DEVELOPMENT OF HIGH-PERFORMANCE CONTROL AND PARALLEL PROCESSOR IMPLEMENTATION FOR INDUCTION MOTOR DRIVES

WALI LI, B.Eng.

Development of High-Performance Control and Parallel Processor Implementation for Induction Motor Drives

by

©Wali Li, B. Eng.

A thesis submitted to the School of Graduate

Studies in partial fulfillment of the

requirements for the degree of

Master of Engineering

Faculty of Engineering and Applied Science

Memorial University of Newfoundland

February 1992

St. John's Newfoundland Canada

ISBN 0-315-73312-8

Canada

To my wife

&

my parents

Abstract

Implementations of the most popular scheme of indirect field-oriented control for AC induction motor drives still suffer from difficulties due to the sensitivity of rotor parameters in decoupling torque and flux and control algorithm complexity in meeting real-time control requirements.

This thesis presents solutions to some of these problems. A new parameter adaptation control scheme, named rotor flux orientation control, is proposed and verified by digital simulations which show a very good result in dynamically correcting the mismatched value in the slip frequency calculator that usually occur when temperature rises or flux saturates. This method is based on the principle of feedback control which uses the detected rotor flux orientation as a feedback control signal.

For processing complex control algorithms in real-time, a new parallel processor architecture, using T800 Transputers, is proposed. Implementation results show that with this parallel processing controller complex control algorithms such as field-oriented control and parameter adaptive control together can be handled in very short processing times thus meeting fast dynamic control demands.

In addition, the design of indirect field-oriented control of AC induction motor is studied. A design method for linear control of AC induction motor control based on the indirect field-oriented control scheme is proposed. Decoupling of torque current and flux current is achieved by applying feed-forward controllers.

A new control scheme, named extended indirect field-oriented control, is proposed and verified by simulation results which show an excellent control performance that can not be achieved by a linear control scheme.

Furthermore, the incorporation of fault tolerance for Transputer communication link failures and processor failures is studied and some simulation tests have been carried out. As a result of the fault tolerance mechanisms incorporated, control system reliability is improved.

Acknowledgement

The author wishes to acknowledge the financial support from the School of Graduate Studies, Dr. R. Venkatesan's research grant and the graduate support from the Faculty of Engineering and Applied Science of Memorial University of Newfoundland which made this work possible.

Special thanks and sincre appreciation to Dr. R. Venkatesan, for acting as my research supervisor and providing advice, guidance and encouragement throughout the study.

Many thanks to Dr. J. E. Quaicoe and Dr. C. R. Moloney, for many valuable discussions and useful suggestions.

I also wish to thank Dr. R. Venkatesan and Dr. C. R. Moloney for their assistance in preparing this thesis.

Help from the technical staff at Computer Aid Engineering Center for using facilities is gratefully appreciated.

Finally, the dedication is due to my wife and my parents, for their constant encouragement and support.

Contents

List of Figures

List of Tables

Notation

i_d, i_q : direct- and quadrature-axis stator currents in the synchronous reference frame.

i_{dr}, i_{qr} : direct- and quadrature-axis rotor currents in the synchronous reference frame.

u_d, u_q : direct- and quadrature-axis stator voltages in the synchronous reference frame.

u_{dr}, u_{qr} : direct- and quadrature-axis rotor voltages in the synchronous reference frame.

$i_{\alpha\beta}$: alpha-beta axes current in the stationary reference frame

i_α, i_β : alpha- and beta-axis stator currents in the stationary reference frame.

$u_{\alpha\beta}$: alpha-beta axes voltages in the stational reference frame

u_α, u_β : alpha- and beta-axis stator voltages in the stationary reference frame.

$I_d(k), I_q(k)$: discrete direct- and quadrature-axis stator currents in the synchronous reference frame

$U_d(k), U_q(k)$: discrete direct- and quadrature-axis stator voltages in the synchronous reference frame

w_s : stator angular frequency

w_r : rotor angular frequency

w_{sl} : slip angular frequency

$\vec{\psi}_r$: rotor flux

ψ_{dr} : rotor flux component in d axis

ψ_{qr} : rotor flux component in q axis

$\psi_{\alpha\beta}$: stator flux in $\alpha - \beta$ axis

ψ_α : stator flux component in α axis

ψ_β : stator flux component in β axis

$\psi_{\alpha r}$: rotor flux component in α axis

$\psi_{\beta r}$: rotor flux component in β axis

θ_s : stator electrical angle

θ_r : rotor angle

θ_{sl} : slip angle

$*$: suffix denoting the reference value

R_r : rotor resistance

R_s : stator resistance

L_r : rotor inductance

L_s : stator inductance

M : mutual inductance

D : viscous friction coefficient

J : total inertia

T : motor torque

T_L : load torque

$\hat{T}$: estimated torque

n_p : number of pole pairs

N : number of pole

p : differential operator

s : Laplace operator

mmf : magnetomotive force

ρ : $1 - \frac{M^2}{L_s L_r}$

T_r : $\frac{L_r}{R_r}$

K_p : controller proportional gain

K_F : flux orientation controller proportional gain

τ : controller integral time constant

e : controller input

y : controller output

T_0 : sampling time

rpm : revolution minute per

Nm : Newton meter

δ : departure angle

Chapter 1

Introduction

Electric machines have been a workhorse of industry for many years. The three basic electric machines - DC, induction, and synchronous - have served industrial needs for nearly a century [1]. Although the induction machine is superior to the DC machine with respect to size, weight, rotor inertia, efficiency, maximum speed, reliability, cost, etc., because of the former's highly non-linear dynamic structure with strong dynamic interactions, it requires more complex control schemes than a separately excited DC machine. As for DC machines, torque control is achieved by controlling the motor current. However, in contrast to DC machines, in AC machines, both the phase angle and the modulus of the current have to be controlled, or in other words, the current vector has to be controlled. Furthermore, in DC machines, the orientation of the field flux and armature mmf is fixed by the commutator and the brushes, while in AC machines the field flux and the spatial angle of the armature m.m.f. require external control. In the absence of this control, the spatial angles between the various fields in AC machines vary with the load and yield an unwanted oscillating dynamic response [2]. With field-oriented control of an AC machine, the torque-and flux-producing characteristics are similar

to those of a separately excited DC machine and the system will adapt to any load disturbances and/or reference value variations as fast as a DC machine.

In the past such control techniques would have not been possible because of the complex hardware and software required to solve the complex control problem. Field-oriented control techniques incorporating fast microprocessors have made possible the application of AC motor drives for high performance applications where traditionally only DC motor drives were applied.

Field-oriented control techniques are now being accepted almost universally for high performance control of AC machines. Such control methods were developed in Germany in the early 1970's. Blaschke [3] developed the direct method of field-oriented control and Hasse [4] invented the indirect method. At present, the field-oriented control techniques have been implemented in terms of four types (rotor flux-oriented control, magnetizing- flux-oriented control, rotor-oriented control, and stator flux-oriented control). The most popular method is the rotor flux-oriented control because of its easy implementation, robustness and fast dynamic response [5]. In rotor flux-oriented control, there are two schemes: direct and indirect field-oriented control. As it has easy implementation, practical use and robustness, indirect field-oriented control has been given extensive attention in recent years [6][7][8][9].

However, the system response of indirect field-oriented control is limited by the accuracy with which the rotor time constant is known. Unfortunately, the rotor time constant varies with both airgap flux and temperature so that it is difficult to keep the controller in tune. Means for estimating and adjusting the rotor time constant in the slip frequency calculator is currently an area of intense activity [6].

The researchers are attempting to find a suitable parameter-adaptation scheme for optimum decoupling torque and flux under various operating conditions.

On the basis of field-oriented control, modern control techniques have been employed for obtaining a higher dynamic control performance. Modern control theory is certainly more difficult to apply for AC drives. However, for the AC drives with field-oriented control used in the inner core, there is no difference in the dynamics between the AC and DC drives. Therefore, it is expected that modern control theory will be applied to AC drives with field-oriented control in the inner loop [2]. Applying modern control theory to AC motor drive control remains a challenge because of the large time critical computation work requirement. Increased computational speed is, of course, the primary benefit of parallel processing. This allows the system to be controlled more quickly and gives the choice of added complexity in the control algorithm.

Fault tolerance can be realized in a parallel processing system by organizing computational operations in a distributed sense, so that an operation failure results in performance degradation rather than a complete breakdown of the controller.

Today, motion control is an area of technology that embraces many diverse disciplines, such as electrical machines, power semiconductor devices, converter circuits, dedicated hardware signal electronics, control theory, microcomputers, LSI/VLSI circuits, sophisticated computer-aid design techniques and system reliability techniques [10] (Fig.1.1). Each of the component disciplines is undergoing an evolutionary process, and is contributing to the total advancement of machine drive control technology.

The aim of this thesis is to investigate the characteristics of indirect field-

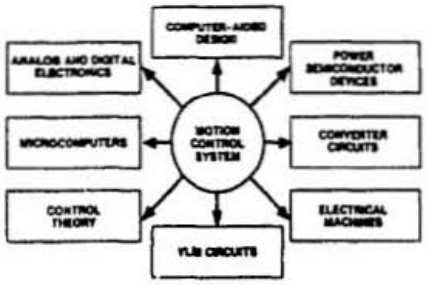


Figure 1.1: Motion control system-An interdisciplinary technology [10]

oriented control of an AC induction motor and to develop new schemes for solving problems in indirect field-oriented control of AC induction motor.

Due to the complexity of algorithms for indirect field-oriented control, most recently developed parameter adaptation control schemes are restricted to motor steady operation for correcting the mismatched value in the slip calculator.

In this thesis, a new easily implemented rotor parameter adaptation control is expected to correct the mismatched value in both motor steady operation and transient operation.

Considering the complexity of indirect field-oriented control and modern control in AC drives, a new VLSI product, known as a Transputer, is chosen as the parallel processing controller for the real-time control requirement. Simulations of indirect field-oriented control on multi-transputer networks have been carried out to examine the parallel processing effectiveness. The corresponding execution times

for control algorithms and some fault tolerance test results for improving control system reliability have been obtained.

A linear control model of an AC induction motor based on indirect field- oriented control have been studied for the purpose of design of PI or PID controllers by using the classical methods such as root-locus approach or frequency domain scheme.

This thesis is organized in eight chapters. In Chapter 2, the principle of indirect field-oriented control is described. An overview of the state-of-the-art of microprocessor-based implementations of field-oriented control of AC induction motor and the development of rotor parameter adaptation control for indirect field-oriented control of AC induction motor is presented.

In Chapter 3, in order to investigate the characteristics of indirect field- oriented control, an analytical model of indirect field-oriented control of AC induction motors has been set up and a series of simulations have been carried out. On the basis of field-oriented control, the conditions for obtaining a truly linear control model of the AC induction motor are studied. Finally, a new proposal for achieving excellent properties from field-oriented control is made and the corresponding simulation has been carried out for verifying this proposal.

In Chapter 4, a new parameter adaptation scheme for indirect field-oriented control of AC induction motors is developed and simulation results are presented, Finally, results are discussed.

In Chapter 5, a brief introduction to the principles of parallel processing and the transputer is presented.

In Chapter 6, a parallel processing controller using five Transputers for indirect field-oriented control of AC induction motor is implemented and corresponding

simulations are carried out. Execution times and some fault tolerance tests are examined.

Chapter 7 presents conclusions.

The last Chapter proposes suggestions for future study.

Chapter 2

Review of Indirect Field-Oriented Control of Induction Motor

2.1 Principles of Indirect Field-Oriented Control [35]

In recent years, the problem of obtaining fast torque response from A.C. machines has received considerable attention. However, unlike the D.C. machines, the voltage, current, torque and speed of an A.C. motor are all interdependent, which results in a highly non-linear dynamic structure. The possibility of obtaining high performance from the A.C. motor is very desirable. A control strategy for achieving this is termed *vector* or *field-oriented* control. This was originally derived from Blackshe; extended and generalized earlier work done by Hasse; similar ideas were expressed by several other researchers: Abbondanti, Nabae et al., Plunkett, Bose and Leonhard [11].

In principle, the strategy involves the transformation of the machine dynamics into those of a *pseudo-DC machine equivalent* in which a torque and field component of machine current are obtained. These current components are independent, if they are measured in the so-called *field co-ordinates*. In this way, the AC motor may

be controlled in the same manner as the DC motor, thus leading to an improved transient reponse. Figure 2.1 explains the indirect field-oriented control with the help of a phasor diagram. The α-β axes are fixed on the stator while the d-q axes rotate at synchronous angular velocity w_s as shown. At any instant, the q electrical axis is at angular position θ_s with respect to the β axis. The angle θ_s is given by the sum of rotor angular position θ_r and slip angular position θ_{sl}, where $\theta_s = w_s t$, $\theta_r = w_r t$, and $\theta_{sl} = w_{sl} t$. The rotor flux $\vec{\psi}_r$, consisting of the air gap flux and the rotor leakage flux, is aligned to the d axis as shown. Therefore, for decoupling control, the stator flux component of current i_d and the torque component of current i_q are to be aligned to the d and q axes, respectively.

We can write the following equations from rotating frame $d-q$ equivalent circuits as shown in Fig. 2.2:

$$\frac{d\psi_{qr}}{dt} + R_r i_{qr} + (w_s - w_r)\psi_{dr} = 0 \tag{2.1}$$

$$\frac{d\psi_{dr}}{dt} + R_r i_{dr} - (w_s - w_r)\psi_{qr} = 0. \tag{2.2}$$

Again,

$$\psi_{qr} = L_r i_{qr} + M i_q \tag{2.3}$$

$$\psi_{dr} = L_r i_d + M i_d. \tag{2.4}$$

From equation (2.3) and (2.4),

$$i_{qr} = \frac{1}{L_r}\psi_{qr} - \frac{M}{L_r} i_q \tag{2.5}$$

$$i_{dr} = \frac{1}{L_r}\psi_{dr} - \frac{M}{L_r} i_d. \tag{2.6}$$

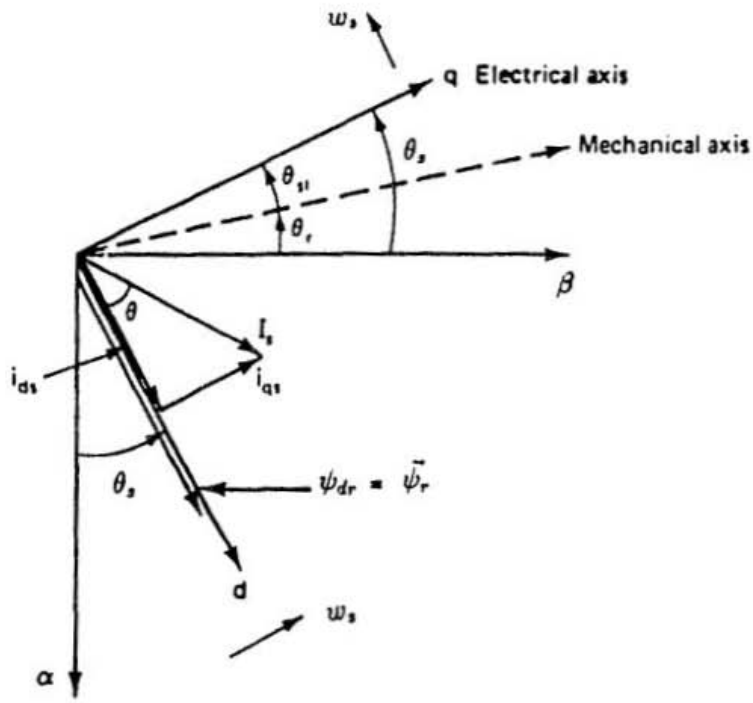


Figure 2.1: Phasor diagram for indirect field-oriented control [35]

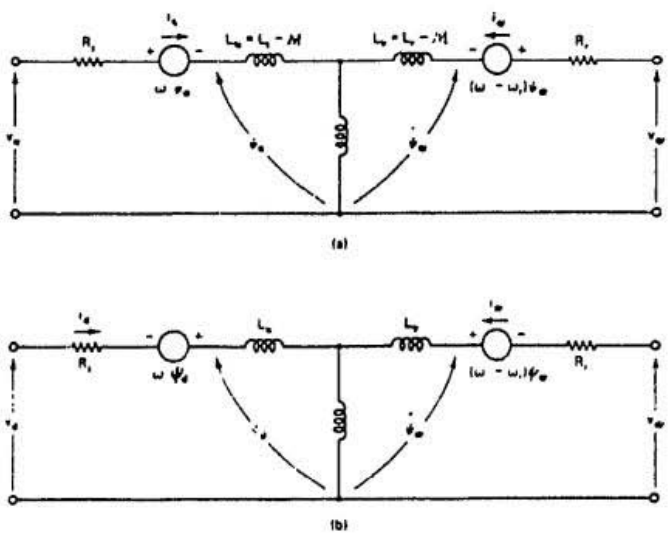


Figure 2.2: D-Q reference frame equivalent circuits. a) Q axis equivalent circuit, b) D axis equivalent circuit [35]

The rotor current from equations (2.1) and (2.2) can be eliminated by substituting equations (2.5) and (2.6) as

$$\frac{d\psi_{qr}}{dt} - \frac{M}{L_r}R_r i_q + \frac{R_r}{L_r}\psi_{qr} + w_{sl}\psi_{dr} = 0 \tag{2.7}$$

$$\frac{d\psi_{dr}}{dt} - \frac{M}{L_r}R_r i_d + \frac{R_r}{L_r}\psi_{dr} - w_{sl}\psi_{qr} = 0, \tag{2.8}$$

where $w_{sl} = w_s - w_r$.

For decoupling control, it is desirable that

$$\psi_{qr} = \frac{d\psi_{qr}}{dt} = 0$$

$$\psi_{dr} = \vec{\psi}_r = \text{constant}$$

$$\frac{d\psi_{dr}}{dt} = 0.$$

Substituting the first one condition, equations (2.7) and (2.8) can be simplified as

$$w_{sl} = \frac{M}{\vec{\psi}_r}(\frac{R_r}{L_r})i_q \tag{2.9}$$

$$\frac{L_r}{R_r}\frac{d\vec{\psi}_r}{dt} + \vec{\psi}_r = Mi_d. \tag{2.10}$$

Again, the torque as a function of rotor flux and stator current can derived as follows: The stator flux linkage relations can be written from Fig. 2.2 as:

$$\psi_q = Mi_{qr} + L_s i_q \tag{2.11}$$

$$\psi_d = Mi_{dr} + L_s i_d. \tag{2.12}$$

Substituting equations (2.5) and (2.6) in equations (2.11) and (2.12), the following equations are obtained:

$$\psi_q = (L_s - \frac{M^2}{L_r})i_q + \frac{M}{L_r}\psi_{qr} \tag{2.13}$$

$$\psi_d = (L_s - \frac{M^2}{L_r})i_d + \frac{M}{L_r}\psi_{dr}. \tag{2.14}$$

The torque equation as a function of stator current and fluxes is

$$T_e = \frac{3}{2}(\frac{P}{2})(i_q\psi_d - i_d\psi_q). \tag{2.15}$$

Equations (2.13) and (2.14) can be substituted in equation (2.15) to eliminate the stator fluxes, therefore

$$T_e = \frac{3}{2}(\frac{P}{2})\frac{M}{L_r}(i_q\psi_{dr} - i_d\psi_{qr}). \tag{2.16}$$

Substituting $\psi_{qr} = 0$ and $\psi_{dr} = \vec{\psi}_r$, the torque expression is:

$$T_e = \frac{3}{2}(\frac{P}{2})\frac{M}{L_r}i_q\vec{\psi}_r \tag{2.17}$$

or, using equation (2.10),

$$T_e = \frac{3}{2}(\frac{P}{2})\frac{M^2}{L_r}i_q i_d. \tag{2.18}$$

The equations above, together with the mechanical equation (2.19) describe the machine model in decoupling control as shown in Fig. 2.3.

$$(\frac{2}{P})J\frac{dw_r}{dt} = T_e - T_L \tag{2.19}$$

The developed torque T_e responds instantaneously with current i_q, but has delayed response due to i_d in equation (2.10). The analogy of the model with a separately excited DC machine is obvious.

For implementation of indirect field-oriented control, it is necessary to take equations (2.9) and (2.18) into consideration. As an example, Fig. 2.4 shows a position servo system using the indirect method of field-oriented control. The flux

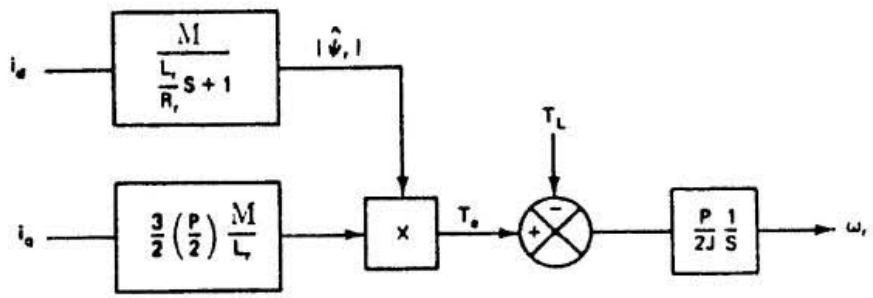


Figure 2.3: Block diagram of machine model with decoupling control [35]

component of current i_d^* for the desired rotor flux $\vec{\psi}_r$ is determined from equation (2.10) and is maintained constant here. The torque component of current i_q^* is derived from the speed control loop as usual. The set value of slip ω_{sl}^* is related to current i_q^* by equation (2.9). The slip-angle vectors $sin\theta_{sl}^*$ and $cos\theta_{sl}^*$, which determine the desired electrical axis with respect to the rotor-mechanical axis, are generated in a feed-forward manner from the ω_{sl}^* signal through a VCO, a counter, and a ROM-based sin/cos generator. The rotor-position vectors $cos\theta_r$ and $sin\theta_r$ are obtained from angle encoder and are added with the slip vectors to obtain the $cos\theta_s$ and $sin\theta_s$ signals as follows:

$$cos\theta_s^* = cos(\theta_r + \theta_{sl}^*) = cos\theta_r cos\theta_{sl}^* - sin\theta_r sin\theta_{sl}^* \tag{2.20}$$

$$sin\theta_s^* = sin(\theta_r + \theta_{sl}^*) = sin\theta_r cos\theta_{sl}^* + cos\theta_r sin\theta_{sl}^* \tag{2.21}$$

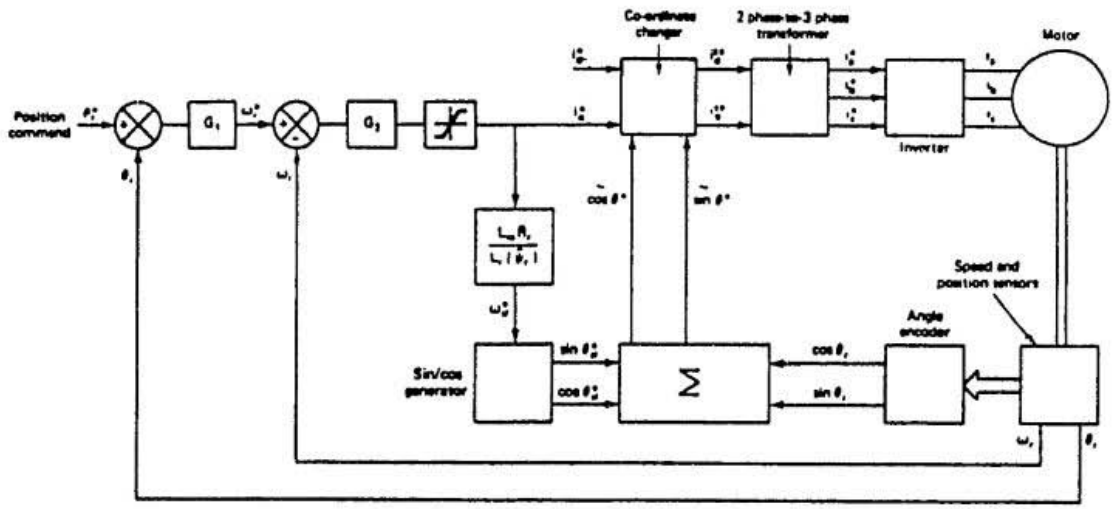


Figure 2.4: Position servo system with indirect field-oriented control [35]

2.2 Field-Oriented Control Using Microprocessors

Control of the AC machine is considerable more complex than control of the DC machine. The complexity increases for higher performance requirements. The reason is that AC machine dynamics are more complex, intricate signal processing is required, and the variable frequency supply requires more complex control than does the DC converter.

Recent achievements in LSI and VLSI technology led to practical utilization of the microprocessor. The microprocessor makes it possible to realize very complicated control strategy that is difficult or even impossible to realize with conventional analog circuitry. When the microprocessor is used in variable speed motor drives, it gives high accuracy surmounting the nonlinearities involved in motor characteristics, power conversion equipment and sensing devices, and providing against the drifts of temperature and source voltage.

The control system shown in Fig. 2.5 has been implemented on an Intel 8086 microcomputer to which a signal processor NEC 7720 was attached as a high speed arithmetic unit. This makes it possible to compute the flux model, the transformation and the inner control loops with 125 μs sampling time which is adequate even for high speed control [11].

It is obvious, of course, that with faster dynamics of the system implementation of optimal control and adaptive control of multi-variable systems will become more and more complex. In this respect, the uniprocessor based motor control is limited in offering higher computing speed. In recent years, multiprocessor based motor control systems have received considerable attention. Increased computa-

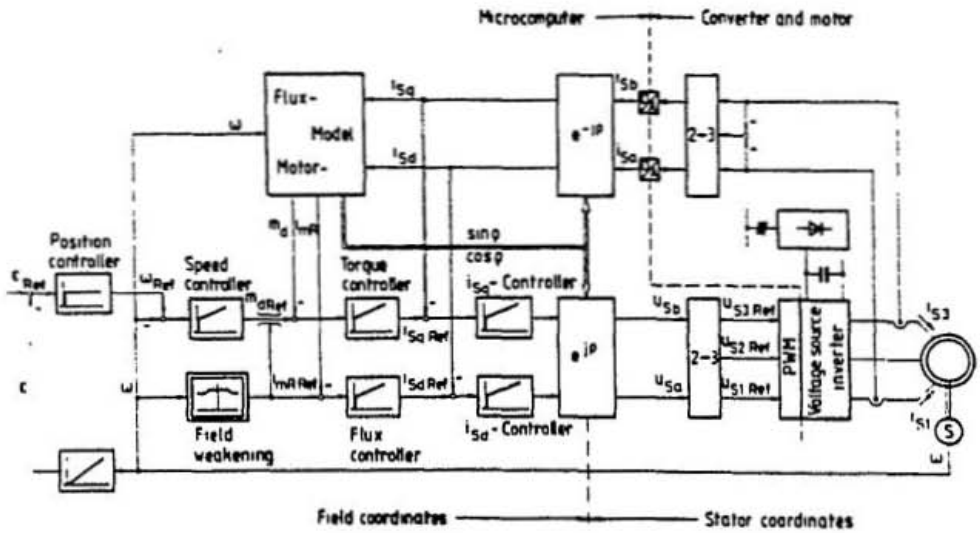


Figure 2.5: 8086-Microcomputer control of AC motor [11]

tional speed is , of course, the primary benefit of parallel processing. This allows faster systems to be controlled and gives the control choice of added complexity in the control algorithm. Easy expansion, within a uniform hardware and software environment, is another feature of concurrent control system because it is possible to add more processors as required. Parallel processing also offers a closer relationship between the inherent parallelism expressed at the design stage and the hardware implementation.

One of the typical applications using multiprocessor applications for field-oriented control was developed in 1985 by Fumio Harashima et al [12]. In their proposal, a three microprocessor based AC motor controller was used as shown in Fig. 2.6. With the parallel controller, two single-chip microcomputers Intel-8031 (12 MHz) are assigned for $3\phi/2\phi$ to $2\phi/3\phi$ signal conversions, respectively. These signal conversions are initiated by the interrupt signal every 1 ms. One 16-bit microprocessor Intel-8086 (5 MHz) executes both the vector control and speed control programs. The execution time for one cycle of the control program is about 2.2 ms. This multiprocessor configuration reduces the workload of the main processor Intel-8086 and permits the realization of more sophisticated control such as adaptive control, optimal control, and so on. Another advantage from this configuration is that it is very flexible to be used in various control types, such as V-type control and I-type control.

In the same year, Kenji Kubo et al [13] proposed a fully digitized AC motor controller by using multiprocessor system. The configuration of the proposed controller is shown in Fig. 2.7. The motor drive system is composed of a power supply, PWM inverter, induction motor, and speed controller.

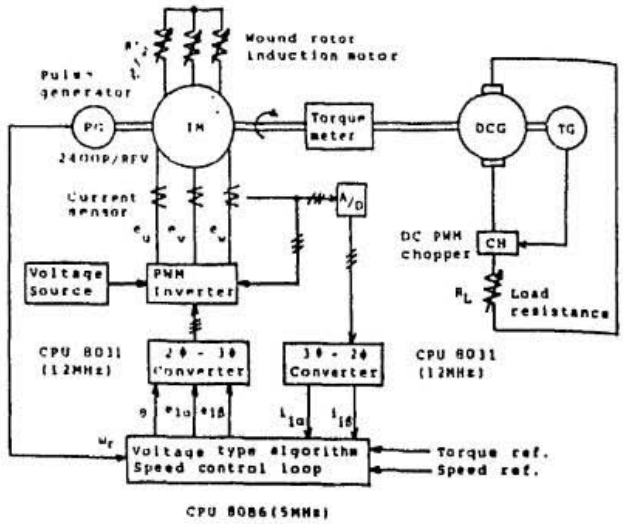


Figure 2.6: Hardware configuration of experimental system [12]

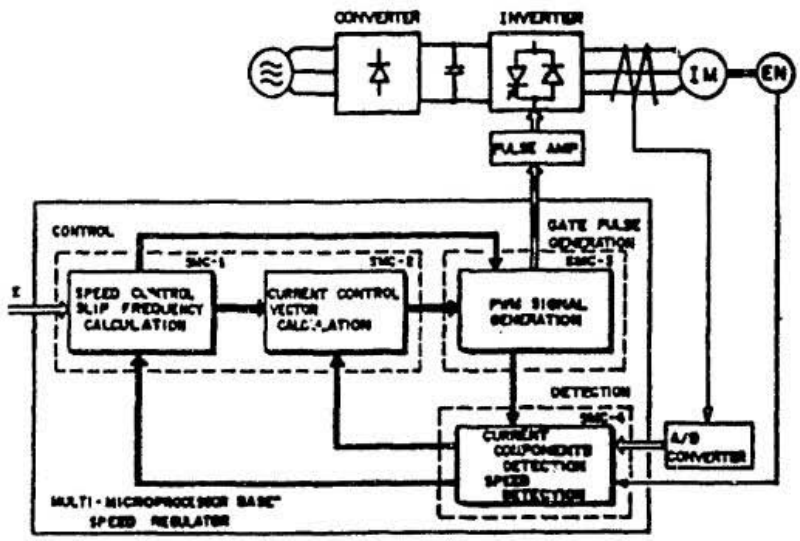


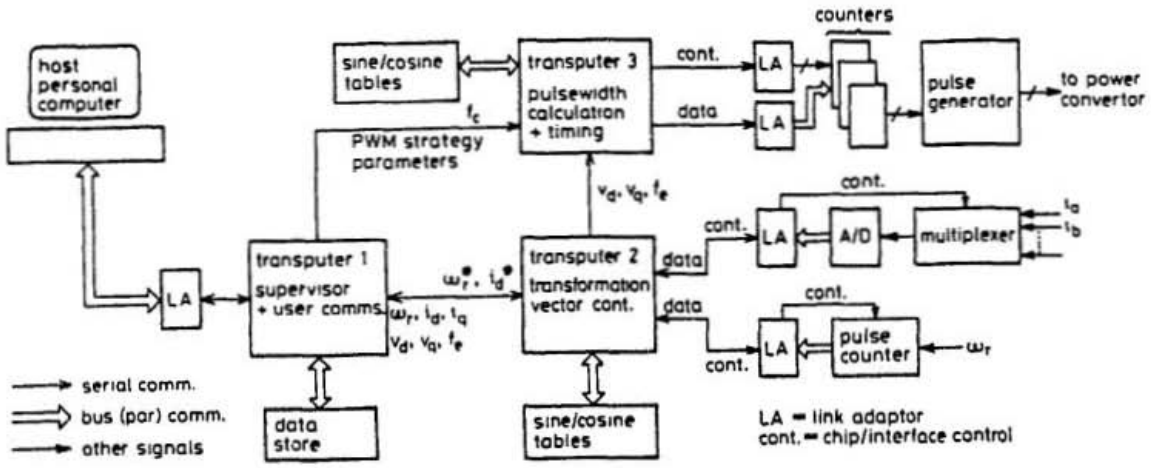
Figure 2.7: Configuration of speed regulator for AC motor [13]

Based on a speed reference and detected signals of motor current and speed, PWM gate pulses are generated according to the control processing. The PWM inverter operates following these gate pulses to regulate motor speed. Four microprocessors are used in the speed controller as indicated by the dotted line in Fig. 2.7. Two microprocessors are assigned to control processing, and other two are used for gate pulse generation and detection processing, respectively.

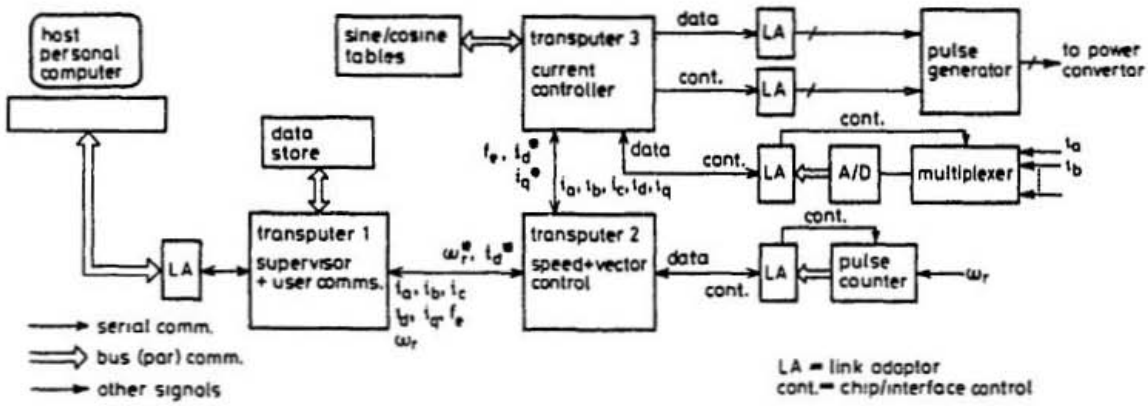
The execution times of SMC 1, SMC 2, SMC 3 and SMC 4 were 750, 700, 380 and 600μs, respectively [13].

With the development of VLSI, the 32-bit transputer systems have been used as promising parallel processing systems for real-time digital control applications, such as DC motor or AC motor control.

The first attempt in using Transputer for AC motor control was made by D.I. Jones and P.J. Fleming [14]. Another Transputer-based high performance AC motor control system, was developed by G.M. Asher et al [15]. In their proposal, a parallel processing controller has been exploited so that very complex control algorithms could be implemented in real-time to improve the AC motor control performance. Three vector control types are presented and implemented by a uniform Transputer-based parallel processing system. The three vector control types are V-type, current-controlled V-type and I-type. Implementations using parallel processing system are illustrated in Fig. 2.8(a) and 2.8(b). The Transputer T_2 handles speed and line current acquisition, speed control, slip and inverter frequency calculation, open-loop stator dynamic compensation and current acquisition for monitoring. Speed is sampled every $5ms$ while current is sampled at 250 μs. Consequently, we can conclude that the functions of a generalized drive controller



(a)



(b)

Figure 2.8: Transputer-based implementations of induction motor control. (a) V-type, (b) I-type [15]

exhibit a natural parallelism. In particular, a three-way parallelism exits between control, supervisor/communication and background computing functions.

2.3 Parameter Identification and Adaptive Control

The indirect field-oriented control basically involves a feed-forward control scheme in which the motor slip frequency is calculated from the estimated (commanded) stator current and an estimate of the rotor time constant. The desired slip frequency is added to the measured rotor speed to form the frequency command which is fed to the inverter. When properly adjusted, the indirect field-oriented control method provides torque response which can exceed the equivalent DC motor [1]. However, the speed of response is limited by the accuracy with which the rotor time constant varies with both air gap flux and temperature so that it is difficult to keep the controller in tune. Means for estimating and adjusting the rotor time constant in the slip frequency calculator is currently an area of intense activity.

L. J. Garces [17] and M. Koyama et al.[25] proposed the correcting function for progressively estimating the rotor time constant in transient state. The drawback of this method is that a long correcting time is needed for obtaining the actual rotor time constant.

K. Ohnishi et al.[26] and Y. Ueda et al.[27] developed the model reference adaptive control system . These two schemes involve much more computations for estimating the adaptive gain. The estimated results are affected by load disturbances.

L. C. Zai and T. A. Lipo [19] proposed a method which used an extended *Kalman Filter* approach to estimate the rotor time constant. The drawback of this scheme

is that the actual computation for estimating the rotor time constant took about 30 seconds of CPU time on a high processing speed MC68000-based microprocessor.

T. Matsuo and T. A. Lipo [18] developed another method in which a prescribed negative sequence current perturbation signal was injected into the AC motor and then the corresponding negative sequence voltage was detected for obtaining the rotor parameters. This scheme requires additional hardware and may induce a strong second harmonic torque pulsation due to the interaction of the position and negative rotating components of mmf. [7].

The recently reported results have still been concentrated on parameter identification or model reference adaptive control. As an example, C. C. Chan [7] proposed an effective method for rotor resistance identification. This method is based on the proper selection of coordinate axes: the d axis of the rotating frame is set to be coincident with the stator current vector. A direct estimate of rotor resistance can be derived from the reference frame. The advantage of this approach lies in its simplicity and accuracy for rotor resistance identification. However, correct results from this identification method can only be obtained when motor is working in steady state and a load should be applied.

M. B. Zhou and W. Qu [24] developed a scheme where the rotor time constant is derived from the vector diagram of T-I type equivalence. This scheme is more or less similar to that of C. C. Chan which can be applied only when motor is working in steady state.

R. D. Lorenz and D. B. Lawson [8] proposed a simplified approach of model reference adaptive control system to continuous on-line tuning for field-oriented control. Due to its simplicity, the primary advantage of the proposed technique is

the ease of implementation of the adaptive control methodology given the . This method could be affected by applied load [9] and takes long time for control convergency.

Based on the above review of microprocessor-based controller implementation and parameter adaptation control for field-oriented control, it can be concluded that complex field-oriented control of AC induction motor can be realized with the help of microprocessors. In addition, various parameter adaptation control schemes have been developed and applied to solve the problem of parameter sensitivity in indirect field-oriented control.

However, microprocessor-based controllers reported to date in the field-oriented control of AC induction motor have not been developed adequately to meet higher performance requirements. Meanwhile, the developed parameter adaptation control schemes can only be applied when motor works in steady state.

With an intent to solve these problems, some solutions are proposed in the subsequent chapters.

Chapter 3

Modeling and Simulation of Indirect Field-Oriented Control System

3.1 Introduction

In Chapter 2, recent developments in the theory of indirect field-oriented control of AC induction motor and its applications have been reviewed. In this chapter, modeling and simulation of indirect field-oriented control of induction motor with squirrel-cage structure are studied. The model of indirect field-oriented control of AC induction motor is used for the study of parameter adaptation control and parallel processing controller which are intended to satisfy the requirements of high-performance control of AC motor.

The dynamics of the AC machine drive control system is extremely complex and the subject is receiving wide attention in recent years. The complexity arises because of the nonlinearity of AC motor dynamics. For this reason, when a control strategy is developed, it is the usual practice to simulate the drive system on the digital computer and study the performance in detail before proceeding to build an

experiment. The electrical dynamics of an induction motor can be represented by a fourth-order nonlinear differential equation which may be either in a stationary reference frame (Stanley equation), or in a synchronously rotating reference frame [2]. The advantage of the latter model is that the steady state sinusoidally varying parameters appear as DC quantities. The established model with a means of feedback control can be simulated on computer and control parameters, such as PI or PID, can be fine tuned until the desired performance is obtained.

Models of AC induction motor and field-oriented control of AC induction motor have been developed and studied by researchers [7], [29], [31], [20], and [30]. A common method of modeling is that the indirect field-oriented control of AC induction motor can be linearized about a steady state operating point using small signal perturbation principle.

In this chapter, the design and simulation for indirect field-oriented control system are studied. A linear control system is achieved by applying feed-forward control scheme so that design of PI or PID controllers does not depend on system operating point. In addition, a new control scheme has been developed to improve control performance.

3.2 System Analysis and Modelling

The AC drive with field-oriented control is a well-known alternative to the DC drive. To implement the field-oriented control, the sufficient conditions for the quick torque control of an induction motor are summarized as follows [12]:

he secondary flux is controlled to be constant, which is equivalent to keeping

nstant.

2. The AC power supply frequency w_s is determined by:

$$w_s = w_r + (\frac{R_r}{L_r})\frac{i_q}{i_d}. \tag{3.1}$$

3. The primary currents i_q and i_d must be adjusted instantaneously and precisely according to the reference values i_q^* and i_d^*.

Consequently, the arbitrary quick torque control without unnecessary transients is realized by adjusting the torque current i_q while keeping the secondary flux constant (or i_d constant). However, the following problems preventing the satisfaction of the foregoing conditions should be taken into consideration:

1. The secondary flux level may be changed by the parameter variation in the induction motor such as the change of rotor resistance due to the temperature rising in the windings.

2. The slip frequency w_{sl} determined by equation (3.1) may have some error when the time constant value of the secondary circuit used in the controller is mismatched with the actual value. This error causes the coordination of the secondary flux to depart from the d-axis.

3. The instantaneous and precise adjustment of the primary current necessitates some control means, such as the current feedback control loops.

To achieve instantaneous and precise adjustment of the primary current, design for the inner current control loops plays an important role.

As usual, for the consideration of system stability, the inner loop (current control loop) design should be carried out first. After the inner control loops have been

designed satisfactorily, the design of outer control loop (speed control loop) can be carried out.

3.2.1 Analysis and Modelling of Flux Current and Torque Current Control Loops

To analyse the indirect field-oriented control for AC induction motor, some necessary assumptions are given below [38]:

ASSUMPTIONS:

1. Symmetrical armatures
2. Constant airgap
3. Sinusoidal induction repartition
4. Saturation, hysteresis and eddy effects negligible
5. Indirect field-oriented controlled (this means that the direction of the rotor flux is aligned to the synchronous rotating axis d).

Based on the above assumptions, the dynamic equations of a squirrel-cage induction motor in the $d-q$ reference frame are given [7]:

$$\begin{bmatrix} u_d \\ u_q \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} R_s + pL_s & -w_s L_s & pM & -w_s M \\ w_s & R_s + pL_s & w_s M & pM \\ pM & 0 & R_r + pL_r & 0 \\ w_{sl} M & 0 & w_{sl} L_r & R_r \end{bmatrix} \begin{bmatrix} i_d \\ i_q \\ i_{dr} \\ i_{qr} \end{bmatrix}. \tag{3.2}$$

According to the $d-q$ equivalent circuits, shown in Fig. 2.2, the following equations are derived:

$$\psi_{qr} = L_r i_{qr} + M i_q \tag{3.3}$$

$$\psi_{dr} = L_r i_{dr} + M i_d. \tag{3.4}$$

By solving equations (3.1)-(3.4), the following equations are obtained:

$$u_d = R_s i_d + \frac{M^2}{L_r}(\frac{p}{1+T_r p}) i_d - w_s \rho L_s i_q \tag{3.5}$$

$$u_q = R_s i_q + \rho L_s p i_q + w_s(\frac{M^2}{L_r}\frac{i_d}{(1+T_r p)}) + \rho L_s w_s i_d \tag{3.6}$$

or

$$u_d = R_s(1 + T_s p\frac{1+\rho T_r p}{1+T_r p}) i_d - w_s \rho L_s i_q \tag{3.7}$$

$$u_q = R_s(1+\rho T_s p) i_q + L_s(\frac{1+\rho T_r p}{1+T_r p}) w_s i_d. \tag{3.8}$$

In order to control the flux current i_d and torque current i_q, feedback control is used favorably. The purpose of feedback control is to improve the dynamic performance of controlled currents such that the torque current i_q follows the reference values as quickly as possible, while the flux current i_d keeps constant. From the equations (3.7) - (3.8), it is clear that even though the field-oriented control scheme linearizes and decouples the relations between rotor flux and electromagnetic torque, the coupled relations between d-axis current and q-axis current or $d-q$ currents and rotor speed w_r still exist, which presents a non-linear and multivariable structure to the current control.

In order to approach the design of current controllers by using classical method such as root-locus or frequency domain analysis, it is proposed here to use interactive cancellation techniques for decoupling the interactions between i_q and i_d or i_q and i_d and w_r.

