THE UNIVERSITY OF CALGARY

ROBUST CONTROLLED FLUX ESTIMATION FOR INDIRECT FIELD ORIENTED CONTROLLED INDUCTION MOTOR DRIVES

BY

Neil M. Cumbria

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR

THE DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

CALGARY, ALBERTA

April, 1998

Abstract

Field oriented control (FOC) is used to achieve high dynamic performance in inverter-fed induction motor drives. It is necessary, for field oriented control, to know the instantaneous magnitude and position of the rotor flux. These values are approximated based on flux measurements in direct FOC, and estimated in indirect FOC. This thesis presents the design of a novel robust flux controller and, subsequently, a robust speed controller for the purpose of indirect field oriented controlled flux estimation for an induction motor drive. The controllers are designed in terms of stability and performance criteria through a graphical technique, called loopshaping, taking into account the effects of external disturbances and parameter variations, including time delay, rotor resistance variances and inertial deviation from the nominal model. The purpose of these controllers is to provide good tracking and performance despite the uncertainties present in the drive system. The system's ability to accurately estimate the flux magnitude and field angle is verified through a parameter sensitivity study, which indicates that that the robust controllers are quite insensitive to rotor resistance variations (maximum absolute error in the flux magnitude is 0.01 p.u. if the rotor resistance is increased to twice the nominal value). This preliminary work indicates that the robust controllers provide a practical alternative to other control methods.

Acknowledgments

The completion of this work represents the fulfillment of a dream; one that could not have been accomplished if not for the wisdom and support of a number of people.

My supervisor, Dr. E. P. Nowicki, must especially be acknowledged for his guidance, understanding, and patience. His constant encouragement and faith in my abilities were paramount throughout the scope of this work.

My fellow students, Mr. Mustafa Mohamadian, and Mr. Alfred Chu, are also recognized for their invaluable suggestions and insights into problems related to this project.

Many thanks are extended to the professors and supporting staff, especially the secretaries in the Department of Electrical and Computer Engineering at the University of Calgary, for all their help during my course of study.

I am deeply grateful to all my dear friends who have helped me in so many ways during this undertaking.

Finally, thanks must go out to my parents, Joanne and Walter Cumbria, and my brother, Marko, for their tireless support and constant encouragement in all my endeavors. Their knowledge, sacrifice and love, has helped me achieve all my goals to date.

To my family

TABLE OF CONTENTS

Appendix A

Appendix B

List of Figures

Chapter 1

Introduction

Field oriented control (FOC) was originally developed in the early 1970's for the purpose of high dynamic performance induction motor control. Due to the FOC linearization technique, an induction motor drive gains the capability of performance previously attainable only with DC motors used in servo drive applications. It is often desirable, in field oriented control, for the rotor flux to reach its maximum value as quickly as possible and maintain that magnitude throughout the normal operating range of the induction motor. This is done by taking advantage of the relationship between the shaft torque and the motor voltages or currents.

Induction motors have gained popularity in industrial applications due to their advantage over other electric machines in terms of structural simplicity, ease of maintenance, ruggedness, and durability. Unlike the DC motor, the squirrel cage induction motor operates without the need for carbon brushes. Instead, as the stator magnetic field rotates, current is induced into the rotor windings. However, due to the fact that the windings are made up of conducting bars embedded in slots in the rotor iron and short-circuited at each end by conducting end-rings (often called a squirrel cage), they are inaccessible to the extraction flux information needed for FOC.

Two primary techniques have been developed in order to alleviate this problem. Direct FOC, which is based on the direct measurement of the air-gap flux; and indirect FOC, in which the desired quantities are estimated, rather than measured with sensors. The various existing methods of indirect FOC share the need of a relationship between stator currents, stator voltages and rotor speed, in some combination, in order to determine the magnitude and angle of the rotor flux.

In this thesis, indirect FOC is used to estimate various rotor flux quantities. At the same time, robust controllers are employed in order to take into account, and compensate for, any plant parameter deviation and disturbance, thereby effectively reducing performance degradation during such deviation from, or disruption with respect to, ideal performance.

1.1. Field Oriented Control

Field orientation implies that the stator current is oriented with respect to the rotor flux so as to attain independently controlled flux and torque. This decoupling of the current components supplied to the machine can be derived from the *d-q* axis theory [1-4], in which the time dependence of some quantities is removed for steady-state operation (i.e. sinusoidal quantities become DC quantities) and variables are expressed in orthogonal or mutually decoupled direct (*d*) and quadrature (*q*) components.

