanging data with MATLAB and by passing commands to it. This method was used to send the parameters to MATLAB so that the system could be simulated using Simulink [82]. This was initially tried on positive direction data using 5 different test data sets having maximum speeds of 0.1, 0.3 and 0.5 m/s and maximum accelerations of 3 m/s^2 and 5 m/s^2. It was seen that the model obtained gives a good match with the experimental data for stage speeds of 0.3 and 0.5 m/s at an acceleration of 5 m/s^2. However, the match obtained is not very good at low speeds and acceleration. The parameters obtained for motion in the positive direction were:

$$F_c = 0.827906 \text{ V}$$

$F_v = 0.973754$ V/(m/s)

$F_{s,\infty} = 2.42512$ V

$\tau_l = 6.94867$ milliseconds

$\dot{x}_s = 10.8895$ mm/s

$m_e = 0.717673$ V/(m/s^2)

All the force units are in V which corresponds to the torque reference command input to the servo amplifier in V. All the experimental data was collected after the stage had been stationary for a long time. Hence, the 2 parameters associated with rising static friction with time were not used in the model and were not identified. The identification was performed using a standard floating point, evolutionary program (EP).

Once the model is identified, the feedforward controller is then defined as an inverse of the plant model and is given by:

$$u_{FF} = F_s(\dot{x}_d) + m_e \ddot{x}_d \tag{8.10}$$

where, $\ddot{x}_d$ and $\dot{x}_d$ are the speed and acceleration commands obtained by differentiating the position reference command. This is possible since the position reference track is generated with a given maximum acceleration and the maximum speed so that the second derivative of the position command is finite.

This method gives much better system identification than the linear system identification method. However, to obtain a good fit in all regions of operation, a large amount of data needs to be used. This increases the time spent in calculating the fitness function given a set of parameters. Hence a large amount of time (about 2 days on a 450 MHz, Pentium II based PC) was required for the evolutionary program to converge. This limitation makes it impractical to use this method and a simpler modification is required.

If the Stribeck effect in the friction model is neglected, the system model becomes:

$$m_e \ddot{x} + F_v \dot{x} + F_c \operatorname{sgn}(\dot{x}) = F_m \tag{8.11}$$

where F_c and F_v represent the Coulomb and viscous friction constants respectively. If data for the actual stage velocity and acceleration as well as the torque reference is available, least squares can be used to find the three parameters, m_e, F_v and F_c, since the

simpler model is linear in these three parameters. The measurement from the stage is the position from which the speed and acceleration are obtained using the Kalman filter that was developed for the friction/elasticity compensator in Section 8.3. The parameters obtained using this method are:

$$F_c = 1.1189 \text{ V}$$

$$F_v = 1.8472 \text{ V/(m/s)}$$

$$m_e = 0.7012 \text{ V/(m/s}^2\text{)}$$

These values are quite close to those obtained using the EP based method while requiring less computation. Hence, this method is preferable to the EP based method. Using the parameters obtained, the feedforward controller is defined as:

$$u_{FF} = F_v \dot{x}_d + m_e \ddot{x}_d + F_c sign(\dot{x}_d) \qquad (8.12)$$

Comparison of results for the two feedforward controllers show that the least squares based method is able to achieve the same results as the more complex EP based method. Figure 8.19 shows the results obtained using the complete controller which combines the feedforward with the fuzzy PD and PI feedback controllers. For the plot shown, the RMS

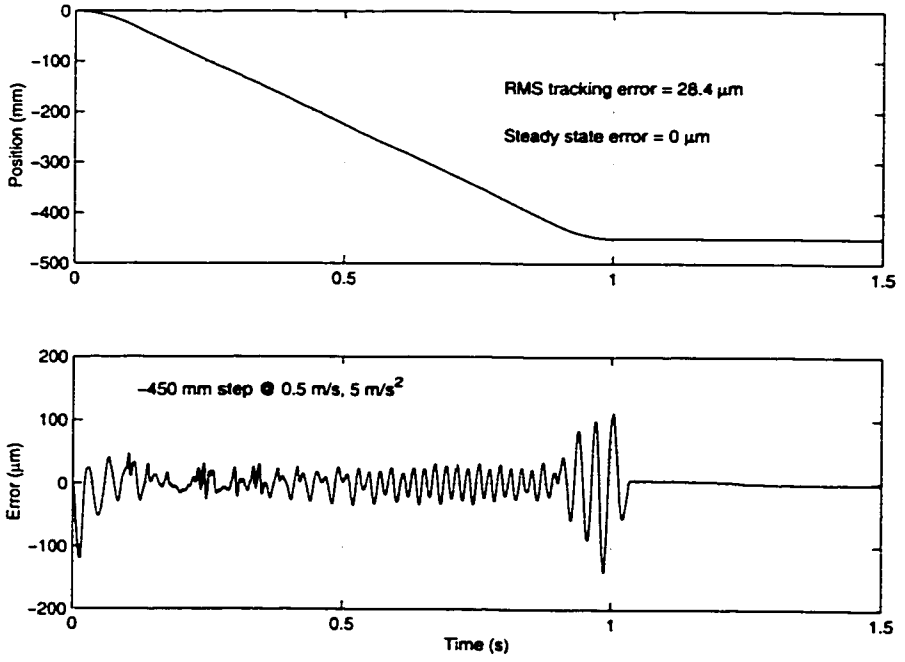


Figure 8.19 Results for the X-Y table using complete control system

tracking error is 28.4 μm, while the final positioning error is zero. The RMS tracking error ranges from 20 to 65 μm for a series of tests conducted by varying the step size, maximum speed and maximum acceleration. This is a reduction of more than 50% over the results obtained for the X-Y table without the feedforward controller.

8.6 Summary

The results obtained using the feedforward-feedback composite control system for both the single-axis setup and the X-Y setup show that high precision positioning with good tracking performance can be obtained from belt drive systems. These results are obtained for speeds up to 0.75 m/s and travel lengths up to 600 mm (for the X-Y setup) and 800 mm for the single-axis setup. Given these results, the developed control system can be implemented in many applications where belt drives can replace existing screw driven rigid link systems. Table 9 summarizes the results obtained for the two test setups using the different controllers developed.

Table 9 Summary of results

Controller	**Single Axis Test Setup**		**X-Y Table**	
	Steady state Error(μm)	**RMS Tracking error (μm)**	**Steady state Error(μm)**	**RMS Tracking error (μm)**
Fuzzy PD	25-90	300-400	20-80	100-225
Fuzzy PD + (fuzzy PI or friction/elasticity compensation)	5-30	225-300	0-0.5	100-200
Feedback + Feedforward	5-25	110-165	0-0.5	20-65

The design methodology can be briefly summarized as follows:

1. Design and tune a fuzzy PD type feedback controller for any given belt drive stage. The output of this controller should be scaled over the full operating range so as to effectively implement a fuzzy sliding mode controller.
2. Add a fuzzy PI type feedback controller with anti-windup to obtain good final positioning. If this causes a limit cycle around the final position, remove this controller and add the acceleration based fuzzy friction/elasticity compensator
3. To reduce the tracking error, design a feedforward controller based on a simplified friction model using least squares or evolutionary programming.
4. Apply the feedforward controller in combination with the feedback controller. The feedback controller gains will have to be re-tuned after such an addition.

The above steps can be used to develop a controller for any belt drive system. It can also be applied to other elastic transmission elements such as plastic lead screws since the controller is not dependent on a system model. The next chapter describes the development of a self-tuning adpative controller which can adapt and adjust to changes in the belt drive system due to aging.

Chapter 9 Self-tuning Control

While developing the control system described in the previous chapter, it was observed that the rule base did not have to be extensively tuned. Also most of the tuning for the rule base is very intuitive since desired changes are localized based on the range of error and change of error. The membership functions for all fuzzy controllers were also kept substantially fixed and did not have to be modified. The only parameters that have to be changed are the scaling factors and the limit on the output of the fuzzy PI controller. There are a total of six such factors (five if the fuzzy PI controller is replaced by the friction/elasticity compensator):

1. Error gain and change of error gain for the fuzzy PD controller
2. Error gain, change of error gain, change of control gain and control limit for the fuzzy PI controller. If the fuzzy compensator is used in place of the fuzzy PI controller, the parameters are acceleration gain, control gain and compensator gain.