It is noted that if the interactive variables $w_s i_q$ and $w_s i_d$ as shown in equations (3.7)-(3.8) are considered as external disturbances to the control of i_d and i_q, the flux

current and torque current control can be configured as a linear control structure and, therefore, the transfer function can be used to analyze the closed-loop control, as shown in Fig. 3.1, where:

$$G_d(s) = \frac{1 + T_r s}{\rho T_s T_r R_s s^2 + (T_r R_s + T_s R_s)s + R_s} \tag{3.9}$$

$$G_q(s) = \frac{1}{R_s(1 + \rho T_s s)} \tag{3.10}$$

$$D_1(s) = \rho L_s \tag{3.11}$$

$$D_2(s) = L_s(\frac{1 + \rho T_r s}{1 + T_r s}). \tag{3.12}$$

It is obvious that the disturbances $w_s i_q$ and $w_s i_d$ are easily computable from the measured quantities i_d, i_q and w_r; therefore, an effective control approach for cancelling this kind of disturbances is to use the feed-forward control technique.

Feed-forward control means the control of undesirable effects of measurable disturbance by approximately compensating for it before it materializes. This feature is advantageous, because, in a usual feedback control system, the corrective action starts only after the output has been affected. This technique achieves the objective of instantaneous compensation control of external disturbances $i_d w_s$ and $i_q w_s$ and minimizes the transient error. When feedback control is combined with the feed-forward control, it is expected that the feedback control compensates for any imperfections in the functioning of the feedback control and provides corrections for unmeasurable disturbance, and meanwhile the feed-forward control minimizes the transient error caused by measurable disturbances such as the $w_s i_q$ and $w_s i_d$.

The block diagram of this compound control is shown in Fig. 3.2, where:

$$G_f(s) = \rho L_s = D_1(s) \tag{3.13}$$

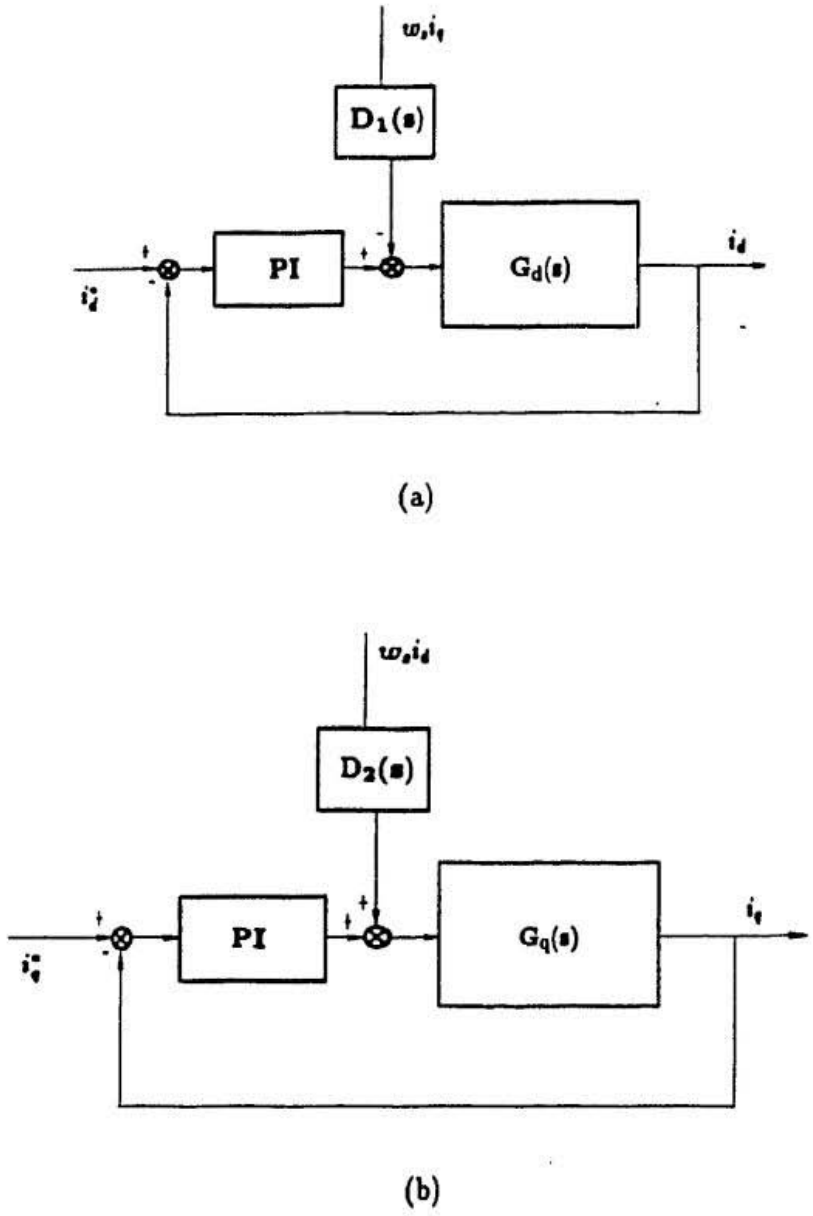


Figure 3.1: Flux current and torque current closed-loop control. (a) Flux current control. (b) Torque current control.

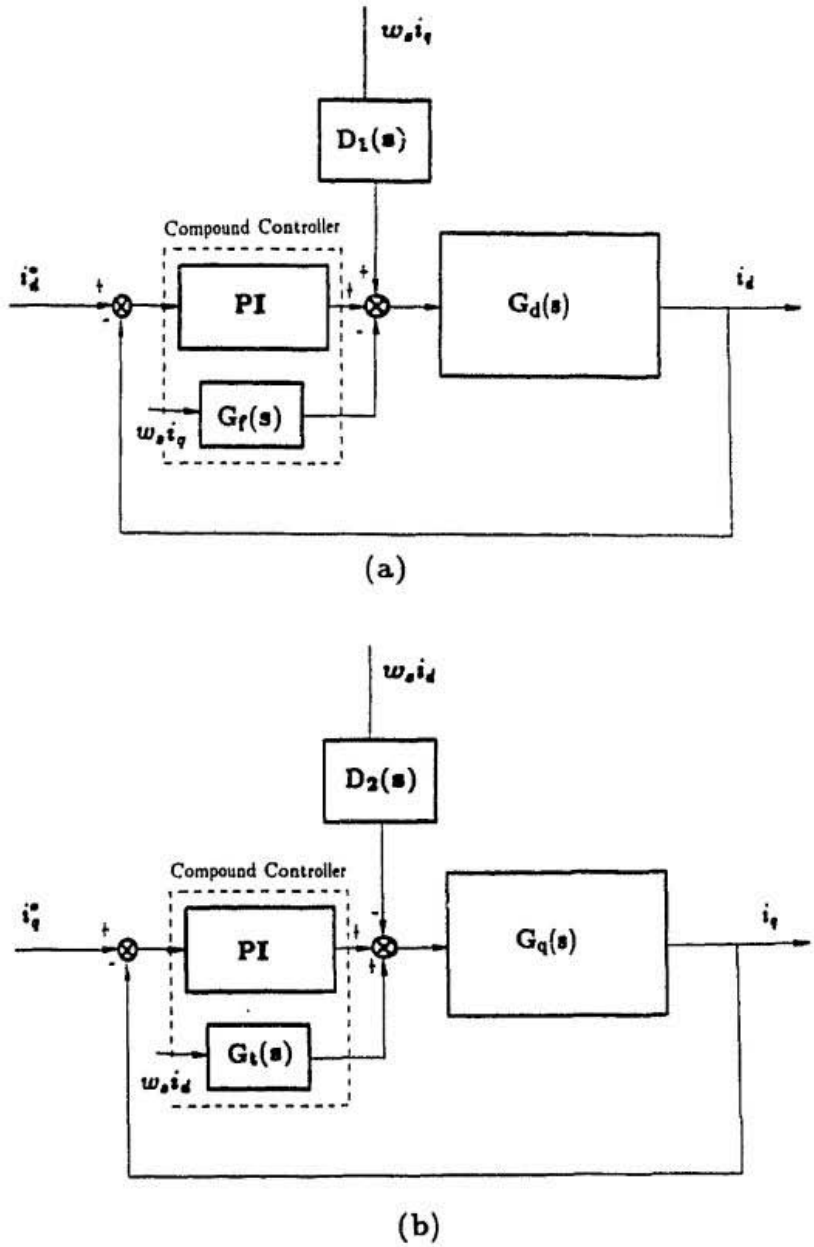


Figure 3.2: Compound Control. a) Flux current with feed forward controller, b) Torque current control with feed forward controller

$$G_t(s) = L_s(\frac{1+\rho T_r s}{1+T_r s}) = D_2(s). \tag{3.14}$$

It can be seen that the disturbances $w_s i_q$ on i_d and $w_s i_d$ on i_q can be cancelled completely by the feed-forward controllers $G_f(s)$ and $G_t(s)$, while the PI controllers effect the current loop control.

3.2.2 Analysis and Modelling of Speed Control Loop

By taking advantage of the feed-forward control, the disturbances can be cancelled so that a linear control structure of AC motor speed control can be simplified as shown in Fig. 3.3.(a),

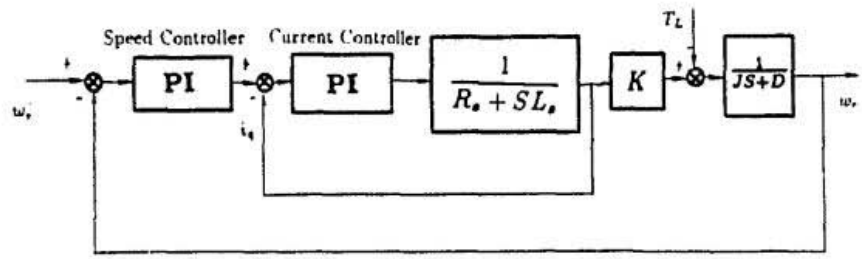
where:

$$K = \frac{3}{2} n_p \frac{M^2}{L_r} i_d. \tag{3.15}$$

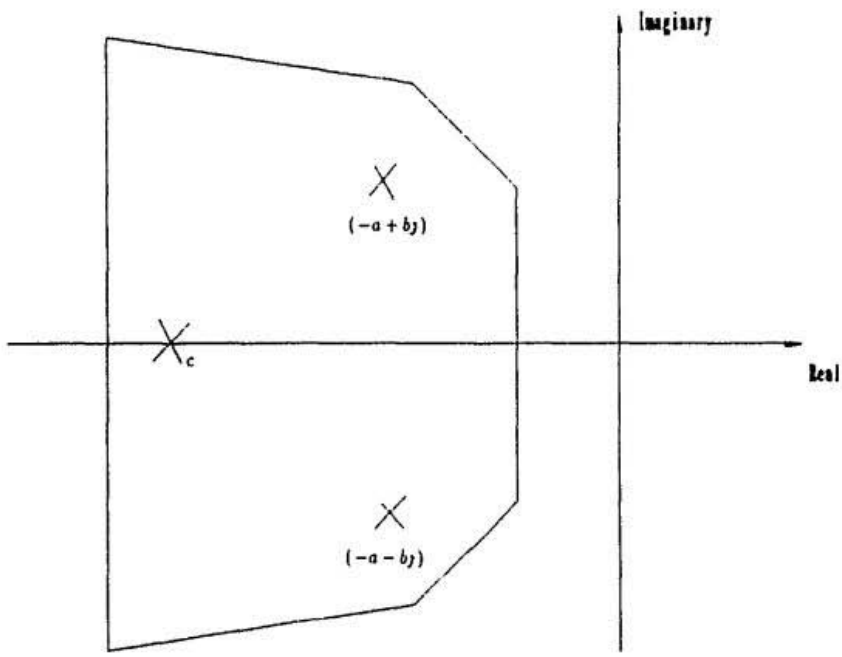
To design the speed loop, several methods are available. The root-locus based approach is popular. In general, according to design specifications, dominant complex poles $(-a+bj)$ and $(-a-bj)$ with a far away real pole $(-c)$ can be specified as shown in Fig. 3.3(b). The parameters of PI controllers can be easily obtained from the locations of dominant poles which provide a desirable control performance.

3.2.3 Modelling of Indirect Field-Oriented Control of Voltage-Fed Squirrel-Cage Induction Motor

For the purpose of analysis of indirect field-oriented control of AC induction motor, it is essential to run simulations of an overall AC induction motor control system model instead of a simplified control model. As a result, the simulation results do not only give control variables in synchronous reference frame, but also give the control variables in stationary reference frame which provides analysis of alternate



(a)



(b)

Figure 3.3: Block diagram of speed loop control and pole location. (a) Speed control loop, (b) Pole location

variables such as stator currents, voltages and rotor flux.

In the simulation set up, the voltage-fed PWM inverter is defined as an ideal sinusoidal power supply. The gain of the inverter is assumed as $K_0 = 30$. The current feedback coefficient and the speed feedback coefficient are defined as K_I and K_s, respectively.

As usual, the motor model is represented in the two-axis system, α-β reference frame. An N-pole induction motor with a short-circuited rotor circuit is characterized by the following equations:

$$\begin{bmatrix} u_\alpha \\ u_\beta \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} R_s + pL_s & 0 & pM & 0 \\ 0 & R_s + pL_s & 0 & pM \\ pM & w_r M & R_r + pL_r & w_r L_r \\ -w_r M & pM & -w_r L_r & R_r + pL_r \end{bmatrix} \begin{bmatrix} i_\alpha \\ i_\beta \\ i_{\alpha r} \\ i_{\beta r} \end{bmatrix} \tag{3.16}$$

$$\frac{J}{n_p}\frac{dw_r}{dt} = T - T_L - D\frac{w_r}{n_p} \tag{3.17}$$

$$T = \frac{3}{2}Mn_p(i_\beta i_{\alpha r} - i_\alpha i_{\beta r}) \tag{3.18}$$

$$\frac{d}{dt}\psi_\alpha = u_\alpha - R_s i_\alpha \tag{3.19}$$

$$\frac{d}{dt}\psi_{\alpha r} = -R_r i_{\alpha r} - w_r \psi_{\beta r} \tag{3.20}$$

$$\frac{d}{dt}\psi_{\beta r} = -R_r i_{\beta r} + w_r \psi_{\alpha r} \tag{3.21}$$

$$\frac{d}{dt}\psi_\beta = u_\beta - R_s i_\beta, \tag{3.22}$$

where:

$$\begin{bmatrix} i_\alpha \\ i_{\alpha r} \\ i_{\beta r} \\ i_\beta \end{bmatrix} = \frac{1}{L_s L_r - M^2} \begin{bmatrix} L_r\psi_\alpha - M\psi_{\alpha r} \\ L_s\psi_{\alpha r} - M\psi_\alpha \\ L_s\psi_{\beta r} - M\psi_\beta \\ L_r\psi_\beta - M\psi_{\beta r} \end{bmatrix}. \tag{3.23}$$

Based on equations (3.16) - (3.23), the time domain simulation model for indirect field-oriented control of AC induction motor can be constructed as shown in Fig. 3.4. The VAX 8530 digital computer is chosen for studying the dynamics of AC motor control. Under the indirect field-oriented control, the required stator voltages u_α and u_β are obtained by the following coordinate transformations:

$$\begin{bmatrix} u_\alpha \\ u_\beta \end{bmatrix} = \begin{bmatrix} cos\theta_s & -sin\theta_s \\ sin\theta_s & cos\theta_s \end{bmatrix} \begin{bmatrix} u_d \\ u_q \end{bmatrix}. \tag{3.24}$$

Three phase variables in AC motor can be easily transformed from two axes variables by the following transformations:

$$\begin{bmatrix} u_a \\ u_b \\ u_c \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -\frac{1}{2} & \frac{\sqrt{3}}{2} \\ -\frac{1}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix} \begin{bmatrix} u_\alpha \\ u_\beta \end{bmatrix}. \tag{3.25}$$

The stator current i_α and i_β in the $\alpha - \beta$ reference frame can be transformed to field-oriented unidirection quantities i_d and i_q with the following coordinate transformations:

$$\begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{\sqrt{3}} & -\frac{1}{\sqrt{3}} \end{bmatrix} \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} \tag{3.26}$$

and

$$\begin{bmatrix} i_d \\ i_q \end{bmatrix} = \begin{bmatrix} cos\theta_s & sin\theta_s \\ -sin\theta_s & cos\theta_s \end{bmatrix} \begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix}. \tag{3.27}$$

The frequency and phase of the voltage components u_α and u_β in the stationary reference frame for establishing the required rotor flux and stator current are controlled by the slip frequency w_{sl} which can be calculated from equation (3.1).

The controllers applied in the indirect field-oriented control of AC induction motor include speed and current feedback PI controllers as well as flux and torque current feed-forward controllers. The transfer function that approximates the action

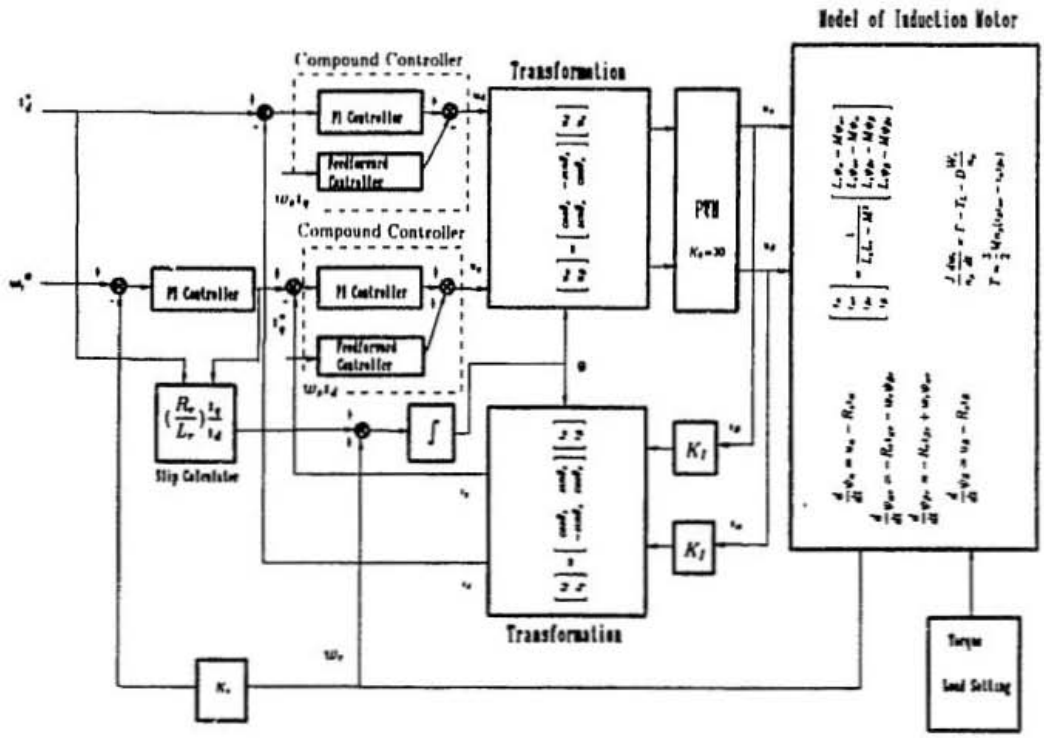


Figure 3.4: Time domain model of indirect field-oriented control system

of a feedback continuous PI controller is given by:

$$G(s) = \frac{Y(s)}{E(s)} = K_p(1 + \frac{1}{\tau s}), \tag{3.28}$$

where τ is the time constant, K_p is the controller gain.

A difference equation of this type of controller is characterized by :

$$y_n = y_{n-1} + K_p(e_n - e_{n-1}) + T_0 * K_i * e_n, \tag{3.29}$$

where $K_i = \frac{K_p}{\tau}$; T_0 is the sampling time.

All the feedback quantities are compared with the corresponding reference quantities, and the error quantity e forms the input to the controller. The feed-forward controller of the flux current is expressed by the following equation:

$$G_f(s) = \rho L_s. \tag{3.30}$$

The feed-forward controller of torque current is expressed by the following equation:

$$G_t(s) = L_s \frac{1 + \rho T_r s}{1 + T_r s}. \tag{3.31}$$

A difference equation of the torque current feed-forward controller is characterized by:

$$y_n = (T_r * y_{n-1} + T_0 * L_r * \rho * T_r(e_n - e_{n-1}))/(T_0 + T_r). \tag{3.32}$$

For different load simulations, the load-setting unit is used to change the applied load to the motor.

3.3 Simulation Results of Indirect Field-Oriented Control of AC Induction Motor

A C program for the simulation of time domain model was designed for simulation purposes (see appendix A-1). The parameters of the AC motor given in [7] are

Table 3.1: *Parameters of controller*

type of controller	P	I
Speed controller	8	0.003
Flux current controller	0.8	0.01
Torque current controller	0.6	0.01

Table 3.1: Parameters of controller

listed in appendix A-2. Using the MatLab to simulate the linearized and simplified AC motor control system as shown in Fig. 3.3(a), the proportional gain and the integral time constant of PI controllers can be determined. The parameters of current controllers and speed controller are listed in Table 3.1. For studying the dynamic properties of indirect field-oriented control of AC motor, a series of simulations have been carried out . By viewing the simulation results, it can be seen that, when the compound control of feed-back and feed forward controllers were applied, the flux current was controlled better than that of only feedback control. In these simulations, the dynamic response of the control system for the following disturbances were studied:

1. Step change in the reference speed

2. Step change in the load

3. Speed reversal.

Step Change in the Reference Speed

The step response of a control system is usually applied to evaluate the control system performance in its response time, overshoot, transient or steady errors and stability.

Nine motor variables were chosen in this simulation for studying the dynamic performance. They are the $d-q$ coordinate components, such as the torque current and flux current, $\alpha-\beta$ coordinate components, such as stator currents and voltages, and motor state variables, such as electromagnetic torque, rotor flux and speed. The behaviour of the field coordinates and the control variables are plotted in Figs. 3.5 - 3.9.

Fig. 3.5(a) shows the rotor speed response with starting speed of 1040 *rpm*. At $t = 2s$, the speed reference was changed to the speed of 512 *rpm*. It can be seen that the rotor speed is accelerated or decelerated smoothly by controlling the motor torque which is proportional to torque current. Fig. 3.5(b) shows the motor torque response. At the beginning, it is obvious that the torque was not linearly proportional to the torque current , because the flux current was not constant due to dynamic transience as shown in Fig. 3.6(b). At $t = 0.6s$, a $2Nm$ load was applied, the corresponding torque and torque current, as illustrated in Fig. 3.5(b) and 3.6(a), show the linear response to the input load because the constant flux has been set up at this time. This proves that under the field-oriented control if the flux is constant, the machine torque is linearly proportional to the torque current . In this simulation, as the feed-forward controller was not applied, the flux current

was disturbed by changes in speed or load, as indicated in Fig. 3.6(b).

Figs. 3.7 - 3.9 show the corresponding responses of stator currents, stator voltages and rotor flux when input speed and load were changed. In order to satisfy the conditions of field-oriented control, the corresponding stator current or voltages changed their magnitude and phase, and the rotor flux kept constant.

Step Change in the Load

The ability to withstand disturbances in a control system is another important feature. A step change in the load is considered as a typical external disturbance. A high performance control system should have a fast dynamic response in adjusting its control variables so that the system outputs affected by the disturbances will recover to their original status as soon as possible. The dropped aptitude of system output such as rotor speed and its recovering time are the important performance specifications.

Figs. 3.10 - 3.14 show the control system response when a step load was applied. In this simulation, the feed-forward controller was applied. In Fig. 3.10(b), a $2Nm$ step load was applied to the motor at $t = 0.6s$. The corresponding motor torque has emerged to balance the load for keeping speed constant as shown in Fig. 3.11(a). When the load was changed from 2 Nm to the rated value of $8Nm$ at $t = 2s$, the motor torque quickly followed the load applied to attain a new equilibrium point. The rotor speed, as shown in Fig. 3.10(a), dropped a little at $t = 0.6s$ and $t = 2s$, and then raised quickly to its original value. In this time, due to the feed-forward control, the flux current remained constant without being disturbed by changes in load.

Figs. 3.12 - 3.14 show the corresponding responses of stator currents, stator voltages and rotor flux when input load were changed.

Speed Reversal

With the field-oriented control, AC motor can be operated in four quadrants like the DC motor.

Figs. 3.15 - 3.19 show the system response for speed reversal. According to input command, the rotor speed reversed at $t = 2s$ from positive rated speed to negative rated speed as shown in Fig. 3.15(a). The stator currents and voltages, as shown in Figs. 3.17 - 3.19, changed their phase, frequency and amplitude when speed reversed. A negative current, as shown in Fig. 3.16(a), occurred to produce negative torque, as shown in Fig. 3.15(b), for speed reversal. The flux current and rotor flux are kept constant with speed reversal. In this simulation, the feed forward control was also applied, therefore, the flux current remained constant without being disturbed by speed reversal.

Figs. 3.5 - 3.19 demonstrate the field-oriented control mechanism of the linearized control of decoupled flux and torque currents in an AC induction motor. The control performance of AC induction motor is found to be like that of DC motor. The motor speed was been controlled to regain the original speed after a small dip for a short duration when load was applied. It also shows that an AC drive of this kind permits operation in four quadrants and allows the motor to be loaded continuously with rated torque at standstill. Because of the slow response of the rotor flux, there is a small dip in magnitude during speed reversal which is regained subsequently.

For the purpose of comparison between feedback control and compound control, Fig. 3.20 - 3.24 present the corresponding simulation results.

Fig. 3.20(a) shows the rotor speed response at a starting speed of 512 rpm. At $t = 2s$, the speed reference was changed to rated speed of 1710 rpm. Fig. 3.20(b) shows the motor torque response when a $2Nm$ load was applied at $t = 0.6s$. The corresponding torque and torque current, as illustrated in Fig. 3.20(b) and 3.21(a), show the linear response to the input load. In this simulation, as the feed-forward controller was applied, the flux current remained constant without being disturbed by changes in speed or load, as indicated in Fig. 3.21(b).

Figs. 3.22 - 3.24 show the corresponding changes of stator currents, stator voltages and rotor flux when input speed and load were in changes.

It can be seen that with feed-forward controller, the flux current is controlled more constant, as shown in Fig. 3.21(b), than that without feed-forward controller, as shown in Fig. 3.6(b).

3.4 Extended Indirect Field-Oriented Control of AC Induction Motor

The concept of series-shunt excited control is well-known in DC motor control techniques. The series-shunt excited control takes advantages of coupled torque and flux current control to enhance the ability of withstanding disturbances and to increase the maximum torque. Implementation of the series-shunt excited control in DC motor is made by connecting flux winding and torque current circuit to let flux current contain some of torque current. The DC motor control theory has proved that a series-shunt excited control system can provide higher torque response than

separately excited control. This performance shows a strong robustness and fast control response.

In AC motor control, the naturally coupled torque and flux current could be easily realized as a series-shunt excited control model, if a proper value of slip is found. However, the coupled torque and flux current characteristics makes the analysis of control performance very complex because of its non-linear and multi-variable structure.

To exploit the possibility of series-shunt excited control in AC motor, first let the AC motor be controlled in field-oriented coordinates, which presents a separately excited type and then the slip is reduced to a smaller value than that for completely decoupling torque and flux. In this case, we can expect to obtain a series-shunt excited control in AC motor like in DC motor.

The proposed series-shunt excited control in AC motor has been simulated in a digital computer to verify the validity and feasibility of this method. The simulation programs and motor parameters [13] are listed in Appendix B-1 and Appendix B-2, respectively. The parameters of PI controller and slip frequency are listed in Table 3.2. Simulation results are illustrated in Figs. 3.25 - 3.27, which indicate an excellent robustness and fast response with almost no overshoot. The features of this control performance can be summarized as follows:

1. There is almost no overshoot during motor transient speed response (see Fig. 3.25(a)).

2. The transient speed response time is equal to that of separately excited control of AC motor (compared Fig. 3.25(a) with Fig. 3.10(a)).

Table 3.2: *Parameters of Controller and Slip Frequency*

type	P	I	Normal	Small
Speed controller	8	0.003		
Flux current controller	0.8	0.01		
Torque current controller	0.6	0.01		
Slip Frequency			100%	10%

Table 3.2: Parameters of Controller and Slip Frequency

3. There is almost no drop or increase in speed (Fig. 3.25(a)) when rated load was applied or removed ($20Nm$, see Fig. 3.25(b)).

4. Rotor flux current is not constant when control system is working in transient status (Fig. 3.26(b), which is similar to the series-shunt excited DC motor control).

5. The stator current is still sinusoidal which can be provided by power electronic devices (Fig. 3.27(a)).

For comparison, Fig. 3.28 shows the linear control response by regular field-oriented control method. The motor is started at a speed of 680 rpm and changed to the rated speed of 1710 rpm at $t = 2s$. It can be seen that, with 20 Nm rated load applied at $t = 0.7s$ and removed at $t = 3s$, the rotor speed dropped much lower or higher than that of series-shunt excited control scheme.

3.5 Discussions

3.5.1 Conditions for a Truly Linear Control System of AC Induction Motor Drives

Generally, an AC induction motor can be controlled linearly by applying the indirect field-oriented control scheme. However, it is a fact that the indirect field-oriented control is only a necessary condition in realizing a linear AC induction motor control system. In other words, the indirect field-oriented control only provides the solution that if the flux current is constant and the rotor flux orientation is aligned to the d-axis in synchronous reference frame, a linear control of AC motor torque can be realized by controlling the torque current.

But the field-oriented control scheme does not ensure that the flux and torque current can be controlled linearly and separately.

In fact, under indirect field-oriented control, the torque current is produced not only by the voltage in the q-axis but also by the voltage in the d-axis and by the rotor speed. The flux current is also produced by both voltages in the d and the q axes as well as by rotor speed.

The coupled relations between flux current and torque current are expressed in equations (3.5) - (3.6). According to the study in this chapter, a linear control model for an AC induction motor under the indirect field-oriented control can be obtained if the feed-forward controller is applied, which results in decoupling the torque current from flux current as shown in Fig. 3.2. Therefore, it can be said that the necessary and sufficient conditions for obtaining a linear control system of AC induction motor is to employ both field-oriented control and feed-forward control.

3.5.2 Extension of Field-Oriented Control

Realization of high-performance control of AC induction motor using field-oriented control is required to satisfy both flux current constant and the alignment between d-axis and rotor flux orientation.

However, high-performance control of AC induction motor could be achieved by an extended field-oriented control that a departure of rotor flux orientation from the d-axis is allowable. In addition, constant rotor flux is not necessary in transient state.

In this, such an extended proposal of field-oriented control has been verified by a series of simulation results which show excellent characteristics of AC motor control. Obviously, the proposed control scheme behaves non-linearly because of its nonconstant rotor flux and the coupling of torque and flux. The principle of such a scheme is similar to that of a series-shunt excited DC motor control. However, the simulation results have shown that the control performance obtained is superior to that of DC motor control. These results probably hint that an AC motor control under the extended field-oriented control could be the most favourable choice over others due to its several excellent characteristics as described in this chapter.

To prove this conjecture, experimental work and further theoretical work should be carried out.

3.6 Summary

In this chapter, based on the model of indirect field-oriented control of voltage-fed squirrel-cage induction motor, a series of simulations have been carried out to verify the control performance which is usually exhibited by DC motor control.

Feed-forward control scheme for decoupling the interactions between flux current control loop and torque current control loop can linearize and simplify the design of AC motor control system and, therefore, improve the control dynamic performance.

Finally, the series-shunt excited control technique is proposed to be applicable to AC motor control. Through simulations, excellent robust control performance was obtained by using this technique.

It is noted that only part of simulation results have been presented in this chapter to limit the number of figures. Actually, we have tried two different types of motors for each simulation. Simulation results have shown a good agreement in control behaviour on these two different motors.

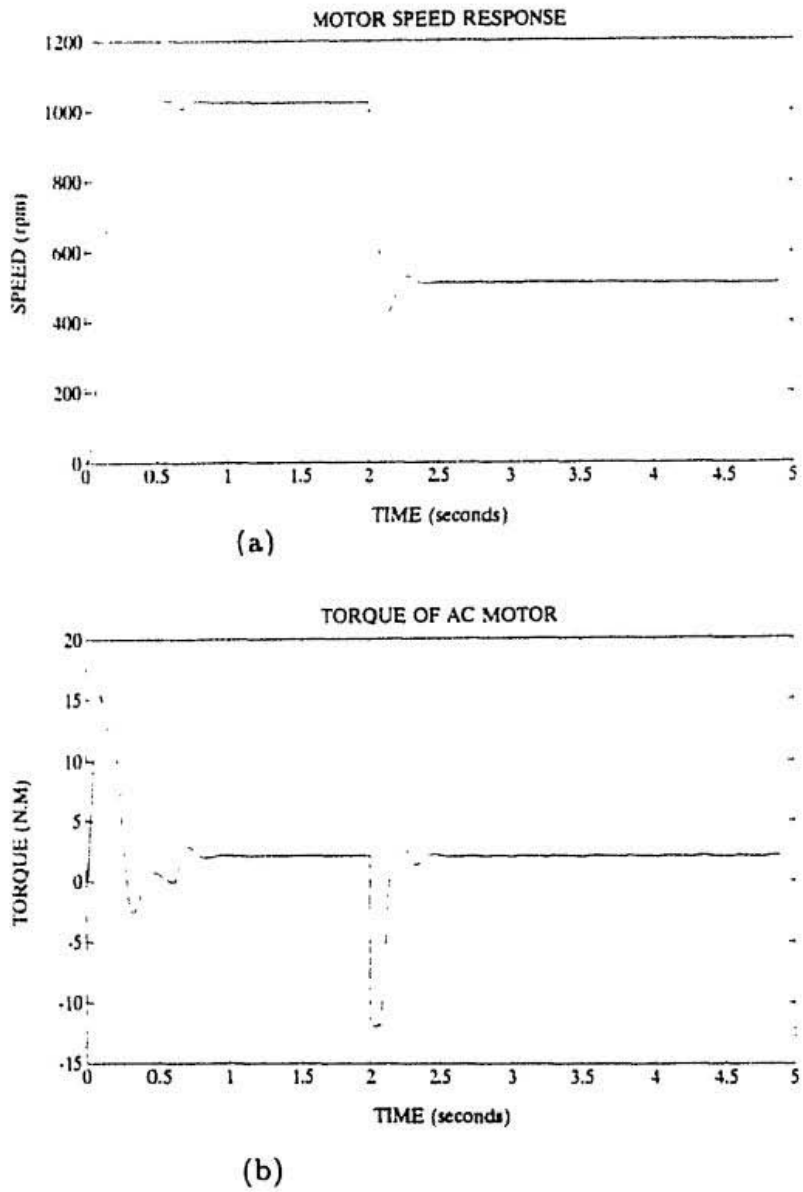


Figure 3.5: System response for step change in the reference speed. a) Rotor speed, b) Motor torque

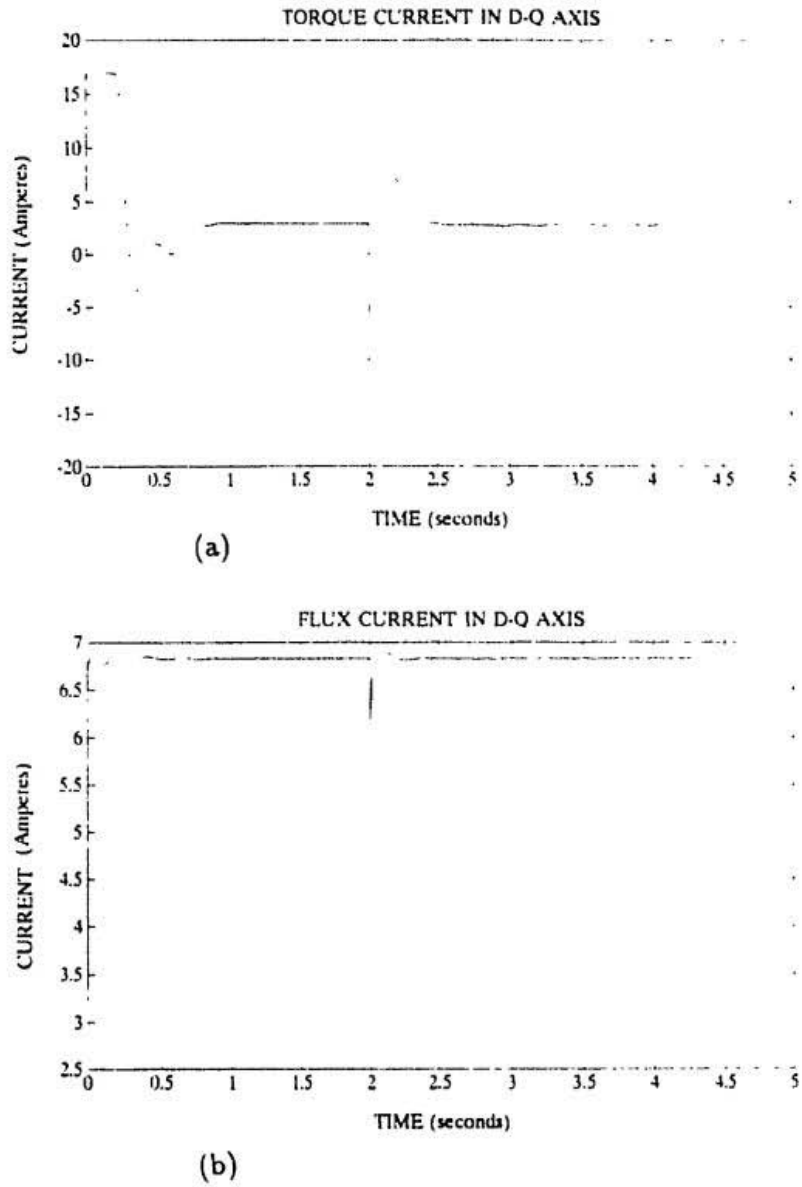


Figure 3.6: System response for step change in the reference speed, a) Torque current, b) Flux current

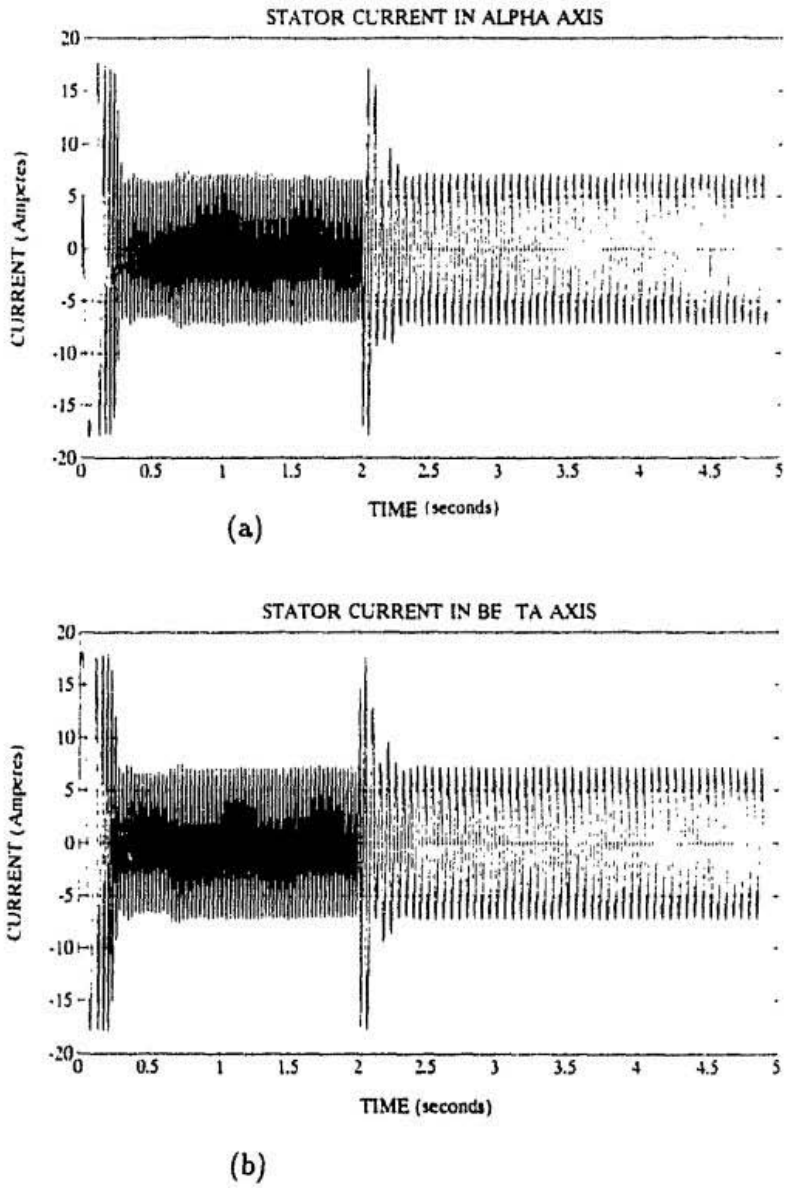


Figure 3.7: System response for step change in the reference speed, a) Stator current in alpha axis, b) Stator current in beta axis

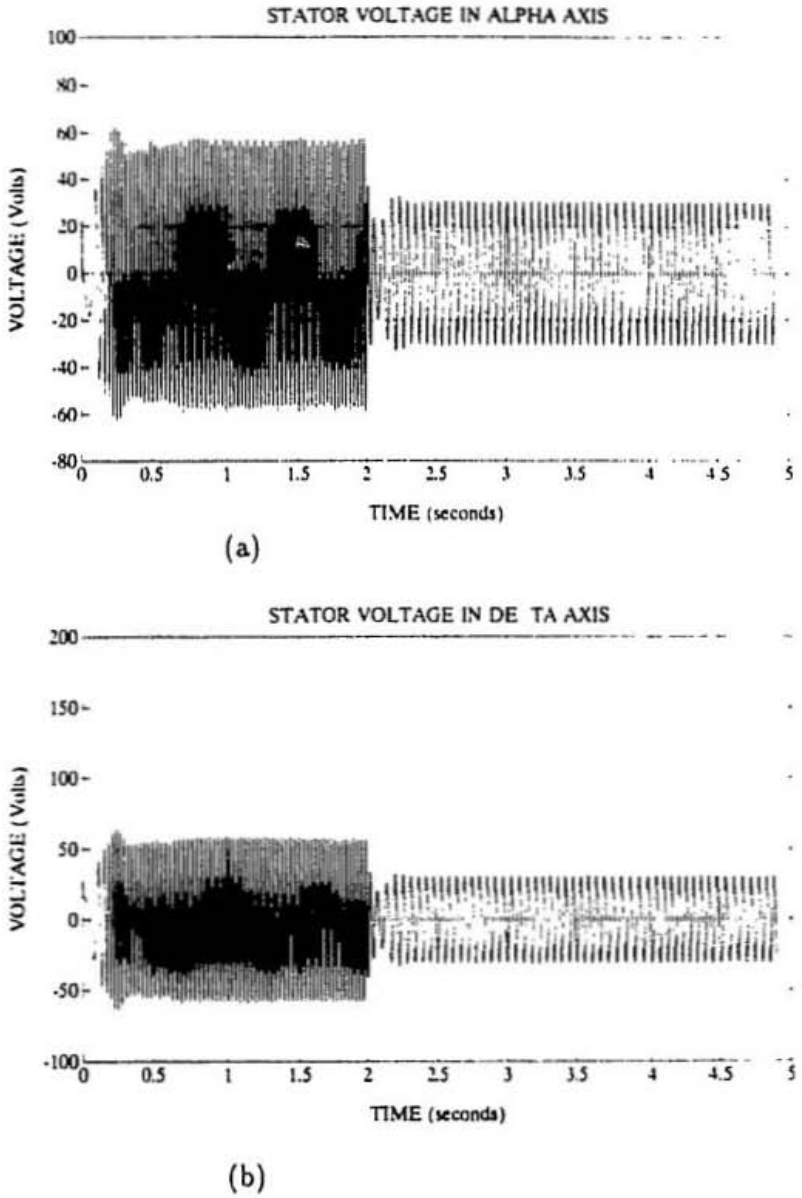


Figure 3.8: System response for step change in the reference speed, a) Stator voltage in alpha axis, b) Stator voltage in beta axis