It is possible to represent the *d-q* model in either a stationary or rotating reference frame. As the term implies, in the stationary reference frame, the direct and quadrature axes are fixed on the stator itself. In the rotor reference frame, the axes are fixed on the rotor. In the synchronously rotating reference frame, the axes rotate at the synchronous speed of the motor and are fixed on the stator magnetic field.

Various superscripts are used to differentiate between the reference frames; in standard notation, d^s and q^s denote the stationary reference frame; d^e and q^e denote the synchronous rotating reference frame; and no superscripts are used for the field oriented reference frame, whereby the synchronously rotating reference frame is fixed in relation to the rotor flux. If machine operation is considered in the synchronously rotating reference frame, sinusoidal variables of other frames appear as DC quantities in this frame for steady state conditions [5].

In the early 1970's, Blaschke developed the concept of field orientation [5, 6]. He realized that the angular position of the rotating magnetic field within the rotor of the induction motor was essential for control. He concluded that if the position of this magnetic field could be measured, it would be possible to induce current into the rotor at the point where the field strength has its maximum. This way, the motor could always deliver full torque, *even under transient conditions*. However, because the magnetic field rotates, the current-carrying rotor winding cannot be constantly located where the magnetic field has its maximum. An impossible, hypothetical solution is to rotate the entire stator in order to let it follow the rotation of the magnetic field, thereby allowing the stator winding to always induce current into the correct position in the rotor.

Blaschke, together with Hasse [7], found a practical solution. Instead of rotating the entire stator, they "rotated" the control signals to the stator (made possible by the vector multiplication of a standard rotation matrix). The flux components of the stator currents are oriented in phase with the total rotor flux linkage of the motor, by fixing the *d*-axis in the rotor synchronous reference frame. Thus, the magnetization of the motor lies directly along the *d*-axis (hence the name). The torque components of the stator currents, meanwhile, are oriented in quadrature (i.e. along the *q*-axis) to the rotor flux linkage, indicating that the magnetization can be maintained constant, while torque may be independently controlled in proportion to the stator current component along the *q*-axis [5-7].

Therefore, field oriented control effectively "converts" the induction motor into a DC motor, that is, permitting independent control of flux and torque. The induction motor will behave in a manner practically identical to a current controlled (or voltage controlled) DC motor [8]. This means that, with field oriented control, it is possible to have the performance of a DC motor in a simple, rugged squirrel cage induction motor.

Of the two basic kinds of FOC, indirect is preferred in practical applications [6]. For indirect FOC, the necessary flux information is estimated, rather than directly measured, as in direct FOC. The advantage to this is that it negates the requirement of an externally mounted device, which, in general, can be subject to severe thermal and mechanical stresses (i.e. Hall sensors) or that flux cannot be sensed at zero speed (i.e. flux sensing coils) [3, 4, 8].

1.2. Robust Control

High disturbance rejection and good command tracking are basic requirements of any high performance drive systems, which are subject to some types of extraneous disturbances and parameter variations during operation. These perturbations typically consist of differences in the dynamics of the model and the actual process, including various external load uncertainties, and plant parameter deviation (e.g. variation of the rotor time constant in an induction machine).

The rotor time constant of an induction motor is dependent on rotor resistance, which tends to vary significantly due to temperature variations and skin effect. This influences the accuracy of the flux magnitude and angle estimation of indirect FOC, leading to a degradation in system performance and quality of control. With the inclusion of robust controllers in the system modeling, it is proposed that rotor resistance perturbations may be compensated for.

In recent years, several advances have been made to reduce the hardware requirements of a field oriented control system [9, 10], and to improve the accuracy in speed and/or position control under a range of operating conditions through the use of robust control methods [11-13], thereby attempting to circumvent problems of plant uncertainties. However, the application of robust control theory to flux control has not been attempted in the past.

Following a commonly used graphical method, called the loopshaping technique [14-16], a novel robust flux controller is proposed, and subsequently, it is proposed that a

robust speed controller be designed in terms of high performance criteria, taking into account the effects of external disturbances and parameter variances, including time delay, rotor resistance variations, and inertial deviation of the plant from the nominal case.