These factors have to be tuned to obtain good performance. Due to the effort required in tuning these factors and due to the possibility of changes in system parameters over time, a self-tuning method has been developed for the controller using evolutionary programming technique. The evolutionary programming approach is used for developing the self-tuning algorithm because of its ability to find global minimum as well as the absence of any need for gradient information. Since a good system model is not available for the belt drive system, the tuning should be performed based on the experimental results obtained and should not need gradient or model information. Hence, the EP approach suits this problem very well. This chapter starts with a description of accelerated evolutionary programming followed by the details of the self-tuning controller and the results obtained.

9.1 Accelerated Evolutionary Programming

One of the main limitations of evolutionary algorithms is the slow speed of convergence. This is because these algorithms are basically random search techniques utilizing some genetic operations to speed up the process. Kim et al have proposed an Accelerated Evolutionary Programming (AEP) algorithm [83] which has many advantages over traditional EP algorithms. The AEP is a modified form of evolutionary programming where the *crossover* operation is eliminated and the mutation is maintained along a particular direction if changes in that direction gave an improvement in the fitness value at the previous step [83]. This gives a directed, random search algorithm and is reported to be able to find the global optimum at much faster rate than traditional genetic algorithms and evolutionary programs. Initially a random, uniformly distributed population of solution vectors or chromosomes is created. This is called the parent population. The fitness for all the parent chromosomes is evaluated and then the parents are mutated to create the children population. Since the *crossover* operation is removed, one child chromosome is created from each parent. The mutation is carried out according to the following rules:

1. For each parent chromosome, the direction of change at the previous iteration is stored for each gene. If such a change resulted in an improvement in the fitness function, the direction is maintained for the next iteration i.e. the perturbation is a random number that maintains the direction of change (either increase or decrease in the floating point value). On the other hand, if the previous direction did not yield an increase in the fitness, the direction is not taken into consideration and the perturbation is truly random (both in magnitude and sign).
2. The random value used in the first rule is obtained from a normal distribution having a mean of zero and a standard deviation equal to the product of the fitness value and the age of the particular chromosome. The age is the number of iterations that the chromosome is alive.

Let the i^{th} chromosome in a evolutionary population be represented by a vector z^i, which is given by:

$$z^i = [z_1^i, z_2^i, , \cdots z_n^i]^T \tag{9.1}$$

where, the members of the vector are the individual parameters (or genes). AEP extends this vector to:

$$z^i = [z_1^i, z_2^i, , \cdots z_n^i, dir(z_1^i), dir(z_2^i), \cdots dir(z_n^i), age^i]^T \tag{9.2}$$

where, $dir(z_j^i)$ is the direction of "evolution" of each gene z_j^i and age^i is the number of iterations for which a chromosome is alive . If $f(\ )$ represents the cost function which is to be minimized by the AEP, Rule 1 can be written as:

$$\begin{aligned} &If \quad f(z^i[k]) < f(z^i[k-1]), \\ &then: \quad dir(z_j^i[k]) = \mathrm{sgn}(z_j^i[k] - z_j^i[k-1]), \\ &\qquad age^i[k] = 1; \\ &else: \quad age^i[k] = age^i[k-1] + 1 \end{aligned} \tag{9.3}$$

where, sgn( ) is the signum function and k is the iteration number.

The mutation is then performed according to Rule 2, which can be written as:

$$\begin{aligned} &If\ age^i[k] = 1, \\ &then: \ \sigma^i = \beta_1\ f(z^i[k]), \\ &\qquad z_j^i[k] = z_j^i[k-1] + dir(z_j^i) \mid N(0, \sigma^i) \mid ; \\ &else: \quad \sigma^i = \beta_2\ f(z^i[k])\, age^i, \\ &\qquad z_j^i[k] = z_j^i[k-1] + N(0, \sigma^i) \end{aligned} \tag{9.4}$$

β_1 and β_2 are two parameters of the AEP that can be adjusted to improve the convergence. Once the children population is created, each child's fitness is compared with its parent and the one (between a child and corresponding parent) having a better fitness is retained as the parent population for the next generation. Then, the new population is arranged in a descending order of fitness and the above steps are repeated till convergence is achieved. This method has been shown to achieve faster convergence compared to standard EP for many problems including identification of friction parameters for rigid link systems [84]. Hence, this method is selected for use in the self-tuning algorithm.

9.1.1 **Modifications**

The AEP algorithm as developed by Kim et al [83] does not have any means of limiting the parameters (genes) to realistic and feasible values. All the controller parameters to be tuned are scaling factors and are hence positive. The AEP has to be constrained to use only positive values. Many methods have been suggested for incorporating constraints in evolutionary programming [85]. These include: clipping of the parameters at the boundary, taking a mirror image across the limit boundary, and removing unrealistic parameters from the population. A hybrid approach is followed here, where the parameters β_1 and β_2 are kept small, and parameters that are set to negative are clipped to a fixed value of 0.5. It is seen experimentally that whenever a parameter gets set to such a low value, the cost of that particular chromosome is high and the direction gets reversed, which increases the clipped parameter in the positive direction in the next iteration. If the number of parameters getting clipped increases due to the high age of a particular chromosome, that age is reduced within the modified AEP algorithm.

9.2 Experimental Method

To apply the AEP to self-tuning, an experimental method has to be developed so that the controller can be tuned on-line. For initial tuning or commissioning of the controller, if a standard set of tests can be carried out with a set of control parameters, the cumulative RMS tracking error and the final positioning error can be used as an indication of the cost function. The number of tests selected should be small so as to minimize the time required for the tuning. For the belt drive stage, the main criterion is its performance when subjected to a step-like position track command. It is seen that the performance of any controller for a 450 mm step limited to 0.5 m/s and 5 m/s^2 is fairly representative of the overall performance for other step profiles. Hence, the cost function for a set of controller parameters is found by running two tests, one in each direction with a step size

of 450 mm with the track limited to 0.5 m/s and 5 m/s^2. Since the final positioning error is more important than the RMS tracking error, the cost function used is:

$$\sqrt{MSE + k \cdot (final\ positioning\ error)^2} \qquad (9.5)$$

where, *MSE* is the mean square tracking error and *k* is a weighting factor > 1.

The AEP performs the same steps as described in section 9.1 with the cost of each chromosome being found experimentally.

9.2.1 **Safety**

Care must be taken when using any evolutionary algorithm directly with a test setup. Although the AEP is initialized to parameters that are uniformly distributed within a

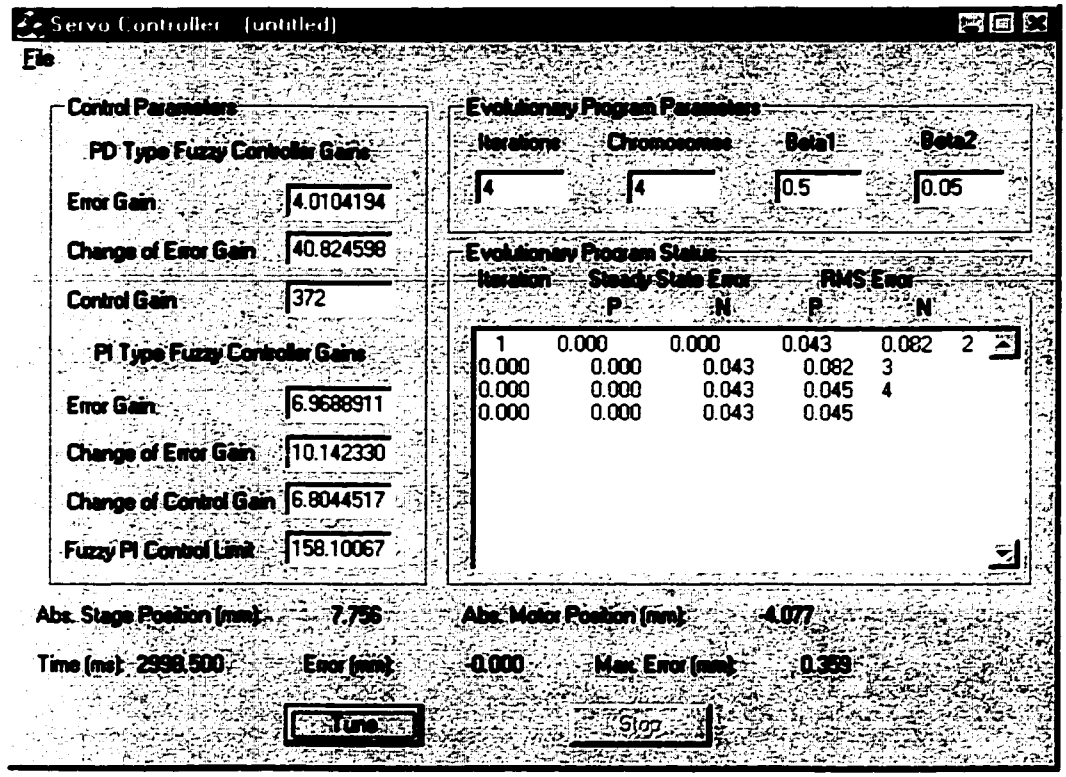


Figure 9.1 Screenshot of the user interface for the self-tuning software

certain range, these parameters can change to unsuitable values since the AEP generates chromosomes using a random perturbation. These unsuitable values can cause instability

and damage to the test setup. Hence, the tuning is kept safe by monitoring the *instantaneous maximum* tracking error for each test. If this error is above 5 mm, the test is stopped (for that particular chromosome only) and the chromosome is given a fixed high cost.