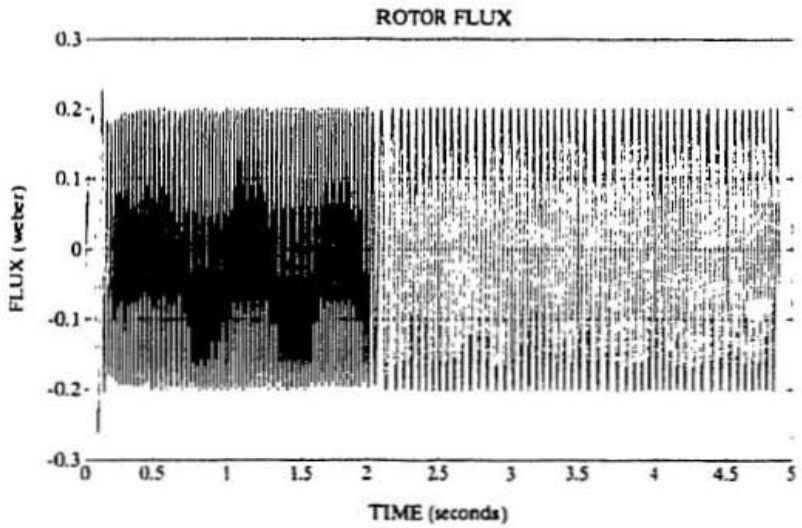


Figure 3.9: Rotor flux response for step change in the reference speed

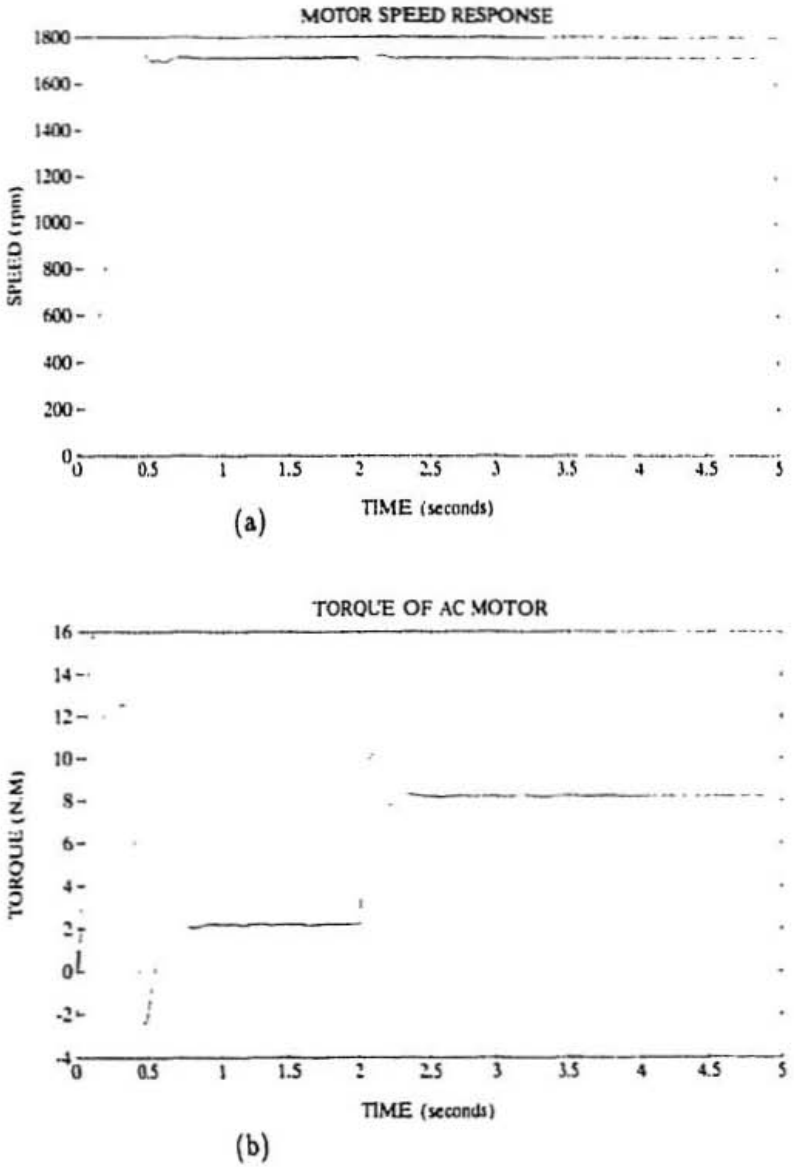
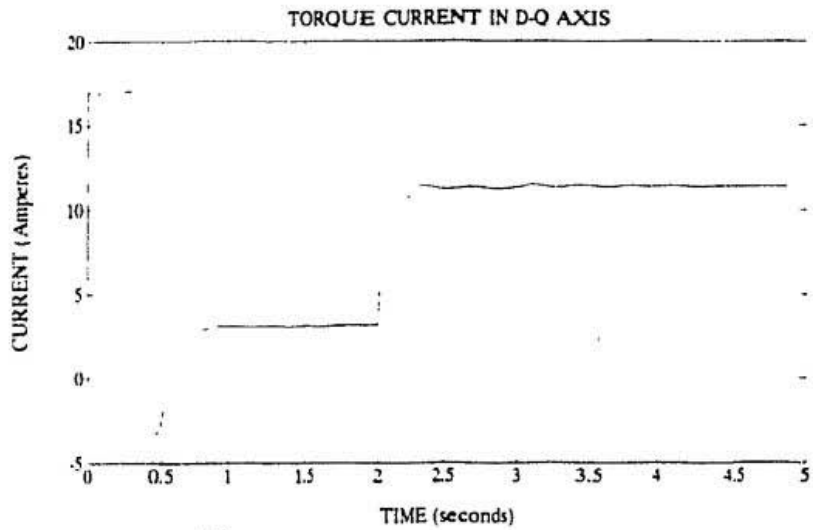
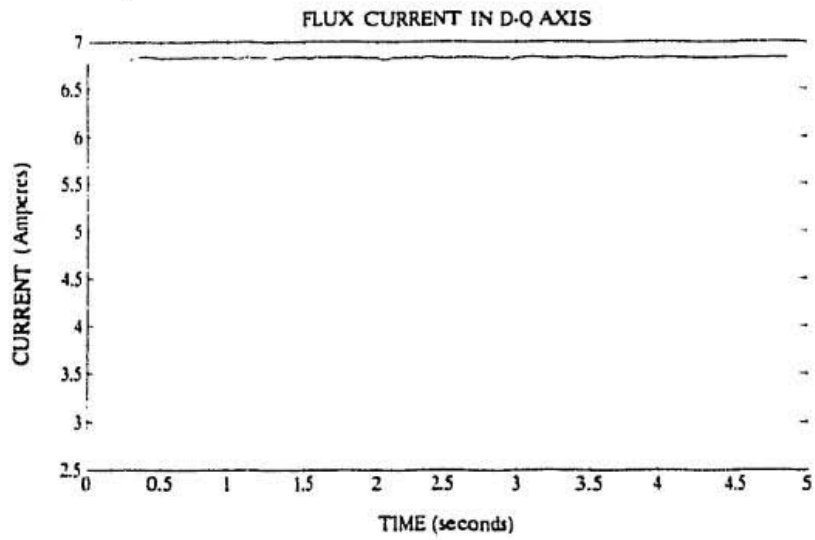


Figure 3.10: System response for step change in the load, a) Rotor speed, b) Motor torque



(a)



(b)

Figure 3.11: System response for step change in the load, a) Torque current, b) Flux current

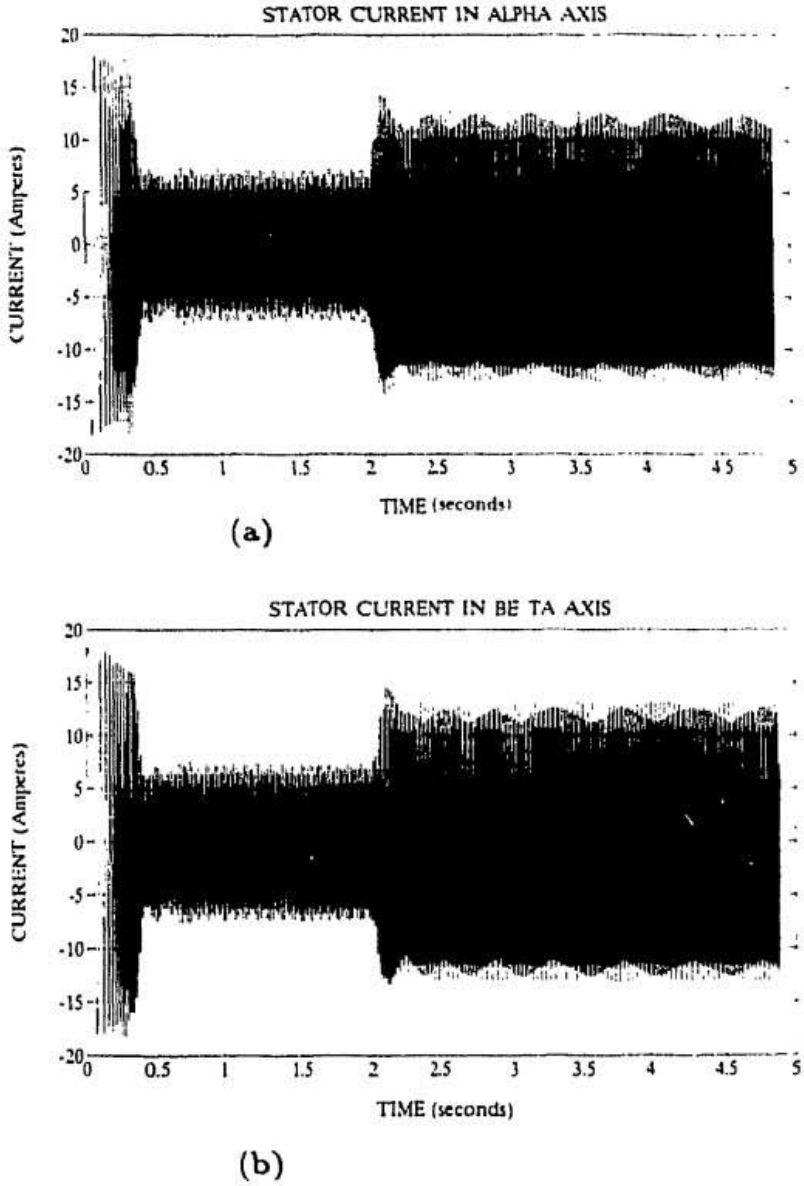


Figure 3.12: System response for step change in the load, a) Stator current in alpha axis, b) Stator current in beta axis

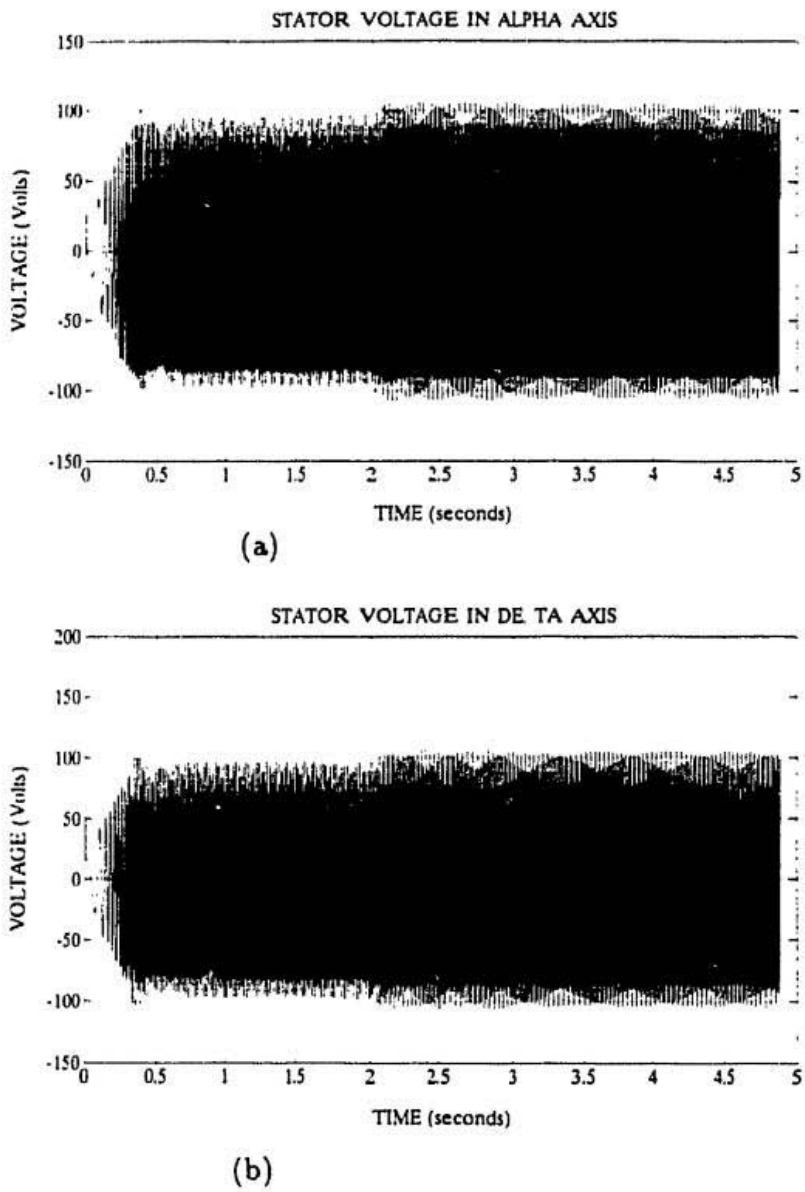


Figure 3.13: System response for step change in the load, a) Stator voltage in alpha axis, b) Stator voltage in beta axis

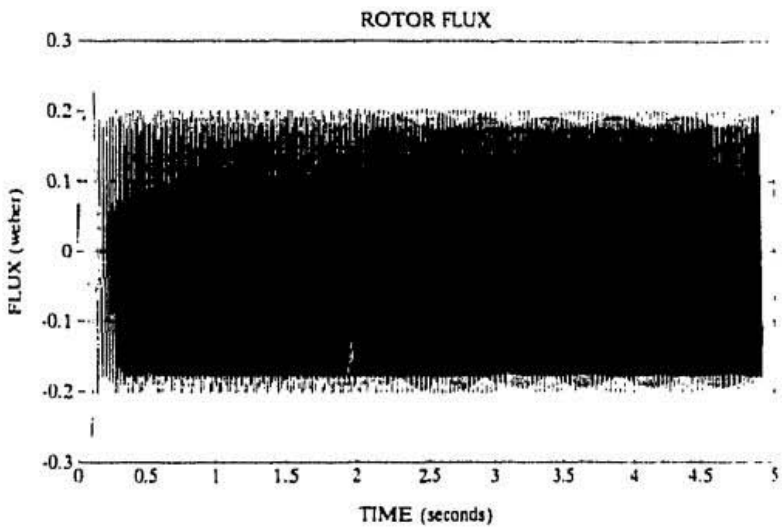


Figure 3.14: Rotor flux response for step change in the load

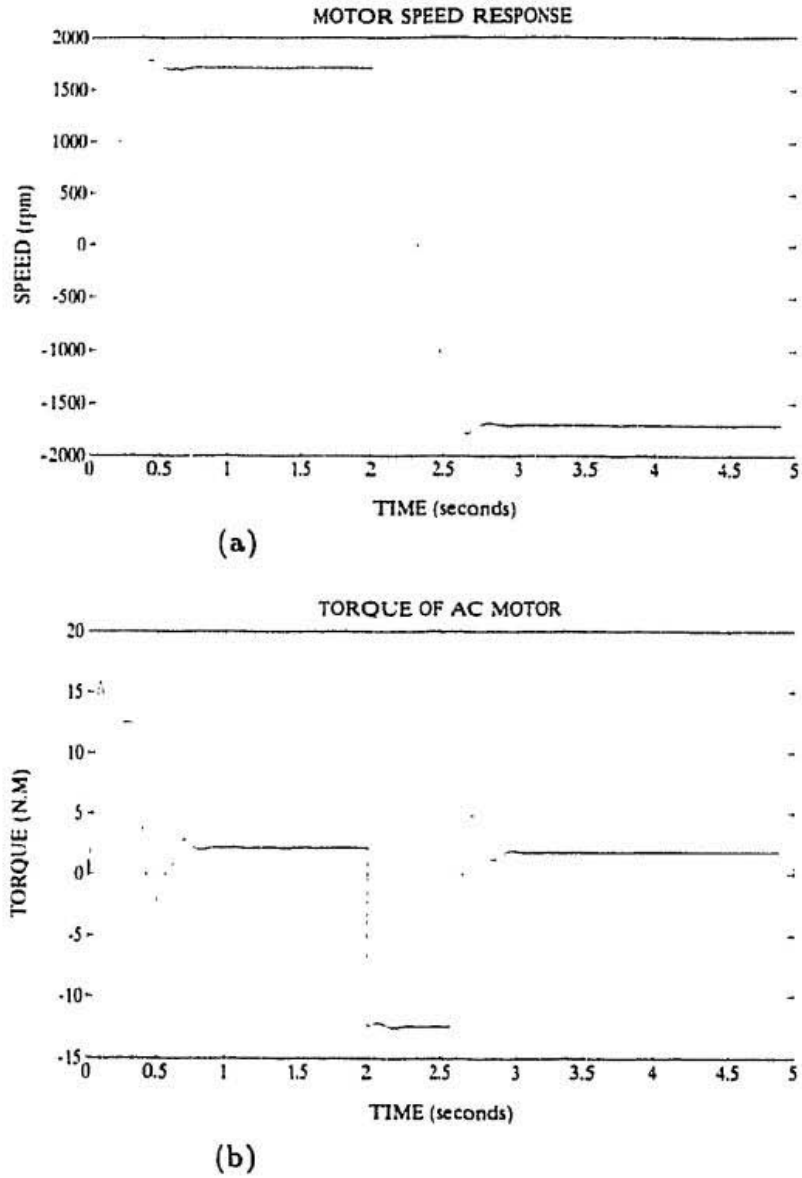
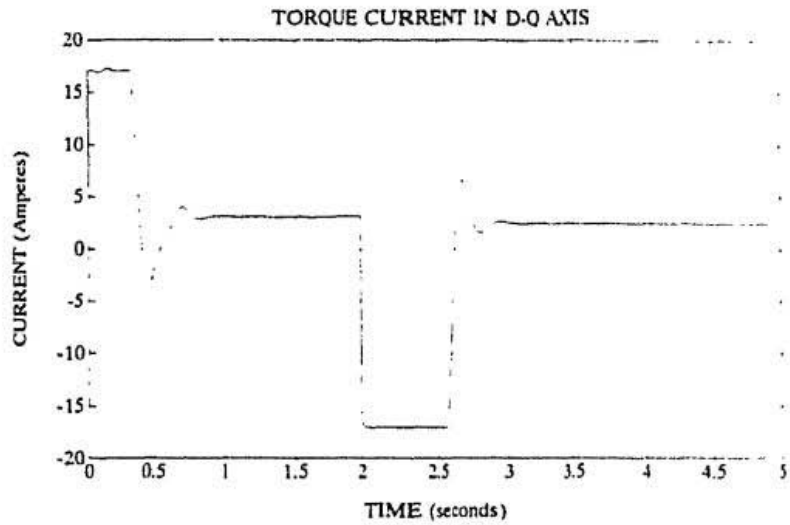
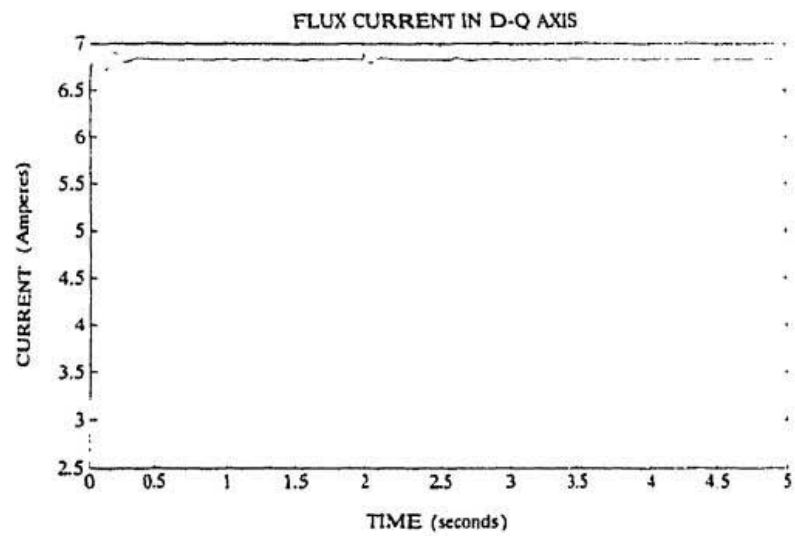


Figure 3.15: System response for speed reversal. a) Rotor speed, b) Motor torque



(a)



(b)

Figure 3.16: System response for speed reversal. a) Torque current, b) Flux current

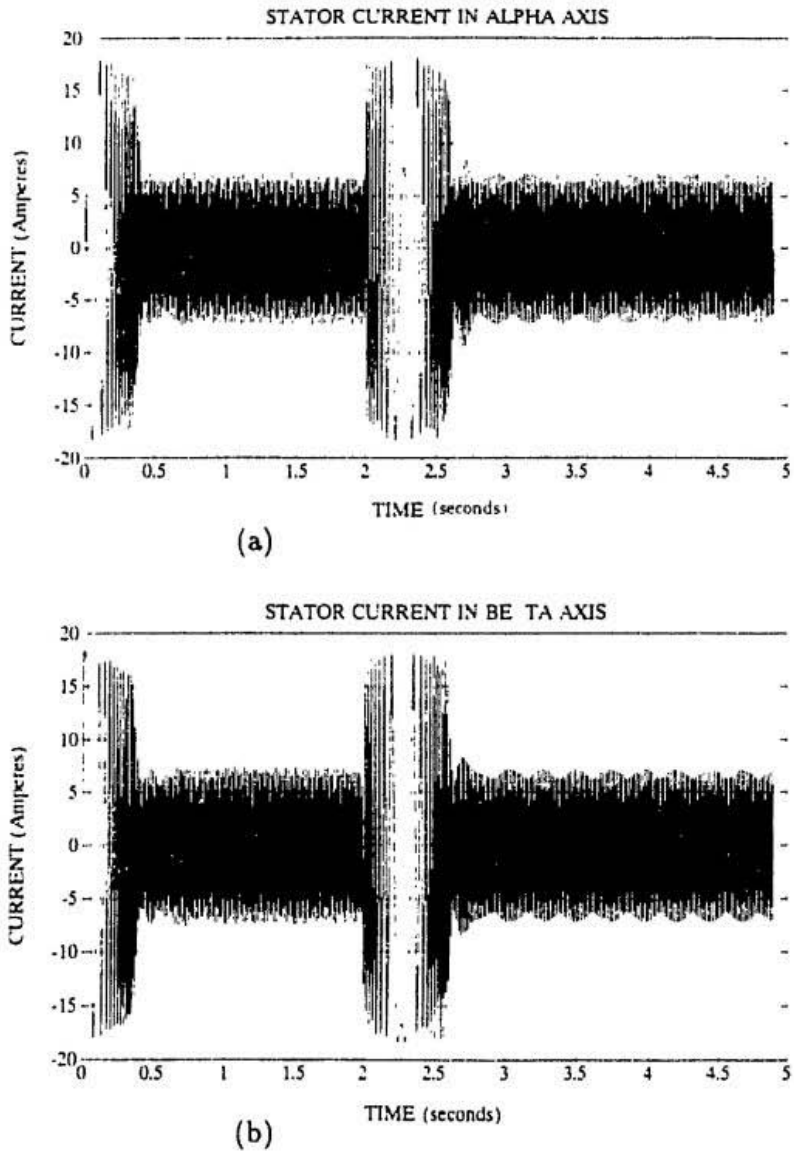
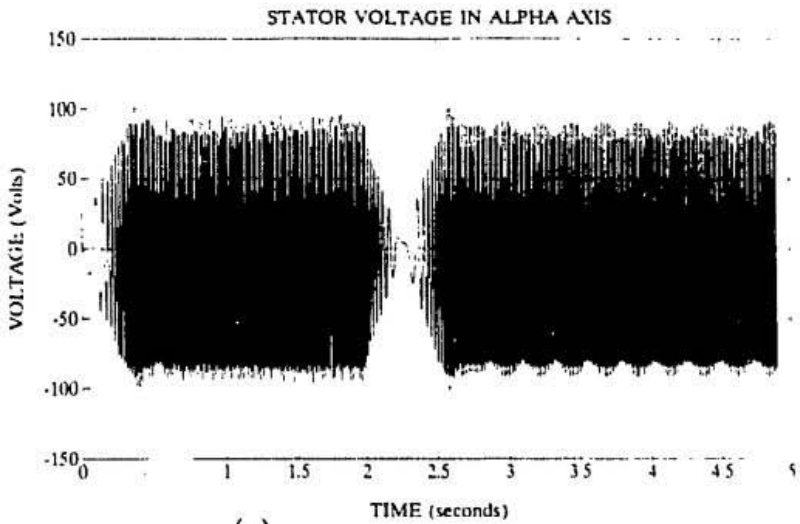
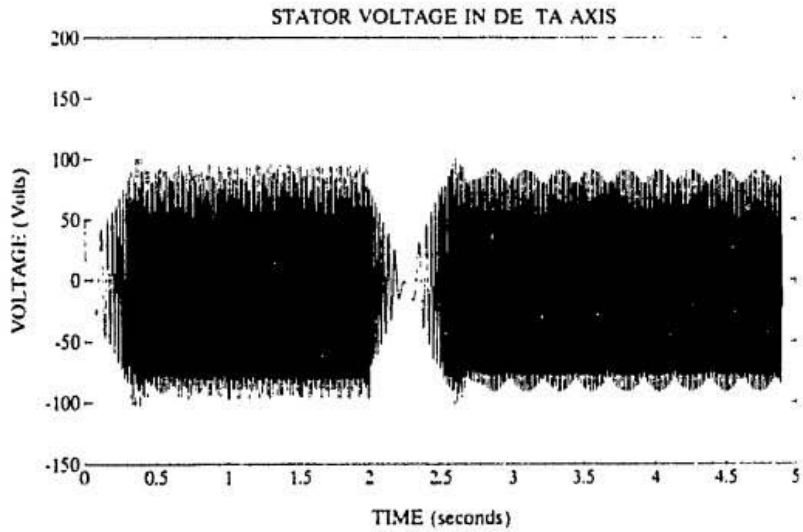


Figure 3.17: System response for speed reversal. a) Stator current in alpha axis, b) Stator current in beta axis



(a)



(b)

Figure 3.18: System response for speed reversal. a) Stator voltage in alpha axis, b) Stator voltage in beta axis

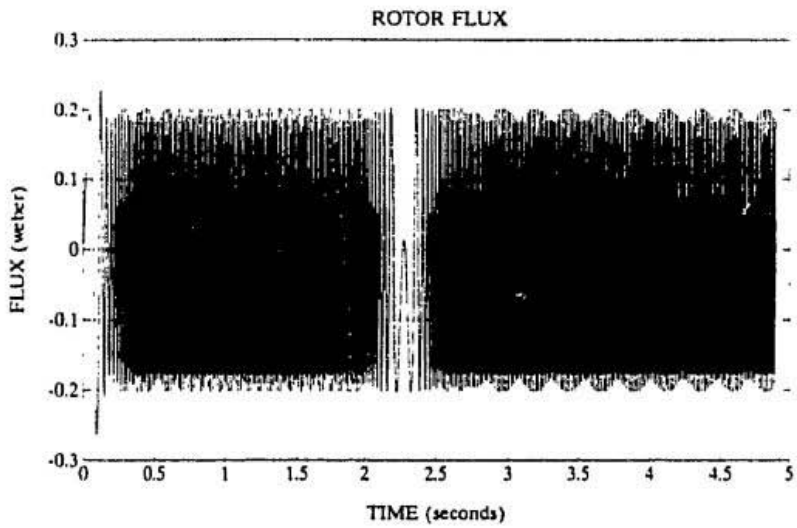


Figure 3.19: Rotor flux response for speed reversal

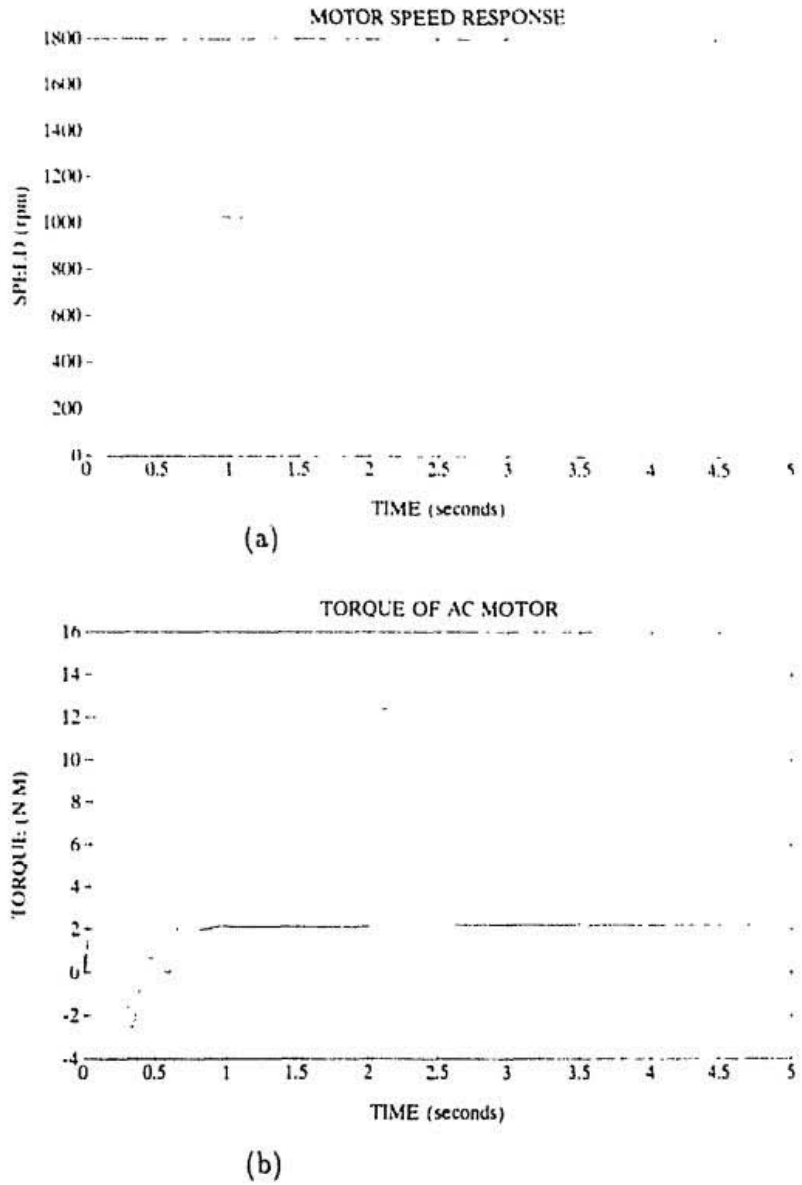


Figure 3.20: System response for step change in the reference speed. a) Rotor speed. b) Motor torque

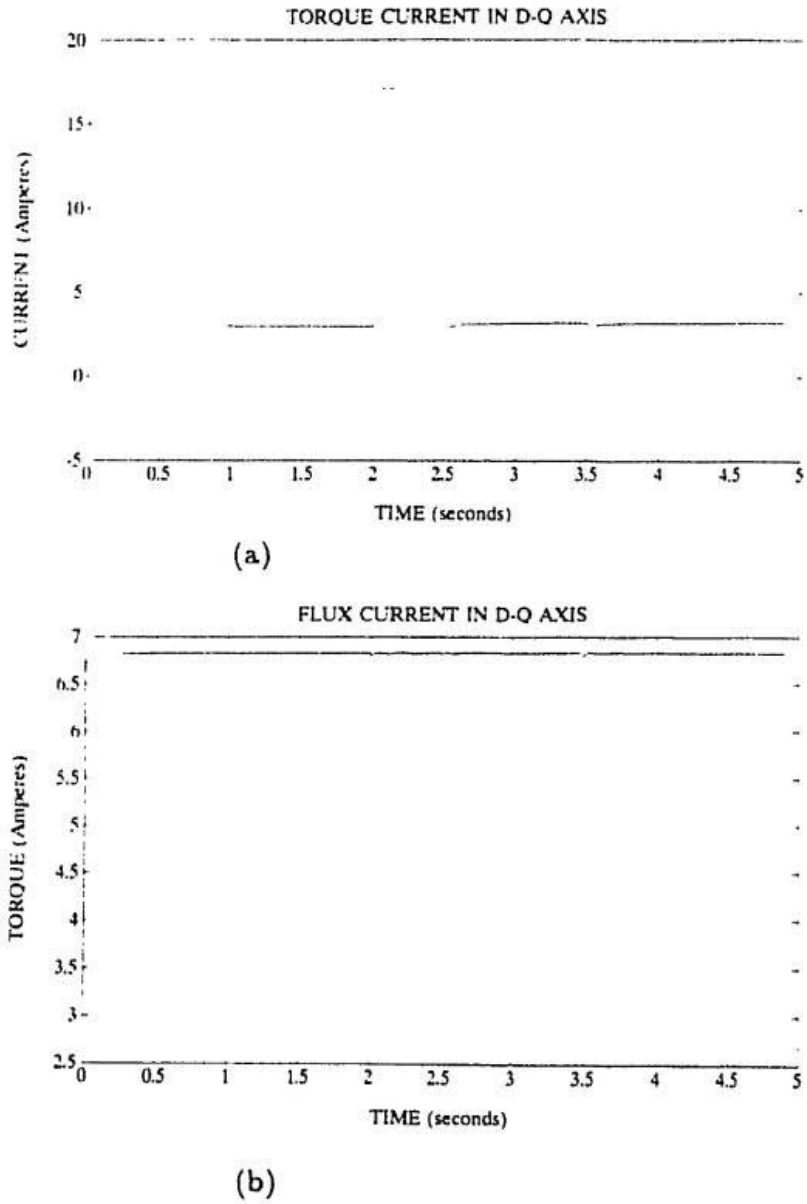


Figure 3.21: System response for step change in the reference speed, a) Torque current, b) Flux current

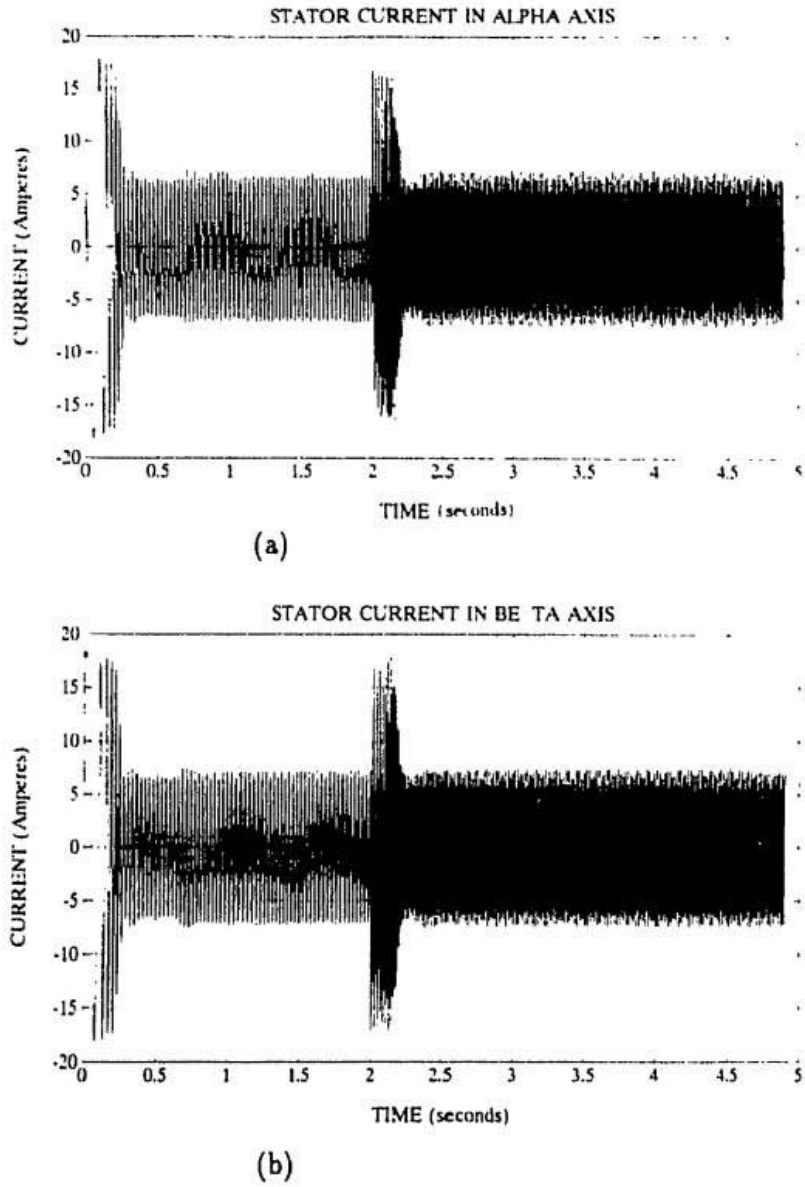


Figure 3.22: System response for step change in the reference speed, a) Stator current in alpha axis, b) Stator current in beta axis

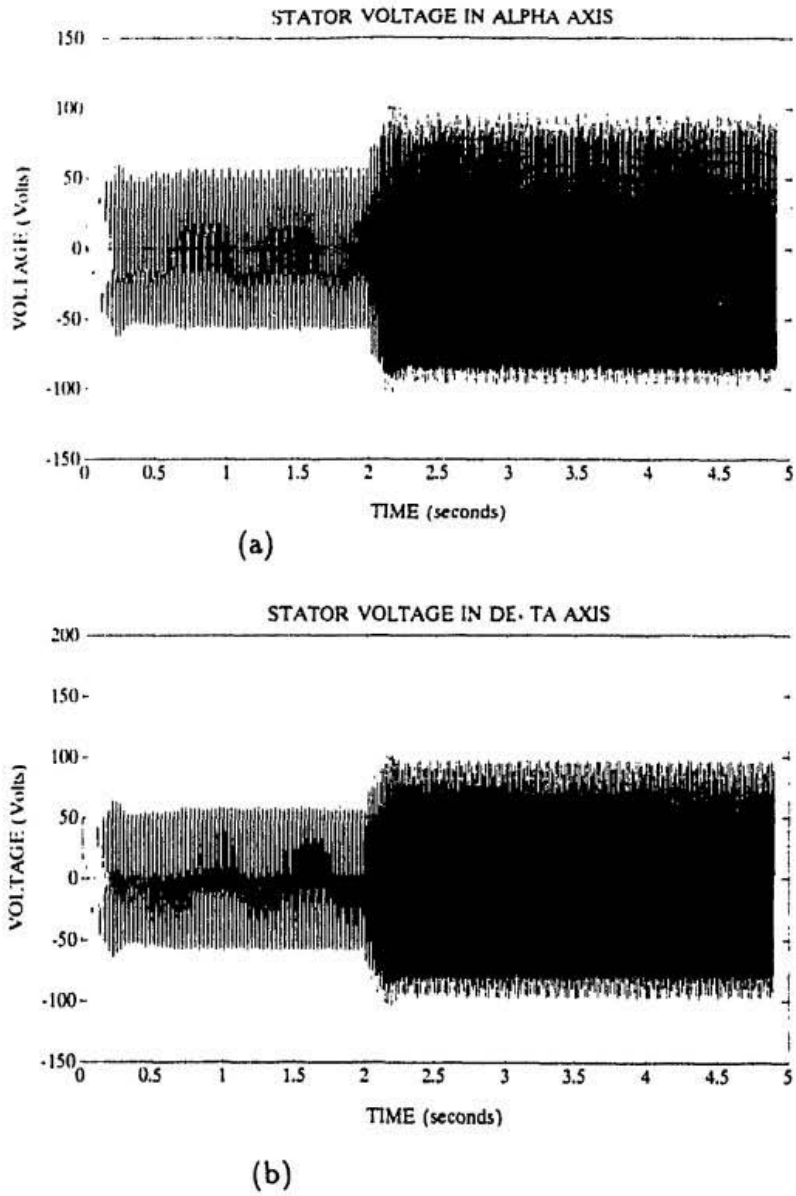


Figure 3.23: System response for step change in the reference speed, a) Stator voltage in alpha axis, b) Stator voltage in beta axis

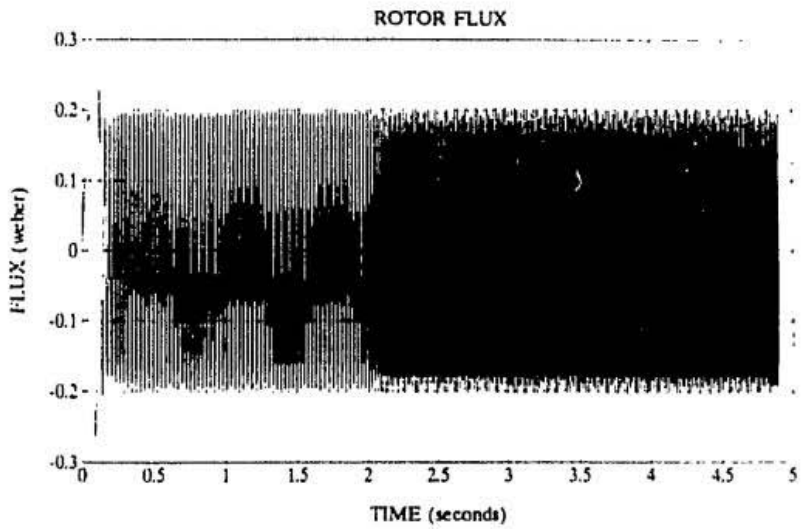
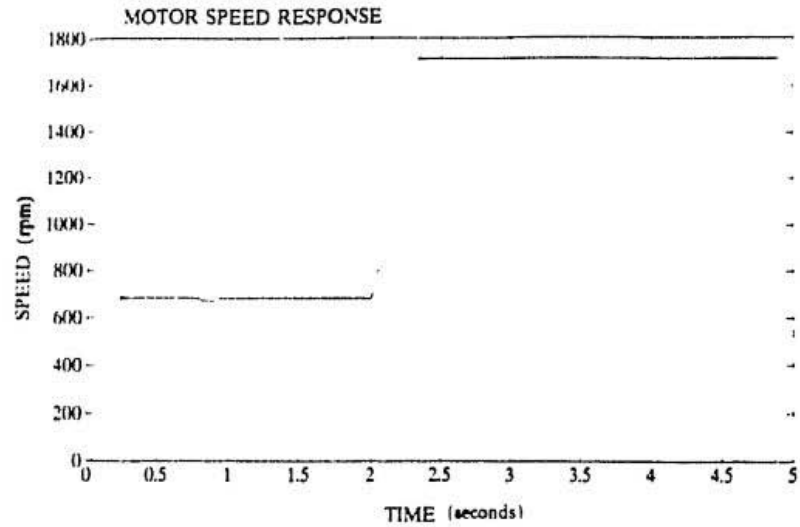
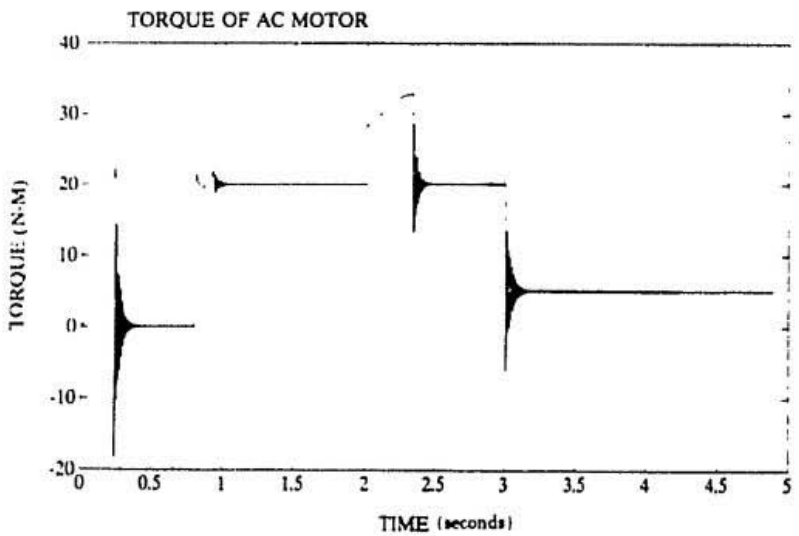


Figure 3.24: Rotor flux response for step change in the reference speed

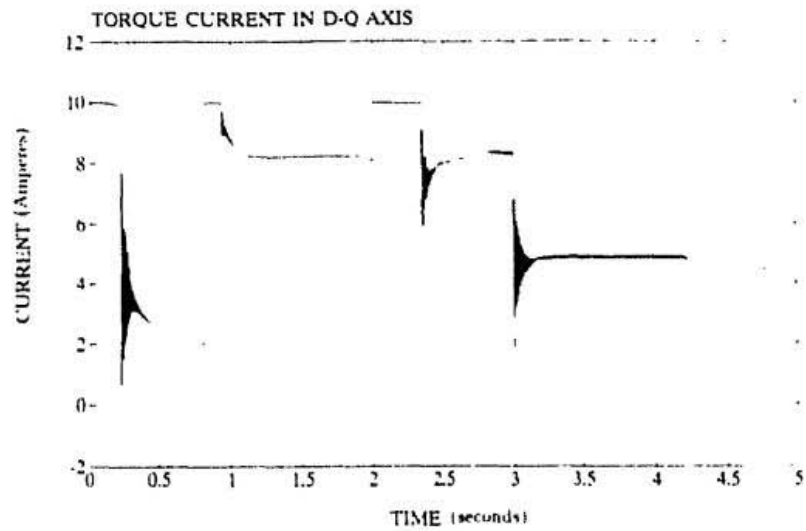


(a)

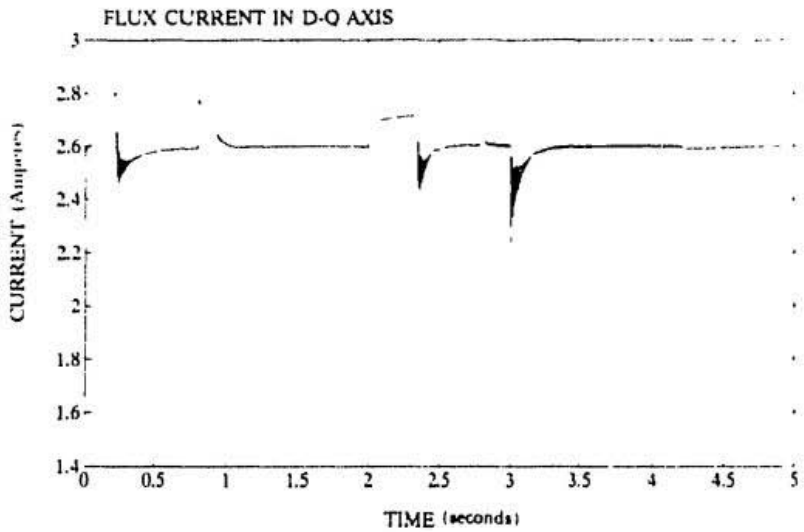


(b)

Figure 3.25: System response with extended field-oriented control, a) Rotor speed, b) Motor torque.