1.3. Thesis Overview/Outline

Chapter 2 serves as a review of the history and basic concepts of field oriented control (including *d-q* axis theory, and matrix transforms), and robust control (including the loopshaping technique).

Presented in Chapter 3 is the implementation of a field oriented control induction motor drive system, utilizing the robust controllers presented in Chapter 2.

Chapter 4 consists of the results of the flux estimator tests, comprising of parameter variation tests to examine the robustness of the estimator and comparisons with alternative control schemes.

A discussion of the results is presented in Chapter 5, followed by conclusions and suggestions for future work in Chapter 6.

Chapter 2

2.1. Field Oriented Control and Flux Estimation

2.1.1. Field Oriented Control Theory

The goal of field oriented control is to control an AC induction motor like a separately excited DC motor, in order to provide a controlled torque over a wide range of operating conditions [3, 4]. This allows for good performance near zero speed, and for the motor flux to be controlled and maintained at its optimum level so as to attain a fast speed response.

In literature, the terms "vector control" and "field oriented control" are used interchangeably. In a DC machine, torque control in an induction machine is achieved by controlling the motor armature current. The induction motor requires external control of the field flux and armature mmf spatial orientation, unlike the DC motor where the field flux and armature mmf orientation is fixed by the commutator and brushes. It is this control that prevents the space angles between the various fields in an AC motor from varying with load (and during transients), which further yields complex interactions and oscillatory dynamic response [4]. Directly controlling these space angles has come to be known as "vector control" or "field oriented control". In this thesis, systems that attempt

to produce a 90° space angle between specifically chosen field components, are referred to as being "field oriented", in order to emulate the behavior of a DC motor.

Field orientation can be best explained by first reviewing the principles of DC motor operation. The basic structure of a DC motor consists of a stationary field structure, which uses either a DC excited winding or permanent magnets, and a rotating armature winding supplied through a commutator and brushes. This structure, along with the motor equivalent circuit, and resulting orientation of the armature mmf and the field flux, is shown in Fig. 2.1.