The software is written in C and C++ using the Win32 and RTX dual process approach for the controller development. The AEP is run in a separate thread within the Win32 process and it calls the main Win32 thread to obtain the cost of any chromosome. The main Win32 thread then activates the real-time subsystem (RTSS) and passes on the controller parameters. It monitors the tracking error at each sample and signals to RTSS to stop the test if the error becomes very high. Figure 9.1 shows a screenshot of the user interface for the self-tuning software. The left hand side shows the parameters which are being tuned and the right hand side shows the AEP parameters as well as the results for the best chromosome in each iteration. This particular run was started from an AEP population previously saved, so the steady state error is zero for all iterations. The RMS error in the negative direction reduces from 82 microns to 45 microns in the third iteration.

9.3 Results

The self-tuning algorithm is able to start from random controller parameters and successfully tunes the parameters to obtain zero final positioning error and an RMS error of less than 30 microns. Figure 9.2 shows the results obtained after self-tuning of the controller. The RMS tracking error obtained after manually tuning the controller is between 20 and 60 microns so that the performance of the controller improves after self-tuning. Also the number of iterations required for convergence to an RMS error of 30 microns is around 15. The whole self-tuning operation takes only about 15 minutes. This self-tuning algorithm is designed for self-commissioning of the control system. However, it can be easily adapted for use as an adaptive controller. The changes in plant parameters such as belt stretch and friction that will appear due to aging are slow

changes. The AEP algorithm can be continuous run on-line during the normal operation of the controller with the parameters β_1 and β_2 set very low. This will allow the AEP to perturb the parameters in the close vicinity around the optimum parameters and the parameters can then adapt slowly while maintaining the control system performance.

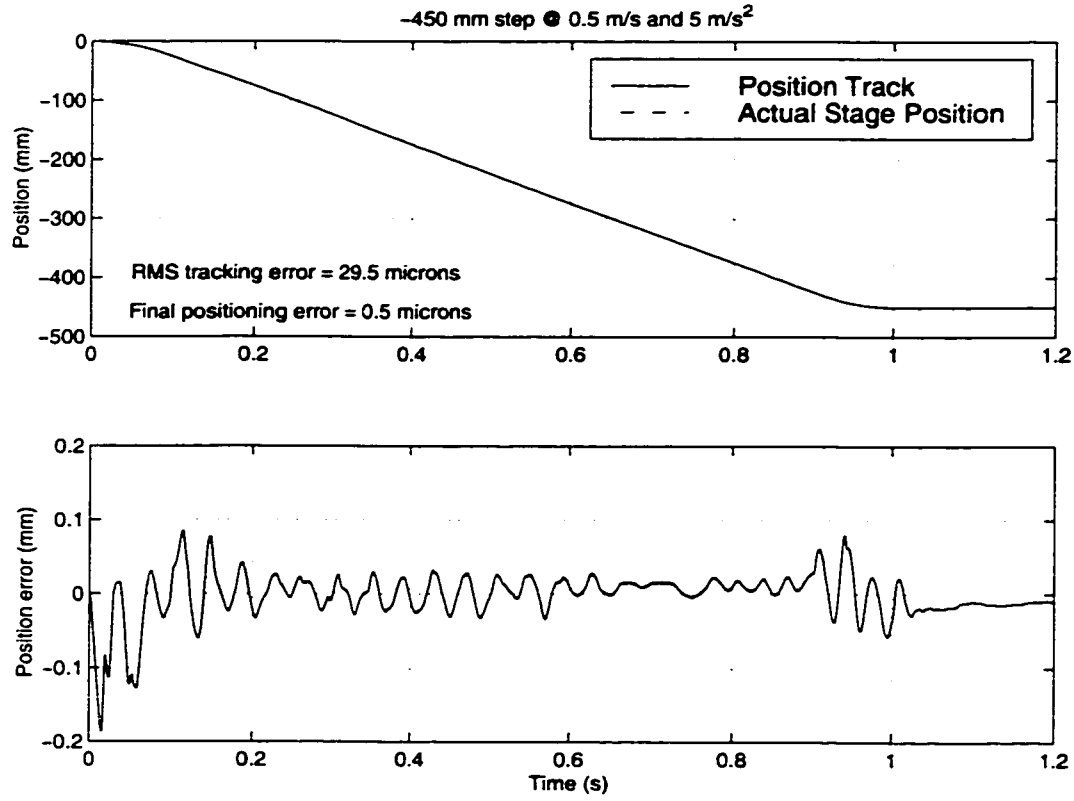


Figure 9.2 Results using parameters obtained after self-tuning

9.4 Discussion

Based on the experimental results presented in the previous chapter and the self-tuning algorithm results, the feasibility of using belt drive systems for high precision positioning using non-collocated control has been established. The developed control system performs well on the single-axis test setup which has a damaged bearing and which uses a very low cost belt. This is an indication that the developed control system is general enough to be applied to any belt drive system.

Chapter 10 Conclusions

Electromechanical drives are widely used for high performance actuation applications. Induction motor drives used for such high performance applications require a speed sensor for effective torque control. Available methods for estimation of induction motor states have many limitations. The first part of this dissertation explored the application of neural networks to state estimation for induction motor drives. Two new speed and state estimation algorithms have been proposed and tested using simulations.

High precision positioning servo applications currently use rigid transmission methods such as lead screws or ball screws to convert the rotary motion of a motor to linear motion of a load. The second part of this dissertation explores the application of fuzzy logic and evolutionary programming to develop a better, non-linear controller for positioning servos that use an elastic transmission such as a belt drive. Experimental results from two test setups are presented to establish the feasibility of using belt drives to obtain positioning accuracy similar to the rigid link systems.

10.1 Contributions

This dissertation makes several contributions to the state of the art. Two key industrial problems have been tackled and effective solutions have been proposed which can be directly applied to industrial products.

- A new open loop speed estimator for induction motor drives has been developed. The method uses a feedforward neural network to estimate the motor speed based on delayed samples of stator voltages and currents. Simulation results presented show the excellent performance of the method and its robustness to motor parameter variations. The proposed estimator does not suffer from many of the limitations affecting existing speed estimation algorithms.

- A new adaptive, closed loop, induction motor state estimator has been developed using a combination of neural networks and extended Kalman filters. This method performs simultaneous speed and rotor flux estimation and can be applied to any control algorithm for motor torque. A feedforward neural network is used to augment the motor state model and this augmented model is used as the non-linear motor model. The estimator uses a joint state and parameter estimation approach to adjust the weights of the neural network. Simulation results show the robustness of the method to variations in motor parameters.
- A combined feedback and feedforward control approach has been used to develop a fuzzy controller for elastic link systems such as belt drives. Experimental results are presented for two belt drive systems which show the high precision obtained. With the developed controller, belt drive systems can be used for many applications where the load does not need to apply thrust. Belt drive systems offer longer travel lengths and higher speeds than lead screw and ball screw systems at much lower cost. Thus, the controller developed in this dissertation has the potential of lowering the cost and increasing productivity for many industrial applications.
- A self-tuning algorithm has been developed using accelerated evolutionary programming to make the developed fuzzy controller self-commissioning. This makes the controller attractive for direct application to an industrial product. The self-tuning algorithm is able to achieve better results than a manually tuned controller since it can simultaneously optimize all the controller parameters.
- The software for the elastic link controller has been written using RTX, a real-time kernel running on Windows NT. A flexible software structure has been created using Win32 and RTX to easily develop any real-time control or signal processing application. The software modules created for fuzzy control and evolutionary programming can be easily reused in other real-time applications. The interprocess communication framework can serve as a breadboard for rapid prototyping of new control systems.