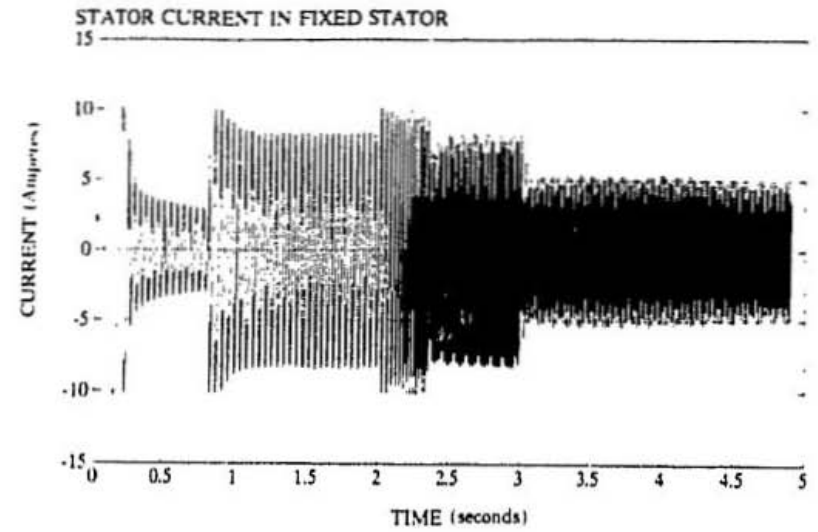


(a)

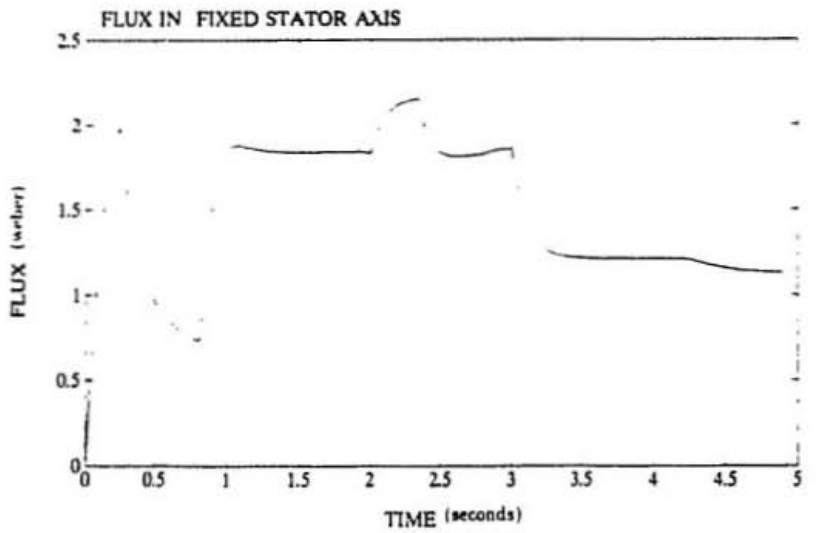


(b)

Figure 3.26: System response with extended field-oriented control, a) Torque current, b) Flux current



(a)



(b)

Figure 3.27: System response with extended field-oriented control, a) Stator current in alpha axis, b) Rotor flux

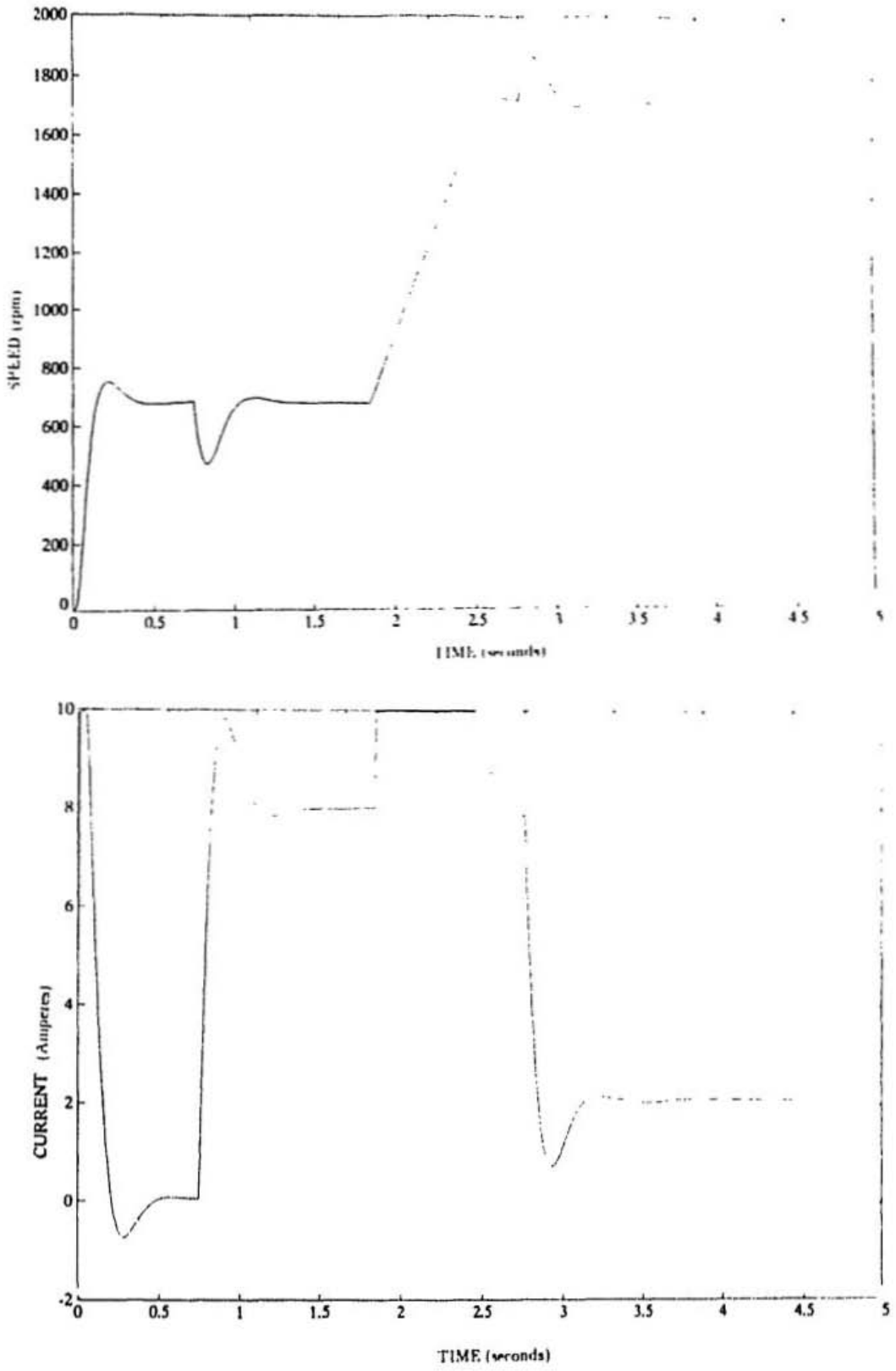


Figure 3.28: System response with field-oriented control

Chapter 4

Rotor Flux Orientation Control

4.1 Introduction

In Chapter 3, a simulation model of indirect field-oriented control of induction motor with squirrel-cage structure was set up. This model has been used to study the control performance of AC motor. It is known that if the rotor parameters keep constant, a linear, decoupled, simplified model of induction motor control system can be derived under the indirect field-oriented control. To implement the indirect field-oriented control, the effects of parameter variations should be considered as discussed in Chapter 2. Efforts to solve this problem are continued by many researchers [1] [6] [2].

In general, the proposals developed so far for adapting parameter variations in the indirect field-oriented control may be divided into two groups. One focuses on the identification of rotor parameters which could be obtained from either indirectly measuring rotor resistance [7] [24] or parameter identification scheme [18] [19]; another is based on the model reference adaptive control scheme which adjusts the control system gain when errors occur between the actual model and the reference model [8] [26] [27].

The purpose of these two approaches is to change the parameters used in slip frequency calculator or to tune the control system gain when rotor parameters change with rising temperature so that the rotor flux can be oriented to the d-axis, which is the essential condition in indirect field-oriented control. The estimated values of rotor parameters obtained from parameter identification are directly used to correct the slip frequency calculation. However, any error resulting from the identification method may lead to a wrong correction. Although the second group makes the control results checkable by feedback control, usually it needs a long time for control convergency, and also the resulting control is affected by load disturbances [6] [9].

So far, most of the schemes developed to cancel the effects of parameter variations on indirect field-oriented control work well in motor steady operation state but not in motor transient operation state, which limits the dynamic control performance.

In this chapter, a new scheme has been proposed to expectedly work for indirect field-oriented control not only in motor steady operation state but also in motor transient operation state.

To describe this new proposal, the principle of the scheme, realization and simulation will be discussed in the subsequent sections.

4.2 The Principle of Rotor Flux Orientation Control

4.2.1 Control Strategy

It is well-known that in the indirect field-oriented control the rotor flux orientation will depart from the d axis if the rotor resistance or inductance changes with increase in rotor temperature or rotor flux saturation. The quantitative description of rotor flux departure from d axis has been studied by Min-Ho, et al [28] and the phasor diagram is presented here (Fig. 4.1) for explanation purposes. Fig. 4.1(a) shows the case where the flux is aligned to d-axis. Fig. 4.1(b) presents the case where the flux departed from d-axis. Defining the angle between flux and d-axis as δ, the following torque and flux current equations can be derived from the vector variable relations as shown in Fig. 4.1(b).

$$i_F = i_d(cos\delta + \frac{i_q}{i_d}sin\delta) \tag{4.1}$$

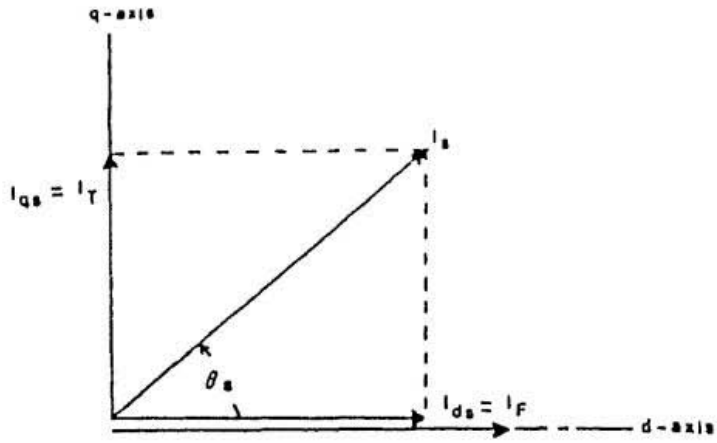
$$i_T = i_q(cos\delta + \frac{i_d}{i_q}sin\delta) \tag{4.2}$$

$$\psi = Mi_F, \tag{4.3}$$

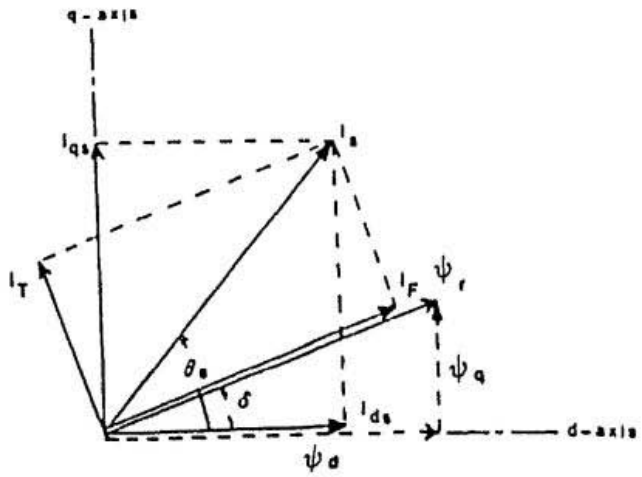
where i_F and i_T are the flux and torque current components in $d-q$ coordinate system; i_{ds} and i_{qs} are the motor stator currents in two phase d, q axis. Due to the departure of flux from d-axis, the developed torque is characterized as:

$$T_e = \frac{3}{2}n_p\frac{M^2}{L_r}i_q i_d(1 - sin\delta cos\delta(\frac{i_q}{i_d} - \frac{i_d}{i_q})). \tag{4.4}$$

In Fig. 4.1(a), as the rotor flux orientation and d-axis coincide, the developed torque is characterized as the same as shown in equation (2.18). In this case, it is clear that



(a)



(b)

Figure 4.1: Flux and torque current vectors in d-q axes. a) Flux orientation aligned to d-axis, b) Flux orientation departed from d axis [28].

the departure angle δ is equal to zero. However, if the rotor flux orientation and d-axis do not coincide, the departure angle is nonzero. The corresponding torque is a complex non-linear function of i_q, i_d and δ.

For controlling the departure angle to zero, a proper feedback control is proposed here. The principle is that when a departure angle occurs due to changes in rotor resistance or flux saturation, this angle is compared with the reference angle (ideally, it is zero). If there is any difference from the comparison, an error signal is fed back to a controller, named flux orientation controller, which sends proper control signals to adjust the rotor time constant T_r used in slip frequency calculator until the departure angle δ becomes zero. Before implementing the proposed scheme, two things have to be studied: one is to analyze the stability of this feedback control system; another is to find a way of detecting the departure angle δ.

4.2.2 Stability Analysis

From the phasor diagram Fig. 4.1(b) and equations (4.1), (4.2) and (3.1), the following equation is obtained [28]:

$$\delta = tan^{-1}\left(\frac{-\frac{\Delta T_r}{T_r}}{\frac{i_d}{i_q} + (1 + \frac{\Delta T_r}{T_r})\frac{i_q}{i_d}}\right), \tag{4.5}$$

where:

1. $T_r = L_r/R_r$ (rotor time constant)

2. T_r^* : normal value of rotor time constant

3. $\Delta T_r = T_r - T_r^*$.

From equation (4.5), it can be seen that the departure angle is a function of flux current, torque current and rotor time constant, which presents a nonlinear and complex problem. Obviously, an accurate analysis is very difficult.

However, in practice, since the time constant of the rotor parameter variation is much larger than that of the induction motor, the departure angle can be adjusted instantaneously by controlling stator current. Therefore, the equation (4.5) can be simplified as in equation (4.6) because departure angle will be instantaneously corrected by adjusting motor stator current. In other words, the presence of departure angle is very small ($\Delta T_r << 1$) due to fast current adjustment.

$$\delta = \frac{-\frac{\Delta T_r}{T_r}}{\frac{i_d}{i_q} + (1 + \frac{\Delta T_r}{T_r})\frac{i_q}{i_d}}. \tag{4.6}$$

Rewriting the equation (4.6), it becomes as the following:

$$\delta = \frac{-\frac{\Delta T_r}{T_r} i_q i_d}{i_d^2 + i_q i_d + \frac{\Delta T_r i_q^2}{T_r}}. \tag{4.7}$$

As the flux current and torque current can be controlled instantaneously, for the departure angle control, the i_d and i_q can be comparatively considered as constants. Therefore, the relations between departure angle and rotor time constant can be simply considered as a linear and non-inertial process. Of course, at the transient adjustment, the relations are much more complex.

4.2.3 Detection of the Departure Angle

By viewing equation (4.4), we see that the developed torque T_e is composed of two parts: one is the motor torque by regular field-oriented control, which presents no parameter variations in rotor; another is an undesirable motor torque which is

caused by rotor parameter variations. The motor torque by regular field-oriented control can be easily obtained using d-q current variables as shown in equation (2.8).

Now, it is clear that if the actual motor torque can be obtained, then the departure angle can be detected by using equation (4.4). There are many methods to obtain actual motor torque. But a simple, easy and accurate scheme introduced by Lorenz [8] is to estimate the motor torque from stator flux, which takes advantage of the low parameter sensitivity of the stator voltage in estimating stator flux:

$$\vec{\psi_{\alpha\beta}} = \int (\vec{u_{\alpha\beta}} - \vec{i_{\alpha\beta}} R_s) dt. \tag{4.8}$$

This solution allows the stator flux to be calculated from terminal voltages and phase current. The actual torque can then be directly estimated from the stator flux and stator current according to:

$$\hat{T} = \frac{3}{2} n_p \vec{\psi_{\alpha\beta}} \vec{i_{\alpha\beta}} \tag{4.9}$$

or

$$\hat{T} = \frac{3}{2} n_p (\psi_\alpha i_\beta - \psi_\beta i_\alpha). \tag{4.10}$$

Fig. 4.2 shows the approach which is well-known as a means of estimating motor torque from terminal variables only and is the basis of the robustness of stator flux oriented control. Now using equation (4.4) and (4.10), the departure angle can be obtained by computing the following equations:

$$\hat{T} - T = \Delta T = \frac{3}{2} n_p \frac{M^2}{L_r} i_d i_q sin\delta cos\delta (\frac{i_q}{i_d} - \frac{i_d}{i_q}) \tag{4.11}$$

or

$$\Delta T = \frac{3}{4} \frac{M^2}{L_r} n_p sin(2\delta)(i_q^2 - i_d^2), \tag{4.12}$$

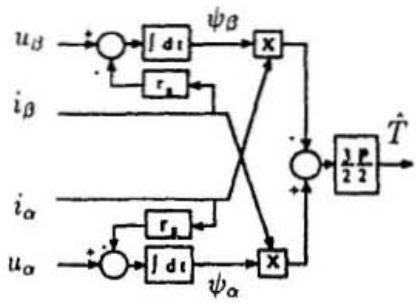


Figure 4.2: Stator-oriented model for estimating torque [8]

then

$$\delta = \frac{1}{2} sin^{-}(\frac{4\Delta T L_r}{3n_p M^2(i_q^2 - i_d^2)}). \tag{4.13}$$

4.3 Realization of Flux Orientation Control

Based on the departure angle detection, a feedback control technique could be applied for the control of flux orientation. The proposed control system and controller structure are shown in Fig. 4.3 and Fig. 4.4. In Fig. 4.3, the departure angle is calculated from torque difference (ΔT) and is then sent to flux orientation controller in which the slip frequency calculation is corrected. The controller is chosen as a proportional regulator:

$$y = K_F * \delta, \tag{4.14}$$

where K_F is the gain of the controller. The controller output y is then added with the normal rotor time constant for adjusting the rotor time constant used in the

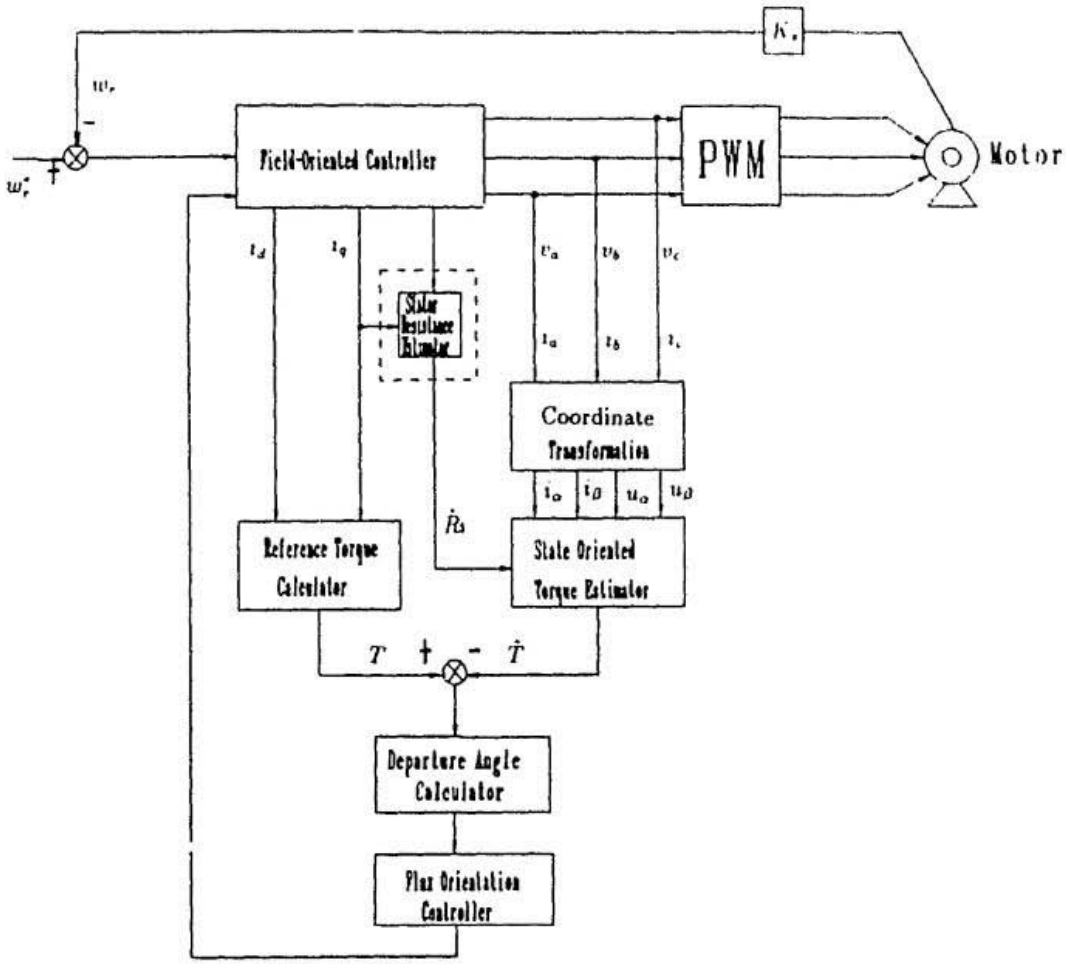


Figure 4.3: Parameter adaptation control for indirect field-oriented control of AC induction motor

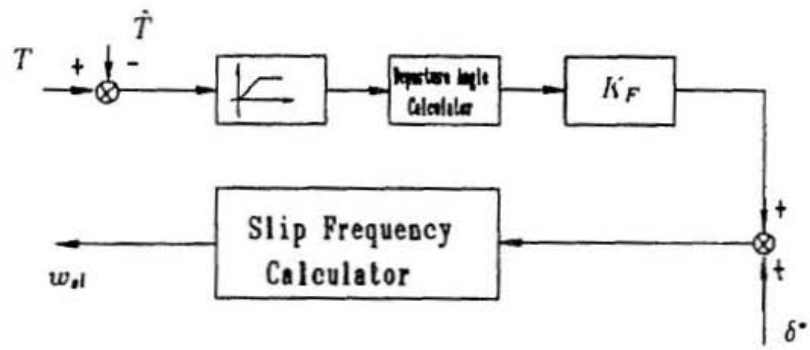


Figure 1.1: Flux orientation controller

slip calculator. The corrected rotor time constant used in slip calculator adjusts the motor stator currents such that the flux orientation is controlled to aligned in the direction of d-axis eventhough the rotor time constant keeps changes. The control process can be explained that when the departure angle is zero, the addition is equal to the value of normal rotor time constant; when nonzero departure angle is resulted by the rotor parameter variations, the addition is equal to the value of normal rotor time constant and the value of departure angle. This results in the changes of the slip frequency and, thus corrects the flux departure.

It is noted that the controller may not just be a proportional controller. It could be a PI or PID controller. However, the simulation results show that the feedback controller works well in both steady and dynamic states for control of the departure angle. If the corrected rotor resistance resulted from the flux orientation control goes to a filter, a better accuracy result can be achieved [39].

4.4 Simulation and Results

In order to verify the validity and the feasibility of this method, the mathematical model of the indirect field-oriented control system with rotor flux orientation control is set up by equations (3.16 - 3.23) and (4.10 - 4.14), and a series of digital computer simulations have been carried out. In the simulation, the motor was operated under variable load and variable speed conditions which verified that whenever the system is working in variable load or constant load and variable speed or constant speed, the proposed scheme for eliminating the effects of rotor parameter variations are correct.

In order to test a worse-case in rotor resistance variations, the R_r setting unit is

set as given by equation (4.15). In Chan's simulation [6], the rotor resistance was assumed to change from 0.3 ohm to 0.7 ohm in about 12 seconds. In Loranz's [7], the rotor resistance was assumed to change from high to low in 25 seconds. In our test, the rotor resistance is assumed to change from high to low and then comes back from low to high in only 4.5 seconds. Such a faster change in rotor resistance could test a worse-case for rotor flux orientation controller. Since the higher rotor resistance the less effect on performance (see Chan's simulation), variations of rotor resistance in our test are supposed to change from a normal value to a smaller value and then return back to its original value, as illustrated in Fig. 4.5.

$$R_r = 0.08 * (t - 3)^2 + 0.125 \tag{4.15}$$

Simulation program and parameters of the motor are the same as in Appendix A-1 and Appendix A-2. Simulation results are shown in Figs. 4.6 - 4.13.

Figures 4.6(a), 4.7(a), 4.8(a), 4.9(a), and 4.10(a) show the responses of indirect field-oriented control system in the condition of no changes in rotor resistance.

Figures 4.6(b), 4.7(b), 4.8(b), 4.9(b), and 4.10(b) show the responses of indirect field-oriented control system in the condition of changes in rotor resistance without flux orientation control. The results indicate that when R_r changes, the flux current, flux level, stator current and motor torque also change with a result of non-linear response in the motor torque and rotor speed. The system behaves non-linearly and the control performance is degraded dramatically.

Figures 4.11 - 4.13 show that the system responses are unaffected by rotor resistance variations when the flux orientation feedback control is applied. The system is linearly controlled as the same with the properties of conventional field-oriented control system as shown in Figures 4.6(a), 4.7(a), 4.8(a), 4.9(a) and 4.10(a).

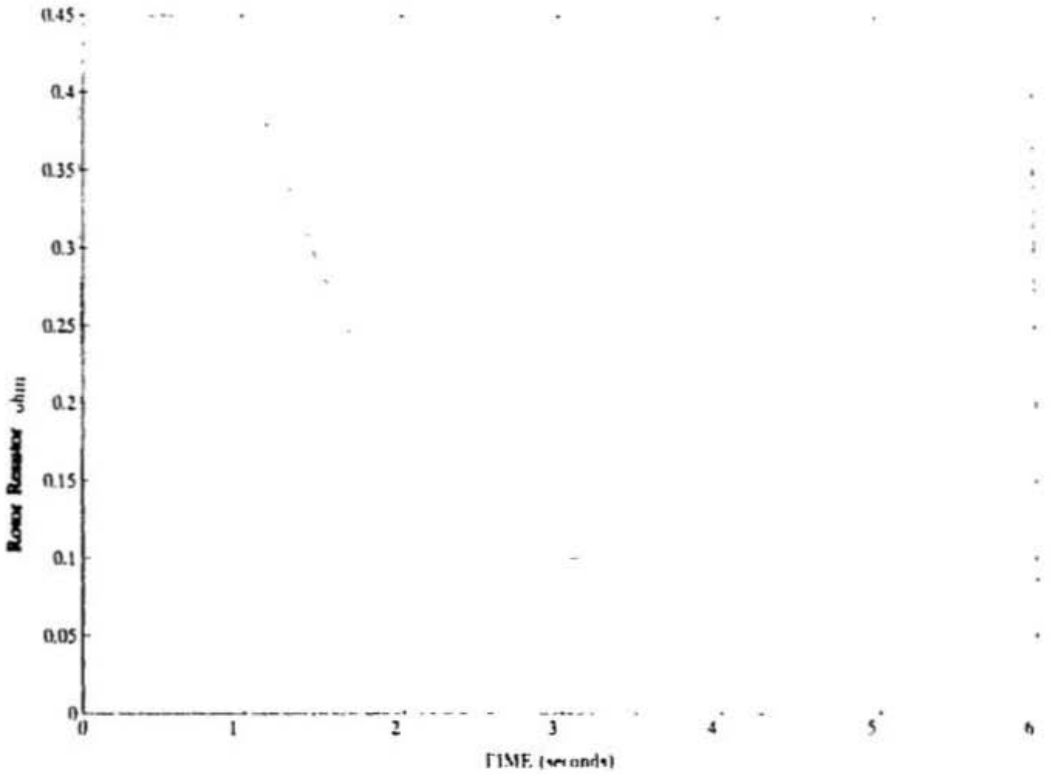


Figure 4.5: Motor rotor resistance variations

The corrected rotor resistance is shown in Fig. 4.13(b) which is very close to the actual rotor resistance as shown in Fig. 4.5. As it can seen that the corrected rotor resistance has certain dynamical errors becouse of the feed-back control adjustment.

4.5 Discussion

4.5.1 Advantages of the Rotor Flux Orientation Control

Parameter identification applied in indirect field oriented control for parameter adaptive control has a feature of forward calculation. The calculation result from parameter identification method is directly sent to the slip frequency calculator for correcting the mismatched value that are used for coordinate transformation calculations. Therefore, if any errors resulted from parameter identification calculation or were caused by some uncertain factors, the calculation of field coordinates will not be accurate.

Techniques developed to date [7] [18] [24] [8] have a limitation to solve uncertain parameter variations or external disturbance such as variable load and unstable power source caused by PWM inverter.

The flux orientation control scheme proposed in this thesis provides solution for these problems, because a feedback control system not only can instantaneously adjust the controlled system outputs to the desired target but also can overcome uncertain factors which may be harmful to control system. For example, if the power frequency is changed by external disturbances, the non-zero departure angle is detected and compared with the reference value through the feedback controller. Then the departed flux orientation caused by power frequency variation can be corrected by feedback controller.

The features of the proposed scheme for correcting flux orientation errors can be summarized as follows:

1. The method considers not only the effects of rotor resistance and rotor self inductance variations, but also any effects which may result in flux orientation departure from its desired position due to its feedback control properties.

2. Implementation of the flux orientation feedback control is easy, because only terminal variables are needed.

3. Regardless of whether the motor is working under full load or no load, transient state or steady state, variable speed or constant speed, this method is suitable.

4. This method works not only for high speed motor control but also for low speed operation when it is incorporated with motor stator resistance identification which will be discussed in next section.

Obviously, these features are excellent and valuable. The proposed parameter adaptation control scheme is expected to be applied widely for indirect field-oriented control systems.

4.5.2 Limitations and Solutions

As a result of stator flux based torque estimation for detecting departure angle, the proposed scheme is limited to high performance velocity-controlled applications that do not require low-speed operation. The reason is that with stator flux based torque estimation this technique loses accuracy and becomes unreliable at low speeds due to the increased dependency of stator voltage on stator resistance

[8] [9]. To overcome such a problem, a simple stator resistance identification can be used if adaption at low operating frequencies is required.

Stator Resistance Identification

To directly obtain the dynamic value of stator resistance during system operation is difficult. So far, no effective methods have been reported in the literature for identifying the dynamic stator resistance. Obviously, this limits the application of stator flux orientation control and stator flux-based estimation of motor torque.

In order to solve this problem, a new and simple approach for dynamic stator resistance identification is developed here.

In Chapter 3, it is known that the torque current control loop can be simplified, if a feed-forward controller is employed. From equation (3.6), the following equations are derived:

$$u_q = R_s(1 + \rho T_s S)i_q \tag{4.16}$$

or

$$u_q = R_s i_q + (L_s - \frac{M^2}{L_r})\frac{di_q}{dt}. \tag{4.17}$$

Then the stator resistance can be simply expressed as:

$$R_s = \frac{u_q - (L_s - \frac{M^2}{L_r})\frac{di_q}{dt}}{i_q}. \tag{4.18}$$

Viewing equation (4.18), we see that the identified stator resistance R_s is dependent on stator inductance L_s, mutual inductance M and rotor inductance L_r as well as u_q and i_q. Item M^2/L_r has been shown not to change with temperature [24]. In general, variations of stator inductance is dependent on stator flux saturation which is caused by a high voltage to stator. As the stator resistance identification is only

tuned in at low speed operation which is corresponding to low stator voltage, R_s can be considered only as a function of u_q and i_q.

Simulation and Results

The simulation results have been shown in Fig. 4.14 - 4.15. The simulation program and motor parameters are listed in Appendix B-1 and Appendix B-2, respectively. The controlled motor was working at low and variable speed. The load was changed from 0 to 5 Nm and 5 Nm to 2 Nm during the tests. Fig. 4.14(a) presents the identified motor stator resistance in the condition of variable speed and load. Fig. 4.14(b) gives the motor speed response. Fig. 4.15(a) indicates the motor torque induced by variable load. Fig. 4.15(b) presents the rotor flux current. The identified stator resistance R_s is very accurate which is very close to the original value of 2.1Ω. It can be seen that the identified stator resistance is not affected by speed variations and changes in load.

4.6 Comparisons

As a relatively easy and inexpensive implementation, the indirect field-oriented control has been used to achieve a high performance control of an AC induction motor in various application fields. However, this favourable scheme suffers from rotor parameter variations with rising temperatures and flux saturation. Many schemes to overcome such problem have been tried out. Unfortunately, the efforts to date have not provided satisfactory results.

The present work applied to indirect field-oriented control place emphasis on the parameter identification and model reference adaptive control. The parame-

ter identification is limited to the estimation of an expected variable, such as the rotor time constant. Any unknown factors will lead to a mis-correction in slip frequency calculation. Model reference adaptive control application is restricted due to complex designs, complex calculations and long convergence times. So far, a wide acceptance of parameter adaptation control scheme has not been found [6] [1].

A standard version for using parameter adaptive control has been suggested by some researchers [24] [6] as described in this chapter. The proposed scheme in this study has shown results very close to the standard version. In order to give a further proof of the proposed scheme, another simulation has be carried out using Chan's scheme.

Chan[7] proposed an identification method in which the rotor resistance can be estimated in motor steady operation. In this simulation, the rotor resistance variations with rising temperature stand for the worst case as defined in equation (4.15).

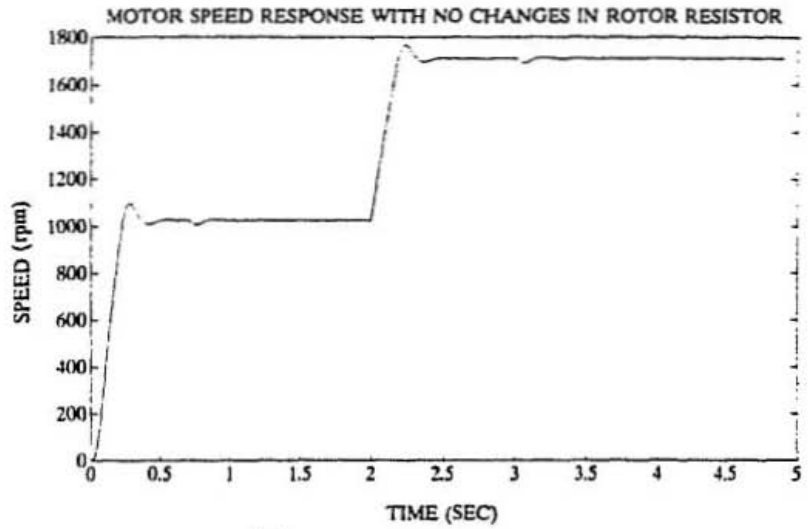
Figs. 4.16 - 4.17 show the simulation results. The simulation program is listed in Appendix-E and motor parameters are the same as in Appendix A-2.

It can be seen that in the transient response of the rotor speed, control performance is affected by variations of rotor resistance even when C.C. Chan's parameter identification scheme is incorporated as shown in Figs. 4.16 - 4.17.

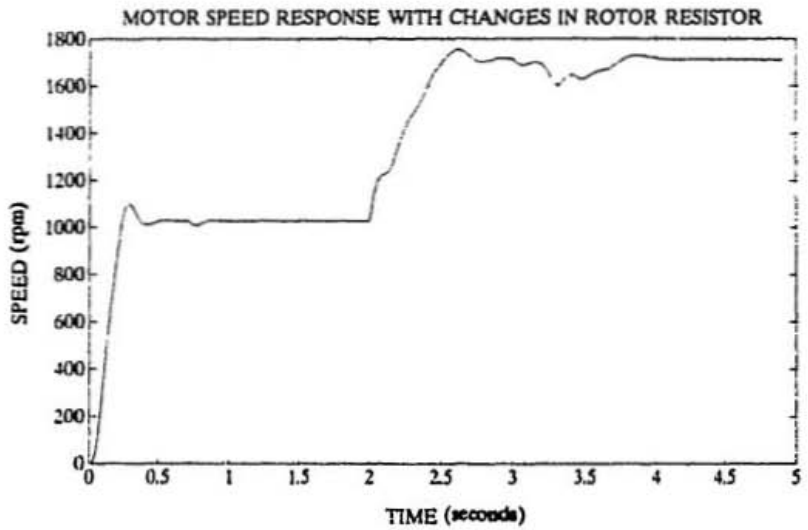
Fig. 4.16(b) shows a wrong identification result of rotor resistance when the control system is operating in transient state. The resulting non-linear control characteristics between torque current and motor torque are shown in Figs. 4.17(a) - 4.17(b).

Compared with Chan's proposal, the new scheme proposed in this thesis is

much better in obtaining dynamic identification results as shown in Fig. 4.13(b). The steady identification results are the same as Chan's. The proposal by Chan is only suitable in the motor steady operation for the adaptive control of rotor resistance variations, but not rotor inductance. The proposal here is not only suitable to correct the rotor time constant in both dynamic and steady state, but is also suitable to correct the rotor time constant variations induced by unknown factors with the help of applied feedback control of flux orientation.