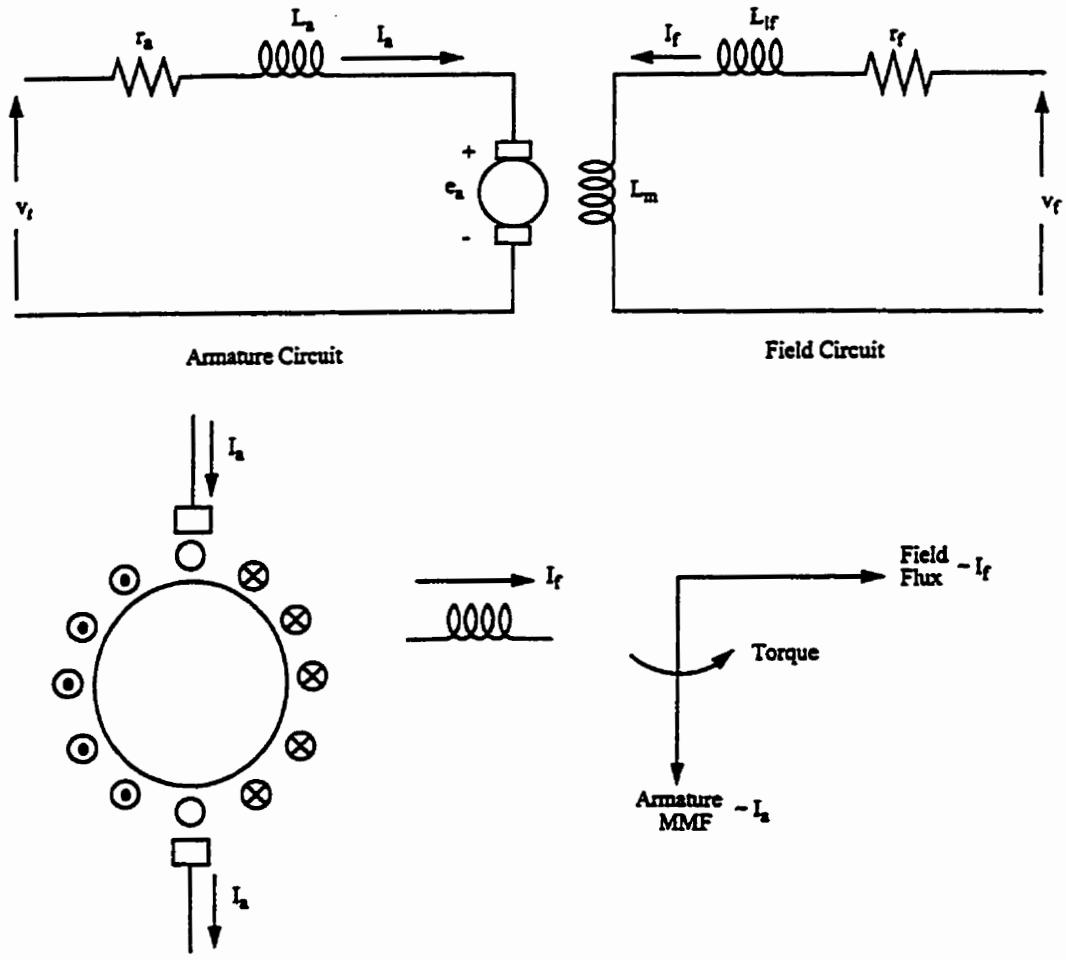


Figure 2.1. DC machine model

As can be seen from Fig. 2.1, the flux and mmf are maintained in a mutually perpendicular orientation independent of rotor speed, resulting in the field flux being unaffected by the armature current and maximum possible torque. The electromagnetic torque, neglecting the armature demagnetization effect and field saturation, is given by

$$T_e = K_t \Psi_g I_a = K_t' I_a I_f \qquad (2.1)$$

where K_t, K_t' = proportionality constants

Ψ_g =airgap flux

I_a =armature current (torque component)

I_f =field current (flux component)

The control variables I_a and I_f are considered as being orthogonal or decoupled vectors. Under normal operating conditions, torque is varied by changing the armature current, and the field current is set to maintain the rated field flux. Since the stator current components responsible for rotor flux and torque production are decoupled, the torque sensitivity remains maximum in both transient and steady-state operation. This DC motor control model becomes valid for induction motor torque control if machine operation is considered in a synchronously rotating reference frame where the sinusoidal variables appear as DC quantities.

The principle of field oriented control can be explained through the use of a phasor diagram [2, 17]. Figure 2.2 shows the equivalent circuit of the induction motor and the corresponding phasor diagram. The equivalent circuit uses only one of the three identical motor phases, as the only difference between them is a 120 degrees phase shift.