10.2 Ideas for Future Work

The work done here can be further extended in many different directions, including:

- The open loop speed estimator developed can be modified to include motor flux estimation. This can be explored so as to develop a low computational cost, all-in-one, state estimation method for induction motor drives.
- Both motor state estimation methods need to be tested on an experimental system so as to verify their real-life performance. Although the data used in simulation testing was made as realistic as possible, experimental application always poses additional challenges. A speed sensorless induction motor control system could then be implemented based on either estimation method.
- The stability and convergence properties of the closed loop state estimator can be analyzed using Lyapunov synthesis or other methods.
- The state space model developed for the belt drive system needs to be validated using a high resolution encoder at the motor shaft.
- A better non-linear feedforward controller can be developed using a better system model, which can also incorporate the belt elasticity properties.
- The rule base and membership functions of the developed fuzzy controller can be adapted and tuned online by modifying the self-tuning algorithm so as to obtain better performance.
- A control system tool set can be developed based on the RTX and Win32 software architecture developed. Additional modules can be added to create an easy real-time prototyping system.

Bibliography

[1] R. D. Lorenz, T. A. Lipo and D. W. Novotny, "Motion control with Induction motors," *Proceedings of the IEEE*, vol. 82, no. 4, pp. 1215-1240, August, 1994.

[2] F. Blaschke, "The principle of field-orientation as applied to the new Transvektor closed-loop control system for rotating machines," *Siemens Review*, vol. 34, pp. 217-20, 1972.

[3] M. Depenbrock, "Direkte Selbstregelung (DSR) für hochdynamische Drehfeldantriebe mit Stromrichterschaltung," *ETZ*, vol. A 7, pp. 211-18, 1985.

[4] I. Takashi and T. Noguchi, "A new quick response and high efficiency control strategy of an induction motor," *Conference Record, 1985 IEEE Industry Applications Society Annual Meeting*, pp. 496-502, 1985.

[5] K. Ohnishi, N. Matsui and Y. Hori, "Estimation, identification, and sensorless control in motion control systems," *Proceedings of the IEEE*, vol. 82, no. 8, pp. 1253-65, August 1994.

[6] J. Holtz, "State of the art of controlled AC drives without speed sensor," *International Journal of Electronics*, vol. 80, no. 2, pp. 249-63, Feb. 1996.

[7] Newport Corporation, *Scientific and Laboratory Products Catalog*, 1994-95.

[8] R. Haus, "Converting rotary motion to linear motion," *PCIM*, pp. 72-4, Nov. 1996.

[9] J. C. Bezdek, "On the relationship between neural networks, pattern recognition and intelligence," *Int. J. Approximate Reasoning*, vol. 6, pp. 85-107, 1992.

[10] L. A. Zadeh, "Fuzzy Sets," *Information and Control*, vol. 8, pp. 338-353, 1965.

[11] G. J. Klir and B. Yuan, *Fuzzy sets and fuzzy logic : theory and applications*, Prentice Hall, 1995.

[12] R. D. Reed and R. J. Marks II, *Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks*, MIT Press, 1999.

[13] D. B. Fogel, *Evolutionary computation: toward a new philosophy of machine intelligence*, IEEE Press, 1995.

[14] Z. Michalewicz, *Genetic algorithms + data structures = evolution programs*, Springer-Verlag, 1996.

[15] D. B. Fogel, *Evolutionary Computation; The Fossil Record*, IEEE Press, 1998.

[16] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Reading, MA, 1989.

[17] S. Haykin, *Neural Networks: A Comprehensive Foundation*, Macmillan College Publishing Company, 1994.

[18] P. C. Krause, *Analysis of Electric Machinery*, McGraw-Hill Book Company, New York, 1986.

[19] W. Leonhard et al, "Microprocessor control of induction motors employing field coordinates", *IEE 2nd International Conference on Electrical Variable-Speed Drives*, London, UK, pp. 146-50, 1979.

[20] R. Krishnan and A. S. Bharadwaj, "A review of parameter sensitivity and adaptation in indirect vector controlled induction motor drive systems," *21st Annual Power Electronic Specialists Conference (PESC) record*, San Antonio, TX, pp. 560-6, June 1990.

[21] J. Holtz and T. Thimm, "Identification of the machine parameters in a vector-controlled induction motor drive," *IEEE Transactions on Industry Applications*, vol. 27, no. 6, pp. 1111-8, Nov./Dec. 1991.

[22] M. Aaltonen et al, "Direct torque control of AC motor drives," *ABB Review*, no. 3, pp. 19-24, 1995.

[23] J. Holtz, "The representation of AC machine dynamics by complex signal flow graphs," *IEEE Transactions on Industrial Electronics*, June 1995.

[24] T. Ohtani, N. Takada and K. Tanaka, "Vector control of induction motor without shaft encoder," *IEEE Transactions on Industry Applications*, vol. 28, no. 1, pp. 157-64, Jan./Feb. 1992.

[25] X. Xu and D. W. Novotny, "Implementation of direct stator flux oriented control on a versatile DSP system," *IEEE Transactions on Industry Applications*, vol. 29, no. 2, pp. 344-48, 1993.

[26] P. Mehrotra, J. E. Quaicoe and R. Venkatesh, "Development of an artificial neural network based induction motor speed estimator," *PESC '96 Record, 27th Annual Power Electronics Specialists Conference*, Baveno, Italy, vol. 1, pp. 682-8, June, 1996.

[27] C. Schauder, "Adaptive speed identification for vector control of induction motors without rotational transducers," *IEEE Transactions on Industry Applications*, vol. 28, no. 5, pp. 1054-61, September/October 1992.

[28] F-Z Peng and T. Fukao, "Robust speed identification for speed sensorless vector control of induction motors," *IEEE Transactions on Industry Applications*, vol. 30, no. 5, September/October 1994.

[29] H. Tajima and Y. Hori, "Speed sensorless field-orientation control of the induction machine," *IEEE Transactions on Industry Applications*, vol. 29, no. 1, pp. 175-80, Jan./Feb. 1993.

[30] L. Ben-Brahim, T. Kudor, K. Shimane and H. Naitoh, "Implementation of an induction motor speed estimator using neural networks," *Proceedings of the International Power Electronics Conference*, Yokohama, Japan, vol. 1, pp. 52-7, April 1995.

[31] H. Kubota, K. Matsuse and T. Nakano, "DSP-based speed adaptive flux observer of induction motor," *IEEE Transactions on Industry Applications*, vol. 29, no. 2, pp. 344-8, Mar./Apr. 1993.

[32] S. Doki, S. Sangwongwanich and S. Okuma, "Implementation of speed-sensorless field-oriented vector control using adaptive sliding observer," *Proc. IECON '92*, vol. 1, pp. 453-458, 1992.

[33] T. Du and M. A. Brdys, "Implementation of extended Luenberger observers for joint state and parameter estimation of PWM induction motor drive," *Proceedings of the Fifth European Conference on Power Electronics and Applications*, Brighton, UK, vol. 4, pp. 439-44, September 1993.

[34] Y-R Kim, S-K Sul and M-H Park, " Speed sensorless vector control of induction motor using extended Kalman filter," *IEEE Transactions on Industry Applications*, vol. 30, no. 5, pp. 1225-33, September/October 1994.

[35] T. Du, P. Vas and F. Stronach, "Design and application of extended observers for joint state and parameter estimation in high-performance AC drives," *IEE Proceedings on Electric Power Applications*, vol. 142, no. 2, pp. 71-8, March 1995.

[36] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamic systems using neural networks," *IEEE Trans. on Neural Networks*, vol. 1, no. 1, pp 4-27, March 1990.

[37] R. Hecht-Nielsen, "Theory of the backpropagation neural network," *Proceedings of the International Joint Conference on Neural Networks*, vol. 1, pp. 593-605, June 1989.

[38] Amol S. Kulkarni and M. A. El-Sharkawi, "Speed estimator for induction motor drives using an artificial neural network," *1997 IEEE International Electric Machines and Drives Conference Record*, Milwaukee, WI, pp. MD2/2.1-3, May 1997.

[39] S. C. Stubberud, R. N. Lobbia and M. Owen, "An adaptive extended Kalman filter using artificial neural networks," *Proceedings of the 34th IEEE Conference on Decision and Control*, New Orleans, LA, pp. 1852-6, December 1995.