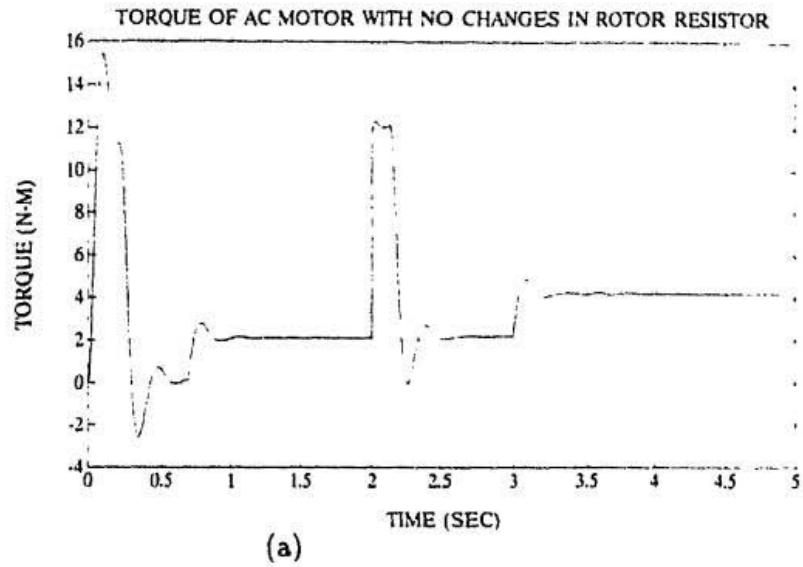


(a)

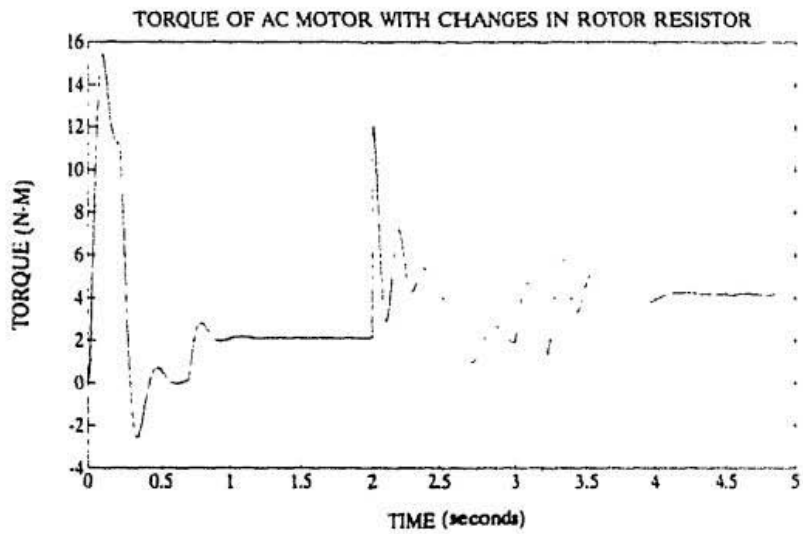


(b)

Figure 4.6: Response of indirect field-oriented control, a) Rotor speed, b) Rotor speed



(a)



(b)

Figure 4.7: Responses of indirect field-oriented control, a) Motor torque, b) Motor torque

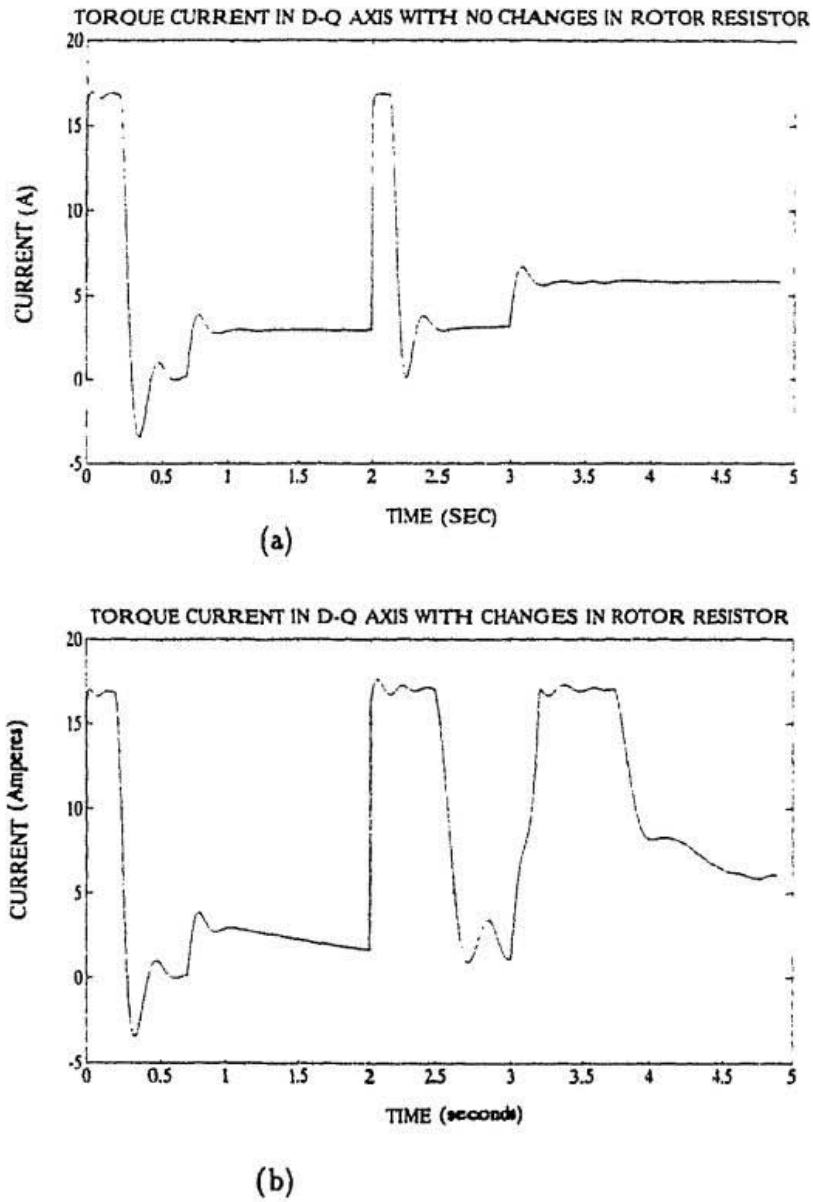


Figure 4.8: Responses of indirect field-oriented control, a) Torque current, b) Torque current

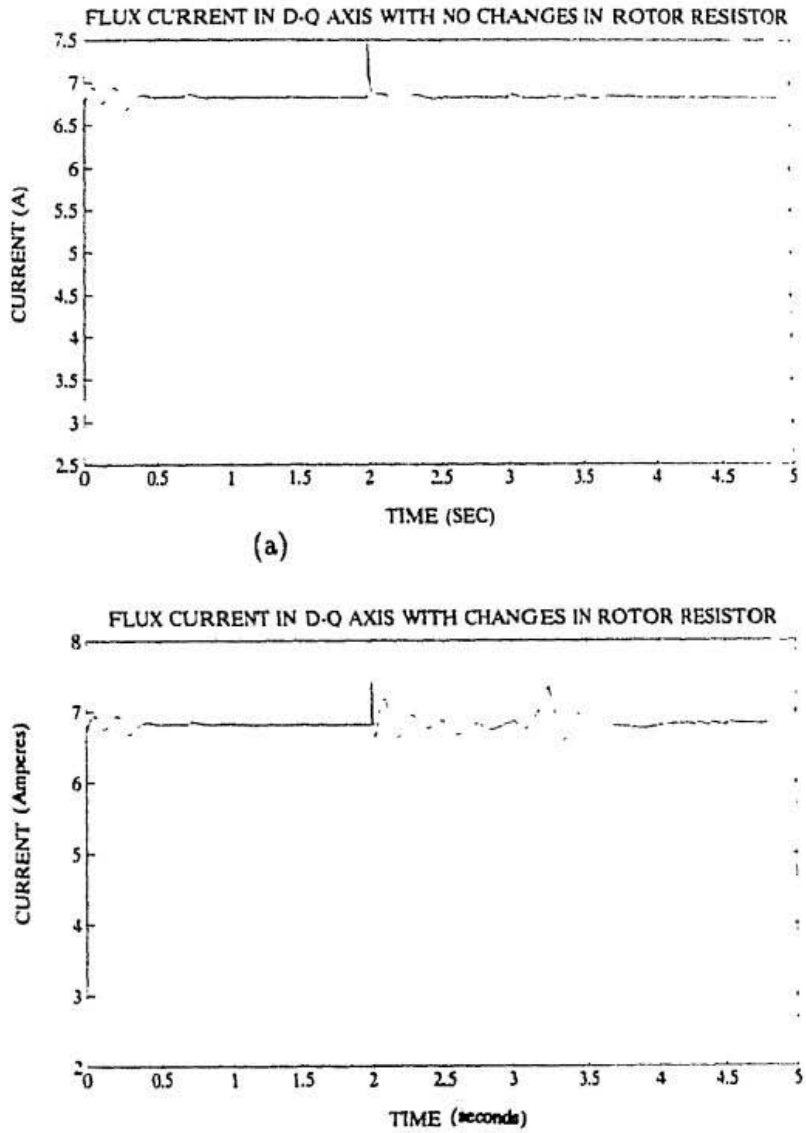
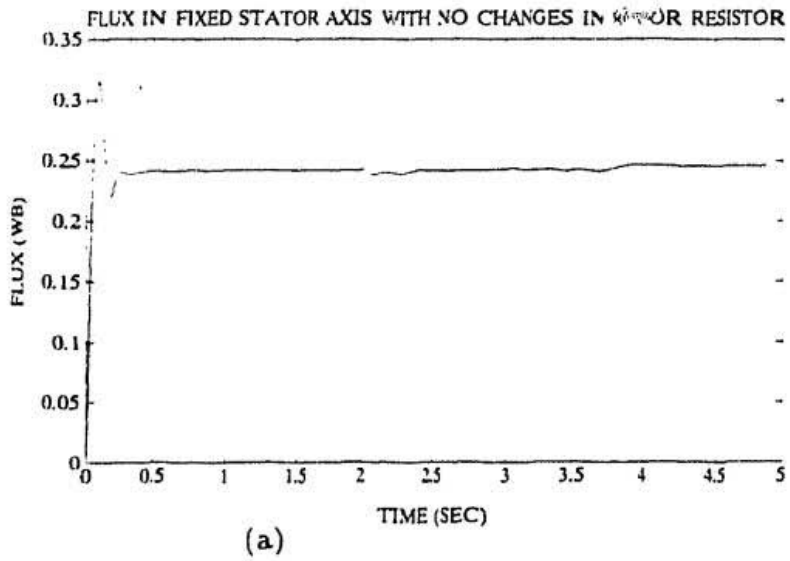
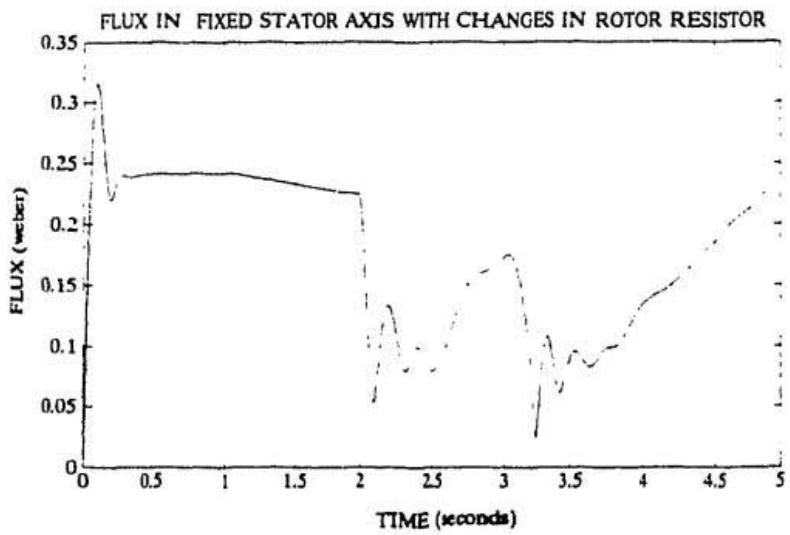


Figure 4.9: Responses of indirect field-oriented control, a) Flux current, b) Flux current

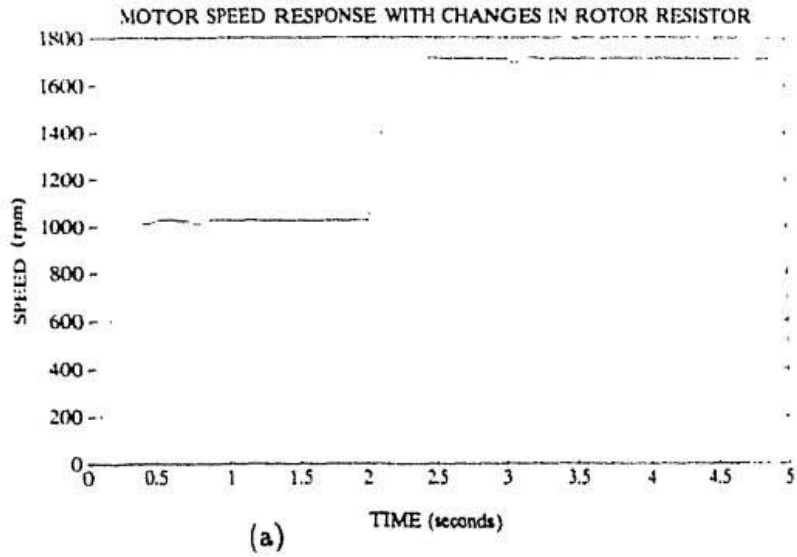


(a)

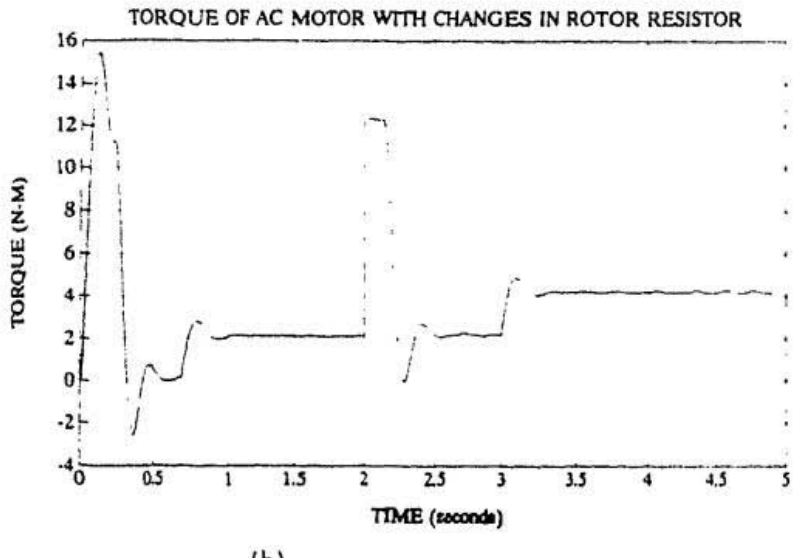


(b)

Figure 4.10: Responses of indirect field-oriented control, a) Rotor Flux, b) Rotor Flux

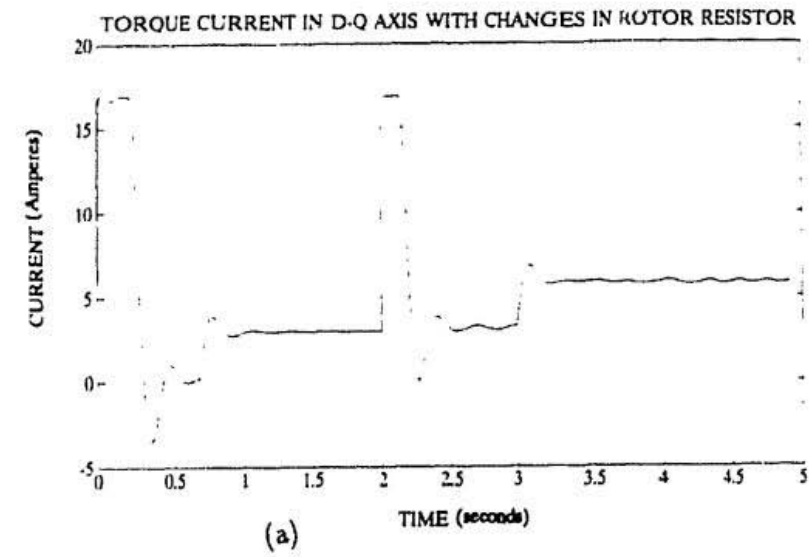


(a)

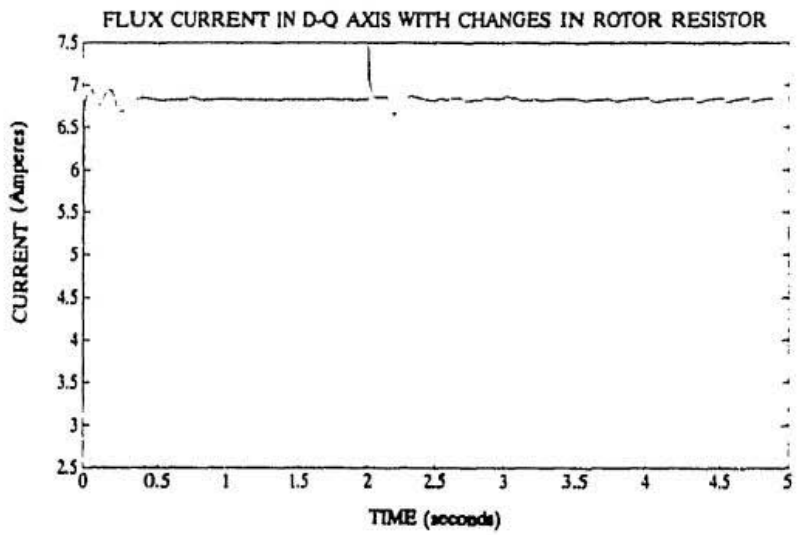


(b)

Figure 4.11: Responses of indirect field-oriented control with flux orientation control, a) Rotor speed, b) Motor torque

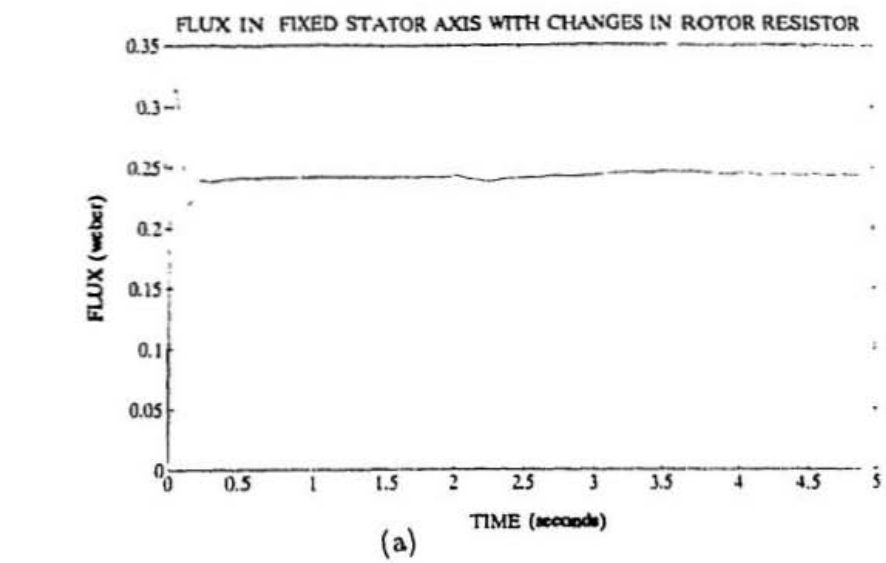


(a)

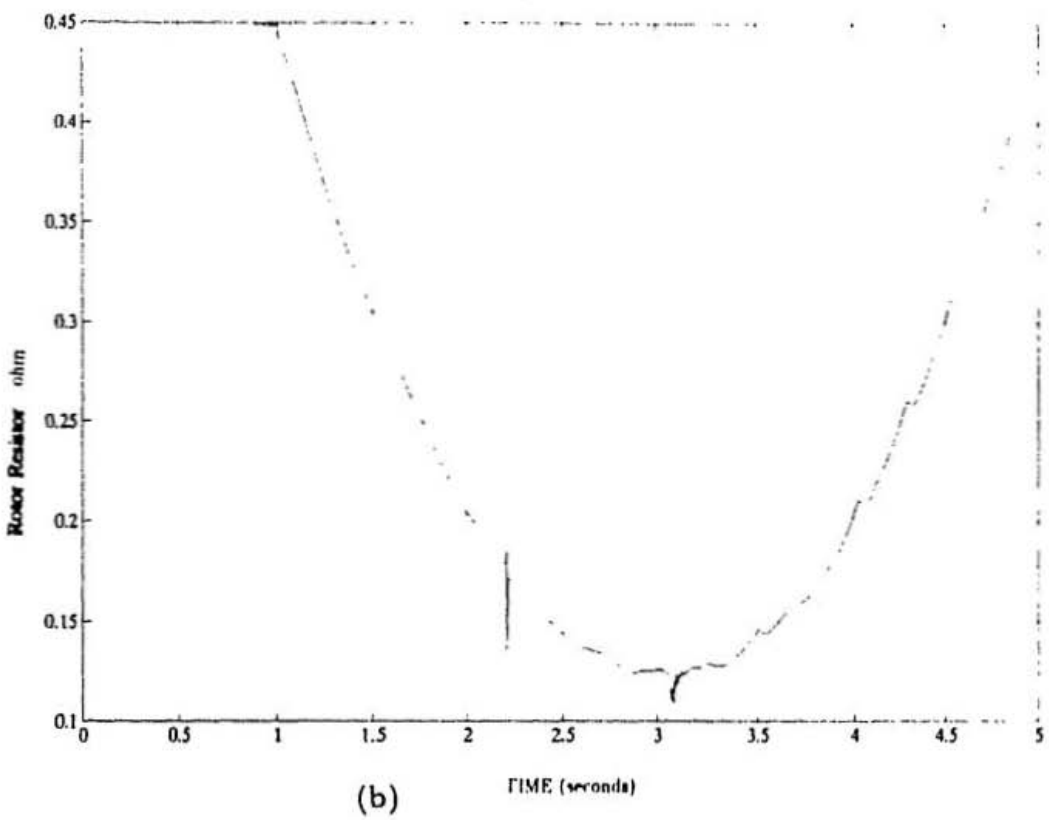


(b)

Figure 4.12: Responses of indirect field-oriented control with flux orientation control, a) Torque current, b) Flux current



(a)



(b)

Figure 4.13: Responses of indirect field-oriented control with flux orientation control, a) Rotor Flux, b) Corrected rotor resistance

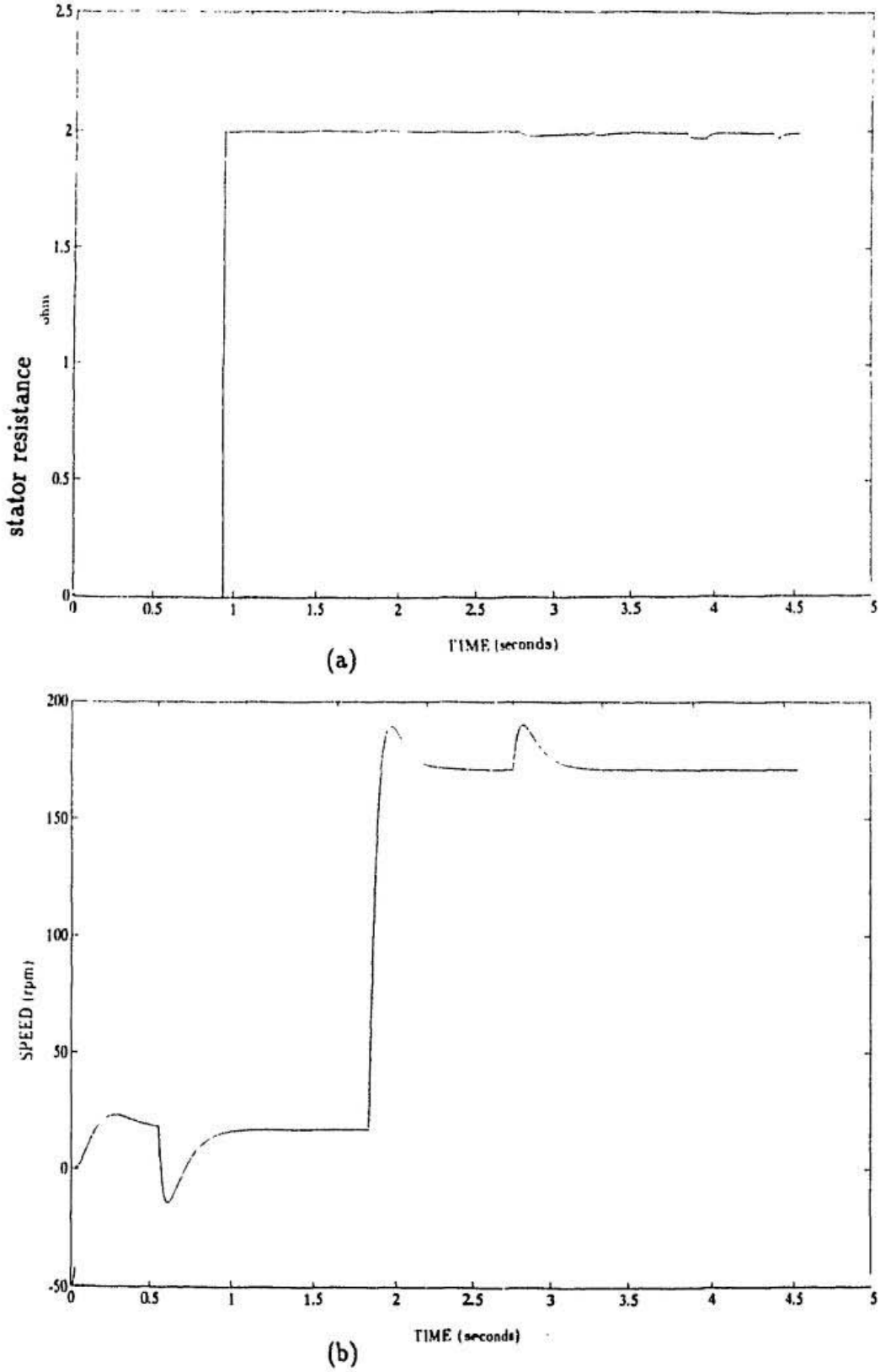


Figure 4.14: Response of indirect field-oriented control, a) Identified stator resistance, b) Rotor speed

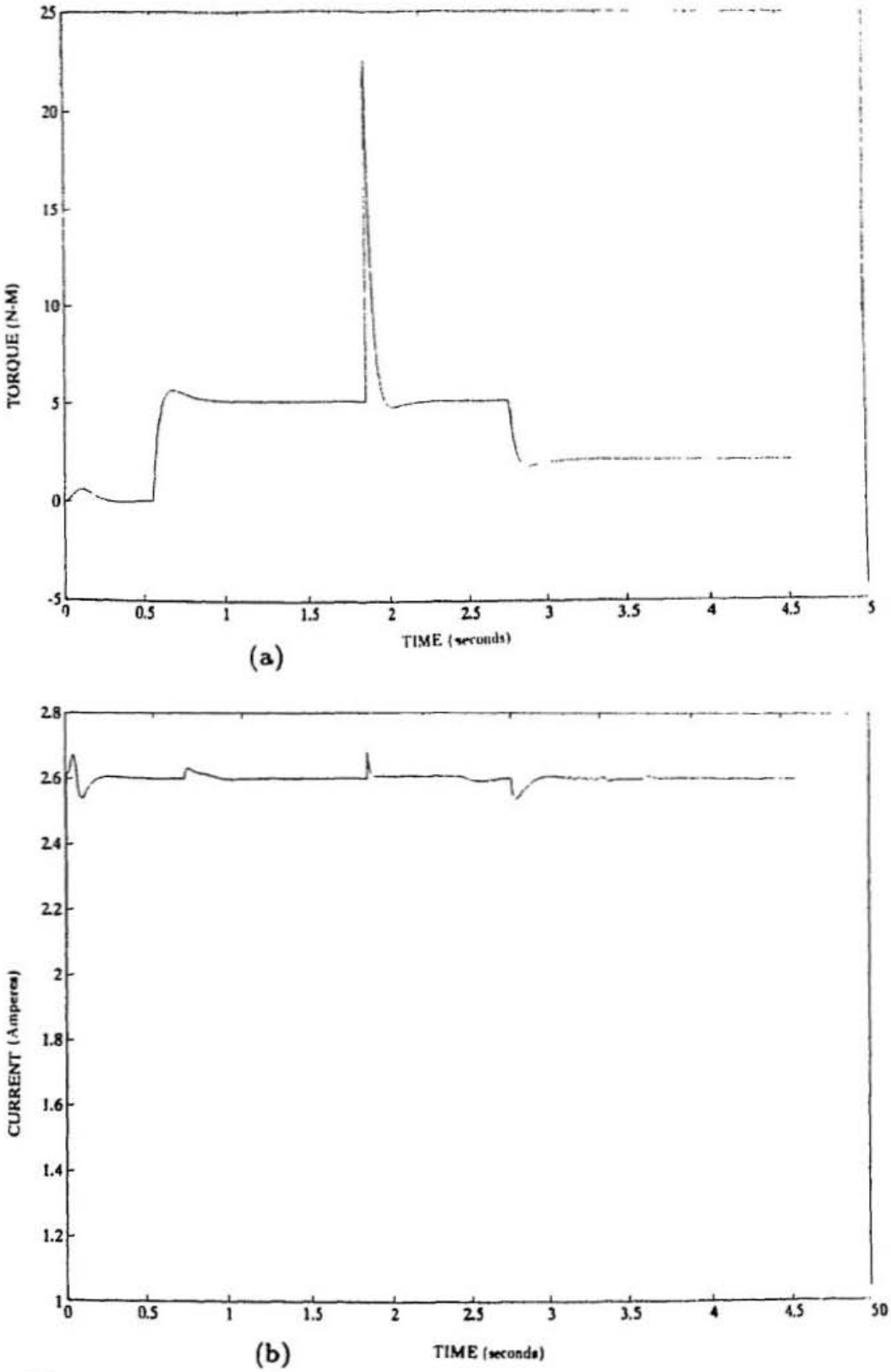
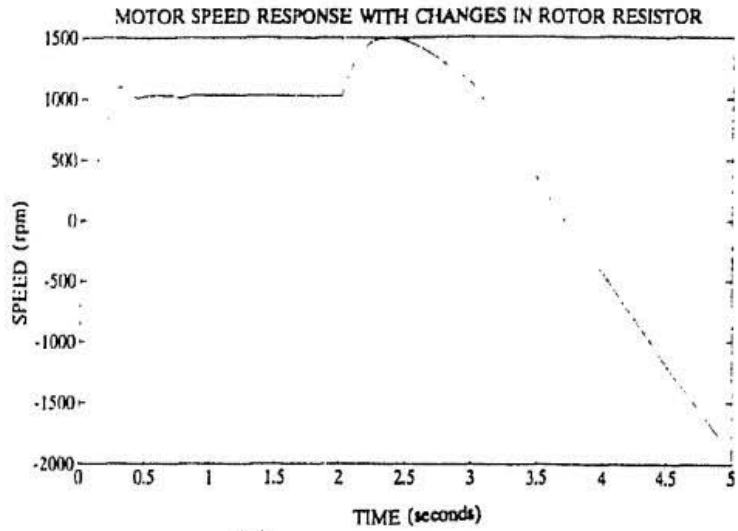
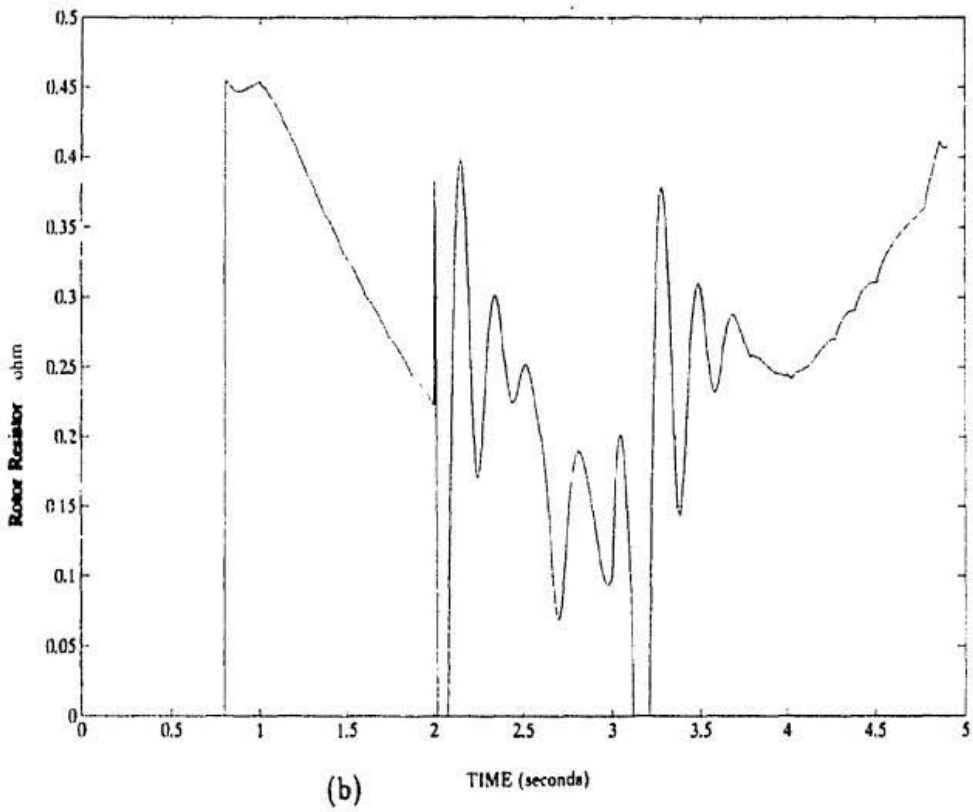


Figure 4.15: Response of indirect field-oriented control, a) Motor torque, b) Flux current

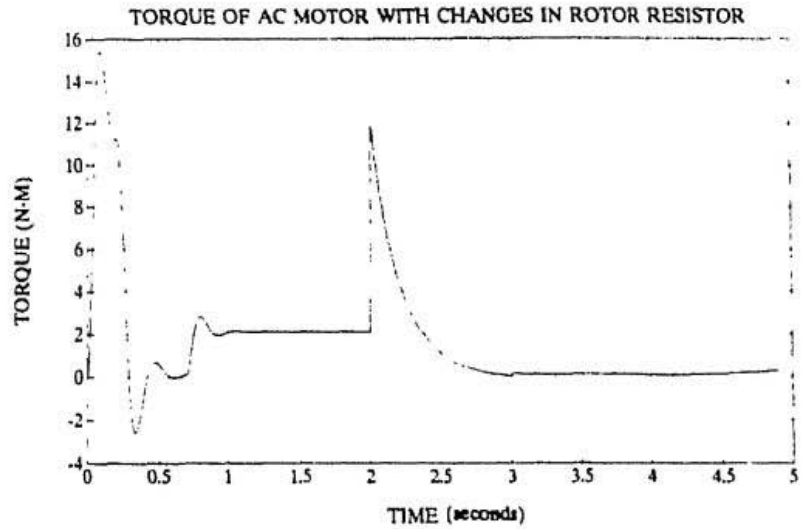


(a)

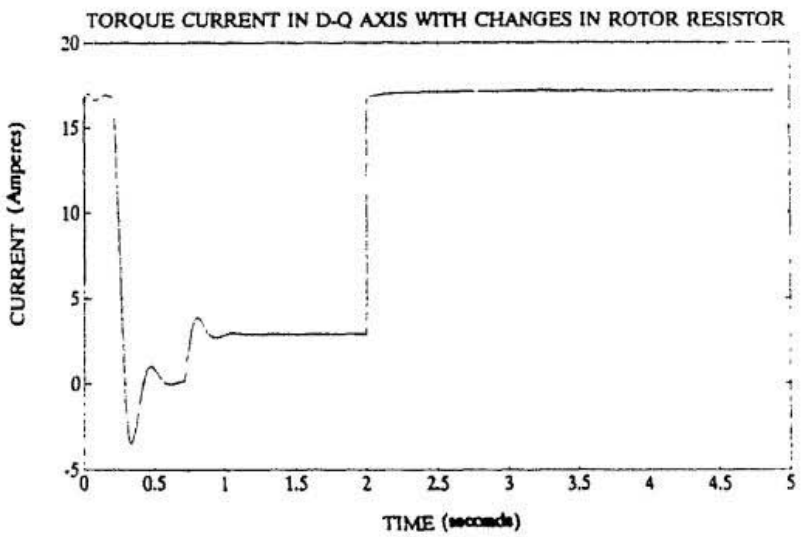


(b)

Figure 4.16: Simulation results by Chan [7], a) Rotor speed response, b) Estimated value of rotor resistance



(a)



(b)

Figure 4.17: Simulation results by Chan [7], a) Torque, b) Torque current

Chapter 5

Parallel Processing and Transputers

5.1 Introduction

The range of processor hardware to handle complex real-time problems of which the field-oriented control of AC motors is an example, can be categorized into the following groups [16] [15]:

1. *Single conventional microprocessor*: In its baseline form, this constitutes the memory and arithmetic unit of the control system. A high interface chip count, together with an external coprocessor or analogue processing, is usually necessary if fast real-time processing is required.

2. *Microcontroller*: Control-oriented processor in which common interface components (A/D, timer, interrupt controllers and pulsewidth-modulated outputs) are carried onboard chip, so reducing off-processor chip count.

3. *ASICs (application specific integrated circuits)*: An extension of microcontroller which minimizes off-processor chip count and onchip silicon for a specific application.

4. *Digital signal processor*: A high-speed development of (1), utilizing high clock rates and specific onchip architecture for fast multiply-shift-add operations.

5. *Concurrent or Parallel processing architectures*: Either a number of conventional processors or Transputers are used in an effort to obtain increased speed and/or flexibility of control processing.

However, as the requirements of high dynamic control, flexible architecture, self-tuning, fault tolerance and user interface in high performance AC motor control increase, an ASIC, microcontroller or signal processor based system is not suitable.

Parallel processing is a plausible alternative to meet these requirements.

5.2 Fundmental Principles of Parallel Processing [40]

Most computer applications, such as adaptive control, speech recognition and neural networks, are explicit models of systems in the real world. Any system that can be viewed as a collection of interconnected parts or events is said to possess concurrence.

Virtually, all known systems have some degree of concurrence because the motion in the world, by its very nature, is immensely parallel.

Events occur in both time and space. Usually multiple events within a system occur at the same time. Rarely do they happen individually in nice sequential order. Therefore, the most workable approach for accurately modelling real-world systems is parallel processing.

Parallel computers achieve their speed by dividing up the composite parts of a problem and working on them simultaneously as they occur.

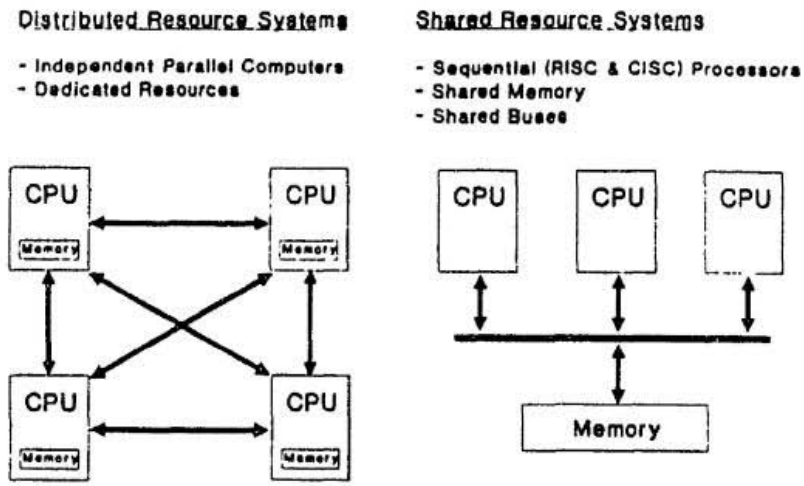


Figure 5.1: Distributed and shared resource systems [40]

Conventional processors, both RISC (Reduced Instruction Set Computer) and CISC (Complex Instruction Set Computer), are not parallel machines and consequently are not equipped to exploit the parallel nature of many applications. Instead, they are sequential architectures, one task must be completed before the next can begin. This means that the parts of a problem are handled one at a time, even if they actually occur together and are interrelated. When parallel systems are modelled with sequential computers, the impact of real-time interactions between events is lost. Artificial constraints and assumptions must be imposed to obtain approximations. The magnitude of this computing limitation increases as applications grow in size and complexity.

Parallel processing can be divided into two basic architectures: *Shared Resource* and *Distributed Resource* Systems, as shown in Fig. 5.1. Shared Resource Systems execute the components of a problem on conventional CPUs. They are connected

by a common bus (a channel over which data and programs can be moved) to shared memory. Concurrence is achieved by time sharing the memory. Sequential processors used in Shared Resource System have a Von Neumann architecture. Although processor performance has increased dramatically with improvements in circuit technologies over the last thirty years, the underlying design of all CISC and RISC devices has remained Von Neumann. It is the basis for virtually all of today's computer designs and software programs, a single instruction stream being sequentially decoded by one processing engine.

Conversely, a Distributed Resource System gains speed by partitioning the parallel parts of a problem among hardware nodes. Each node runs its own program and includes a CPU with local memory and facilities that support concurrence. Inherent drawbacks of Von Neumann machines severely constrain the parallel processing performance of Shared Resource Systems. Each machine operates sequentially by fetching an instruction from memory, reading an operand, executing the instruction, and storing the result. Delays first occur because of demands placed on system memory – a processor can only access this shared memory when no other processor is accessing it.

The performance improvement of Shared Resource Systems may stop at just three or four processors because memory access times begin to limit the speed at which a program can execute. It is only possible to speed up a Shared Resource System by using expansion memory that is very much faster than the CPUs. A Shared Resource System might attempt to overcome its memory access limitation by giving some dedicated memory to each processor, in addition to global memory. This strategy is not a permanent solution because the interconnection data

bus quickly becomes a fatal bottleneck. Conventional microprocessors, in order to communicate with each other, send out messages on a common interface bus. This bus has a finite bandwidth, which means that only a fixed number of transactions can take place each second. When one CPU is sending a message, arbitration is required and the other CPUs are added, the communications bus becomes saturated by simultaneous requests. Contention for the bus and increased capacitance degrade performance.

Shared Resource System may attempt to overcome bandwidth problems by adding buses. A cluster of processors, each with local memory, could share some *cluster memory* via one common bus. More buses could be used to interconnect clusters, perhaps to shared memory again.

The major problems of Van Neumann based architectures , communicating via a common resource, a bus or shared memory, have the following limitations [15] [16]:

1. A shared resource becomes a bottleneck limiting system expansion.
2. Expansion of the system is difficult due to complexity.
3. Bus arbitration logic is necessary, which increases complexity, cost and chip count.
4. Multiprocessor bus layout suffers from high track densities , capacitance and cost problems.
5. Software development is cumbersome and it is left to the programmer to establish the parallelism using languages inherently designed for sequential processors and programs.

5.3 Transputers T800 [39]

The Transputer suffers from none of the disadvantages listed in the previours subsection. It is a parallel microprocessor, generally categorized as a Multiple Instruction Multiple Data (MIMD), with four standard serial communication links. Transputers do not share a common bus, but instead exchange messages through their own high-speed serial links. Each link is a fast, asynchronous, full-duplex channel used to provide pairwise connection of Transputer nodes. These connections can be configured in a variety of topologies such as rings, arrays, and pipelines.

The bandwidth of a Transputer system rises linearly with the number of transputers added. Each of the four bidirectional link engines on the IMS T800 can transfer serial data to and from the transputer at a selected rate of 5, 10, or 20 Megabits/second. The total bandwidth of just one IMS T800 is 4×2.3 or 9.2 MBytes/second of real data. This data throughput is equivalent to 8-27 Ethernets.

Another key point about the links is that ongoing data communication between transputer nodes requires zero processor overhead. Each link has its own DMA controller so it can operate autonomously, in parallel with the other three links, CPU and FPU. This enables communication to take place simultaneously with computation. Transputers can be easily added as building blocks to transputer networks because their modularity and synchronized point-to-point communication create a unified system structure. This flexibility enables transputers to large multiprocessing networks. Some systems today use over 1000 Transputers.

The INMOS T800 transputer is a 32 bit CMOS microcomputer with a 64 bit floating point unit and graphics support, as shown in Fig. 5.2. It has 4 Kbytes on-chip RAM for high speed processing, a configurable memory interface and four

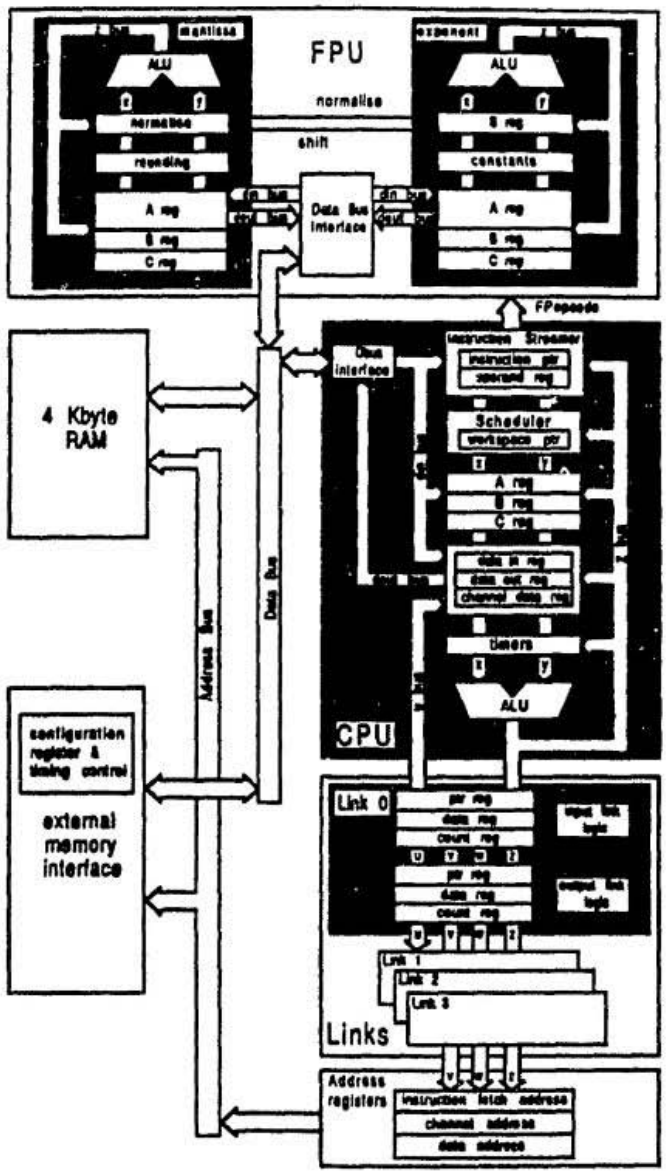


Figure 5.2: INMOS T800 architecture [39]

standard INMOS communication links. The instruction set achieves efficient implementation of high level languages network. Procedure calls, process switching and typical interrupt latency are sub-microsecond operations. The processor speed of a device can be pin-selected in stages from 17.5 MHz up to the maximum allowed for the part. A device running at 30 MHz achieves an instruction throughput of 30 MIPS peak and 15 MIPS sustained. The INMOS T800 provides high performance arithmetic and floating point operations. The 64 bit floating point unit provides single and double length operation to the ANSI-IEEE 754-1985 standard for floating point arithmetic. It is able to perform floating point operations concurrently with the processor, sustaining a rate of 2.2 MFLOPs at a processor speed of 20 MHz and 3.3 MFLOPs at 30 MHz. High performance graphics support is provided by microcoded block move instructions which operate at the speed of memory. The two-dimensional block move instructions provide for contiguous block moves as well as block copying of either non-zero bytes of data only or zero bytes only. Block move instructions can be used to provide graphics operations such as text manipulation, windowing, panning, scrolling and screen updating. Cyclic redundancy checking (CRC) instructions are available for use on arbitrary length serial data streams, to provide error detection where data integrity is critical. The INMOS T800 can directly access a linear address space of 4 Gbytes. The 32 bit wide memory interface uses multiplexed data and address lines and provides a data rate of up to 4 bytes every 100 nanoseconds (40 Mbytes/sec) under a 30 MHz clock. A configurable memory controller provides all timing, control and DRAM refresh signals for a wide variety of mixed memory systems. System service include processor reset and bootstrap control, together with facilities for error analysis. Error signals

may be daisy-chained in multi-transputer systems.

The standard INMOS communication links allow networks of transputer family products to be constructed by direct point to point connections with no external logic. The INMOS T800 links support the standard operating speed of 10 Mbit s/sec, but also operate at 5 or 20 Mbits/set. Each link can transfer data bidirectionally at up to 2.35 Mbytes/sec.

In recent years, Transputer-based real-time processing controller has been applied in many control application fields [14] [15] [16] [23] [32] [33]. The results show that the Transputer is a high speed processor which can be used for complex control in real-time requirement. In next chapter, Transputer-based parallel processing controller has been proposed for the requirements of high-performance control of AC induction motor drives and fault tolerance issues. The effectiveness of parallel processing on the AC induction motor control is examined.

It is suggested that the cost comparison between using conventional microcomputers and Transputers be performed after experiment.

Chapter 6

Parallel Processing Controller for the Indirect Field-Oriented Control of AC Induction Motor

6.1 Introduction

In Chapters 3 and 4, the design of field-oriented control for AC induction motor and the solution to the motor parameter sensitivity in field-oriented control due to temperature rising or flux saturation have been studied.

In this chapter, to achieve fast processing for field-oriented control, a parallel processing controller using one and four T800 Transputers has been proposed. Simulation has been carried out to investigate the effectiveness of parallel processing to implement the field-oriented control. Estimation times of control process in one Transputer and four Transputers are investigated. The effects of sampling time on control system performance are studied. Furthermore, some fault tolerance issues for Transputer failures or link failures are proposed and tested.

As a digital control of high performance AC induction motor using field-oriented control strategy requires controllers with significantly high computational power

and throughput, many AC induction motor control systems to date are controlled by multi-processors. Custom ASICs, microcontrollers and signal processors have also been used effectively in embedded systems. Some control systems, such as servo drives, often require additional features of self-tuning, fault tolerance, on-line diagnostics protection, data capturing, and user interface [16]. These additional requirements are difficult to be met by ASICs, microcontroller or signal processor based systems due to their inflexible hardware architectures, small memory space and inconvenient software development tools. Existing multiprocessor solutions based upon Von Neuman architecture, communicating via a common resource, a bus or shared memory, have many disadvantages as discussed in Chapter 5. The Transputer does not have any of these disadvantages and is proved to be the most suitable fast processor in real-time digital control [15][16][14][33].

To study the effectiveness of parallel processing on the field-oriented control systems, the parallel processing environment for real-time simulations is proposed. In this case, real-time response of the motor control system from the parallel processing simulation has advanced the digital control system development, performance studies, checkout and troubleshooting.