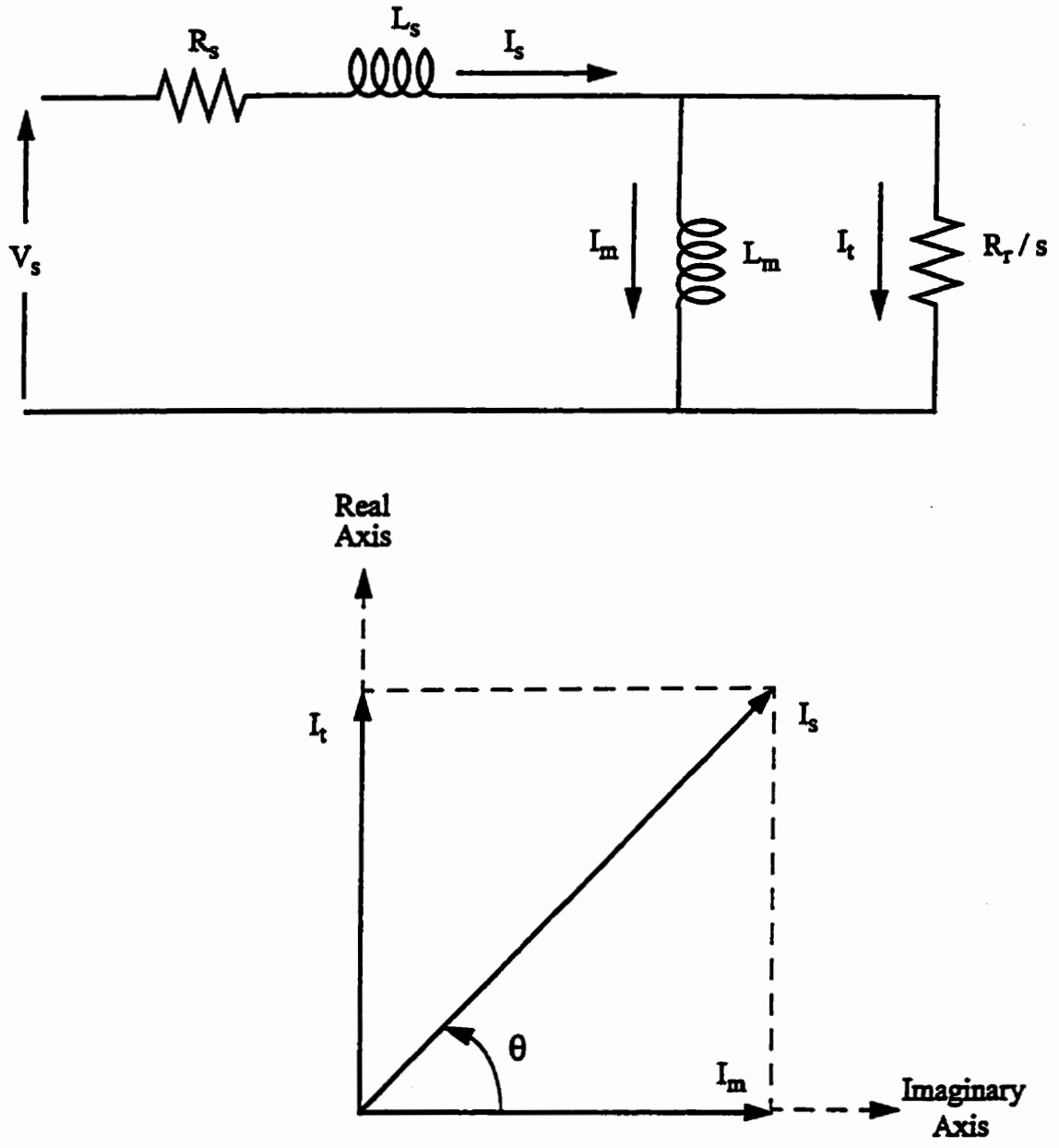


Figure 2.2. Per-phase equivalent circuit and phasor diagram for the induction motor

Neglecting rotor leakage inductance [3], the primary (or stator) current I_s splits into the magnetizing current I_m and the secondary (or torque) current I_t. The magnetizing current induces the rotor field through the inductance L_m. Secondary current I_t produces the motor torque. The amount of secondary current (and torque) is dependent on the secondary resistance and the slip frequency R_2/s.

In the complex plane

$$I_m = I_s \cos\theta \tag{2.2}$$

$$I_t = I_s \sin\theta \tag{2.3}$$

where θ = phase angle.

The torque as a function of primary current I_s is nonlinear, making direct control of the motor torque impossible [17]. However, field oriented control theory allows for the establishment of a control law such that I_m is maintained constant, thereby making the rotor flux constant. The secondary current, I_t, meanwhile, is controlled in proportion to the amount of torque required. Thus, referring to Fig. 2.2 and Eqn. (2.3), both the stator current I_s and the phase angle θ will increase or decrease given a large or small torque, respectively.

This relationship is quantified by

$$I_s = \sqrt{I_m^2 + I_t^2} \tag{2.4}$$

$$\theta = \tan^{-1}\left[\frac{I_t}{I_m}\right] \tag{2.5}$$

The torque is given by

$$T_e = K_t \Psi_m I_t = K_t' I_m I_t \tag{2.6}$$

where K_t, K_t' = proportionality constants

Ψ_m =airgap flux

From the equivalent circuit in Fig. 2.2,

$$L_m I_m \omega_e = \frac{R_r}{s} I_t \qquad (2.7)$$

where $\omega_e = 2\pi f_e$

f_e = stator excitation frequency

s = per unit slip (i.e. synchronous speed of the stator field minus steady speed of the rotor, divided by the synchronous speed of the stator field)

The angular slip is then given by

$$\omega_{sl} = \frac{R_r}{L_m}\frac{I_t}{I_m} = \omega_e s \qquad (2.8)$$

The three key variables in field oriented control are the stator excitation frequency (i.e. synchronous frequency), the magnitude, and the phase angle, of the primary current. The aim of a drive system utilizing FOC , then, is to satisfy Eqns. (2.4), (2.5), and (2.8), at least implicitly. Note that in doing so, not only are the magnetizing and torque producing currents, I_m and I_t respectively, orthogonal phasors (as in Fig. 2.2), but also orthogonal *space vectors*, just as is the case for the DC motor (as in Fig. 2.1).