[40] R. G. Brown, *Introduction to Random Signal Analysis and Kalman Filtering*, John Wiley and Sons, 1983.

[41] M. T. Wishart and R. G. Harley, "Identification and control of induction machines using artificial neural networks," *IEEE Transactions on Industry Applications*, vol. 31, no. 3, pp. 612-9, May/June 1995.

[42] S. Singhal and L. Wu, "Training mutilayer perceptrons with the extended Kalman filter algorithm," in D. S. Touretzky, ed. *Advances in Neural Information Processing Systems*, vol. 1, pp. 133-140, Morgan Kaufmann, San Mateo, CA

[43] R. J. Williams, "Training recurrent networks using the extended Kalman filter," *Proceedings of the 1992 International Joint Conference on Neural Networks*, pp. IV-241-6, Baltimore, 1992.

[44] J. P. DeCruyenaere and H. M. Hafez, "A comparison between Kalman filters and recurrent neural networks," *Proceedings of the 1992 International Joint Conference on Neural Networks*, pp. IV-247-51, Baltimore, 1992.

[45] D. Obradovic, "On-line training of recurrent neural networks with continuous topology adaptation," *IEEE Transactions on Neural Networks*, vol. 7, no. 1, pp. 222-228, January 1996.

[46] "Understanding accuracy and repeatability," *Ideas*, Series A, Issue 2, Industrial Devices Corporation, http://www.idcmotion.com, Novato, CA.

[47] R. Haus, "Converting rotary motion to linear motion," *Power Conversion and Intelligent Motion (PCIM)*, pp. 72-4, November, 1996.

[48] I. A. Andermo, T. Shimomura, T. Kiriyama, and Y. Yamaguchi, "An absolute linear encoder utilizing a combination of capacitive and optical encoder technology," *Japan Society of Precision Engineering*, vol. 29, no. 1, March 1995.

[49] M. S. Kang and B. Yang, "Discrete time noncollocated control of flexible mechanical systems using time delay," *Transactions of the ASME, Journal of Dynamic Systems, Measurement and Control*, vol. 116, pp. 216-222, June 1994.

[50] VenturCom, Inc., *Real-time Extension (RTX) for Windows NT User's Guide*, version 4.2, http://www.vci.com, 1998.

[51] R. H. Cannon, Jr., *Dynamics of Physical Systems*, McGraw-Hill, New York, 1967.

[52] B. Armstrong-Helouvry, P. Dupont and C. Canudas De Wit, "A survey of models, analysis tools and compensation methods for the control of machines with friction," *Automatica*, vol. 30, no. 7, pp. 1083-1138, 1994.

[53] J. Christian Gerdes and Vijay Kumar, "An impact model of mechanical backlash for control system analysis," *Proceedings of the American Control Conference*, Seattle, WA, pp. 3311-3315, June 1995.

[54] S. Shankar Sastry and A. Isidori, "Adaptive control of linearizable systems," *IEEE Transactions on Automatic Control*, vol. 34, no. 11, pp. 1123-31, Nov. 1989.

[55] K. W. Lim, M. F. Rahman and J. X. Xu, "Comparing controllers for a flexible link," *Proceedings of the 1996 IEEE IECON, 22nd International Conference on Industrial Electronics, Control, and Instrumentation*, vol. 3, pp. 1978-83, 1996.

[56] Karl J. Åström and Tore Hägglund, *PID controllers: Theory, design and tuning*, Research Triangle Park Publications, 1995.

[57] J. H. Kim, J. Y. Jeon, S. W. Lee and K. Koh, "High-precision control of positioning systems with nonsmooth non-linearities," *Proceedings of the 35th IEEE Conference on Decision and Control*, pp. 4375-80, Kobe, Japan, 1996.

[58] W. Li and M. Rehani, "Modeling and control of a belt-drive positioning table," *Proceedings of IECON, 22nd IEEE International Conference on Electronics, Control and Instrumentation*, Taipei, Taiwan, vol. 3, pp. 1984-9, August 1996.

[59] C. C. Lee, "Fuzzy logic in control systems," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 20, no. 2, pp. 404-18, 1990.

[60] T. C. H. Huang, *High Performance Electric Drive Systems using Fuzzy Control*, Ph.D. Thesis, University of Washington, 1995.

[61] D. Driankov, H. Hellendoorn, M. Reinfrank, *An Introduction to Fuzzy Control*, Springer-Verlag, Berlin, 1993.

[62] J. J. Slotine and W. Li, *Applied nonlinear control*, Prentice-Hall, 1994.

[63] M. A. El-Sharkawi and C. H. Huang, "Variable structure tracking of DC motor for high performance applications", *IEEE Transactions on Energy Conversion*, pp. 643-650, December 1989.

[64] M. A. El-Sharkawi, "Development and implementation of high performance variable structure tracking control for brushless motor", *IEEE Transactions on Energy Conversion*, vol. 6, no. 1, pp. 114-119, March 1991.

[65] D. S. Reay, M. M. Moud and B. W. Williams, "On the appropriate use of fuzzy systems: fuzzy sliding mode position control of a switched reluctance motor," *1995 IEEE Intl. Symp. on Industrial Control*, Monterey CA, pp. 371-6, Aug. 1995.

[66] S. Assilian and E. Mamdani, "An experiment in linguistic synthesis with a fuzzy logic controller," *Int. J. of Man-Machine Studies*, vol. 7, pp. 1-13, 1975.

[67] K. F. Man, K. S. Tang, S. Kwong and W. A. Halang, *Genetic Algorithms for Control and Signal Processing*, Springer-Verlag, London, 1997.

[68] S. W. Kim and J. J. Lee, "Design of a fuzzy controller with a fuzzy sliding surface," *Fuzzy Sets and Systems*, vol. 71, pp. 359-67, 1995.

[69] K. J. Åström and B. Wittenmark, *Adaptive Control,* Addison-Wesley, 1989

[70] E. Ostertag and M. J. Carvalho-Ostertag, "Fuzzy control of an inverted pendulum with fuzzy compensation of friction forces," *International Journal of Systems Science*, vol. 24, no. 10, pp. 1915-1921, 1993.

[71] A. Tzes and P. Y. Peng, "Fuzzy neural network control for DC-motor micro-maneuvering," *Transactions of the ASME, Journal of Dynamic Systems, Measurement and Control*, vol. 119, no. 2, pp. 312-5, June 1997.

[72] C. G. Baril and P. O. Gutman, "Performance enhancing adaptive friction compensation for uncertain systems," *IEEE Transactions on Control Systems Technology*, vol. 5, no. 5, pp. 466-79, Sept. 1997.

[73] D. R. Seidl, T. L. Reineking and R. D. Lorenz, "Use of neural networks to identify and compensate for friction in precision, position controller mechanisms," *Conference Record of the IEEE Industry Applications Society Annual Meeting*, vol. 2, pp. 1937-44, 1992.

[74] D. Misir, H. A. Malki and G. Chen, "Design and analysis of a fuzzy proportional-integral-derivative controller," *Fuzzy Sets and Systems*, vol. 79, pp. 297-314, 1996.

[75] R. Palm, "Robust control by fuzzy sliding mode," *Automatica*, vol. 30, no. 9, pp. 1429-37, 1994.

[76] G. F. Franklin et al, *Digital Control of Dynamic Systems*, Addison-Wesley, 1998.

[77] E. D. Tung and M. Tomizuka, "Feedforward tracking controller design based on the identification of low frequency dynamics," *ASME Journal of Dynamic Systems, Measurement and Control*, vol.115, pp. 348-356, Sept. 1993.

[78] M. Tomizuka, "Zero phase error tracking algorithm for digital control," *ASME Journal of Dynamic Systems, Measurement and Control*, vol. 109, pp. 65-8, 1987.

[79] C. H. Chen, H. V. Brussel and J. Swevers, "Extended pole placement method with noncausal reference model for digital servo-control," *ASME Journal of Dynamic Systems, Measurement and Control*, vol. 117, pp. 641-4, Dec. 1995.

[80] G. C. Luh and G. Rizzoni, "Nonlinear system identification using genetic algorithms with application to feedforward control design," *Proceedings of the American Control Conference*, Philadelphia, pp. 2371-5, June 1998.

[81] *MATLAB System Identification Toolbox User's Guide*, version 4, The MathWorks, Inc., http://www.mathworks.com, 1998.