The approach that has been proposed is to connect several processors in a parallel arrangement and to provide a means of communication between the processors. The software is then partitioned over the several processors forming the parallel processing control. However, partitioning necessitates a careful and thorough consideration in the dynamic coupling relations within the control algorithms to determine the optimal breakdown of the system functions. In some case, inherent parallelism in the system may simplify the partitioning. The issue of how many

processors to use then can be addressed. For efficient operation, the portions of the simulation that are allocated to the individual processors should use approximately the same amount of compute time per processor. This will ensure correct updating of system variables and avoid wasted time in the calculation cycle. The updating of variables within the partitioned simulation will require not only careful timing consideration but also efficient data transfer between processors to avoid inadvertent phase shift.

6.2 Two Transputer Based Simulation of Field-Oriented Control System

To study the effectiveness of parallel processing on field-oriented control of AC motor, it is helpful to check the control algorithm execution times in single processor for comparison with multiprocessors.

In this section, two Transputer are employed to simulate the motor control system. one Transputer functions as the controller; another functions as the motor by executing its dynamic equations. Based on the experiment, various execution times of control algorithms, such as the PI controller, slip frequency calculation and the coordinate transformation, are examined in one Transputer. Comparisons in control algorithm execution times between conventional single processor and single Transputer are presented. The results show that the Transputer is a higher processing speed controller.

6.2.1 Hardware Implementation

In the simulation set up, an IBM-PC microcomputer (386 model) and two T800 Transputers are used to form the simulation system. The IBM-PC microcomputer is connected with the two Transputers, and functions as a compiler.

In the simulation, the executable files compiled and linked by the IBM-PC microcomputer were loaded down into each Transputer. Transputer with loaded executable files worked individually unless communications were required between Transputers. The Transputer communication works such that when a processor performs input or output to a communication channel, the processor is blocked until the corresponding processor performs its respective output or input. Channels can be used as a synchronization mechanism in addition to a communication mechanism.

Fig. 6.1 shows the hardware structure for the simulation of control system. In Transputer 1, the motor speed w_r was *sampled* from Transputer 2, which represents the AC induction motor, through channel 1. The motor $\alpha - \beta$ current I_α and I_β are *sampled* from Transputer 2 through channel 2. Transputer 2 works like a real motor, if the dynamics computation time is less than the required current sampling time. In this way, at each dynamics computation loop, Transputer 2 calculates all the motor state variables and sends them out to Transputer 1 for closed-loop control.

In Fig. 6.2, it shows that the two parallel processors can simulate the real-time motor control system. Fig. 6.2(a) indicates a real motor control system in which the dynamic processes of AC motor can be emulated by a Transputer as shown in Fig. 6.2(b). The Transputer computation time for motor dynamics is obviously

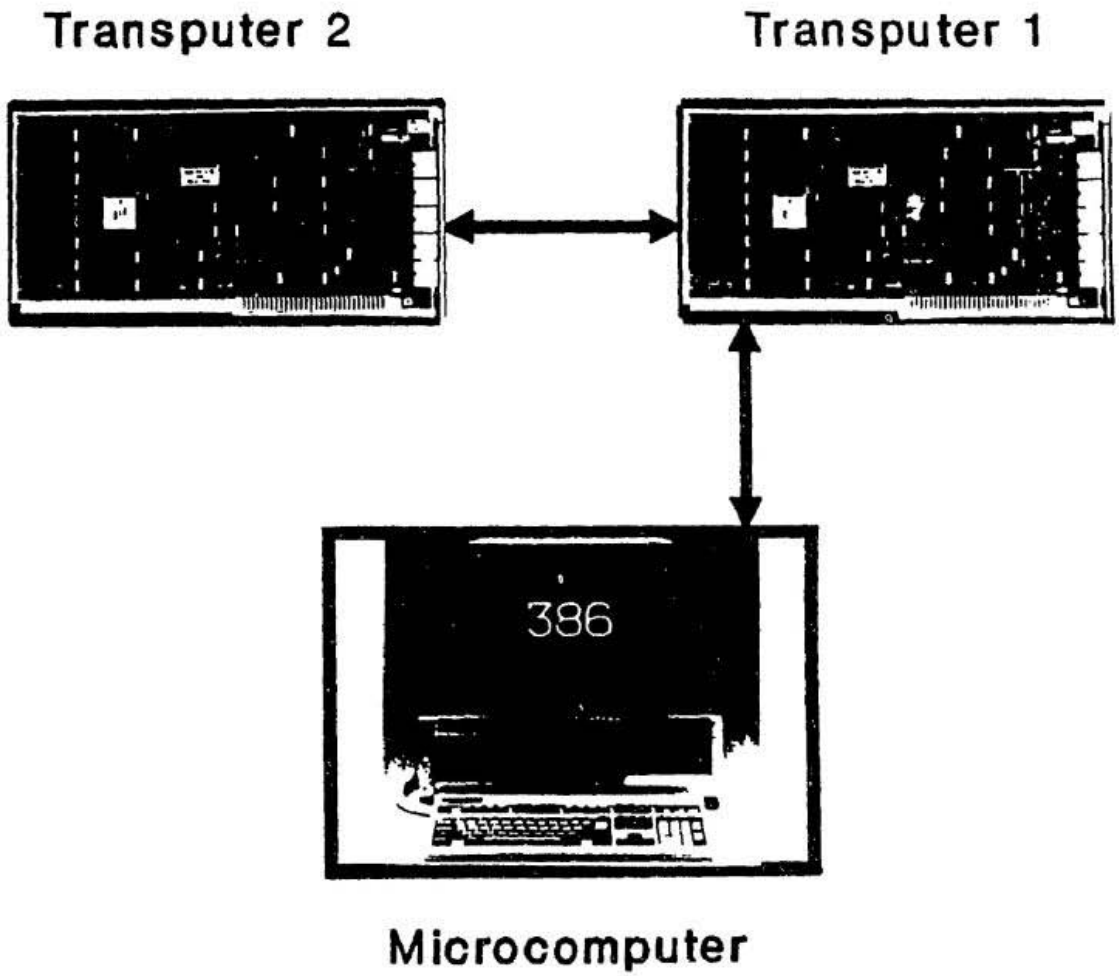
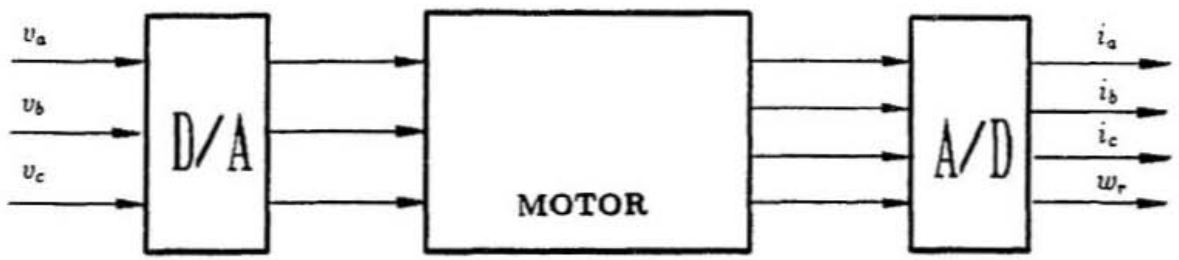
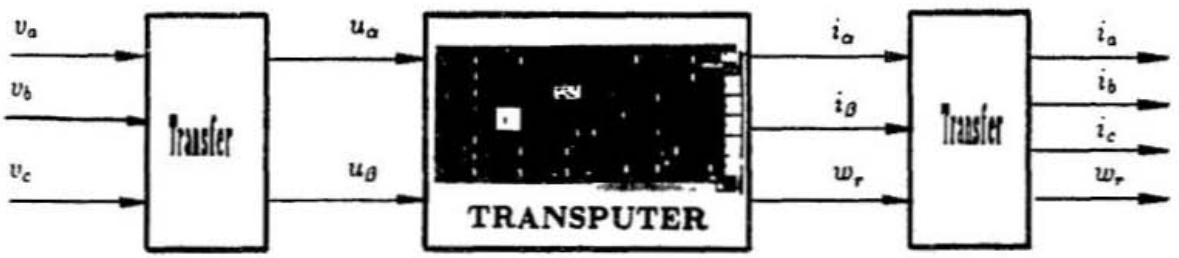


Figure 6.1: Parallel processing hardware configuration for real-time simulation of AC motor control



(a)



(b)

Figure 6.2: Transputer based real-time emulating of indirect field-oriented control of AC induction motor. a) Real motor control, b) Real-time emulation of motor control

longer than the real motor response time, but it is acceptable if it is equal to 10-20 times of the motor stator time constant [16].

6.2.2 Software Design

The simulation software flow chart is illustrated in Fig. 6.3 in which the programs consist of three subroutines: set-up routine, process control routine and motor dynamics computation routine. In Transputer 1, the current I_α and I_β are sampled from Transputer 2, and then the I_α and I_β are transformed to I_q and I_d in $d-q$ coordinate. According to the values of I_q and I_d, the current controller algorithms as expressed in equation (6.1) are computed. The calculation results of current controller are then transformed back to U_α and U_β in α-β coordinate according to equation (3.24). These two variables are then sent to the Transputer 2 as the required stator voltages. The PWM calculation is not included here as it is beyond the scope of this research topic. The control algorithm program and motor dynamics program are listed in Appendix C-1 and Appendix C-2, respectively. The parameters of motor are the same as in Appendix A-2.

Set-Up Routine

This routine enables Transputers to get commands from the IBM-PC microcomputer and to set up the motor speed reference, computing step size, load setting time, adaptive control tuning time and the starting time for rotor resistance change. All the control system state variables are reset to the initial values.

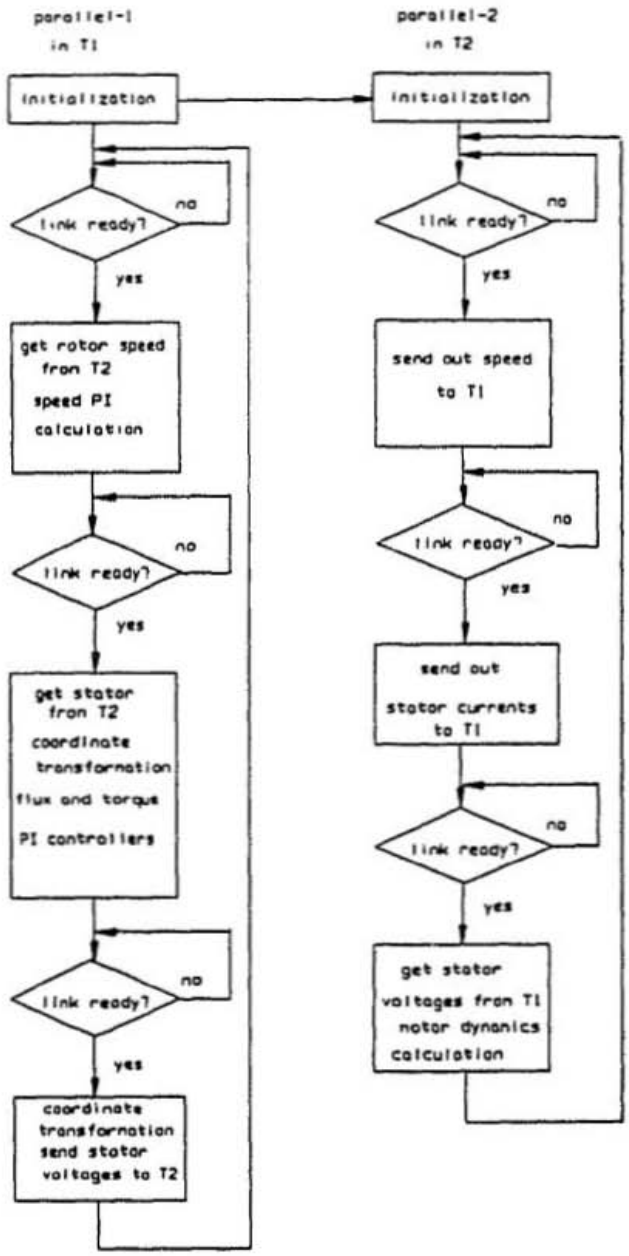


Figure 6.3: Software structure for simulation of AC motor control system

Speed Controller and Slip Frequency Calculation Routine

The purpose of this routine is to calculate the speed PI control algorithm, current limitation and slip frequency. The difference equation of the PI controller in the increment approximation of equation (3.29) can be expressed as:

$$I_q(k) = K_p * (e(k) - e(k-1)) + K_p * K_i * e(k) + I_q(k-1), \tag{6.1}$$

where:

1. $K_i = \frac{\tau}{T_0}$;
2. T_0 is the sampling time;
3. τ is the integration time constant.

The PI control algorithm is computed in Transputer 1 when rotor speed w_r is sampled. The output of PI controller is compared with the current limitation value and goes through the limitation function which produces the final PI controller output. Following the calculation of i_q, the calculations of slip frequency and angle are handled. The difference equations of slip frequency and angle equation are written as:

$$w_{sl}(k) = \frac{I_q(k) * R_r}{L_r * I_d(k)} \tag{6.2}$$

$$\theta(k) = (w_{sl}(k) + w_r(k) + w_{sl}(k-1) + w_r(k-1)) * T_0/2 + \theta_{sl}(k-1). \tag{6.3}$$

The calculated slip frequency is added to the rotor frequency as the required motor stator frequency for field-oriented control. The calculated slip angle is added to the rotor angle for field orientation coordinate transformation.

Current PI Controller and Coordinate Transformation Calculation Routine

The difference equations of flux current controller and torque current controller from equation (3.28) are written as:

$$U_d(k) = K_d * (e(k) - e(k-1)) + K_d * K_1 * e(k) + U_d(k-1) \tag{6.4}$$

$$U_q(k) = K_q * (e(k) - e(k-1)) + K_q * K_2 * e(k) + U_q(k-1), \tag{6.5}$$

where:

1. $K_1 = \frac{T_1}{T_0}$;
2. $K_2 = \frac{T_2}{T_0}$;
3. T_0 is the sampling time.

The coordinate transformation for changing I_α and I_β to I_d and I_q is expressed in equation (3.27). The coordinate transformation for changing U_d and U_q to U_α and U_β is expressed in equation (3.24).

Flux Orientation Feedback Control

The actual and reference motor torques are calculated according to the motor terminal variables, such as the stator voltages, stator current and motor rotor speed. The torque error results from the difference between actual and reference torque is fed back to the field orientation controller that adjusts the slip frequency.

Motor Dynamics Calculation

The difference equations for motor dynamics equations (3.16) - (3.23) are expressed in software program (see Appendix C-2). To obtain a high accuracy of motor dynamics calculation, the predictor-corrector algorithm is used for solving the first order differential equations. The simulation shows that the computing step size determines the accuracy of motor dynamics calculation. With a proper choice in computing algorithms, the computing step size could be increased without losing computation accuracy. This helps the simulation not to take so much CPU time.

6.2.3 Execution Times

To measure different calculation times for different control algorithms and channel communications, the T800 Transputer internal high privilege timer with 1μs resolution is used. The execution times are measured by using Transputer time measuring instructions. The measured execution times are listed in table 6.1. In order to make comparison with conventional 16 or 32 bit microcomputers, the typical control algorithm execution times in 16 or 32 bit microcomputers are given in Table 6.2 [22]. It is obvious from comparing table 6.1 and table 6.2 that the processing speed by Transputer is much faster than that of conventional microcomputers.

If the field-oriented control system incorporates flux orientation feedback control, the overall control system execution time is found to be 260 μs. It can be seen that, even though the Transputer has very fast processing speed, one Transputer controller is still not fast enough to meet the requirement in high dynamic servo control system in which 100 μs current sampling time is needed [16].

Table . Single Transputer Execution Times
(Time in Microseconds)

Algorithm	Execution Time
Addition	0.35
Substraction	0.35
Multiplication	1
Division	1.65
One Link Communiation	5.6
Coordinate Transformation	5.5
Triangle Function Computing	45
Speed Controller	15.5
Current Controller	9.5
Field-Oriented Control	161.5
Field-Oriented Control With Field Orientation feedback Control	230

Table 6.1: Execution times in single Transputer

Operation	P-Code Instructions	Execution time in microseconds					
		TDC316	LSI-11	PDP 11/40	TI9900	Z8000	8086
Addition	ADI T_1	5.2	7.7	3.4	10.9	5.7	4.6
Multiplication	LDCI	5.2	4.6	3.3	16.9	3.0	2.8
	LDOI	6.6	9.4	3.9	15.4	3.5	5.4
	MPI	22.6	74.8	14.3	99.2	21.5	34.2
	T_2	34.4	90.8	21.5	131.5	28.0	42.4
Update	LDOI	6.6	9.4	3.9	15.3	3.5	5.4
	SRO I	6.6	8.8	4.0	13.6	4.0	5.6
	T_3	13.2	18.2	7.9	27.9	7.5	11.0
Procedure Call	BNT	6.2	8.4	3.3	12.0	2.5	1.2
	RET	17.4	27.3	9.4	29.6	6.2	9.8
	T_4	23.6	35.7	12.7	41.6	8.7	11.0

Table 6.2: Execution times in conventional single processor [22]

6.3 Multi-Transputer Based Simulation of Field-Oriented Control System

One Transputer controller is not suitable in servicing very fast dynamic servo control systems. Owing to their speed of computing and flexibility for real-time digital control, parallel processing systems have made implementation of complex control algorithms feasible for critical time requirement. The fast processing speed and hardware flexibility provided by multi-transputer also make it possible for system fault tolerance.

6.3.1 Hardware Implementation

To simulate the field-oriented control with incorporation of fault tolerance in a parallel processing environment, five Transputers named Transputer 1, Transputer 2, Transputer 3, Transputer 4 and Transputer 5 have been employed. This parallel processing hardware architecture is shown in Fig. 6.4. Each Transputer has four communication links named link 1, link 2, link 3 and link 4. Each link is a bidirection transmission line with $20Mbit/sec$. The IBM-PC (model 386) microcomputer compiles and links all C-coded parallel programs and loads them down into each Transputer for concurrent processing. Results are reported back to the IBM-PC microcomputer from each Transputer for display or storage.

Transputer 1, connected to Transputer 2, Transputer 4, Transputer 5 and IBM-PC microcomputer through Channel 0, Channel 1, Channel 4, Channel 5, respectively as shown in Fig. 6.4, performs two functions: (1) it works as an interactive interface which receives any information from other Transputers and sends it to the IBM-PC microcomputer, or transfers any command from IBM-PC microcomputer

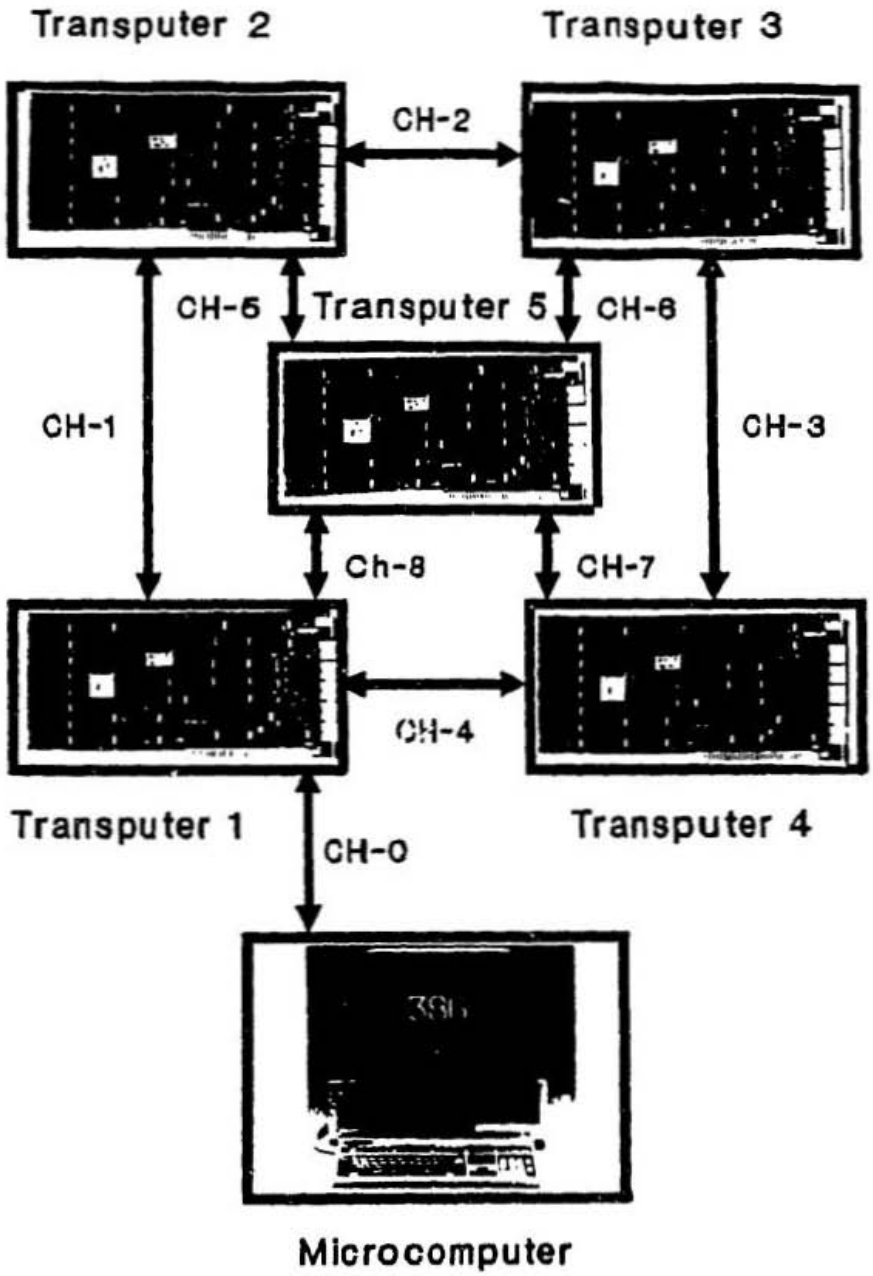


Figure 6.4: Five Transputer based parallel processing architecture for real-time simulation of AC motor control

to other Transputers; (2) it also incorporates system fault tolerance by acting as a redundant processor which will make the parallel processing architecture reconfigurable. This provides the possibility to improve the system reliability. Transputer 2 is connected to Transputer 1, Transputer 3, Transputer 4 and Transputer 5. It handles the motor speed controller, slip calculation and sine triangle function of slip angle calculation. The calculated results in Transputer 2 are sent to Transputer 3 for computing the current PI control algorithm. Transputer 3 is connected to Transputer 2, Transputer 4 and Transputer 5. The flux and torque current PI control algorithms, sampled I_α and I_β, and coordinate transformation are handled in Transputer 3. The control outputs U_α and U_β are sent to the motor which is represented by Transputer 5. Transputer 4 is connected to Transputer 1, Transputer 3 and Transputer 5. It performs the flux orientation feedback control. The feedback control will adjust the slip frequency so that the field-oriented control condition as indicated in equation (3.1) is always satisfied even when the motor rotor resistance is changed with rising temperature or flux saturation. The actual and reference torques are calculated in Transputer 4 at first. Any errors resulting from comparing actual and reference torque will result in correcting the flux orientation errors. Transputer 5 is connected to Transputer 1, Transputer 2, Transputer 3 and Transputer 4. It performs the dynamics calculation of the motor, as expressed in equations (3.16) - (3.23). The calculated results of motor speed, flux and torque currents, stator currents and voltages are sent to other Transputers for control purposes.

6.3.2 Parallel Processing Software Design

Proper design of parallel processing software is important to achieve minimum execution time. Based on the inherent parallelism in motor control, the control algorithm programs are assigned to Transputer 2, Transputer 3 and Transputer 4 such that the total execution time expected is less than 100 μs which can meet very fast dynamic servo control requirements. The designed parallel processing program flow-chart for real-time motor control simulation is illustrated in Fig. 6.5. The programs are coded in C and operated in Transputers featuring 64-bit floating point operations. The C codes for these programs are listed in Appendices D-1 to D-5. The parameters of motor are the same as in Appendix A-2.

As shown in Fig. 6.5, the parallel processing programs consist of set-up routine and five parallel processing routines run in five Transputers. These five parallel processing programs are named as parallel-1, parallel-2, parallel-3, parallel-4 and parallel-5. All sub-programs are concurrently executed in the corresponding Transputers.

Set-Up Routine

The function of the set-up routine is to initialize each Transputer. Commands from host computer (IBM-PC) are passed to each individual Transputer for setting motor speed reference, flux current reference, timing for loading and field orientation feedback control tuning. All the control and motor state variables are initialized at this time. The computing step size and the starting time for rotor resistance variation are also defined here.

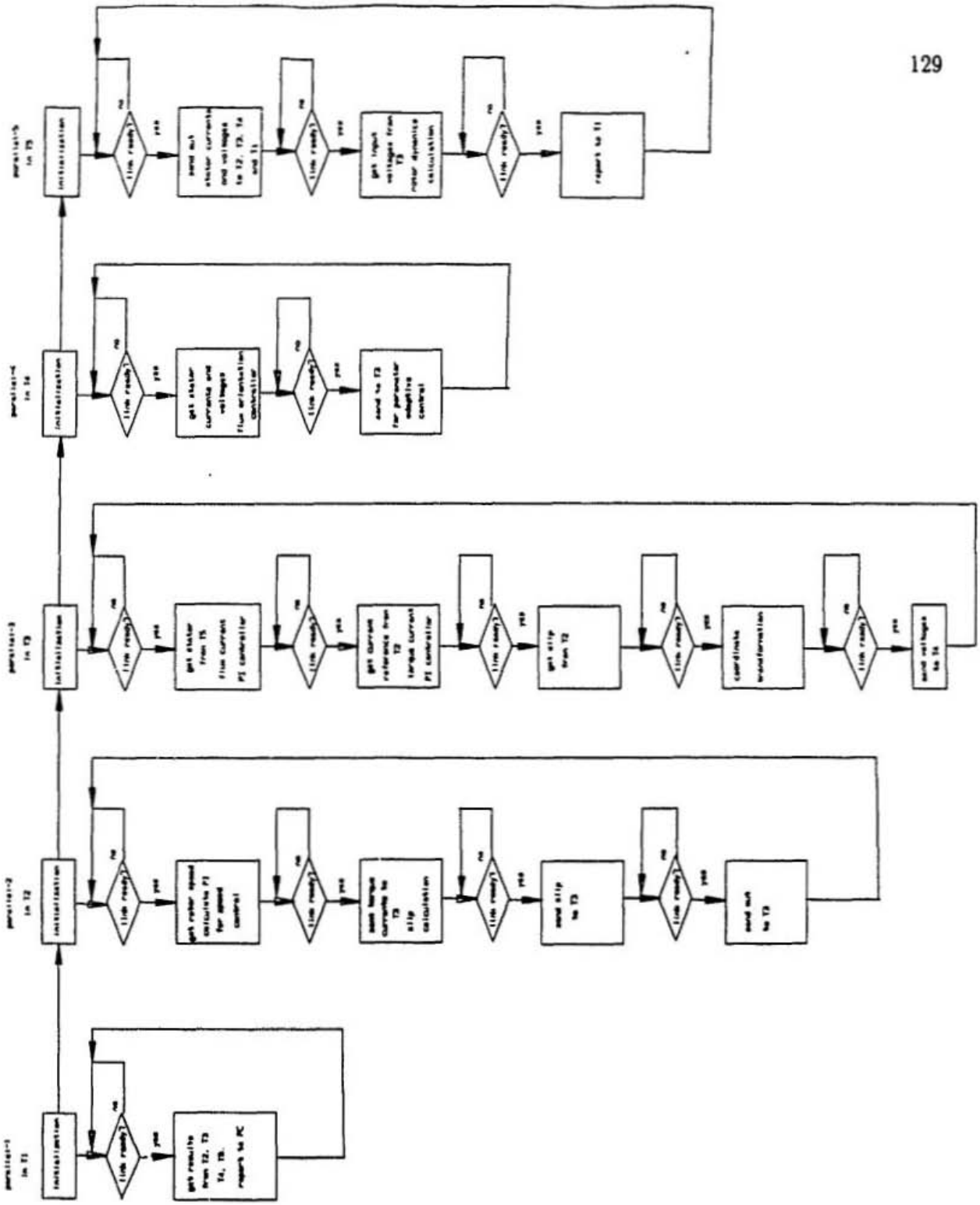


Figure 6.5: Parallel processing program structure for simulation of AC motor control

Parallel-1 Routine

The main purpose of this routine run in Transputer 1 is to work as an interactive interface which passes host computer commands to other Transputers and receives calculated results and status information from any of the other Transputers and then sends them to host computer. This program is listed in Appendix D-1.

Parallel-2 Routine

The listing for this routine is given in Appendix D-2. This routine run in Transputer 2 performs the speed PI controller, torque current limitation and slip frequency calculation. The sine triangle function of slip angle is also included in this routine. The calculated speed PI controller output, slip frequency and sine function are sent to Transputer 3 through Channel 2 as shown in Fig. 6.4.

Parallel-3 Routine

This routine run in Transputer 3 handles the flux and torque current PI control. In addition, it calculates coordinate transformations of α-β to d-q and d-q to α-β as well as the cosine triangle function of slip angle as shown in equations (3.26) and (3.27). The processing speed of this routine plays a critical role in determining the overall control system sampling time. According to data dependency in the field-oriented control, this routine is carefully decomposed from system control algorithms. It is expected that the Transputer 2 takes tasks as much as possible so that the Transputer 3 works as little as possible to achieve a fast control of stator currents. In the simulation, it shows that the triangle function takes most of the processing time. The listing of this routine is given in Appendix D-3.

Parallel-4 Routine

This routine run in Transputer 4 performs the flux orientation feedback control. The calculated control signal is sent to Transputer 3 for correcting the slip frequency. The listing of this routine is given in Appendix D-4.

Parallel-5 Routine

This program is run in Transputer 5. It simulates the motor dynamic behaviour. For each computing loop, it sends out all the necessary motor state variables to other Transputers for the control purposes. In each computing loop, Transputer 5 responds nearly as fast as a real motor does. This means that for each computing loop in Transputer 5 the other Transputer should have already prepared to sample the motor state variables, such as the motor speed, stator current and voltages. Otherwise, the processing speed may be not high enough to match such fast dynamic motor control. The listing of the motor dynamic program is given in Appendix D-5.

6.3.3 Execution Times

Execution times are measured using the $1\mu s$ timer provided in each Transputer board. The execution time at Transputer 3 which performs the inner loop current control gives the minimum bound for the sampling time. To explain the methodology of multi-Transputer processing time measurement, Fig. 6.6 shows the task dependent relations among processes on different Transputers in which the minimum execution time can be easily calculated. P1, P2 and P3 represent different control processes. In Transputer 2, P1 is the motor speed controller, slip frequency calculation and communication to motor. P2 is the sine function computation and

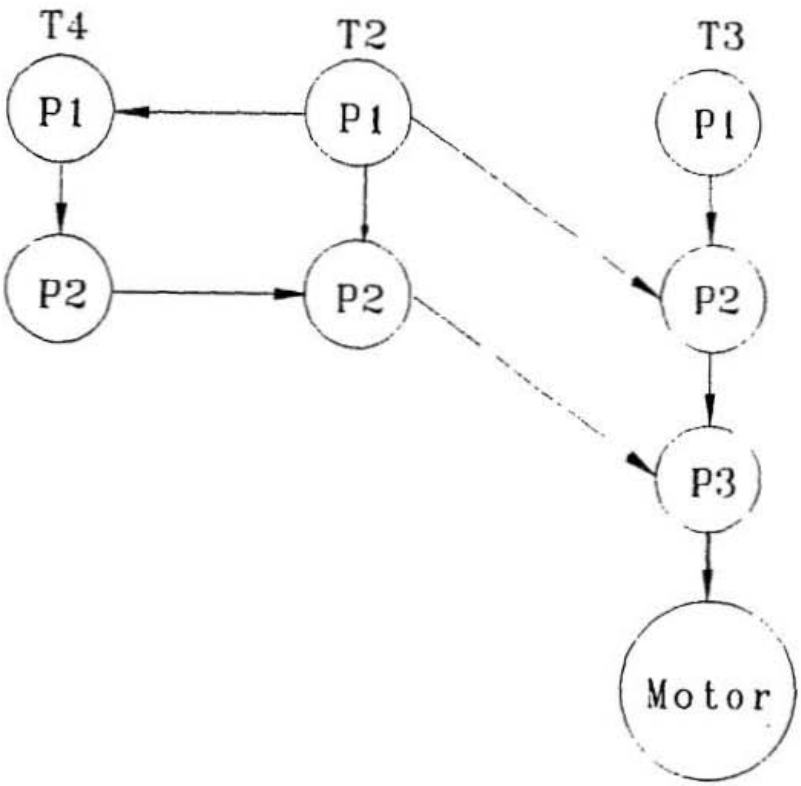


Figure 6.6: Control task dependency in AC motor control

(Time in Microseconds)

Processes	T2	T3	T4	
P1	34.5	17	27.5	Excution Times
P2	50	24.5	37.5	
P3		50.5		
Total	84.5	92	65	

Table 6.3: Execution times in five transputers

communications to Transputer 3. In Transputer 3, P1 represents the current sampling, the α-β to d-q transformation, flux current and torque current controller. P2 is the cosine function calculation. P3 is the coordinate transformation of d-q to α-β for motor stator voltages. In Transputer 4, P1 and P2 represent the flux orientation feedback controller. In Fig. 6.6, The lines between processes indicate the data dependency. For example, The process P2 in Transputer 3 can not start until the precess P1 in Transputer 2 has been completed. The Table 6.3 illustrates the execution times for each process; the minimum processing time for the indirect field-oriented control is shown as 92μs. In order to compare to other multiprocessor based parallel processing system, Table 6.4 presents the execution times in different parallel processors. Compared with single Transputer and other multiprocessors, the proposed multi-Transputer controller system offers the fastest processing speed (92μs).

Controller Type	Flexibility	Software Development	Parameter Adaptation Control	Sampling Time
Intel-8086 + signal processor NEC77720 [6]	No	Cumbersome	No	125
Inter-8086 + two Intel-8031 [17]	Yes	Cumbersome	No	2200
Four Inter-8086 [9]	Yes	Cumbersome	No	700
Three Transputers (T400) [7]	Yes	Easy	No	250
Five Transputer (T800)	Yes	Easy	Yes	92 or 83

Table 6.4: Execution times in multiprocessors (microseconds)

6.4 Fault Tolerance

Another major advantage of parallel processing that can explicitly improve the reliability for real-time control systems is the possibility of incoporating the fault tolerance. The fault-tolerant control system has the ability to continue performing its tasks after the occurrence of faults. The most common technique used to achieve fault tolerance is design redundancy, which is the addition of hardware, resources, or time in excess of that needed for normal system operation [37]. Hardware redundancy requires two or more processors which offers a very high processing speed but requires a high cost. Time redundancy needs no additional hardware processors that are required in hardware redundancy, but it takes extra time for fault detection, avoidance and isolation. Considering the advantages of both hardware and time redundancy, a hybrid fault-tolerant architecture using Transputers for field-oriented control is proposed. The proposed architecture is fully based on the parallel processing controller as indicated in Fig. 6.4. It takes advantages of Transputer's fast processing speed, which gives possibility for time redundancy, and flexible architecture, which offers the ability for reconfiguring the communication links and processors when faults occur. According to the Transputer's high processing speed, the execution time in one sampling is divided into two parts: one is for control algorithm computation; the other is for the purpose of fault tolerance.

Based on above concepts, the link failures and processor failures are studied in the following subsections.

6.4.1 Link Failure

Most frequently, failures in a multi-Transputer system can be attributed to failures in communication links. For example, broken links, unreliable connections between Transputers and loosen pin contacts will result in a communication failure. For overcoming these problems, Transputer instruction sets provide special link communication instructions which use the *time-out* concept to improve communication reliability. Having examined Transputer-based parallel processors for field-oriented control, it is clear that the Transputer 2 and Transputer 3 play a very important role in achieving high performance real-time motor control.

Fig. 6.7 shows the principle of reconfiguration of Transputer links for fault tolerance concerns where the dotted lines represent the reconfigured communication channel path. The four communication links in each Transputer make the dynamic reconfiguration possible. The strategy in the motor control system is that the system should survive any link failure between Transputer 2 and Transputer 3 as well as the links to Transputer 5 which represents the motor. As shown in Fig. 6.7(a), Channel 2 formed by link 1 of Transputer 2 and link 0 of Transputer 3 will be taken over by Channel 9 formed by link 3 of Transputer 2 and link 3 of Transputer 3, if Channel 2 failed. The principle is that the link failure would result in communication *time-out* causing the Transputer to reconfigure its communication through other channels. The failed Channel 5 will be taken over by Channel 1 and Channel 8 as the dotted line shown in Fig. 6.7(b). Similarly, the failed Channel 6 will be serviced by Channel 3 and Channel 7 as shown in Fig. 6.7(c). By implementing this scheme, any single link failure among the Channel 2, Channel 5 and Channel 6 is isolated and recovered by other links.

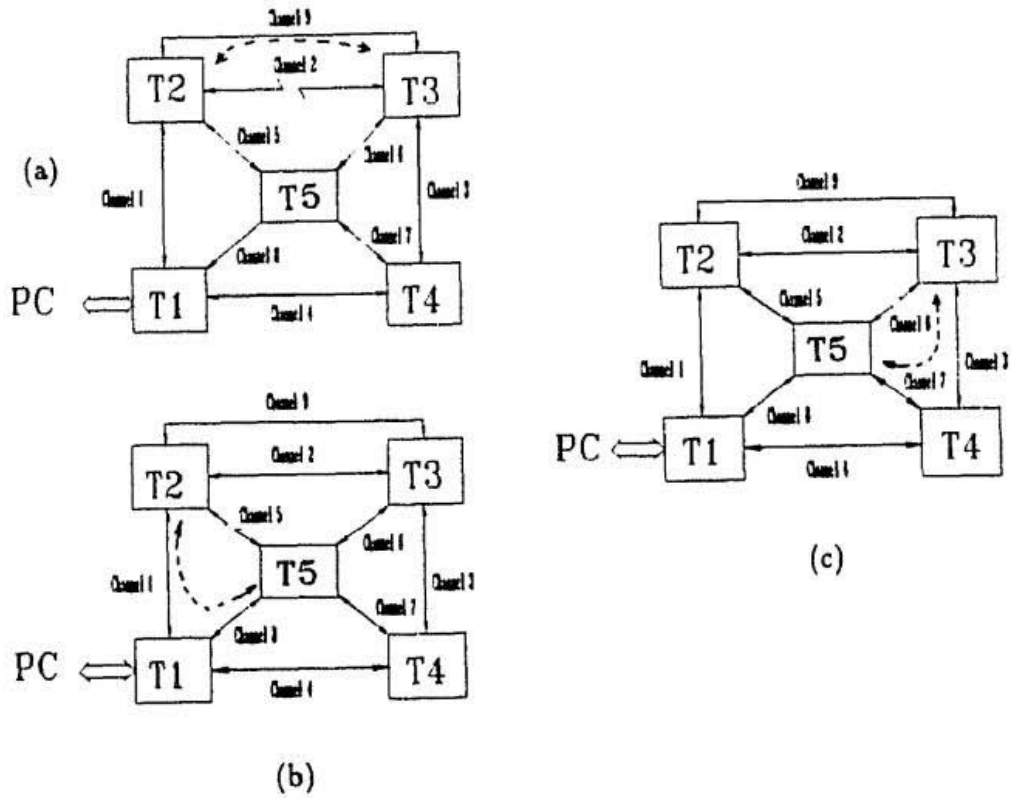


Figure 6.7: Reconfigurable communication channels in Transputer network

Another significant feature of this parallel architecture is its multi-fault tolerance which allows Channel 2, Channel 5 and Channel 6 to fail at the same time. This case is shown in Fig. 6.8 in which the Channel 2, Channel 5 and Channel 6 are replaced by Channel 1, Channel 8, Channel 3, Channel 7 and Channel 9, respectively, as shown in the dotted lines. Obviously, the communication reconfiguration in Transputer is superior to that of bus-shared parallel processor architecture. In the latter, the failure in the common shared bus will result in a total system communication failure.

It is noted that, for the rest of communication link failures, the strategy is to give up any communication among failed links instead of trying to reconfigure existing channels. The failed links with Transputer 4 results in no correction for slip calculation when motor resistance changes due to temperature. The failed links with Transputer 1 will shut down the communication between working Transputers and IBM-PC microcomputer, but motor control is kept going by Transputer 2 and Transputer 3.

System Testing

The fault-tolerant controller for the indirect field-oriented control of AC motor is tested by disconnecting any one of Channel 2, Channel 5 or Channel 6 during the simulation. The results show that the communication reliability is improved by dynamic channel reconfiguration. It should be noted that reliable communication instructions do take more time than general communication instructions. Using the reliable communication instructions, processing time in each Transputer is increased. The increases in execution time depend on how many times of the

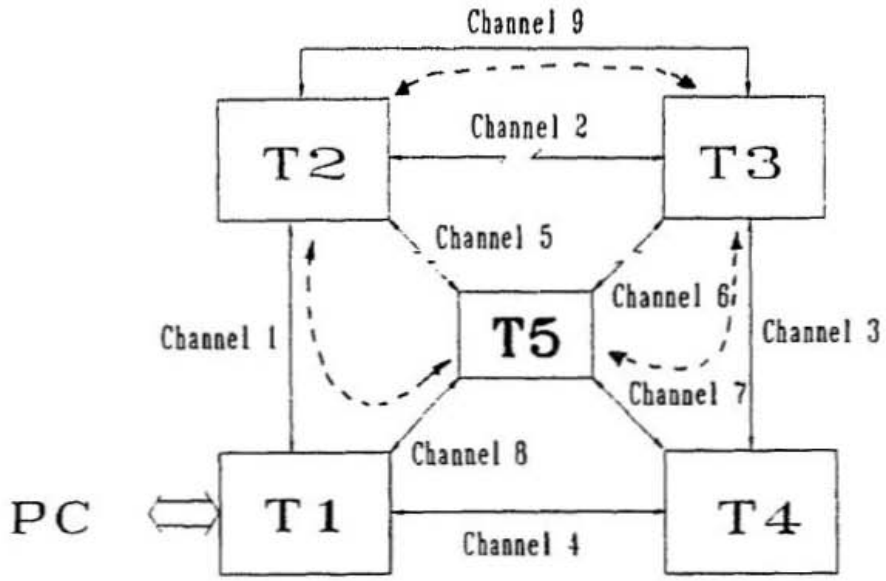


Figure 6.8: Recovery of multi-communication channel failures

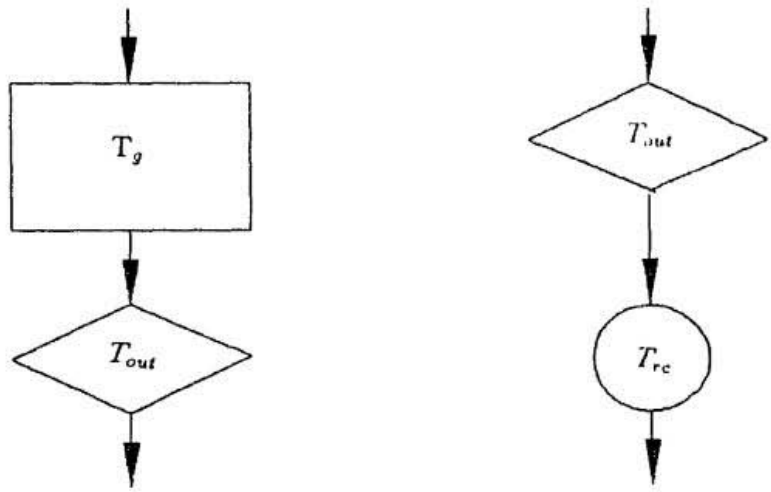


Figure 6.9: Reliable communication times

reliable communication are needed in one sampling period. In Transputer T800, one general communication time between two Transputers is 5.5μs. One reliable communication time between two Transputers takes about 10-15 μs. The communication *time out* is chosen as 8μs which should be equal to or longer than one general communication time. The following formula gives the calculation of one reliable communication time.

$$\textbf{Reliable Comm. Time} = \textbf{Gen. Comm. Time} + \textbf{Time Out} .$$

Let us take the process in Transputer 3 as an example to explain how much extra times is taken by reliable communication. In Parallel-3 routine, there are four times of Channel 6 communication, as shown in Appendix D-3. Fig. 6.9 shows one Channel 6 communication calculation. Fig. 6.9(a) indicates the normal communication case without link failure. The square represents the normal communication time. The diamond represents time-out communication. The communication time

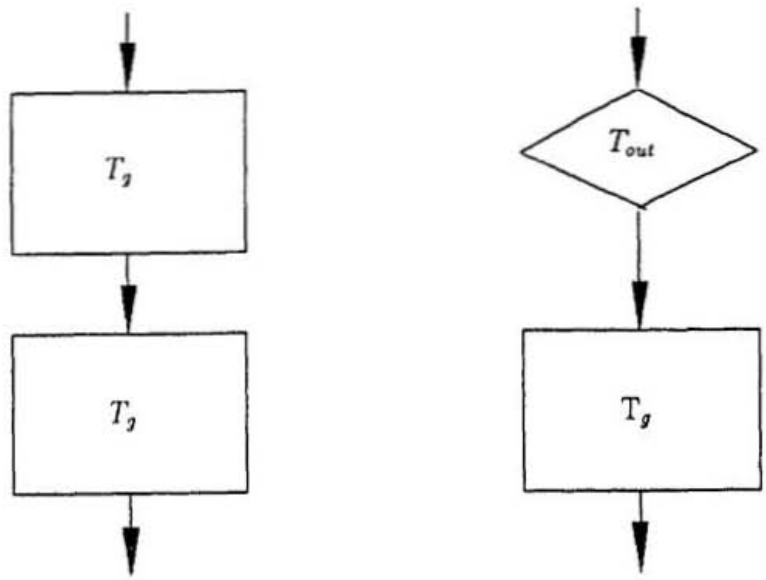


Figure 6.10: Communication time with auxiliary channel

of Channel 6 without failure is totally 15 μs. Fig. 6.9(b) indicates the communication with Channel 6 failure. The circle represents the communication time for channel reconfiguration. So the communication time when Channel 6 is failed is about 19μs. For four Channel 6 failure communications, the total communication time for channel reconfiguration is 76 μs. Fig. 6.10 shows one Channel 2 communication time calculation. There are four Channel 2 communications in Parallel-3. For normal communication, the four Channel 2 communications take 44 μs. For Channel 2 failure, the total reconfigured channel communication time is 54 μs which is different from the results shown in Fig. 6.9 because the auxiliary Channel 9 is used. It is clear that the improved communication reliability is a benefit, but the processing speed slows down by channel reconfiguration and channel time out.