In Fig. 2.3, field orientation is illustrated with the control inputs i_{qs}^* and i_{ds}^*. The currents are expressed in orthogonal direct-axis and quadrature-axis components, i_{ds} and i_{qs}, respectively, where both are in a synchronously rotating reference frame. When comparing with the DC motor, i_{ds} is analogous to the field current I_f, and i_{qs} is

analogous to the armature current I_a. Thus, the torque of the induction motor in the synchronously rotating reference frame may be

expressed as

$$T_e = K_t \Psi_m \, i_{qs} = K_t' \, i_{qs} \, i_{ds} \qquad (2.9)$$

where K_t, K_t' = proportionality constants

Ψ_m = airgap flux

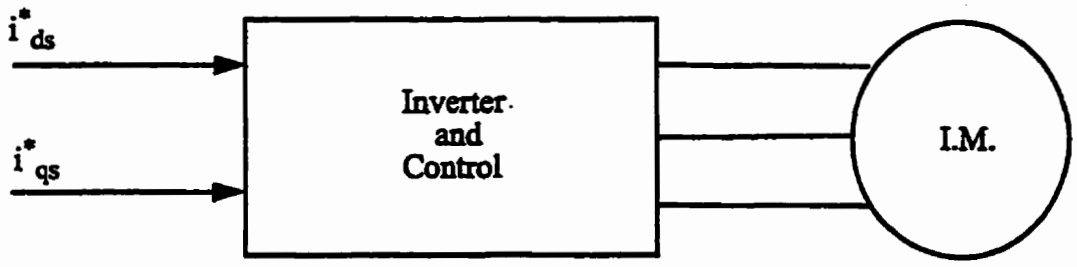


Figure 2.3. FOC induction motor

Adjustable speed operation can be accounted for by the use of an element within a feedback loop [4], so dynamic behavior of the induction machine has to be taken into consideration. Due to the coupling effect between the stator and rotor phases, the dynamic performance is a complex one, whereby the coupling coefficients tend to vary with rotor position. It is possible to describe the machine in terms of differential equations with time-varying coefficients.

2.1.2. Variable Transformations

The *d-q* axis theory is generally used for dynamic modeling of an induction motor, whereby variables and parameters are expressed in orthogonal or mutually decoupled direct (*d*) and quadrature (*q*) components [1]. This model may be formulated in one of several reference frames, either stationary or rotating. In a stationary reference frame, the direct and quadrature axes are fixed on the stator, while in a rotating reference frame, the axes can be fixed on the rotor or can be rotating at synchronous speed (referenced to the stator or rotor flux).

For the stationary reference frame, the *d-q* axes are denoted by d^s and q^s, where the *s* superscript denotes variables and transformations associated with circuits which are stationary, as opposed to rotor circuits which are free to rotate. The synchronously rotating reference frame, whereby the *d-q* axes are denoted by d^e and q^e, can be defined as the reference frame rotating at the electrical angular velocity corresponding to the fundamental frequency of the variables associated with the stationary circuits (denoted by ω_e). For AC machines, ω_e is the electrical angular velocity of the air gap rotating magnetic field established by stator currents of fundamental frequency. For the field oriented reference frame, in which the synchronously rotating reference frame is fixed in relation to the rotor flux, no superscripts are used. The advantage of considering machine operation in the synchronously rotating reference frame, or the field oriented reference frame, is that sinusoidal variables of other frames are transformed into DC quantities in this frame for steady-state operation.

Figure 2.4 can be used to explain the axiom of variable transformation. This diagram shows three physical phases a_s, b_s, c_s (fixed relative to the stator datum), the stationary reference frame axes d^s and q^s and the synchronously rotating reference frame axes d^e and q^e. The arbitrary angle x lies between the a_s-axis and the stator datum (i.e. the q^s-axis).