[82] *MATLAB Application Programming Interface User's Guide*, version 5, The MathWorks Inc., http://www.mathworks.com, 1998.

[83] J. H. Kim, J. Y. Jeon and K. Koh, "A novel evolutionary algorithm with fast convergence," *Proceedings of IEEE International Conference on Evolutionary Computation*, Perth, Australia, Nov. 1995.

[84] J. H. Kim, H. K. Chae, J. Y. Jeon and S. W. Lee, "Identification and control of systems with friction using accelerated evolutionary programming," *IEEE Control Systems Magazine*, pp. 38-47, August 1996.

[85] L. Davis (ed.), *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York, 1991.

[86] *MATLAB User's Guide*, version 5, The MathWorks, Inc., http://www.mathworks.com, 1998.

Appendix

A.1 Introduction to Neural Networks

Artificial Neural Networks (ANNs) constitute one of the important computational intelligence techniques, which try to emulate the human brain or biological neural systems. ANNs are made up of layers of neurons connected to each other by means of synapses or weights. Each neuron implements a multiple-input, single-output function, which is generally non-linear. Neural networks have been shown to be good non-linear approximators, and have been used for duplicating the dynamic input-output relationship of non-linear systems. In addition, neural networks perform very well when used for parameter estimation. Two classes of neural networks which have received considerable attention in the area of identification and control of dynamic systems are:

- multilayer feedforward neural networks, which are also known an multilayer perceptrons, and,
- recurrent networks, which can be viewed as generalized feedforward networks with some of the outputs being used as inputs after some delay [36].

Figure A.1 shows a typical three-layer feedforward neural network where the empty circles denote neurons whose output is a non-linear function of the sum of its inputs. All the lines represent connections between layers and each such connection has an associated weight. The filled circles in the input layer represent direct feedthrough connections i.e. no processing is done at these nodes. The outputs of the neurons in the last layer are the outputs of the whole neural network, which is why that layer is called the output layer. The intermediate layer does not directly receive any input and its outputs are connected to the neurons in the output layer. Hence it is called the hidden layer. In addition to the three inputs, a fixed bias input is also connected through weights to all the hidden and output neurons (all connections are not shown in the figure). The

multilayer perceptron shown here has 3 inputs, 1 hidden layer with 5 neurons and 3 outputs. It is referred to as having a 3-5-3 structure.

It has been shown that such a three-layer feedforward neural network can approximate

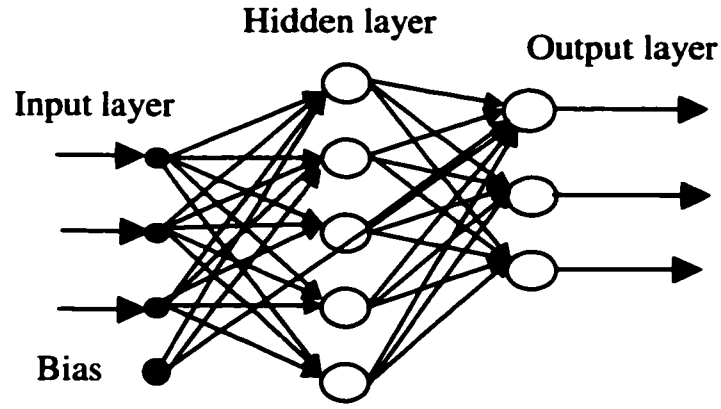


Figure A.1 Three layer feedforward neural network

any square integrable, non-linear, multiple-input, multiple-output function to any desired degree of accuracy under certain conditions [37].

Figure A.2 shows one form of a recurrent network known as a Jordan network, where some of the outputs are fed back as inputs to the network through a delay element. Many

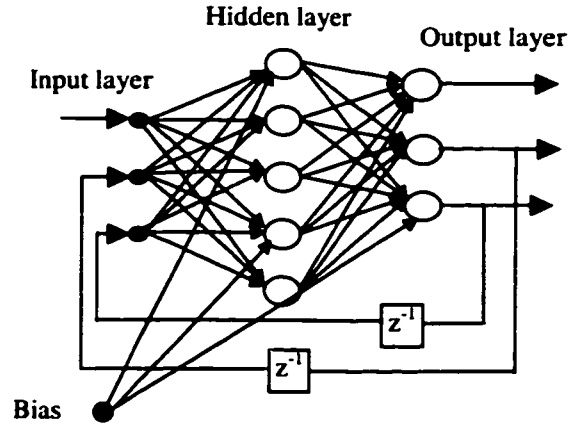


Figure A.2 Jordan recurrent network

other forms of recurrent networks have been proposed including Elman networks, where the hidden neuron outputs are fed back to themselves through a delay, fully recurrent networks and time-delay recurrent networks. While feedforward networks can

approximate any static, non-linear, MIMO function, recurrent networks are capable of accurately representing any non-linear, *dynamic* function.

A feedforward neural network processes data in a very simple fashion. The hidden layer neurons perform a weighted sum of all their inputs and then perform a non-linear function on the summation.. If the weight connecting the i^{th} input to the j^{th} hidden neuron is denoted by w_{ij}, and the inputs are denoted by x_i, the output of the j^{th} hidden neuron is given by:

$$y_j = f\left(\sum_i w_i x_i\right) \qquad \text{(A.1)}$$

where, f denotes the non-linear function. For neurons in the output layer, the same equation applies, with x_i now denoting the output of the i^{th} hidden neuron and w_{ij} denoting the weight connecting the i^{th} hidden neuron and the j^{th} output neuron. Many different functions are used for the non-linear function f, the most common being the sigmoidal function which is given in a general form by:

$$f = \frac{a}{1+\exp(-\lambda * net + c)} - b \qquad \text{(A.2)}$$

where, a, b, c and λ are the parameters of the sigmoidal function and net is the argument of the function i.e. the weighted sum of all the inputs to the neuron. In particular, λ is called the threshold of the function. With b and c equal to zero and a and λ equal to 1, the logistic function is obtained, while the hyperbolic tangent function is obtained when b and λ are equal to 1, a is equal to 2 and c is equal to zero.

A.1.1. Neural Network Training

The weights of a neural network are adjusted using some algorithm so as to reduce the error between the desired outputs and the NN outputs. This phase is known as neural network training and it is this operation which makes neural networks attractive for use in a wide variety of applications. Backpropagation (or steepest descent) is the most

commonly used algorithm for adjusting the weights of a neural network. If the outputs of a neural network are denoted by o_k, the desired outputs are denoted by d_k and the vector of all errors o_k - d_k is represented by $\boldsymbol{e}$, then, the performance index used for adjusting the weights is given by:

$$E = (1/2)\, \boldsymbol{e}^T \boldsymbol{e} \qquad \text{(A.3)}$$

This performance index is minimized by adjusting the weights of the network. If $\boldsymbol{m}$ is a vector containing all the weights of the network and $\boldsymbol{m}^*$ is the optimum weight vector, then a Taylor series expansion of E in the neighborhood of $\boldsymbol{m}$, neglecting higher order terms gives:

$$E(\boldsymbol{x}) = E(\boldsymbol{m}) + \nabla E\, \Delta \boldsymbol{m} + (1/2)\, \Delta \boldsymbol{m}^T \boldsymbol{H} \Delta \boldsymbol{m} \qquad \text{(A.4)}$$

where, $\Delta \boldsymbol{m}$ is the perturbation of weights around $\boldsymbol{m}$. ∇E and $\boldsymbol{H}$ are the gradient vector (first order) and the Hessian matrix (second order) of E with respect to $\boldsymbol{m}$. The optimum set of weights is found when we set $\partial E / \partial \boldsymbol{m}^* = 0$, which gives:

$$\Delta \boldsymbol{m} = \boldsymbol{m}^* - \boldsymbol{m} = - \nabla E\, \boldsymbol{H}^{-1} \qquad \text{(A.5)}$$

The main variation in the different methods used for adjusting the weights of the neural network is due to the approximation of the Hessian matrix. The backpropagation algorithm approximates the Hessian matrix as the identity matrix, multiplied by a constant, which is called the learning rate. This algorithm keeps the threshold of the sigmoidal function constant. The adaptive neuron algorithm also belongs to the category of gradient descent methods, but it also adjusts the threshold of the neurons in the hidden layer, during the training process. Algorithms which use the Hessian matrix by finding its inverse are called Newton methods, since they use the second derivative of the function $E(\boldsymbol{m})$. Newton methods are hence, computationally intensive. Some algorithms try to avoid this inversion by approximating the inverse of the Hessian matrix by some other matrix, which is easier to evaluate. These methods are called quasi-Newton methods, into which category falls the extended Kalman filter method.