6.4.2 Transputer Failure

To improve the reliability in presence of a processor failure, the proposed software is illustrated in Fig. 6.11. Before control algorithm programs start, the diagnosis programs in each Transputer (for fault detection, voting, and isolation) are executed. Methods to achieve fault tolerance in presence of different Transputer failures are shown in Figures 6.12, 6.13, 6.14, 6.15, and 6.16.

The diagnosis results are reported to the Transputer which performs the task of a voter; it determines which processor is failed and marks the failed one and then sends its decision to each working Transputer. The failed Transputer would not get information from the voter and is isolated as the other Transputers reconfigure their communication channels according to voter's decision.

Fig. 6.12 shows the case when Transputer 2 is failed. T1, T2 and T3 execute diagnosis programs and then send their results to the fourth Transputer T5, a voter. In T5, the three inputs from T1, T2 and T3 are compared with each other. If there is any difference (only a single fault is assumed), the failed one (Transputer 2) can be allocated and isolated. The function of the failed Transputer will be taken over by the voter, Transputer 5. Fig. 6.12 also shows the reconfigured software programs for the failure of Transputer 2. Any one of the Transputers is possible to be a voter or a substitute. The principle of fault tolerance illustrated in Figures 6.13, 6.14, 6.15 and 6.16 is similar to that of Fig. 6.12.

It is noted that, in order to improve the voter reliability, at least three voters are available for detecting the same faulty processor, if time is allowed. For example, if the T2 is assumed to be failed, the three possible voters, namely T5, T3, and T1, can be applied. These three voters also can be compared with each other before

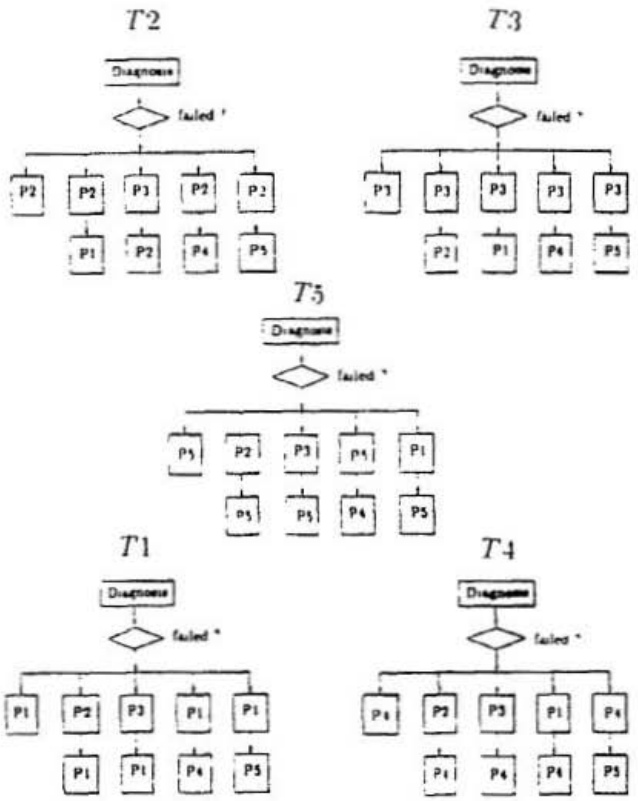


Figure 6.11: Parallel processing program structure incorporated with processor fault tolerance

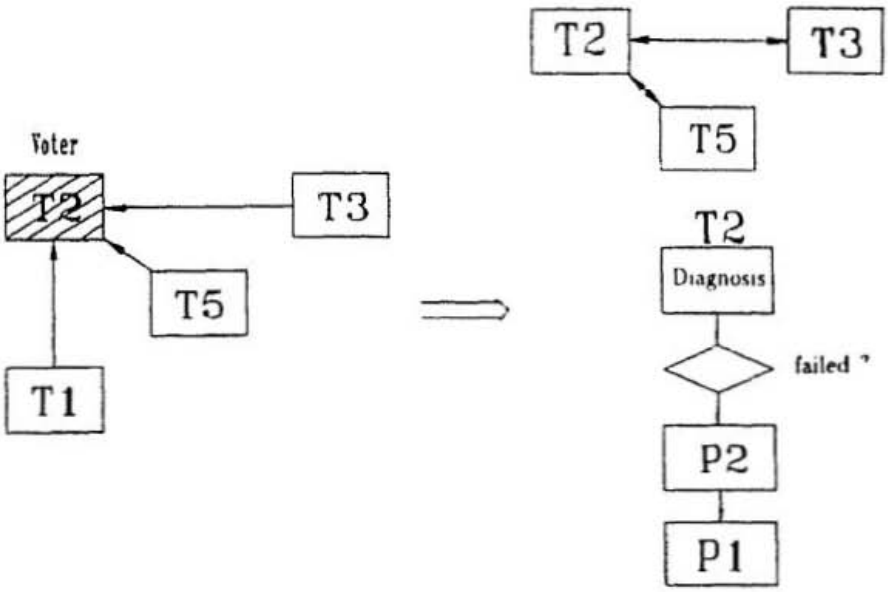


Figure 6.12: Transputer 1 failure

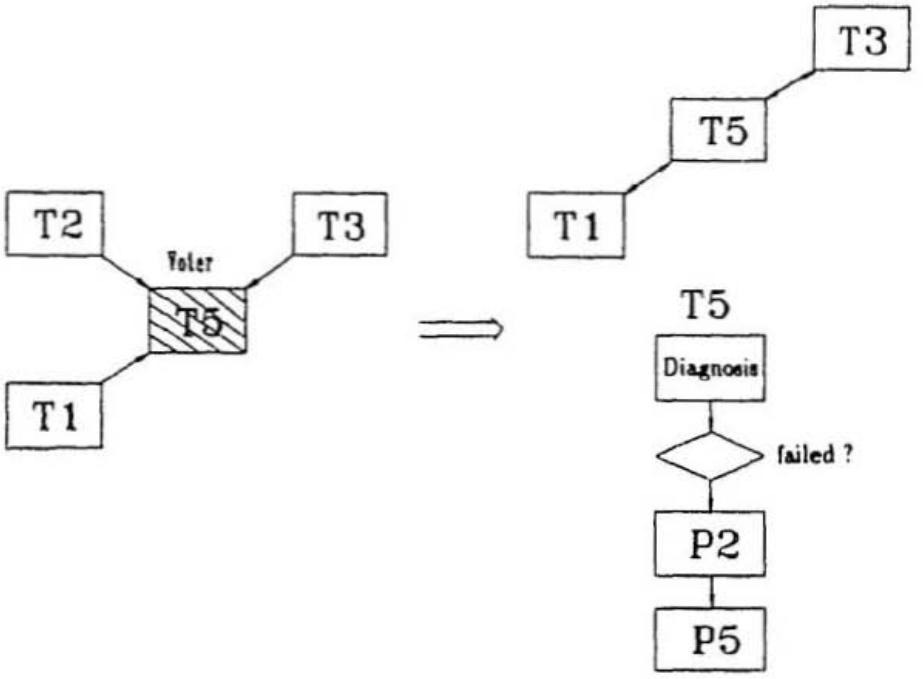


Figure 6.13: Transputer 2 failure

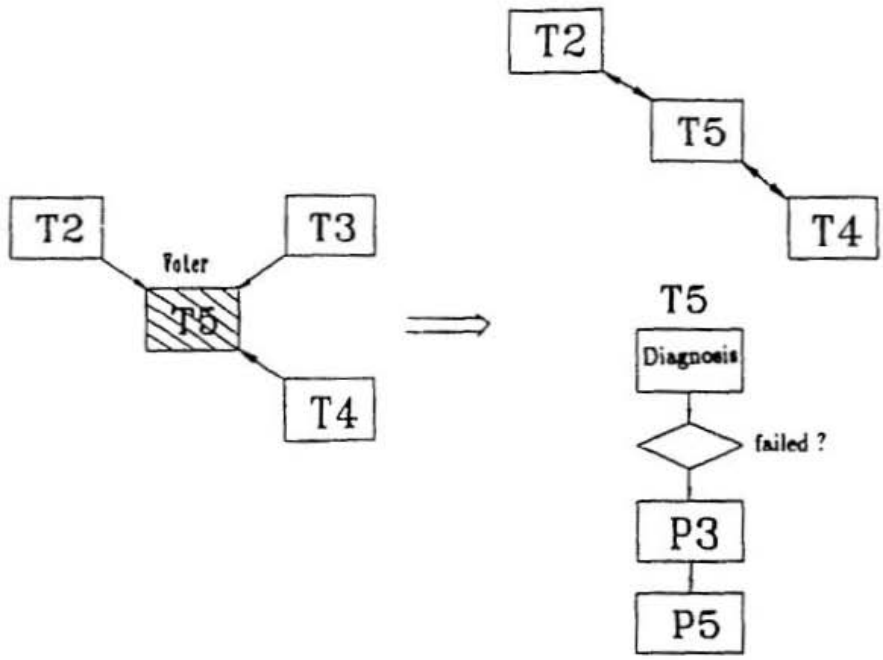


Figure 6.14: Transputer 3 failure

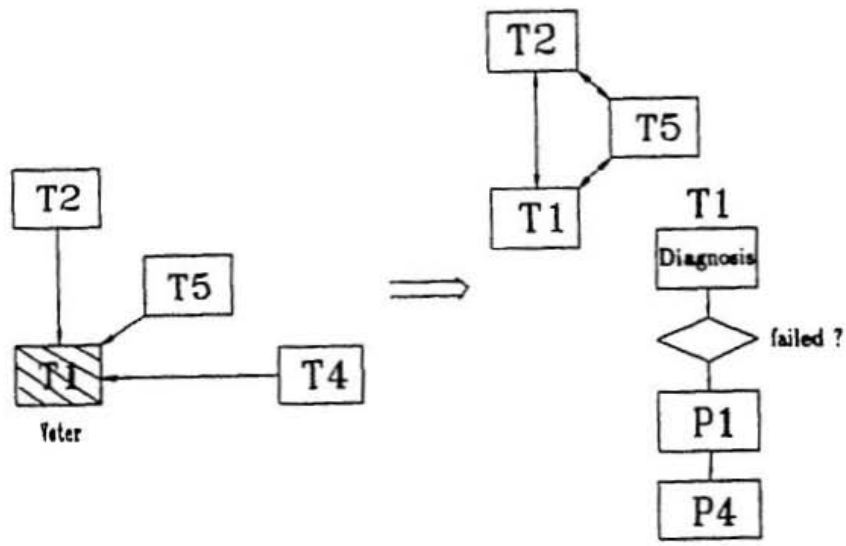


Figure 6.15: Transputer 4 failure

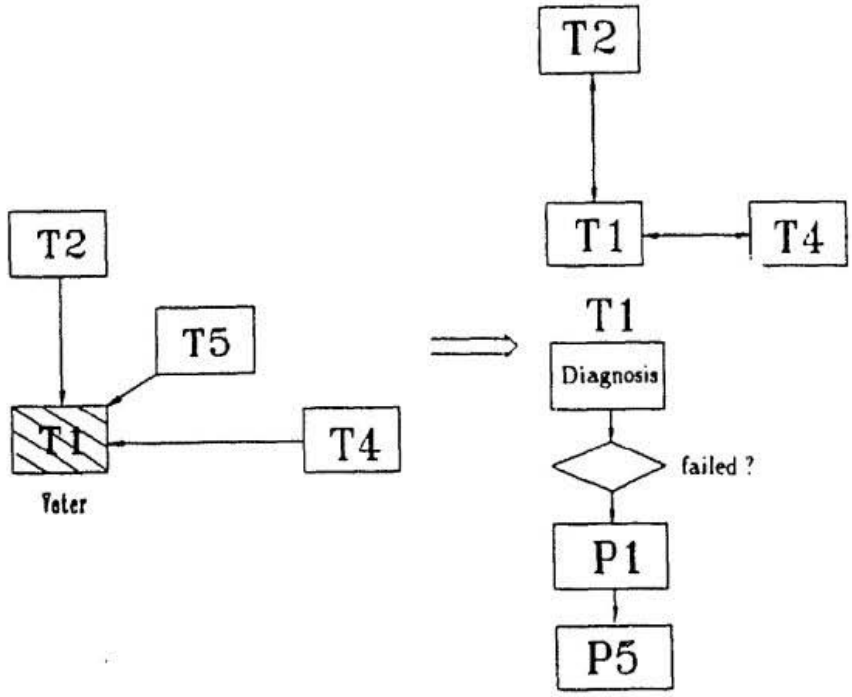


Figure 6.16: Transputer 5 failure

passing a verdict on the failed processor T2. Fig. 6.17 shows such a strategy.

6.5 Effects of Sampling Time On Control System Performance

In real-time digital control, sampling time is a performance factor. To evaluate the sampling effects on the performance of field-oriented control system, a simulation based on the parallel processing architecture with five Transputers, was carried out. The motor dynamics algorithm is performed by Transputer 5. In each processing loop, Transputer 5 sends out the motor state variables, such as motor speed, stator currents and voltages. If the Transputer 3 is not fast enough to catch up with the Transputer 5 in every processing loop, one or two more sample values may be lost. For example, if the processing time for one loop in Transputer 5 is 100μs, losing two processing loops in Transputer 3 means that sampling time becomes

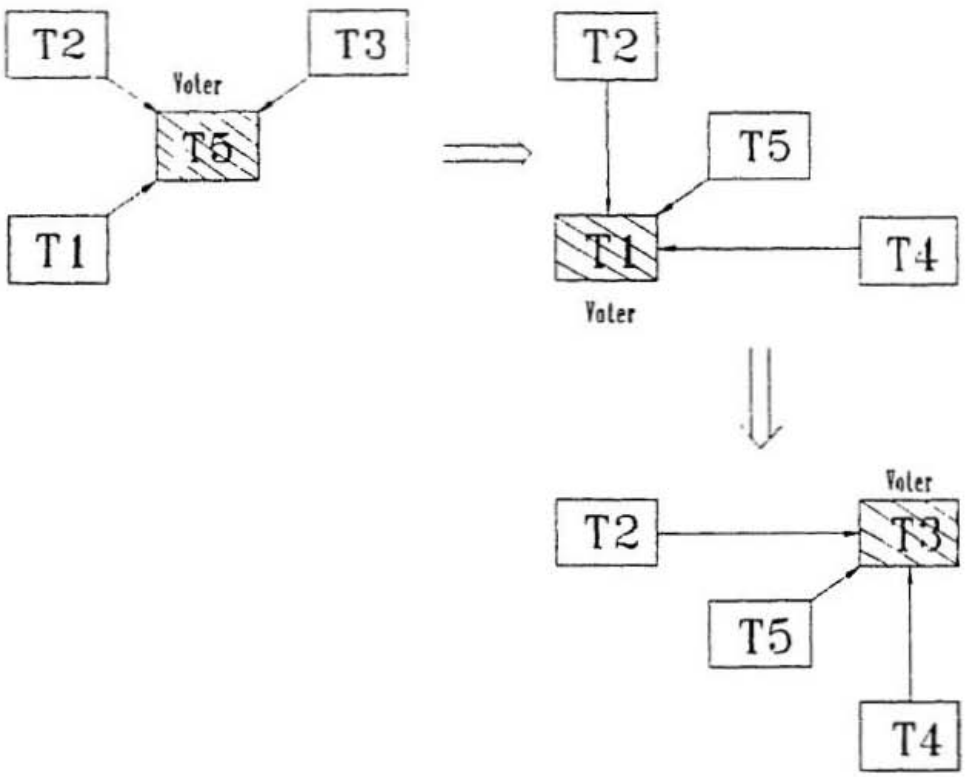


Figure 6.17: Multi-voting for transputer failure

300μ. To indicate the sampling time effects, four sampling times (100μs, 150μs, 200μs and 400μs) are selected for testing. The simulation results are shown in Fig. 6.18. In Fig. 18(a), by using 100 μs sampling time, the motor torque is controlled instantaneously and the control behaviour is satisfactory. In Fig. 6.18(b) (150 μs), the controlled torque current presents somewhat higher overshoot compared with Fig. 6.18(a). In Fig. 6.18(c), the controlled torque current overshoot is dramatically increased due to a relatively slow sampling time (200 μs). This high torque current overshoot may be not acceptable. In Fig. 6.18(d), the torque current is totally out of control due to a very slow sampling (400 μs).

The simulation for sampling time effects on control performance has indicated that the stability of digital control system not only depends on the control system gain, but also depends on the control system sampling time. To insure a stable control system, the system gain has to be reduced if the sampling time is not short enough due to processor speed. We know that reduced control system gain can result in a slow dynamic control response. Usually, the current sampling time is chosen be 10-20 times the stator time constant to satisfy a high dynamic response [16]. In order to obtain a very fast system dynamic control, high processing speed processors are necessary so that a very short sampling time is achievable.

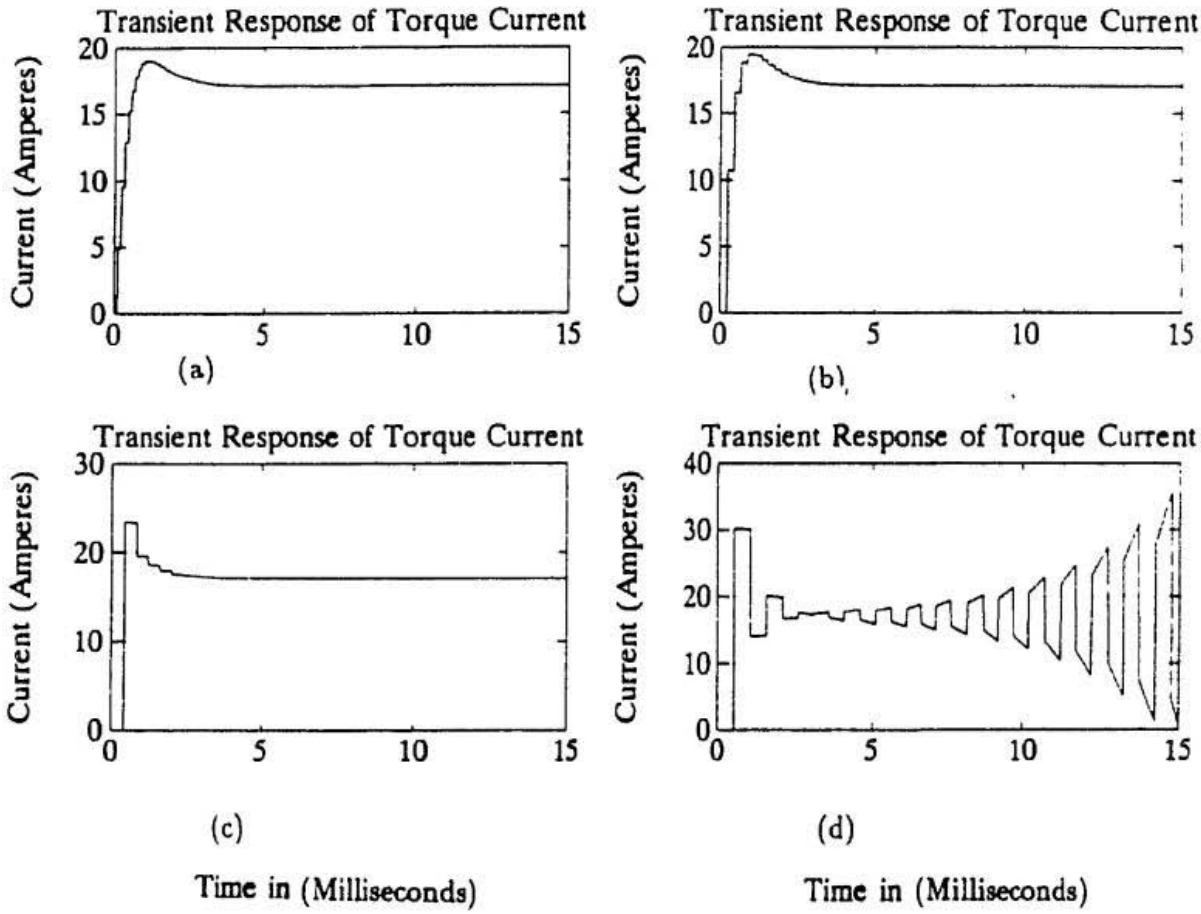


Figure 6.18: Motor torque current response

Chapter 7

Conclusions

The objective of achieving a high performance AC induction motor control has been approached through the following three major original contributions:

1. Investigation of the control characteristics of AC induction motor based on the principle of indirect field-oriented control.

2. Development of a new parameter adaptation control method.

3. Proposal of a parallel processing controller for complex control and system reliability requirements in the time critical condition.

An analytical model for indirect field-oriented control of AC induction motor has been developed. A series of digital computer simulations have been carried out to verify that using the field-oriented control the control performance of AC induction motor is similar to that of DC motor.

The necessary and sufficient conditions for obtaining a linear model of indirect field-oriented control of AC induction motor have been derived and verified by simulation results, which can support the design of PI or PID controllers for field-oriented control of AC induction motor by using the classical Nyquist, Bode or root

locus techniques.

In addition, the definition of field-oriented control has been extended in a broad sense to show that a high performance control can also be realized. Excellent dynamic control response using such an extended scheme has been shown by corresponding simulation trials. In principle, the extended field-oriented control is similar to the series-shunt excited DC motor control, but the resulting performance is superior to that of DC motor, which can be seen from a series of simulation results.

Next, a new parameter adaptation scheme termed *flux orientation feedback control* for indirect field-oriented control of AC induction motor has been developed. The simulation results of the proposed method show an excellent performance of parameter adaptive control when the simulated rotor resistance is varied in a worst case and under various operation conditions, such as step change in rotor speed or in load. It is expected that incorporated with the proposed stator resistance dynamic identification, such a scheme can work well under zero or crawling speed conditions. Furthermore, the variations of rotor parameter can be identified dynamically so that the actual rotor parameters can be obtained. The simulation results of the proposed scheme of dynamic correction for rotor flux orientation and dynamic identification for rotor parameters have clearly shown that the method is superior to any other recently reported schemes [7] [8] [24].

Lastly, a Transputer based parallel processing controller has been proposed for indirect field-oriented control, and a set of experiment has been carried out for parallel processing simulation purposes. Results show that with the parallel processing controller very complex control algorithms, such as the field-oriented control and

parameter adaptation control, can be handled in a very short time which is enough to satisfy a fast dynamic control system.

The potential of parallelism in a task dependent AC drive control system has been studied. Simulation results show that if sampling time is small enough, the modification of task dependency in an AC drive control system might provide a further decomposition of control algorithms, so that the parallel processing speed could be increased again.

It is believed that the Transputer system has significant advantages, particularly if a flexible architecture to meet multifarious specifications is the main criterion. Furthermore, real-time simulations can be realized with a parallel processing environment, which leads to a more practical design of a parallel processing controller.

The effect of link failures and processor failures have been studied. As a result of flexible and reconfigurable architecture, system reliability can be increased by incorporating fault tolerance techniques.

A simulation test shows that any link failure can be recovered by dynamically reconfiguring the link network.

A combination of hardware redundancy and time redundancy has been proposed to improve the processor reliability such that the failed processor could be found out and replaced by a voter .

Finally, it is expected that all the results in this study will be further examined by experimental testing.

Chapter 8

Suggestions for Future Work

The present research contributes simulation results related to indirect field-oriented control of AC induction motors. Obviously, all the simulation results should be further verified by experimental work. The proposed schemes also need more theoretical analysis. The suggestions are outlined as follows:

1. It is strongly recommended that the proposal of the extended field-oriented control be verified by extensive experimental works. Theoretical analysis of the effectiveness of lower slip frequency on field-oriented control is required.

2. The truly linear control structure of the AC induction motor should be examined by experimental work. To validate the linear design of the PI or PID controller for AC induction motor control, comparisons should be carried out between the parameters of the PI or PID controller from the linearized model and the parameters of the PI or PID controller from an experimental model.

3. It is strongly recommended that the proposed parameter adaptation control, named rotor flux orientation feedback control be verified by experimental study. The theoretical analysis of the control stability is required.

4. A full experimental set-up for the indirect field-oriented control of AC induction motors using parallel processing controller - Transputer network - should be put together to verify the feasibility and effectiveness of parallel processing control under a time critical condition.

   Modern control techniques, such as the adaptive control, model reference adaptive control, parameter identification, sliding model control and optimal control are expected to be realized by such a parallel processing controller for the real-time control requirement.

   In addition, the incorporation of fault tolerance techniques could be attempted in the parallel processing environment to meet the increased requirement of system reliability. Analysis of the increased controller reliability should be studied by using a Markov model.

Bibliography

[1] P. C.Sen, *Electric Motor Drives and Control-Past, Present and Future.* IEEE Transactions on Industrial Electronics, Vol.37, No.6, December 1990, pp.[562-575].

[2] B. K. Bose, *Motion Control Technology Present and Future*, IEEE Transactions on Industry Applications, Vol.IA-21, No.6, November/December 1985, pp.[1337-1342].

[3] F. Blaschke, *The Principle of Field Orientation as Applied to the New Transvektor Closed-Loop Control System for Rotating-Field Machines*, Siemens Review, No.5, 1972, pp.[217-220].

[4] K. Hasse, *Zur Dynamik Drehzahlgeregelter Antriebemit Stromrichtergespeisten Asynchron-Kurzschlusslaufermaschinen. Dissertation* Techn. Hochsch., Darmstadt, West Germany, 1969.

[5] E. Y. Y. Ho and P. C. San, *Decoupling Control of Induction Motor Drives*, IEEE Transactions on Industrial Electronics, Vol. 35, No. 2, pp.[253-262], 1988.

[6] T. A.Lipo, *Recent Progress in the Development of Solid-State AC Motor Drives*, IEEE Transactions on Power Electronics, Vol.3, No.2, April 1988, pp.[105-117].

[7] C.C.Chan and H. Wang, *An Effective Method for Rotor Resistance Identification for High-Performance Induction Motor Vector Control*, IEEE Transactions on Industrial Electronics, Vol.37, No.6, December 1990, pp.[477-482].

[8] R. D. Lorenz and D. B. Lawson, *A Simplified Approach to Continuous On-Line Tuning of Field- Oriented Induction Machine Drives*, IEEE Transactions on Industry Applications, Vol.26, No.3, May/June 1990, pp.[420-424].

[9] T. M. Rowan, R. J. Kerkman and D. Leggate *A Simple On-Line Adaption for Indirect Field Orientation of an Induction Machine*, IEEE Transactions on Industry Applications, Vol.27, No.4, July/August 1991, pp.[720-727].

[10] B. K. Bose, *Technology Trends in Microcomputer Control of Electrical Machines*, IEEE Transactions on Industrial Electronics, Vol.35, No.1, February 1988, pp.[160- 177].

[11] W. Leonhard, *Microcomputer Control of High Dynamic Performance - A Survey*, Automatica, Vol. 22, No. 1, pp.[1-19], 1986.

[12] F. Harashima, Seijikondo, K. Ohnishi, Masatoshikajita and M. Susono, *Multimcroproceeor-Based Control System for Quick Response Induction Motor Drive*, IEEE Transactions on Industry Applications, Vol.IA-21, No.4, May/June 1985, pp.[602-608].

[13] K. Kubo, M. Watanabe, T. Onmae and K. Kamiyama, *A Fully Digitalized Speed Regulator Using Multimicroprocessor System for Induction Motor Drives*, IEEE Transactions on Industry Applications, Vol.IA-21, No.4, July/August 1985, pp.[1001-1007].

[14] P. L. Fleming, *Parallel Processing in Control*, Peter Peregrinus Ltd. 1988

[15] G.M.Asher and M.Summer, *Parallelism and the Transputer for Real-Time High-Performance Control of Induction Motors*, IEE Proceedings, Vol.137. Pt.D, No.4, July 1990, pp.[179-188].

[16] R.G. Harleg, M.R. Webster, G. Diana and D.C. Levy, *A High Performance, Interactive, Real-Time Control for a Field Oriented Control AC Servo Drive*, IEEE 1990 Industry Application Society Annual Meeting, Part I, pp.[613-618], Seattle, U.S.A.

[17] L. J. Garces, *Parameter Adaption for the Speed-Controlled Static AC Drive with a Squirrel-Cage Induction Motor*, IEEE Transactions on Industry Applications, Vol.IA-16, No.2, March/April 1980, pp.[173-178].

[18] T. Matsuo and T. A. Lipo, *A Rotor Parameter Identification Scheme for Vector-Controlled Induction Motor Drives*, IEEE Transactions on Industry Applications, Vol.IA-21, No.4, May/June 1985, pp.[624-632].

[19] L. C. Zai and T. A. Lipo, *An Extended Kalman Filter Approach to Rotor Time Constant measurement in PWM Induction Motor Drives*, Conference Record, IEEE IAS Annual Meeting, October 1987, pp.[177-183].

[20] C.H. Liu, C..C Hwu and Y.F. Feng, *Modelling and Implementation of a Microprocessor-Based CSI- Fed Induction Motor Drive Using Field-Oriented Control*, IEEE Transactions on Industry Applications, Vol.25, No.4, July-August 1989, pp.[588-597].

[21] Y.Y. Tzou and H.J. Wu, *Multimicroprocessor-Based Robust Control of an AC Induction Servo Motor*, IEEE Transactions on Industry Applications, Vol.26, No.3, May/June 1990, pp.[441-449].

[22] R.G. Rajulu and V. Rajaraman, *Estimation of Execution Times of Process Control Algorithms on Microcomputers*, IEEE Transactions on Industrial Electronics, Vol.IE-31, No.1, February 1984, pp.[56-59].

[23] F. Garcia-Nocetti, H.A. Thompson, M.C.F. De Olivesra, C.M. Jones and P.J. Fleming, *Implementation of a Transputer-Based Flight Controller*, IEE Proceedings, Vol.137, Pt.D, No.3, May 1990, pp.[130-136].

[24] M. Zhou and W. Qu, *A Novel Method of Detecting Rotor Time Constant of Motor on Line in Vector Control System*, Proceedings of 1990 International Power Electronics Conference, pt. 1, pp.[29-36], Shinjuku, Tokyo.

[25] M. Koyama, M. Yano, I. Kamiyama and S. Yano, *Microprocessor-Based Vector Control System for Induction Motor Drives with Rotor Time Control Identification Function*, IEEE Transactions on Industry Applications, Vol.IA-22, No.3, May/June 1986, pp.[453-459].

[26] K. Ohnishi, Y. Ueda and K. Miyachi, *Model Reference Adaptive System Against Rotor Resistance Variation in Induction Motor Drive*, IEEE Transactions on Industrial Electronics, Vol.IE-33, No.3, August 1986, pp.[217- 223].

[27] Y. Ueda *et al*, *A Digital Compensation Method for Rotor Resistance Variation in the High Performance Slip Frequency Control of Induction Motor* , Report of IEE of Japan, RM-84-34, June 1984.

[28] P. Min-Ho, K. Young-Real, W. Chung-Yuen, K. Hack-Seong and K. Yuen-Jun, *Microprocessor-Based Field- Oriented Control of a Current Controlled PWM Inverter-Fed Induction Motor Servo-Drive*, Proceedings of 1990 International Power Electronics Conference, Vol.1, April 2-6, 1990, Shinjuku, Tokyo, Japan, pp.[37-48].

[29] S. Sathikumar and J. Vithayathil, *Digital Simulation of Field-Oriented Control of Induction Motor*, IEEE Transactions on Industrial Electronics, Vol. IE-31, No. 2, pp.[141-148], May 1984

[30] G. R.Slemon, *Modelling of Induction Machines for Electric Drives*, IEEE Transactions on Industry Applications, Vol.25, No.6, November/December 1989, pp.[1126- 1131].

[31] S. N. Ghani, *Digital Computer Simulation of Three-Phase Induction Machine Dynamics-A Generalized Approach*, IEEE Transactions on Industry Applications, Vol.24, No.1, January/February 1988, pp.[106-114].

[32] A. Parameswar and N. K. Sinha, *A Transputer Based Parallel Processing Architecture for Robotics Applications*, Canadian Conference on Electrical and Computer Engineering, September 1990.

[33] G.S. Stavrakakis, C. Lefas and A. Pouliezos, *Parallel Processing Computer Implementation of a Real Time DC Motor Drive Fault Detection Algorithm*, IEE Proceedings, Vol.137, Pt.B, No.5, September 1990, pp.[309-313].

[34] H.A. Thompson, P.J. Fleming, *Fault-Tolerant Transputer-Based Controller Configurations for Gas-Turbine Engines*, IEE Proceedings, Vol.137, Pt.D, No.4, July 1990, pp.[253-260].

[35] B.K. Bose, *Power Electronics and AC Drives*, Prentice-Hall, Englewood Cliffs, New Jersey 07632, 1986.

[36] B. K. Bose, *Microcomputer Control of Power Electronics and Drives*, IEEE PRESS, 1987.

[37] J. Wu, *A Fault-Tolerant Task Scheduling Method for Parallel Processing Systems*, Proceedings of the ISMM International Conference, Parallel and Distributed Computing, and Systems. pp.[198-200], New York, Oct. 10 - 12, 1990.

[38] Y. D. Li and B. D. Fornel, *A Voltage-Oriented Control of PWM VSI-Fed Induction Motor Drives*, Proceedings of 1990 International Power Electronics Conference. pt. 1, pp.[49- 53], Shinjuku, Tokyo.

[39] W. Li and R. Venkatesan, *A New Adaptive Control Scheme for Indirect Vector Control System* , IEEE 1992 Industry Application Society Annual Meeting(accepted), U.S.A.

[40] *Transputer Technical Specifications* Computer System Architects 1990.

[41] *Transputer Handbook* INMOS Limited 1989

Bibliography

1. K. B. Nordin, D. W. Novotny and D. S. Zinger, *The Influence of Motor Parameter Deviations in Feedfarward Field Orientation Drive Systems*, IEEE Transactions On Industry Application. Vol. IA-21, No. 4, July/August 1985.

2. C. M. Liaw, Y. S. Kung and C. M. Wu, *Design and Implementation of a High-Performance Field-Oriented Induction Motor Drive*, IEEE Transactions on Industiral Electronics, Vol. 38, No. 4, August 1991.

3. T. Kume and T. Iwakane, *High- Performance Vector-Controlled AC Motor Drives: Applications and New Technologies*, IEEE Transactions on Industry Applications, Vol.IA-23, No.5, September/October 1987, pp.[872-880].

4. R. Gabriel, W. Leonhard and C. Nordby, *Field-Oriented Control of a Standard AC Motor Using Microprocessors*, IEEE Transactions on Industry Applications, Vol.IA-16, No.2, March/April 1980.

5. C. Wang, D. W.Novotny and T. A.Lipo, *An Automated Rotor Time Constant Measurement S ystem for Indirect Field-Oriented Drives*, IEEE Transactions on Industry Applications Vol.24, No.1, January/February 1988.

6. N. Mutoh, H. Nagase, K. Sakai, and H. Ninomiya, *High-Response Digital Speed-Control System for Induction Metros*, IEEE Transactions on Industrial Electronics, Vol.IE-33, No.1, February 1986, pp.[52-58].

7. T. Murata, T. Tsuchiya and I. Takeda, *Vector Control for Induction Machine on the Application of Optimal Control Theory*, IEEE Transactions on Industrial Electronics, Vol.37, No.4, August 1990, pp.[283-290].

8. J.W. Ponton and R. Mckinnel, *Nonlinear Process Simulation and Control Using Transputers*, IEE Proceedings, Vol.137, Pt.D, No.4, July 1990, pp.[189-195].

9. M. P. Kazmierkowski and W. Sulkowski, *A Novel Vector Control Scheme for Transistor PWM Inverter-Fed Induction Motor Drive*, IEEE Transactions on Industrial Electronics, Vol.38, No.1, February 1991, pp.[41-47].

10. I. Takahashi, T. Noguchi, *A New Quick-Response and High-Efficiency Control Strategy of an Induction Motor*, IEEE Transactions on Industry Applications, Vol.IA-22, No.5, September/October 1986, pp.[820-827].

11. K. Ramamritham, J. A.Stankovic and P.F. Shiah, *Efficient Scheduling Algorithms for Real-Time Multiprocessor Systems*, IEEE Transactions on Parallel and Distributed Systems, Vol.1, No.2, April 1990, pp.[184-194].

12. C.Barrientos and C.A.Klein, *Design of a Multimicroprocessor-Based Control Using a Structured Design Approach*, IEEE Transactions on Industrial Electronics, Vol.IE- 31, No.4, November 1984, pp.[292-298].

13. M. Macdonald and P. C.Sen, *Control Loop Study of Induction Motor Drives Using DQ Model*, IEEE Transactions on Industrial Electronics and Control

Instrumentation, Vol.IECI-26, No.4, November 1979, pp.[237-243].

14. M. R. Stojic and S. N. Vukosavic, *Design of Microprocessor-Based System for Positioning Servomechanism with Induction Motor*, IEEE Transactions on Industrial Electronics, Vol.38, No.5, October 1991, pp.[369-378].

15. A. M. Khambadkone and J. Holtz, *Vector- Controlled Induction Motor Drive with a Self-Commissioning Scheme*, IEEE Transactions on Industrial Electronics, Vol.38, No.5, October 1991, pp.[322-327].

16. J. Zhang and T. H. Barton, *A Fast Variable Structure Current Controller for an Induction Machine Drive*, IEEE Transactions on Industry Applications, Vol.26, No.3, May/June 1990, pp.[415-419].

17. R.W. Daniel and P.M. Sharkey, *Transputer Control of a Puma 560 Robot via the Virtual Bus*, IEE Proceedings, Vol.137, Pt.D, No.4, July 1990, pp.[245-252].

18. P. Vas, *Vector Control of AC Machines*, Oxford Science Pulications, 1990.

19. *Logical System C* Computer System Architects 1990.

20. A. DeCegrama, *The Technology of Parallel Processing*, Vol. 1, Prentice Hall Englewood Cliffs, New Jersey, 1989.

21. A. Bellini, G. Figalli and G. Ulivi, *A Microcomputer-Based Optimal Control System to Reduce the Effects of the Parametric Variations and Speed Measurement Error in Induction Motor Drives.* IEEE Trans. on Ind. App., Vol IA-22, No. 1, Jan./Feb. 1986, pp.[42-50].

22. R. D. Lorenz, *Turning of Field-Oriented Induction Motor Controllers for High-Performance Applications.* IEEE Trans. on Ind. App., Vol IA-22, No. 2, March/April 1986, pp.[293-297].

23. R. D. Lorenz and D. Lawson, *Performance of Feedforward Current Regulators for Field Oriented Induction Machine Controller.* IEEE Trans. on Ind. App., Vol IA-23, No. 4, July/Aug. 1987, pp.[597-602].

24. *PRO-MATLAB for VAX/VMS Computers, User's Guide*, The MathWorks Inc. June 1989.