The three-phase to two-phase transformation of the stator voltages is given by

$$V_{s_{d^s q^s}} = T\, V_{s_{ph}} \tag{2.10}$$

where $V_{s_{d^s q^s}} = [v^s_{qs},\ v^s_{ds}]^T$ (2.11)

$$V_{s_{ph}} = [v_{as},\ v_{bs},\ v_{cs}]^T \tag{2.12}$$

$$T = \frac{2}{3}\begin{bmatrix} \cos(x) & \cos(x-\frac{2\pi}{3}) & \cos(x-\frac{4\pi}{3}) \\ \sin(x) & \sin(x-\frac{2\pi}{3}) & \sin(x-\frac{4\pi}{3}) \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \tag{2.13}$$

and conversely

$$V_{s_{ph}} = T^{-1} V_{s_{d^s q^s}} \tag{2.14}$$

where

$$T^{-1} = \begin{bmatrix} \cos(x) & \sin(x) & 1 \\ \cos(x-\frac{2\pi}{3}) & \sin(x-\frac{2\pi}{3}) & 1 \\ \cos(x-\frac{4\pi}{3}) & \sin(x-\frac{4\pi}{3}) & 1 \end{bmatrix} \tag{2.15}$$

(A similar type of operation can be done for the stator currents.)

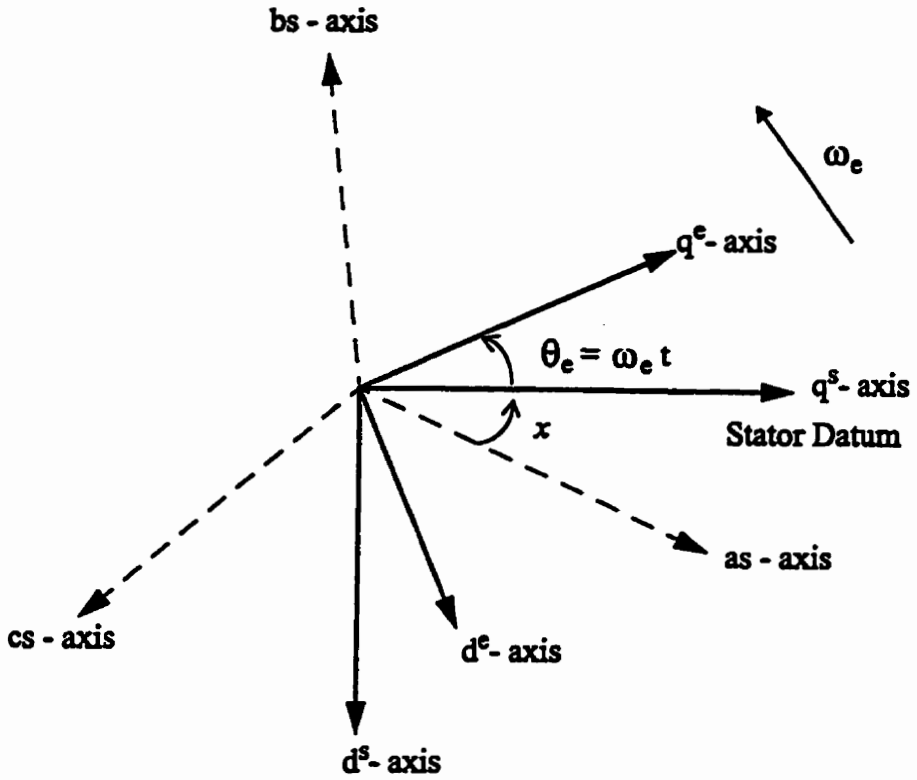


Figure 2.4. Axes transformations

Setting $x = 0$ results in the q^s-axis coinciding with the a_s-axis. The transform matrix then becomes

$$T = \frac{2}{3}\begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & -\frac{\sqrt{3}}{2} & \frac{\sqrt{3}}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \qquad (2.16)$$

and

$$T^{-1} = \begin{bmatrix} 1 & 0 & 1 \\ -\frac{1}{2} & -\frac{\sqrt{3}}{2} & 1 \\ -\frac{1}{2} & \frac{\sqrt{3}}{2} & 1 \end{bmatrix} \qquad (2.17)$$

The stationary reference frame to rotating reference frame transformation is given by

$$V_{s_{dq}} = U V_{s_{d^s q^s}} \tag{2.18}$$

where $V_{s_{dq}} = [v_{qs}, v_{ds}]^T$ (2.19)

$$V_{s_{d^s q^s}} = [v^s_{qs}, v^s_{ds}]^T \tag{2.20}$$

$$U = \begin{bmatrix} \cos\theta_e & -\sin\theta_e \\ \sin\theta_e & \cos\theta_e \end{bmatrix} \tag{2.21}$$

and conversely

$$V_{s_{d^s q^s}} = U^{-1} V_{s_{dq}} \tag{2.22}$$

where $U^{-1} = \begin{bmatrix} \cos\theta_e & \sin\theta_e \\ -\sin\theta_e & \cos\theta_e \end{bmatrix}$ (2.23)

and $\theta_e = \omega_e t$ (i.e. angle of the stator synchronously rotating reference frame).