A.2 Introduction to Fuzzy Control

Human beings use linguistic terms like *very hot* or *slightly old* in normal conversations. Although the implied meaning of such linguistic terms is known, it is not possible to express this meaning mathematically using conventional set theory. Zadeh introduced Fuzzy Sets in his landmark paper in 1965 [10]. The main difference between fuzzy and conventional sets is in the definition of membership of a given set. For conventional set theory, a number either is or is not a member of any given set. The membership can then be represented as a binary quantity: either 1 or 0. On the other hand, the membership of a number in a fuzzy set is defined in terms of a membership function, which generally varies between 0 and 1. The membership function represents a degree of membership of the value in the particular fuzzy set. For example, consider a set of old people. In conventional set theory, one can define such a set as having all people above a certain

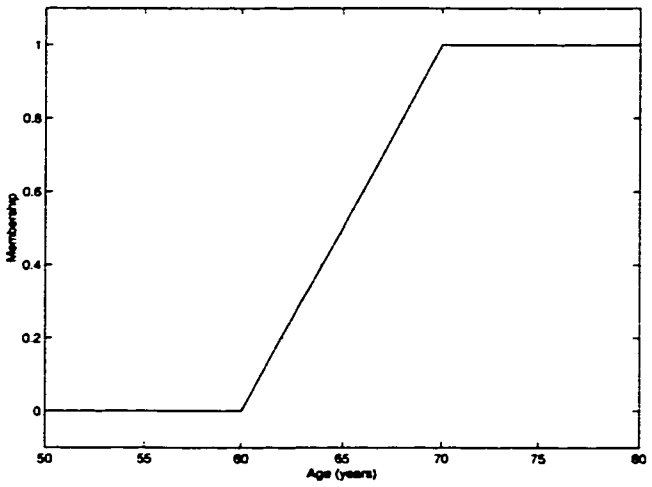


Figure A.3 Example of a membership function
for fuzzy sets

age, say 65, as members. If person A is 64 years old, he does not belong to this set of old people, while a person B who is 66 years old belongs to this set. Actual linguistic definitions are not so rigid. A fuzzy set of old people can then be defined as a membership function which is zero for any age below 60 and then increases steadily with a value of 1 for any age above 70. Such a membership function is shown in Figure A.3.

For this fuzzy set, the membership of person A is 0.4 and that of person B is 0.6. Thus, the membership represents a degree to which a person is considered old. Any person with an age above 70 has a membership of 1, indicating that the person is certainly old. Fuzzy set theory lays down a systematic way of dealing with linguistic constructs and defines all operations similar to conventional set theory [11].

Fuzzy control uses fuzzy set theory to define a non-linear controller and was first

Figure A.4 Components of a fuzzy controller

developed by Mamdani in 1975 [66]. A fuzzy controller has three components: a fuzzifier, a rule base and a defuzzifier as shown in Figure A.4 [61]. The inputs to the controller are crisp numbers and the outputs are also crisp number, but all the processing inside the controller is done using fuzzy variables. The first step is to convert the crisp inputs to memberships of each fuzzy set defined for the inputs. This operation is called *fuzzification* and the block that performs this operation is called a *fuzzifier*. These membership functions are used in the *rule base* which relates fuzzy values of the inputs to the output of the rule base. The output of the rule base is also a set of membership values of the fuzzy sets defined for the output variable. To interface with the physical world, this fuzzy variable has to be converted back to a crisp number. This is done by the *defuzzifier*.

A.2.1. Fuzzifier

The fuzzifier converts a crisp input to membership functions of the fuzzy sets defined for that input. Figure A.5 shows an example of membership functions defined for a fuzzy variable. Typically fuzzy sets are named linguistically as *LargeNegative*, *Zero* or some variation of these names. Given a crisp input value, the membership for each fuzzy set can be found from the membership functions. The figure shows five fuzzy sets labeled *LN*, *SN*, *Z*, *SP*, and *LP* using triangular membership functions. The membership

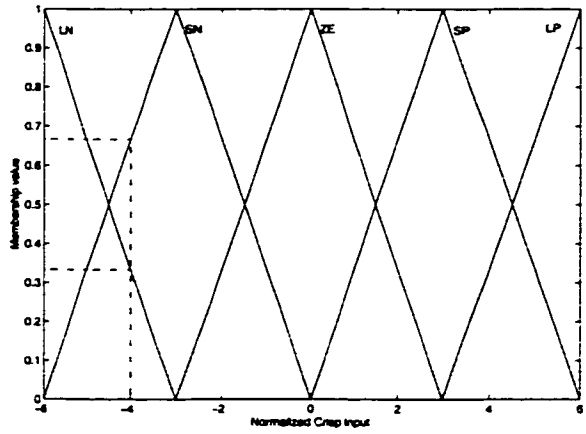


Figure A.5 Sample membership functions used by a fuzzifier

functions are defined over an input range of [-6, 6]. This is generally a normalized range selected for ease of programming. The actual range in terms of physical quantities is dependent on the normalization parameter (scaling factor or gain) for the particular variable. For example, if the error gain is 0.1, an error in the range of [-60, 60] is mapped to the normalized values of [-6, 6]. The membership functions at the two limits (LN and LP) extend to the physical limits of the variables. Thus, if error is > 60, with an error gain of 0.1, the membership for LP will be 1, with all other sets having zero membership values.

The figure shows an example of fuzzification. A crisp input of –4 is shown for which two sets have non-zero membership: *LN* and *SN*. The membership of *LN* is about 0.333 while that of *SN* is about 0.666. Thus, given the crisp input –4, the fuzzy variable is given by the set of memberships: [0.333 0.666 0 0 0].

A.2.2. Rule base

The rule base maps the fuzzy sets for the inputs to the fuzzy sets for the output and constitutes the core of a fuzzy controller. Table 10 shows a sample rule base. *SN*, *LN*, *LP* are notations for fuzzy sets named *Small Negative, Large Negative, Large Positive* respectively. The rule base has two dimensions corresponding to a fuzzy controller with

two inputs and one output. Each row gives the resultant output fuzzy set for each combination of input fuzzy sets. Thus, if input1 is LN and input2 is SP, then the output is SN and so on. The entire rule base can be described in terms of such IF-THEN statements or rules, which are OR-ed together. Given the input fuzzy set membership values, the rules are evaluated using the *compositional rule of inference* to obtain the output fuzzy sets.

Different compositional rules have been defined which use different functions for the AND and OR operations. Let *A* denote the fuzzy variable input1, *B* denote input2, *O* denote the output, *a* denote the value of input1, *b* denote the value of input2 and *o* denote

Table 10 Sample Rule base for a fuzzy controller

		Input 2				
		LN	SN	ZE	SP	LP
INPUT1	LN	LN	LN	LN	SN	ZE
	SN	LN	SN	SN	ZE	SP
	ZE	LN	SN	ZE	SP	LP
	SP	SN	ZE	SP	SP	LP
	LP	ZE	SP	LP	LP	LP

the value of the output. Then, using the minimum function for the AND operation, the rule:

IF *A* is LN AND *B* is SP, THEN *O* is SN, is evaluated by,

$$\mu_O^{SN-k}(o) = \min\left(\mu_A^{LN}(a), \mu_B^{LP}(b)\right) \qquad \text{(A.6)}$$

where μ denotes the membership function. The membership function of the output for the set SN is obtained as a result of evaluation of the above rule and is denoted by μ_O^{SN-k}. The superscript *SN_k* denotes that the membership is for the fuzzy set SN as a result of evaluating the k^{th} rule. All rules having the output SN are then combined using the OR operation to get the final membership of the output in the fuzzy set SN. If the maximum function is used for the OR operation, we get:

$$\mu_O^{SN}(o) = \max_i\left(\mu_O^{SN-i}\right) \qquad \text{(A.7)}$$

The same procedure is adopted for all rules to get all the fuzzy memberships of the output. This min-max inference is the most common although other kinds such as the min-sum inference and the product-sum inference are also widely used.