Appendix-A1

This program is designed for the simulation of indirect field-oriented control of AC induction motor.

```
#include<stdio.h>
#include<math.h>
main( )
   {  FILE  *fp;
      int i, T3, T4, T5, T, Np=2, Kp=8, T6, T7;
      float  Rs=0.49, Ls=0.0388, Lr=0.0354, Lb=0.0021, M=0.0354,
             J=0.024, D=0.0011, TR, Si,o,ua,R=0.45,n,phai,
             Rr, t, Ws, Is, T2, TL, N1, N2, Kd, Kq, Ki, K1, K2,
             Ka, Q, y1, y2, y9, y14, y15, y16, y17, y18,y19, y20, y21,
             y22, y23, y24, y25, y26, y27, y28, y29, y30, y31, y32, y33,
             y39,y3[50010], y4[5001C], y5[50010], y6[50010], y7[50010],
             y8[50010], y10[50010], y11[50010], y12[50010], y13[50010],
             y34[50010], y35[50010], y36[50010], UT1[50010],S1[50010],
             y37[50010], y38[50010], y40[50010], UT[50010], TM[50010],
             y41[50010], y42[50010], Ud[50010], Uq[50010], V[50010],
             U[50010], X[50010], Ta[50010], ph[50010], S[50010],
             Ia[50010], y[50010],e[50010],p[50010];
      printf("Enter input T,T3,T4,T5,T6,T7\n");
```

```
scanf("%d %d %d %d %d",&T,&T3,&T4,&T5,&T6,&T7);
printf("T=%d,T3=%d,T4=%d,T5=%d,T6=%d,T7=%d\n",T,T3,T4,T5,T6,T7);
printf("Enter input T2,TL,N1,N2\n");
scanf("%f %f %f %f",&T2,&TL,&N1,&N2);
printf("T2=%f,TL=%f,N1=%f,N2=%f\n",T2,TL,N1,N2);
i=1; Kd=0.8; Kq=0.6; Rr=0.45; Ki=0.003; Ka=0.01; K1=0.01;
K2=0.01;
Q=1.0/(Ls*Lr-M*M);
y14=0; y15=0; y16=0; y17=0; y24=0; y25=0; y26=0; y27=0; y28=0;
y29=0; y30=0; y31=0; y32=0; S[i]=0;S1[i]=0;e[i]=0;p[i]=0;
y3[i]=0; y4[i]=0; y5[i]=0; y6[i]=0; y7[i]=0; y8[i]=0; y10[i]=0;
y11[i]=0; y12[i]=0; y13[i]=0; y35[i]=0; y36[i]=0; y37[i]=0;
y38[i]=0; y40[i]=0; y41[i]=0; y42[i]=0; UT[i]=0; TM[i]=0;
if (!(fp=fopen("ud.d","wb")))
   {printf("cannot open file\n");
    exit(1);
    }

t=0;
while (t<=T)
   {  i=i+1;
      if (i>T3)
        TL=4;
      else TL=0;
```

```
if (i>T6)
   TL=2;
else TL=TL;
if (i>T4)
  Rr=0.08*(t-3)*(t-3)+0.125;     * Rotor resistance variations *
else Rr=Rr;
y1=N1;  y2=N2;
if (i>T7)
   y2=10;
else y2=N2;
y3[i]=y1-y42[i-1]*0.585652;
y4[i]=y2-y40[i-1]*0.027922;
y5[i]=Kp*(y4[i]-y4[i-1])+Kp*Ki*y4[i]+y5[i-1];
y6[i]=Kd*(y3[i]-y3[i-1])+Kd*K1*y3[i]+y6[i-1];
if (y5[i]>10)
    y5[i]=10;
else if (y5[i]<(-10))
    y5[i]=(-10);
else y5[i]=Kp*(y4[i]-y4[i-1])+Kp*Ki*y4[i]+y5[i-1];
y7[i]=y5[i]-y41[i-1]*0.585652;
y8[i]=Kq*(y7[i]-y7[i-1])+Kq*K2*y7[i]+y8[i-1];
y9=y5[i]*R/(y1*Lr);
y10[i]=y9+y40[i-1];
y[i]=(y10[i]+y10[i-1])*T2/2+y[i-1];
```

```
y11[i]=y[i];
Ud[i]=y6[i]-(Ls-(M*M/Lr))*y10[i]*y41[i-1]/30; * Feed-forward
Uq[i]=y8[i]+Ls*y10[i]*y42[i-1]/30;              controller *
y12[i]=(Ud[i]*cos(y11[i])-Uq[i]*sin(y11[i]))*30;
y13[i]=(Ud[i]*sin(y11[i])+Uq[i]*cos(y11[i]))*30;
y18=y12[i]-Rs*(Lr*y14-M*y15)*Q;
y19=(-1)*Rr*(Ls*y15-M*y14)*Q-y40[i-1]*y17;
y20=y13[i]-Rs*(Lr*y16-M*y17)*Q;
 21=(-1)*Rr*(Ls*y17-M*y16)*Q+y40[i-1]*y15;
y22=y14+y18*T2;
y23=y15+y19*T2;
y24=y16+y20*T2;
y25=y17+y21*T2;
y26=y12[i]-Rs*(Lr*y22-M*y23)*Q;
y27=(-1)*Rr*(Ls*y23-M*y22)*Q-y40[i-1]*y25;
y28=y13[i]-Rs*(Lr*y24-M*y25)*Q;
y29=(-1)*Rr*(Ls*y25-M*y24)*Q+y40[i-1]*y23;
y30=y14+T2*(y18+y26)/2;
y31=y15+T2*(y19+y27)/2;
y32=y16+T2*(y20+y28)/2;
y33=y17+T2*(y21+y29)/2;
y14=y30; y15=y31; y16=y32; y17=y33;
y34[i]=(Lr*y30-M*y31)*Q;
y35[i]=(Ls*y31-M*y30)*Q;
```

```
y36[i]=(Lr*y32-M*y33)*Q;
y37[i]=(Ls*y33-M*y32)*Q;
n=sqrt(y31*y31+y33*y33);
phai=n*sin(y33/n);
y38[i]=3*M*(y36[i]*y35[i]-y34[i]*y37[i]);
y39=(y38[i]-TL)*Np;
y40[i]=T2*y39/(T2*D+J)+J*y40[i-1]/(T2*D+J);
y41[i]=y36[i]*cos(y11[i])-y34[i]*sin(y11[i]);
y42[i]=y34[i]*cos(y11[i])+y36[i]*sin(y11[i]);
UT[i]=y13[i]-y36[i]*Rs;
UT1[i]=y12[i]-y34[i]*Rs;
S[i]=(UT[i]+UT[i-1])*T2/2+S[i-1];
S1[i]=(UT1[i]+UT1[i-1])*T2/2+S1[i-1];
TM[i]=3*(S1[i]*y36[i]-S[i]*y34[i]);
if (i<T5)                              * Flux orientation control *
   {TR=0, Si=0,o=0,e[i-1]=0,p[i-1]=0;}
else {
      TR=3*M*6.83*y5[i]/0.585652-TM[i];
      Si=TR*4/(3*Np*M*(y41[i]*y41[i]-y42[i]*y42[i]));
      if (Si>0.75)
         {Si=0.75;}
      else {Si=Si;}
      o=0.5*asin(Si);
      e[i]=0-o; p[i]=0.2*e[i]; \* The controller may be PI
                                  or PID */
```

```
                    R=0.45+p[i];

                    }

fprintf(fp,"%f %f %f %f %f %f %f %f %f\n",y40[i],y38[i],y41[i],y42[i],
        y34[i],y36[i],y12[i],y13[i],phai);

            t=t+T2;
          }

      }
```

Appendix-A2 [6]

This parameters are used in the simulation of indirect field-oriented control of AC induction motor.

rated power	1 KW
rated speed	1710 rpm
number of poles	4
Rs	0.49 Ohm
Rr	0.45 Ohm
Ls	0.0388 H
Lr	0.0354 H
M	0.0354 H
total inertia J	0.024 Nm s/rad

Appendix-B1

This program is designed for the simulation of indirect field-oriented control of AC induction motor.

```
#include<stdio.h>
#include<math.h>
main( )
   {  FILE  *fp;

      int i, T3, T4, T5, T, Np=3, Kp=8, T6, T7;

      float  Rs=2.0, Ls=0.235, Lr=0.234, Lb=0.0021, M=0.224,

             J=0.051, D=0.0011, R=2.1,n,RS,

             Rr, t, Ws, Is, T2, TL, N1, N2, Kd, Kq, Ki, K1, K2,

             Ka, Q, y1, y2, y9, y14, y15, y16, y17, y18,y19, y20, y21,

             y22, y23, y24, y25, y26, y27, y28, y29, y30, y31, y32, y33,

             y39,y3[50010], y4[50010], y5[50010], y6[50010], y7[50010],

             y8[50010], y10[50010], y11[50010], y12[50010], y13[50010],

             y34[50010], y35[50010], y36[50010], UT1[50010],S1[50010],

             y37[50010], y38[50010], y40[50010], UT[50010], TM[50010],

             y41[50010], y42[50010], Ud[50010], Uq[50010], V[50010],

             U[50010], X[50010], Ta[50010], ph[50010], S[50010],

             R2[50010], R3[50010], R4[50010], Ia[50010];

     printf("Enter input T,T3,T4,T5,T6,T7\n");

     scanf("%d %d %d %d %d %d",&T,&T3,&T4,&T5,&T6,&T7);

     printf("T=%d,T3=%d,T4=%d,T5=%d,T6=%d,T7=%d\n",T,T3,T4,T5,T6,T7);
```

```
printf("Enter input T2,TL,N1,N2\n");
scanf("%f %f %f %f",&T2,&TL,&N1,&N2);
printf("T2=%f,TL=%f,N1=%f,N2=%f\n",T2,TL,N1,N2);
i=1; Kd=3; Kq=6; Rr=2.1; Ki=0.0007; Ka=0.01; K1=0.01;
K2=0.01;
Q=1.0/(Ls*Lr-M*M);
y14=0; y15=0; y16=0; y17=0; y24=0; y25=0; y26=0; y27=0; y28=0;
y29=0; y30=0; y31=0; y32=0; S[i]=0;S1[i]=0;R2[i]=2.1;
y3[i]=0; y4[i]=0; y5[i]=0; y6[i]=0; y7[i]=0; y8[i]=0; y10[i]=0;
y11[i]=0; y12[i]=0; y13[i]=0; y35[i]=0; y36[i]=0; y37[i]=0;
y38[i]=0; y40[i]=0; y41[i]=0; y42[i]=0; UT[i]=0; TM[i]=0;
if (!(fp=fopen("vd.d","wb")))
   {printf("cannot open file\n");
    exit(1);
    }

t=0;
while (t<=T)
   {  i=i+1;
      if (i>T3)
        TL=5;
      else TL=0;
      if (i>T6)
        TL=2;
```

```
else TL=TL;
if (i>T4)
  Rr=(0.47)*(t-3)*(t-3)+0.1;   * Rotor resistance variations *
else Rr=Rr;
y1=N1;  y2=N2;
if (i>T7)
   y2=1;
else y2=N2;
y3[i]=y1-y42[i-1];
y4[i]=y2-y40[i-1]*0.027922;
y5[i]=Kp*(y4[i]-y4[i-1])+Kp*Ki*y4[i]+y5[i-1];
y6[i]=Kd*(y3[i]-y3[i-1])+Kd*K1*y3[i]+y6[i-1];
if (y5[i]>10)
    y5[i]=10;
else if (y5[i]<(-10))
    y5[i]=(-10);
else y5[i]=Kp*(y4[i]-y4[i-1])+Kp*Ki*y4[i]+y5[i-1];
y7[i]=y5[i]-y41[i-1];
y8[i]=Kq*(y7[i]-y7[i-1])+Kq*K2*y7[i]+y8[i-1];
y9=y5[i]*R/(y1*Lr);
y10[i]=y9+y40[i-1];
y11[i]=(y10[i]+y10[i-1])*T2/2+y11[i-1];
Ud[i]=y6[i]-(Ls-(M*M/Lr))*y10[i]*y41[i-1]/30;     * Feed-forward
Uq[i]=y8[i]+Ls*y10[i]*y42[i-1]/30;                  controller *
```

```
y12[i]=(Ud[i]*cos(y11[i])-Uq[i]*sin(y11[i]))*30;
y13[i]=(Ud[i]*sin(y11[i])+Uq[i]*cos(y11[i]))*30;
y18=y12[i]-Rs*(Lr*y14-M*y15)*Q;
y19=(-1)*Rr*(Ls*y15-M*y14)*Q-y40[i-1]*y17;
y20=y13[i]-Rs*(Lr*y16-M*y17)*Q;
y21=(-1)*Rr*(Ls*y17-M*y16)*Q+y40[i-1]*y15;
y22=y14+y18*T2;
y23=y15+y19*T2;
y24=y16+y20*T2;
y25=y17+y21*T2;
y26=y12[i]-Rs*(Lr*y22-M*y23)*Q;
y27=(-1)*Rr*(Ls*y23-M*y22)*Q-y40[i-1]*y25;
y28=y13[i]-Rs*(Lr*y24-M*y25)*Q;
y29=(-1)*Rr*(Ls*y25-M*y24)*Q+y40[i-1]*y23;
y30=y14+T2*(y18+y26)/2;
y31=y15+T2*(y19+y27)/2;
y32=y16+T2*(y20+y28)/2;
y33=y17+T2*(y21+y29)/2;
y14=y30; y15=y31; y16=y32; y17=y33;
y34[i]=(Lr*y30-M*y31)*Q;
y35[i]=(Ls*y31-M*y30)*Q;
y36[i]=(Lr*y32-M*y33)*Q;
y37[i]=(Ls*y33-M*y32)*Q;
n=sqrt(y31*y31+y33*y33);
```

```
            y38[i]=4.5*M*(y36[i]*y35[i]-y34[i]*y37[i]);
            y39=(y38[i]-TL)*Np;
            y40[i]=T2*y39/(T2*D+J)+J*y40[i-1]/(T2*D+J);
            y41[i]=y36[i]*cos(y11[i])-y34[i]*sin(y11[i]);
            y42[i]=y34[i]*cos(y11[i])+y36[i]*sin(y11[i]);
fprintf(fp,"%f %f %f %f %f %f %f %f %f %f\n",y40[i],y38[i],y41[i],
            y42[i],y34[i],y36[i],y12[i],y13[i],n);
            t=t+T2;
        }
    }
```

Appendix-B1 [13]

This parameters are used in the simulation of indirect field-oriented control of AC induction motor.

rated power	5 KW
number of poles	6
Rs	2 Ohm
Rr	2.1 Ohm
Ls	0.235 H
Lr	0.234 H
M	0.224 H
total inertia J	0.32 Kg m

Appendix-C1

This program run in Transputer 1 is designed for the simulation of indirect field-oriented control of AC induction motor.

```
#include<stdio.h>
#include<math.h>
#include<conc.h>
#undef Time
#undef SetTime
main( )

   {  FILE   *fp;

      int    i=1,j=1, T3, T4, T5, T, Np=2, T6, T7, z,

             ProcToHigh(void), Kp=8, start, end;


      float

             R=0.45, n, Rr,TL, N1, N2, y40a, y40, y18, y41, y42, t,

             Ud,Uq, y1, y2, y9, ya,w1,w2,

             y3, y3a, y4a, y4, y5a, y5, y7, y7a, y8a, y8, y10a, y10,

             y42a, y, y41a, Kd, Kq, Ki, K1, K2, T2, y11, Ka, y6a, y6,

             Rs=0.49, Ls=0.0388, Lr=0.0354, Lb=0.0021, M=0.0354,

             J=0.024, D=0.0011,Si,o,ua,Ws, Is,

             Q, y14, y15, y16, y17, y19, y20, y21,y22,

             y23, y24, y25, y26, y27, y28, y29, y30, y31, y32, y33,

             y39, y12, y13, y34, y35, y36, y37, y38, TR,

             S1a, UT1a, y34a, y36a, S, Sa=0,UT1, S1, UT, TM, V,

             UTa=0, Ia, e, p, x;
```

```
S=0; S1a=0; e=0; p=0; y12=0; y13=0; y36=0; y34=0; S1=0;
UT1; UT1a=0; UT=0; TM=0; Rr=0.45; t=0; ya=0;

y14=0; y15=0; y16=0; y17=0; y24=0; y25=0; y26=0;
y27=0; y28=0; y29=0; y30=0; y31=0; y32=0; y40a=0;
y41a=0; y42a=0; y12=0; y13=0; y35=0; y36=0; y37=0; t=0;
Rr=0.45; Q=1.0/(Ls*Lr-M*M); y34a=0; y36a=0;
Kd=0.8; Kq=0.6; Ki=0.003; Ka=0.01; K1=0.01;
K2=0.01; y41a=0; y42a=0; y7a=0; y8a=0; y4a=0;
y5a=0; y3a=0; y6a=0; Lr=0.0354; y10a=0; R=0.45;
printf("Enter input T,T3,T4,T5,T6,T7\n");
scanf("%d %d %d %d %d %d",&T,&T3,&T4,&T5,&T6,&T7);
printf("T=%d,T3=%d,T4=%d,T5=%d,T6=%d,T7=%d\n",
       T, T3, T4, T5, T6, T7);
printf("Enter input T2,TL,N1,N2\n");
scanf("%f %f %f %f",&T2, &TL, &N1, &N2);
printf("T2=%f,TL=%f, N1=%f, N2=%f\n", T2, TL, N1, N2);
if (!(fp=fopen("ud.d","wb")))
{printf("cannot open file\n");
exit(1);}
ChanOut(LINK1OUT, (char*) &T,sizeof(T));
ChanOut(LINK1OUT, (char*) &T2,sizeof(T2));
ChanOut(LINK1OUT, (char*) &T3,sizeof(T3));
ChanOut(LINK1OUT, (char*) &T4,sizeof(T4));
ChanOut(LINK1OUT, (char*) &T5,sizeof(T5));
```

```
ChanOut(LINK1OUT, (char*) &T6,sizeof(T6));
ChanOut(LINK1OUT, (char*) &TL,sizeof(TL));
ProcToHigh();
t=0;
while (t<=T)
{
SetTime(o);
i=i+1;
y2=N2; y1=N1;
if (i>T7)
          y2=3;
else      y2=N2;
if (j<6)
{j=j+1;}
else {
ChanIn(LINK1IN, (char*) &y40a,sizeof(y40a));
j=1; }
ChanIn(LINK1IN, (char*) &y34a,sizeof(y34a));
ChanIn(LINK1IN, (char*) &y36a,sizeof(y36a));
y4=y2-y40a*0.027922;
y5=Kp*(y4-y4a)+Kp*Ki*y4+y5a;
y4a=y4;
if (y5>10)
          y5=10;
```

```
else if (y5<(-10))
          y5=(-10);
else      y5=y5;
y9=y5*R/(y1*Lr);
y5a=y5;
y10=y9+y40a;
y=(y10+y10a)*T2/2+ya;
y10a=y10;
y11=y;
ya=y;
w1=sin(y11);
w2=cos(y11);
y41a=y36a*w2-y34a*w1;
y42a=y34a*w2+y36a*w1;
y3=y1-y42a*0.585652;
y6=Kd*(y3-y3a)+Kd*K1*y3+y6a;
y3a=y3;
y6a=y6;
y7=y5-y41a*0.585652;
y8=Kq*(y7-y7a)+Kq*K2*y7+y8a;
y7a=y7;y8a=y8;
start = Time();
end = Time();
z= end - start;
```

```
y12=(y6*w2-y8*w1)*30;
y13=(y6*w1+y8*w2)*30;
ChanOut(LINK1OUT, (char*) &y12,sizeof(y12));
ChanOut(LINK1OUT, (char*) &y13,sizeof(y13));
UT=y13-y36a*Rs;
UT1=y12-y34a*Rs;
S=(UT+UTa)*T2/2+Sa;
S1=(UT1+UT1a)*T2/2+S1a;
TM=3*(S1*y36a-S*y34a);
Sa=S;UTa=UT;UT1a=UT1;S1a=S1;
if (i<T5)
    {TR=0, Si=0,o=0,e=0,p=0;}
else {
     TR=3*M*6.83*y5a/0.585652-TM;
     Si=TR*4/(3*Np*M*(y41a*y41a-y42a*y42a));
     if (Si>0.75)
        {Si=0.75;}
     else {Si=Si;}
     o=0.5*asin(Si);
     e=0-o; p=0.2*e;
     R=Rr+p;
     }
printf("difftime=%d\n",z);
ChanIn(LINK1IN, (char*) &Rr,sizeof(Rr));
```

```
            fprintf(fp,"%f %f %f %f %f\n\r",y40a,y41a,y42a,y34a,Rr);
            t=t+12;
        }
}
```

Appendix-C2

This program run in Transputer 2 is designed for the simulation of indirect field-oriented control of AC induction motor.

```
#include<stdio.h>
#include<math.h>
#include<conc.h>
#undef Time
#undef SetTime
main( )

   {

      int    i=1,j=1,Np=2,T,T6,T3,T4,T5;

      float  Rs=0.49, Ls=0.0388, Lr=0.0354, Lb=0.0021, M=0.0354,

             J=0.024, D=0.0011, TR, Si, o, ua, R=0.45, n, Ws, Is,

             T2, w2, Q, y14, y15, y16, y17, y18, y19, y20, y21,

             y23,y24, y25,y26, y27, y28, y29, y30, y31, y32, y33,

             y39, y11, y12, y13, y34, y35, y36, y37, y38, y40, t,

             y40a, y41, y41a, y42, y42a, Rr, y34a, y36a, TL, y22;

             y14=0; y15=0; y16=0; y17=0; y24=0; y25=0; y26=0;

             y27=0; y28=0; y29=0; y30=0; y31=0; y32=0; y40a=0;

             y41a=0; y42a=0; y12=0; y13=0; y35=0; y36=0; y37=0;

             t=0; Rr=0.45; Q=1.0/(Ls*Lr-M*M); y34a=0; y36a=0;

             ChanIn(LINK0IN, (char*) &T,sizeof(T));

             ChanIn(LINK0IN, (char*) &T2,sizeof(T2));

             ChanIn(LINK0IN, (char*) &T3,sizeof(T3));
```

```
        ChanIn(LINK0IN, (char*) &T4,sizeof(T4));
        ChanIn(LINK0IN, (char*) &T5,sizeof(T5));
        ChanIn(LINK0IN, (char*) &T6,sizeof(T6));
        ChanIn(LINK0IN, (char*) &TL,sizeof(TL));

while (t<=T)
    {
        SetTime(0);
        i=i+1;
        if (j<6)
        {j=j+1;}
        else {
        ChanOut(LINK0OUT, (char*) &y40a,sizeof(y40a));
        j=1; }
        ChanOut(LINK0OUT, (char*) &y34a,sizeof(y34a));
        ChanOut(LINK0OUT, (char*) &y36a,sizeof(y36a));
        ChanIn(LINK0IN, (char*) &y12,sizeof(y12));
        ChanIn(LINK0IN, (char*) &y13,sizeof(y13));
        if (i<T5) {Rr=0.45;}
        else { Rr=0.08*(t-3)*(t-3)+0.125;}
        y18=y12-Rs*(Lr*y14-M*y15)*Q;
        y19=(-1)*Rr*(Ls*y15-M*y14)*Q-y40a*y17;
        y20=y13-Rs*(Lr*y16-M*y17)*Q;
        y21=(-1)*Rr*(Ls*y17-M*y16)*Q+y40a*y15;
        y22=y14+y18*T2;
```

```
y23=y15+y19*T2;
y24=y16+y20*T2;
y25=y17+y21*T2;
y26=y12-Rs*(Lr*y22-M*y23)*Q;
y27=(-1)*Rr*(Ls*y23-M*y22)*Q-y40a*y25;
y28=y13-Rs*(Lr*y24-M*y25)*Q;
y29=(-1)*Rr*(Ls*y25-M*y24)*Q+y40a*y23;
y30=y14+T2*(y18+y26)/2;
y31=y15+T2*(y19+y27)/2;
y32=y16+T2*(y20+y28)/2;
y33=y17+T2*(y21+y29)/2;
y14=y30; y15=y31; y16=y32; y17=y33;
y34=(Lr*y30-M*y31)*Q;
y35=(Ls*y31-M*y30)*Q;
y36=(Lr*y32-M*y33)*Q;
y36a=y36;y34a=y34;
y37=(Ls*y33-M*y32)*Q;
n=sqrt(y31*y31+y33*y33);
y38=3*M*(y36*y35-y34*y37);
if (i<T3) {TL=0;}
else {TL=2;}
y39=(y38-TL)*Np;
y40=T2*y39/(T2*D+J)+J*y40a/(T2*D+J);
y40a=y40;
```

```
            ChanOut(LINK00UT, (char*) &Rr,sizeof(Rr));
            t=t+T2;
        }
    }
```

Appendix-D1

This program run in Transputer 1 is designed for the simulation of indirect field-oriented control of AC induction motor in parallel processing environment.

```
#include<stdio.h>
#include<math.h>
#include<conc.h>
#undef Time
#undef SetTime

main( )

   {  FILE   *fp;

      int    i=1, T3, T4, T5, T, Np=2, T6, T7,z,z1,z2,

             ProcToHigh(void),start,end;

      float

             R=0.45,n,Rr, T2, TL, N1, N2, y12, y13,y5,y8,y34,

             y40a,y11, y38, y40,y18, y41a,y42a,t;

             printf("Enter input T,T3,T4,T5,T6,T7\n");

             scanf("%d %d %d %d %d %d",&T,&T3,&T4,&T5,&T6,&T7);

             printf("T=%d,T3=%d,T4=%d,T5=%d,T6=%d,T7=%d\n",
                    T,T3,T4,T5,T6,T7);

             printf("Enter input T2,TL,N1,N2\n");

             scanf("%f %f %f %f",&T2,&TL,&N1,&N2);

             printf("T2=%f,TL=%f,N1=%f,N2=%f\n",T2,TL,N1,N2);

             if (!(fp=fopen("ud.d","wb")))

             {printf("cannot open file\n");
```

```
    exit(1);}
    ChanOut(LINK1OUT, (char*) &T,sizeof(T));
    ChanOut(LINK1OUT, (char*) &T7,sizeof(T7));
    ChanOut(LINK1OUT, (char*) &N1,sizeof(N1));
    ChanOut(LINK1OUT, (char*) &N2,sizeof(N2));
    ChanOut(LINK1OUT, (char*) &T2,sizeof(T2));
    ChanOut(LINK1OUT, (char*) &T5,sizeof(T5));
    ChanOut(LINK2OUT, (char*) &T,sizeof(T));
    ChanOut(LINK2OUT, (char*) &T2,sizeof(T2));
    ChanOut(LINK2OUT, (char*) &T3,sizeof(T3));
    ChanOut(LINK2OUT, (char*) &T4,sizeof(T4));
    ChanOut(LINK2OUT, (char*) &T5,sizeof(T5));
    ChanOut(LINK2OUT, (char*) &T6,sizeof(T6));
    ChanOut(LINK2OUT, (char*) &TL,sizeof(TL));
    ChanOut(LINK3OUT, (char*) &T5,sizeof(T5));
    ChanOut(LINK3OUT, (char*) &T,sizeof(T));
    ChanOut(LINK3OUT, (char*) &T2,sizeof(T2));
    t=0;
    ProcToHigh();
while (t<=T)
  {
    SetTime(0);
    i=i+1;
    ChanIn(LINK1IN, (char*) &z,sizeof(z));
```

```
        printf("difftime=%d\n",z);
        ChanIn(LINK1IN, (char*) &z1,sizeof(z1));
        printf("         difftime=%d\n",z1);
        ChanIn(LINK3IN, (char*) &y41a,sizeof(y41a));
        ChanIn(LINK3IN, (char*) &y42a,sizeof(y42a));
        ChanIn(LINK2IN, (char*) &y40,sizeof(y40));
        ChanIn(LINK2IN, (char*) &y38,sizeof(y38));
        ChanIn(LINK2IN, (char*) &y34,sizeof(y34));
        ChanIn(LINK2IN, (char*) &n,sizeof(n));
        ChanIn(LINK2IN, (char*) &Rr,sizeof(Rr));
        if (i<T5) {R=0.45;}
        else { ChanIn(LINK3IN, (char*) &R,sizeof(R));}
        ChanIn(LINK3IN, (char*) &z2,sizeof(z2));
        printf("              difftime=%d\n",z2);
        fprintf(fp,"%f\n\r",y41a);
        t=t+T2;
    }
}
```

Appendix-D2

This program run in Transputer 2 is designed for the simulation of indirect field-oriented control of AC induction motor in parallel processing environment.

```
#include<stdio.h>
#include<math.h>
#include<conc.h>
#undef Time
#undef SetTime

main( )

   {

      int    i=1,j=1, T3, T4, T5, T, Np=2, Kp=8, T6, T7,z,

             z1,ProcToHigh(void),start,end;

      float

             t, T2, TL, N1, N2,Ud,Uq, y1, y2, y9, ya,w1,w2,y36,

             y3, y3a, y4a,y4, y5a,y5, y7,y7a, y8a,y8, y10a,y10,

             y11, y12, y13,y34,R,Lr,Ki,y38,y40a,y40,y6,y41a,y41,

             y42a,y42,y;

             y41a=0;y42a=0;t=0;y40a=0;ya=0;y4a=0;y10a=0;y5a=0;

             Ki=0.003;  Lr=0.0354;R=0.45;


             ChanIn(LINK0IN, (char*) &T,sizeof(T));

             ChanIn(LINK0IN, (char*) &T7,sizeof(T7));

             ChanIn(LINK0IN, (char*) &N1,sizeof(N1));
```

```
        ChanIn(LINK0IN, (char*) &N2,sizeof(N2));
        ChanIn(LINK0IN, (char*) &T2,sizeof(T2));
        ChanIn(LINK0IN, (char*) &T5,sizeof(T5));
        ChanOut(LINK10UT, (char*) &N1,sizeof(N1));
        ChanOut(LINK10UT, (char*) &N2,sizeof(N2));
        ChanOut(LINK10UT, (char*) &T,sizeof(T));
        ChanOut(LINK10UT, (char*) &T2,sizeof(T2));
        ChanOut(LINK10UT, (char*) &T7,sizeof(T7));
        ChanOut(LINK10UT, (char*) &T5,sizeof(T5));
ProcToHigh();
while (t<=T)
  {
        SetTime(0);
        i=i+1;
        start = Time();
        y2=N2;y1=N1;
        if (i>T7)
                y2=3;
        else    y2=N2;
        ChanIn(LINK2IN, (char*) &y40a,sizeof(y40a));
        y4=y2-y40a*0.027922;
        y5=Kp*(y4-y4a)+Kp*Ki*y4+y5a;
        y4a=y4;
        if (y5>10)
```

```
                y5=10;
        else if (y5<(-10))
                y5=(-10);
        else    y5=y5;
        ChanOut(LINK1OUT, (char*) &y5,sizeof(y5));
        y9=y5*R/(y1*Lr);
        y5a=y5;
        y10=y9+y40a;
        y=(y10+y10a)*T2/2+ya;
        y10a=y10;
        y11=y;
        ya=y;
        ChanOut(LINK1OUT, (char*) &y11,sizeof(y11));
        w1=sin(y11);
        ChanOut(LINK1OUT, (char*) &w1,sizeof(w1));
        end = Time();
        z= end - start;
        ChanOut(LINK0OUT, (char*) &z,sizeof(z));
        ChanIn(LINK1IN, (char*) &z1,sizeof(z1));
        ChanOut(LINK0OUT, (char*) &z1,sizeof(z1));
        t=t+T2;
    }
}
```

Appendix-D3

This program run in Transputer 3 is designed for the simulation of indirect field-oriented control of AC induction motor in parallel processing environment.

```
#include<stdio.h>
#include<math.h>
#include<conc.h>
#undef Time
#undef SetTime

main( )

   {

      int    i=1,T,T7,T5,start,end,ProcToHigh(void),

             j=1,Kp=8,z,z1;

      float

             N1,N2, Kd, Kq, Ki, K1, K2,t,T2,y11,y34a,y36a,

             Ka, y1, y41,y41a,y42,y42a,y7,y7a,y8,y8a,y5,

             y3,y3a, y6a,y6,y2,y4,y4a,y5a,ya,Lr,y40a,

             y10,y10a,y,R,y9,w1,w2,y12,y13;

             Kd=0.8; Kq=0.6; Ki=0.003; Ka=0.01; K1=0.01;

             K2=0.01;y41a=0;y42a=0; y7a=0;y8a=0;y4a=0;

             y5a=0;y3a=0; y6a=0;Lr=0.0354;y10a=0;R=0.45;

             ya=0;y34a=0;y36a=0;w1=0;w2=0;t=0;

             ChanIn(LINK0IN,(char*) &N1,sizeof(N1));

             ChanIn(LINK0IN,(char*) &N2,sizeof(N2));
```

```
        ChanIn(LINKOIN,(char*) &T,sizeof(T));
        ChanIn(LINKOIN,(char*) &T2,sizeof(T2));
        ChanIn(LINKOIN,(char*) &T7,sizeof(T7));
        ChanIn(LINKOIN,(char*) &T5,sizeof(T5));
ProcToHigh();
while (t<=T)
    {
        SetTime(0);
        i=i+1;
        if    (j!=51)
              {j=j+1;}
        else  {
        ChanIn(LINK2IN, (char*) &y34a,sizeof(y34a));
        ChanIn(LINK2IN, (char*) &y36a,sizeof(y36a));
        j=1;  }
        start = Time();
        y41=y36a*w2-y34a*w1;
        y42=y34a*w2+y36a*w1;
        y41a=y41;y42a=y42;
        y1=N1;
        y3=y1-y42a*0.585652;
        y6=Kd*(y3-y3a)+Kd*K1*y3+y6a;
        y3a=y3;
        y6a=y6;
```

```
        ChanIn(LINK0IN, (char*) &y5,sizeof(y5));
        y7=y5-y41a*0.585652;
        y8=Kq*(y7-y7a)+Kq*K2*y7+y8a;
        y7a=y7;y8a=y8;
        ChanIn(LINK0IN, (char*) &y11,sizeof(y11));
        w2=cos(y11);
        ChanIn(LINK0IN, (char*) &w1,sizeof(w1));
        y12=(y6*w2-y8*w1)*30;
        y13=(y6*w1+y8*w2)*30;
        end = Time();
        z1=end-start;
        ChanOut(LINK00UT, (char*) &z1,sizeof(z1));
        ChanOut(LINK20UT, (char*) &y12,sizeof(y12));
        ChanOut(LINK20UT, (char*) &y13,sizeof(y13));
        ChanOut(LINK10UT, (char*) &y41a,sizeof(y41a));
        ChanOut(LINK10UT, (char*) &y42a,sizeof(y42a));
        ChanOut(LINK10UT, (char*) &y5,sizeof(y5));
        if (i<T5)
                {R=0.45;}
         else       {ChanIn(LINK1IN,(char*) &R,sizeof(R));}
        t=t+T2;
      }
}
```

Appendix-D4

This program run in Transputer 4 is designed for the simulation of indirect field-oriented control of AC induction motor in parallel processing environment.

```
#include<stdio.h>
#include<math.h>
#include<conc.h>
#undef Time
#undef SetTime

main( )
   {
      int    i=1,j=1,Np=2,T,T6,T3,T4,T5,ProcToHigh(void),start,
             end,z;

      float  Rs=0.49, Ls=0.0388, Lr=0.0354, Lb=0.0021, M=0.0354,
             J=0.024, D=0.0011, TR, Si,o,ua,R=0.45,n,Ws, Is, TL,
             T2,w2, Q, y14, y15, y16, y17, y18,y19, y20, y21,y22,
             y23,y24, y25,y26, y27, y28, y29, y30, y31, y32, y33,
             y39, y11, y12, y13,y34,y35,y36,y37, y38, y40,t,y40a,
             y41,y41a, y42,y42a,Rr,y34a,y36a;
             y14=0; y15=0; y16=0; y17=0; y24=0; y25=0; y26=0;
             y27=0; y28=0; y29=0; y30=0; y31=0; y32=0;y40a=0;
             y41a=0;y42a=0;y12=0;y13=0;y34a=0;y36a=0;y37=0;t=0;
             Rr=0.45; Q=1.0/(Ls*Lr-M*M);
             ChanIn(LINK2IN, (char*) &T,sizeof(T));
```

```
        ChanIn(LINK2IN, (char*) &T2,sizeof(T2));
        ChanIn(LINK2IN, (char*) &T3,sizeof(T3));
        ChanIn(LINK2IN, (char*) &T4,sizeof(T4));
        ChanIn(LINK2IN, (char*) &T5,sizeof(T5));
        ChanIn(LINK2IN, (char*) &T6,sizeof(T6));
        ChanIn(LINK2IN, (char*) &TL,sizeof(TL));
ProcToHigh();
while (t<=T)
    {
        SetTime(0);
        i=i+1;
        if    (j!=51)
              {j=j+1;}
        else  {
        ChanOut(LINK1OUT, (char*) &y34a,sizeof(y34a));
        ChanOut(LINK1OUT, (char*) &y36a,sizeof(y36a));
        j=1;  }
        ChanOut(LINK3OUT, (char*) &y40a,sizeof(y40a));
        ChanIn(LINK1IN, (char*) &y12,sizeof(y12));
        ChanIn(LINK1IN, (char*) &y13,sizeof(y13));
        if (i<T5) {Rr=0.45;}
        else { Rr=0.08*(t-3)*(t-3)+0.125;}
        y18=y12-Rs*(Lr*y14-M*y15)*Q;
        y19=(-1)*Rr*(Ls*y15-M*y14)*Q-y40a*y17;
```

```
y20=y13-Rs*(Lr*y16-M*y17)*Q;
y21=(-1)*Rr*(Ls*y17-M*y16)*Q+y40a*y15;
y22=y14+y18*T2;
y23=y15+y19*T2;
y24=y16+y20*T2;
y25=y17+y21*T2;
y26=y12-Rs*(Lr*y22-M*y23)*Q;
y27=(-1)*Rr*(Ls*y23-M*y22)*Q-y40a*y25;
y28=y13-Rs*(Lr*y24-M*y25)*Q;
y29=(-1)*Rr*(Ls*y25-M*y24)*Q+y40a*y23;
y30=y14+T2*(y18+y26)/2;
y31=y15+T2*(y19+y27)/2;
y32=y16+T2*(y20+y28)/2;
y33=y17+T2*(y21+y29)/2;
y14=y30; y15=y31; y16=y32; y17=y33;
y34=(Lr*y30-M*y31)*Q;
y35=(Ls*y31-M*y30)*Q;
y36=(Lr*y32-M*y33)*Q;
y37=(Ls*y33-M*y32)*Q;
y34a=y34;y36a=y36;
n=sqrt(y31*y31+y33*y33);
y38=3*M*(y36*y35-y34*y37);
if (i<T3) {TL=0;}
else {TL=2;}
```

```
            y39=(y38-TL)*Np;
            y40=T2*y39/(T2*D+J)+J*y40a/(T2*D+J);
            y40a=y40;
            ChanOut(LINK0OUT, (char*) &y36,sizeof(y36));
            ChanOut(LINK0OUT, (char*) &y34,sizeof(y34));
            ChanOut(LINK0OUT, (char*) &y13,sizeof(y13));
            ChanOut(LINK0OUT, (char*) &y12,sizeof(y12));
            ChanOut(LINK2OUT, (char*) &y40,sizeof(y40));
            ChanOut(LINK2OUT, (char*) &y38,sizeof(y38));
            ChanOut(LINK2OUT, (char*) &y34,sizeof(y34));
            ChanOut(LINK2OUT, (char*) &n,sizeof(n));
            ChanOut(LINK2OUT, (char*) &Rr,sizeof(Rr));
            t=t+T2;
        }
}
```

Appendix-D5

This program run in Transputer 5 is designed for the simulation of indirect field-oriented control of AC induction motor in parallel processing environment.

```
#include<stdio.h>
#include<math.h>
#include<conc.h>
#undef Time
#undef SetTime

main( )

   {

     int    i=1,T,T5,ProcToHigh(void),start,end,z,z2;

     float  Rs=0.49,Ls=0.0388,Lr=0.0354,M=0.0354,TR,Si,o,ua,

            R=0.45,n,Rr,Ws, Is,Ka,Np,S1a,y5a,y41a,y42a,UT1a,

            y12, y13,S,Sa=0,y34, y36, UT1,S1,t,T2, UT,TM,V,

            UTa=0,Ia, y,e,p;

            S=0;S1a=0;e=0;p=0;y12=0;y13=0;y36=0;y34=0;S1=0;

            UT1;UT1a=0;UT=0;TM=0;Rr=0.45;t=0;

            ChanIn(LINK2IN, (char*) &T5,sizeof(T5));

            ChanIn(LINK2IN, (char*) &T,sizeof(T));

            ChanIn(LINK2IN, (char*) &T2,sizeof(T2));

            ProcToHigh();

     while (t<=T)

         {
```

```
SetTime(0);
i=i+1;
ChanIn(LINK0IN, (char*) &y41a,sizeof(y41a));
ChanIn(LINK0IN, (char*) &y42a,sizeof(y42a));
ChanIn(LINK0IN, (char*) &y5a,sizeof(y5a));
ChanIn(LINK1IN, (char*) &y36,sizeof(y36));
ChanIn(LINK1IN, (char*) &y34,sizeof(y34));
ChanIn(LINK1IN, (char*) &y13,sizeof(y13));
ChanIn(LINK1IN, (char*) &y12,sizeof(y12));
ChanOut(LINK2OUT, (char*) &y41a,sizeof(y41a));
ChanOut(LINK2OUT, (char*) &y42a,sizeof(y42a));
start = Time();
UT=y13-y36*Rs;
UT1=y12-y34*Rs;
S=(UT+UTa)*T2/2+Sa;
S1=(UT1+UT1a)*T2/2+S1a;
TM=3*(S1*y36-S*y34);
Sa=S;UTa=UT;UT1a=UT1;S1a=S1;
if (i<T5)
     {TR=0, Si=0,o=0,e=0,p=0;}
else {
      TR=3*M*6.83*y5a/0.585652-TM;
      Si=TR*4/(3*Np*M*(y41a*y41a-y42a*y42a));
      if (Si>0.75)
```

```
                {Si=0.75;}
              else {Si=Si;}
              o=0.5*asin(Si);
              e=0-o; p=0.2*e;
              R=0.45+p;              /* The controller may be PI or PID */
              ChanOut(LINK00UT, (char*) &R,sizeof(R));
              ChanOut(LINK20UT, (char*) &R,sizeof(R));
             }
        end = Time();
        z2=end-start;
        ChanOut(LINK20UT, (char*) &z2,sizeof(z2));
             t=t+T2;
    }
}
```

Appendix-E

This program is designed for the simulation of indirect field-oriented control of AC induction motor.

```
#include<stdio.h>
#include<math.h>
main( )
   {  FILE  *fp;
      int i, T3, T4, T5, T, Np=2, Kp=8, T6, T7;
      float  Rs=0.49, Ls=0.0388, Lr=0.0354, Lb=0.0021, M=0.0354,
             J=0.024, D=0.0011, R=0.45,n,
             Rr, t, Ws, Is, T2, TL, N1, N2, Kd, Kq, Ki, K1, K2,
             Ka, Q, y1, y2, y9, y14, y15, y16, y17, y18,y19, y20, y21,
             y22, y23, y24, y25, y26, y27, y28, y29, y30, y31, y32, y33,
             y39,y3[50010], y4[50010], y5[50010], y6[50010], y7[50010],
             y8[50010], y10[50010], y11[50010], y12[50010], y13[50010],
             y34[50010], y35[50010], y36[50010], UT1[50010],S1[50010],
             y37[50010], y38[50010], y40[50010], UT[50010], TM[50010],
             y41[50010], y42[50010], Ud[50010], Uq[50010], V[50010],
             U[50010], X[50010], Ta[50010], ph[50010], S[50010],
             R2[50010], R3[50010], R4[50010], Ia[50010];
      printf("Enter input T,T3,T4,T5,T6,T7\n");
      scanf("%d %d %d %d %d %d",&T,&T3,&T4,&T5,&T6,&T7);
```

```
printf("T=%d,T3=%d,T4=%d,T5=%d,T6=%d,T7=%d\n",T,T3,T4,T5,T6,T7);
printf("Enter input T2,TL,N1,N2\n");
scanf("%f %f %f %f",&T2,&TL,&N1,&N2);
printf("T2=%f,TL=%f,N1=%f,N2=%f\n",T2,TL,N1,N2);
i=1; Kd=0.8; Kq=0.6; Rr=0.45; Ki=0.003; Ka=0.01; K1=0.01;
K2=0.01;
Q=1.0/(Ls*Lr-M*M);
y14=0; y15=0; y16=0; y17=0; y24=0; y25=0; y26=0; y27=0; y28=0;
y29=0; y30=0; y31=0; y32=0; S[i]=0;S1[i]=0;R2[i]=0.45;
y3[i]=0; y4[i]=0; y5[i]=0; y6[i]=0; y7[i]=0; y8[i]=0; y10[i]=0;
y11[i]=0; y12[i]=0; y13[i]=0; y35[i]=0; y36[i]=0; y37[i]=0;
y38[i]=0; y40[i]=0; y41[i]=0; y42[i]=0; UT[i]=0; TM[i]=0;
if (!(fp=fopen("ua.d","wb")))
   {printf("cannot open file\n");
    exit(1);
    }

t=0;
while (t<=T)
   {  i=i+1;
      X[i]=t;
      if (i>T3)
        TL=2;
      else TL=0;
      if (i>T6)
```

```
  TL=4;
else TL=TL;
if (i>T4)
  Rr=0.08*(t-3)*(t-3)+0.125;
else Rr=Rr;
y1=N1;  y2=N2;
if (i>T7)
   y2=3;
else y2=N2;
y3[i]=y1-y42[i-1]*0.585652;
y4[i]=y2-y40[i-1]*0.027922;
y5[i]=Kp*(y4[i]-y4[i-1])+Kp*Ki*y4[i]+y5[i-1];
y6[i]=Kd*(y3[i]-y3[i-1])+Kd*K1*y3[i]+y6[i-1];
if (y5[i]>10)
    y5[i]=10;
else y5[i]=Kp*(y4[i]-y4[i-1])+Kp*Ki*y4[i]+y5[i-1];
if (y5[i]<-10)
    y5[i]=-10;
else y5[i]=Kp*(y4[i]-y4[i-1])+Kp*Ki*y4[i]+y5[i-1];
y7[i]=y5[i]-y41[i-1]*0.585652;
y8[i]=Kq*(y7[i]-y7[i-1])+Kq*K2*y7[i]+y8[i-1];
y9=y5[i]*R/(y1*Lr);
y10[i]=y9+y40[i-1];
y11[i]=(y10[i]+y10[i-1])*T2/2+y11[i-1];
```

```
Ud[i]=y6[i];
Uq[i]=y8[i];
y12[i]=(Ud[i]*cos(y11[i])-Uq[i]*sin(y11[i]))*30;
y13[i]=(Ud[i]*sin(y11[i])+Uq[i]*cos(y11[i]))*30;
y18=y12[i]-Rs*(Lr*y14-M*y15)*Q;
y19=(-1)*Rr*(Ls*y15-M*y14)*Q-y40[i-1]*y17;
y20=y13[i]-Rs*(Lr*y16-M*y17)*Q;
y21=(-1)*Rr*(Ls*y17-M*y16)*Q+y40[i-1]*y15;
y22=y14+y18*T2;
y23=y15+y19*T2;
y24=y16+y20*T2;
y25=y17+y21*T2;
y26=y12[i]-Rs*(Lr*y22-M*y23)*Q;
y27=(-1)*Rr*(Ls*y23-M*y22)*Q-y40[i-1]*y25;
y28=y13[i]-Rs*(Lr*y24-M*y25)*Q;
y29=(-1)*Rr*(Ls*y25-M*y24)*Q+y40[i-1]*y23;
y30=y14+T2*(y18+y26)/2;
y31=y15+T2*(y19+y27)/2;
y32=y16+T2*(y20+y28)/2;
y33=y17+T2*(y21+y29)/2;
y14=y30; y15=y31; y16=y32; y17=y33;
y34[i]=(Lr*y30-M*y31)*Q;
y35[i]=(Ls*y31-M*y30)*Q;
y36[i]=(Lr*y32-M*y33)*Q;
```

```
y37[i]=(Ls*y33-M*y32)*Q;
n=sqrt(y31*y31+y33*y33);
y38[i]=3*M*(y36[i]*y35[i]-y34[i]*y37[i]);
y39[i]=(y38[i]-TL)*Np;
y40[i]=T2*y39/(T2*D+J)+J*y40[i-1]/(T2*D+J);
y41[i]=y36[i]*cos(y11[i])-y34[i]*sin(y11[i]);
y42[i]=y34[i]*cos(y11[i])+y36[i]*sin(y11[i]);
UT[i]=y13[i]-y36[i]*Rs;
UT1[i]=y12[i]-y34[i]*Rs;
S[i]=(UT[i]+UT[i-1])*T2/2+S[i-1];
S1[i]=(UT1[i]+UT1[i-1])*T2/2+S1[i-1];
TM[i]=3*(S1[i]*y36[i]-S[i]*y34[i]);
if (i<T5)
   {R2[i]=0.45; Ws=0; Is=0; U[i]=0; Ia[i]=0; R3[i]=0; R4[i]=0;}
else {Ws=sqrt(y9*y9);
      Is=y41[i]*y41[i]+6.83*6.83;
      U[i]=y13[i]*y34[i]-y12[i]*y36[i];
R2[i]=Ws*sqrt(Lr*y10[i]*0.0013*Is/(y10[i]*Ls*Is-U[i])-Lr*Lr);
if (R2[i]<0.1)
   {R=0.1;}
else
R=R2[i];
      }
fprintf(fp,"%f  %f  %f  %f  %f  %f  %f\n",y40[i],y38[i],y41[i],
```

```
                    y42[i],y34[i],n,R2[i]);
            t=t+T2;
        }
    }
```