From the sample rule base it can be seen that many rules can give the same resulting fuzzy set for the output. e.g. the output is SN for four combinations of input1 and input2. However, it can be seen from the membership functions in Figure A.5 that for a given crisp value, only two membership functions overlap, so that at the most two sets will have non-zero memberships for each input. This implies that at most four rules will *be activated* simultaneously. This is typical of most fuzzy controllers. The advantages of this characteristic are:

- the rule base can be tuned very easily, since for some given inputs, depending on the performance of the controller, only the *active* rules need to be adjusted. This is in contrast to linear controllers, where changing any gain affects the system performance over its entire range of operation.
- the output of the rule base is converted to a crisp value using the defuzzifier. This provides for inherent smoothening of the controller output. Thus, a fuzzy controller is a true non-linear controller while gain-scheduling or multiple gain PID controllers obtain their non-linear behavior by using multiple linear regions and hence do not display the smoothness in transitions from one region/gain to another.

A.2.3. Defuzzifier

The output of the rule base is a fuzzy variable which is converted to a crisp value by the defuzzifier. Many defuzzification methods have been proposed, of which the center-of-area or *centroid* defuzzification is the most common. Using this method, the crisp output is given by:

$$o = \frac{\sum_{i=1}^{p} \mu_i A_i c_i}{\sum_{i=1}^{p} \mu_i A_i} \qquad \text{(A.8)}$$

where, μ_i = membership value of control input for the i^{th} fuzzy set of the output

A_i = area of the membership function for the i^{th} fuzzy set of the output

c_i = centroid of the membership function for the i^{th} fuzzy set of the output

p = number of fuzzy sets for the output

If the areas of all membership functions for the output are the same, the A_i term drops out of the defuzzification equation.

A.2.4. PD and PI type of fuzzy control

A two-input, one-output fuzzy feedback controller generally uses as inputs the error at a given instant and the change of error between the current and the previous instant [61]. The output can either be the control signal or a change in control signal. If the output is the control signal itself, then the fuzzy controller effectively implements a non-linear, variable gain PD controller. For example, if the output of the fuzzy controller is denoted by *u(k)*, and the inputs are denoted by *e(k)* and *ce(k)* for error and change of error respectively, then:

$$u(k) = f(e(k), ce(k)) \qquad \text{(A.9)}$$

where *f()* denotes the non-linear function that the fuzzy controller represents. On the other hand, if the output is a *change in control*, it is added to the control of the previous instant to obtain the control at the current instant. Then *u(k)* is given by:

$$u(k) = u(k\text{-}1) + \Delta u(k) \qquad \text{(A.10)}$$

where $\Delta u(k)$ is the output of the fuzzy controller. This yields a PI type of fuzzy controller, since the output of the controller is effectively integrated to obtain the actual control signal. The controller obtained in this manner is effectively a non-linear, variable gain PI controller. It should be noted that this kind of controller does not perform an

actual integration of the error or the change of error and in this regard is quite different from a linear PI controller.

A.3 Introduction to Evolutionary Algorithms

Evolutionary algorithms are a class of iterative, stochastic search techniques loosely tied to the Darwinian theory of evolution that is based on the *survival of the fittest* [16]. These techniques use multiple search agents to find the maximum value of a fitness function. The fitness function evaluates the match of a set of parameters to the problem under consideration. Each search agent is a set of such unknown parameters being searched and is referred to as a *chromosome*. Each parameter in the search agent is called a *gene*. Many evolutionary operators such as *crossover*, *mutation* and *selection* are used to create a new population of *chromosomes* for each iteration. The population at the start of the iterations is known as the *parent* population and the modified population obtained at the end is known as the *child* population. Evolutionary algorithms can be used in complex problems where traditional search techniques cannot be used. Examples are discontinuous, non-linear systems where optimization methods dependent on gradient information cannot be applied.

Evolutionary Algorithms are of two main types: Genetic Algorithms and Evolutionary Programs [14, 67]. Genetic Algorithms use strings of binary digits (bits) as chromosomes to encode the genes, while Evolutionary Programs work with floating point numbers. In this dissertation, Evolutionary Programs have been used and the principles used for these are explained below.

A.3.1. Evolutionary Programming

Evolutionary Programming (EP) uses an array of floating point numbers as a *chromosome* with each number representing a parameter being searched, or in EP

parlance, a *gene*. Selection, Crossover and Mutation operators are used in each iteration to find the next generation of the search population. The steps taken are as follows:

1. Initialize a population of chromosomes, with each chromosome being an array of floating point numbers. Ideally, the population is initialized to random values uniformly distributed over the possible range of variation of the parameters. This is generally not practically possible since the range of variation of parameters is not known beforehand.
2. Find the fitness value for each chromosome.
3. Perform the *Selection* operation. This is done by finding the sum of the fitness values of all chromosomes and then redistributing the chromosomes according to the ratio of the fitness of each chromosome to the total fitness. For example, if chromosome A has a fitness value of 2 and the total fitness of all chromosomes is 10, then after selection, 20% of the population will be copies of chromosome A.
4. Perform the *Crossover* operation by selecting chromosomes in pairs and then producing 2 child chromosomes. This is done by selecting a random dividing point P

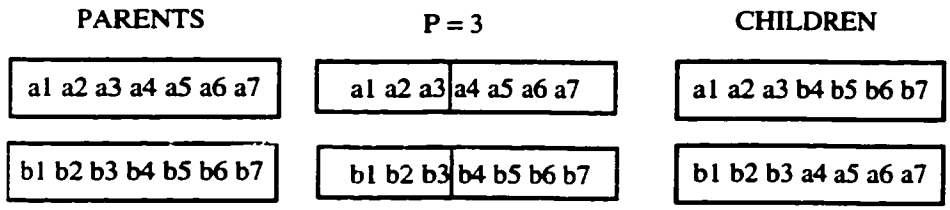


Figure A.6 The Crossover operation

within the array of chromosomes (e.g. if the number of parameters or genes is 7, any integer less than 7 can be selected as P). Then one new child chromosome is formed by combining the first P parameters from the first parent chromosome and the remaining chromosomes from the second parent chromosome. Similarly, the second child chromosome is formed by combining the first P chromosome from the second parent and the remaining chromosomes from the first parent. This is shown in Figure A.6 for P = 3.

5. *Mutation* is then applied to the child population either by randomly perturbing some of the parameters or by totally changing some parameters. The amount of mutation is determined by a mutation probability, which is a constant number between 0 and 1. For each gene, a random number is generated, if the number is greater than the mutation probability, the gene is perturbed otherwise it is left alone.
6. Finally the fitness values are found for all chromosomes in the children's population. Steps 3 through 6 are then repeated with the children's population for the previous iteration serving as the new parent population. These iterations are continued till the fitness function satisfies a stopping criterion or the maximum number of iterations specified is exceeded. A modification called the *king of the hill* approach is widely used, where, after the fitness values for the child population have been found, the best chromosome among the parent and child population combined is retained in the new population for the next generation. This assures that the solution will not worsen and can only improve or stay the same with every iteration, while at the same time the randomness of the search is maintained.

VITA

Amol Kulkarni was born on February 13, 1970 in Dhule, Maharashtra, India. He received his Bachelor of Engineering (B.E.) degree in Electrical Engineering from the University of Pune, Pune, India and the Master of Technology (M.Tech.) degree in Energy Systems Engineering from the Indian Institute of Technology, Mumbai, India in 1990 and 1992 respectively. Subsequently he worked as a Development Executive at The National Radio and Electronics Company Ltd. (NELCO), Mumbai, India from 1992-1994. His research interests include electric drives, motion control, power electronics, computational intelligence and control systems.

Education:

- B.E. Electrical Engineering, University of Pune, Pune, India, June, 1990.
- M.Tech. Energy Systems Engineering, Indian Institute of Technology, Mumbai, India, February 1992.
- Ph.D. Candidate, Electrical Engineering, University of Washington, Seattle, WA, 1994-Present.

Publications:

- Amol S. Kulkarni and M. A. El-Sharkawi, "Speed estimator for induction motor drives using an artificial neural network", *IEEE International Conference on Electric Machines and Drives (IEMDC '97)*, Milwaukee, 18-21 May 1997, pp. MD2-2.1-3.
- Amol S. Kulkarni, M. A. El-Sharkawi, R. J. Marks II, J. Xing, G. Andexler and I. Kerszenbaum, "Development and testing of a technique for on-line detection of shorts in field windings of turbine-generator rotors," *IEEE Transactions on Energy Conversion, In Press.*
- Amol S. Kulkarni and M. A. El-Sharkawi, "Intelligent Precision Position Control of Elastic Drive Systems," *In review IEEE Transactions on Energy Conversion.*