For a balanced, fixed voltage three-phase supply let

$$v_{as} = V_{sm} \cos(\omega_e t) \tag{2.24}$$

$$v_{bs} = V_{sm} \cos(\omega_e t - \tfrac{2\pi}{3}) \tag{2.25}$$

$$v_{cs} = V_{sm} \cos(\omega_e t + \tfrac{2\pi}{3}) \tag{2.26}$$

where V_{sm} = stator peak voltage.

If it is further assumed that the phase voltages are balanced and sinusoidal, then simplifying Eqns. (2.10) and (2.14) in the synchronously rotating reference frame yields

$$v_{as} = v_{qs}^{s} \tag{2.27}$$

$$v_{bs} = -\frac{1}{2}v_{qs}^{s} - \frac{\sqrt{3}}{2}v_{ds}^{s} \tag{2.28}$$

$$v_{cs} = -\frac{1}{2}v_{qs}^{s} + \frac{\sqrt{3}}{2}v_{ds}^{s} \tag{2.29}$$

and

$$v_{qs}^{s} = \frac{2}{3}v_{as} - \frac{1}{3}v_{bs} - \frac{1}{3}v_{cs} = v_{as} \tag{2.30}$$

$$v_{ds}^{s} = -\frac{1}{\sqrt{3}}v_{bs} + \frac{1}{3}v_{cs} \tag{2.31}$$

(zero components in T and T^{-1} have been dropped.)

Substituting Eqns. (2.24), (2.25), and (2.26) into (2.30) and (2.31) yields

$$v_{qs}^{s} = V_{sm}\cos(\omega_e t) \tag{2.32}$$

$$v_{ds}^{s} = -V_{sm}\sin(\omega_e t) \tag{2.33}$$

Then, substituting Eqns. (2.32) and (2.33) into (2.18) results in

$$v_{qs}^{e} = V_{sm} \tag{2.34}$$

$$v_{ds}^{e} = 0 \tag{2.35}$$

It can be seen that the sinusoidal variables appear as DC quantities in the stator synchronously rotating reference frame for steady-state operation. This is the main advantage of using the *d-q* axis theory.

2.1.3. Synchronously Rotating Reference Frame Model

The *d-q* equivalent circuit is shown in Fig. 2.5, from which the stator voltages in the stator synchronously rotating reference frame are given by

$$v_{qs}^{e} = R_s\, i_{qs}^{e} + s\Psi_{qs}^{e} + \omega_e\, \Psi_{ds}^{e} \qquad (2.36)$$

$$v_{ds}^{e} = R_s\, i_{ds}^{e} + s\Psi_{ds}^{e} - \omega_e\, \Psi_{qs}^{e} \qquad (2.37)$$

where s = differential operator

v_{qs}^{e} = instantaneous q^e-axis stator voltage

v_{ds}^{e} = instantaneous d^e-axis stator voltage

R_s = stator resistance

i_{qs}^{e} = instantaneous q^e-axis stator current

i_{ds}^{e} = instantaneous d^e-axis stator current

ω_e = stator frequency (rad/s)

Ψ_{qs}^{e} = q^e-axis stator flux linkage

Ψ_{ds}^{e} = d^e-axis stator flux linkage

If the rotor moves at a speed of ω_r, the *d-q* axes fixed on the rotor move at a speed of $(\omega_e - \omega_r)$ relative to the synchronously rotating reference frame. Then, the rotor voltages can be expressed as

$$v_{qr}^{e} = R_r\, i_{qr}^{e} + s\Psi_{qr}^{e} + (\omega_e - \omega_r)\, \Psi_{dr}^{e} \qquad (2.38)$$

$$v_{dr}^{e} = R_r\, i_{dr}^{e} + s\Psi_{dr}^{e} - (\omega_e - \omega_r)\, \Psi_{qr}^{e} \qquad (2.39)$$

where v^e_{qr} = instantaneous q^e-axis rotor voltage

v^e_{dr} = instantaneous d^e-axis rotor voltage

R_r = rotor resistance

i^e_{qr} = instantaneous q^e-axis rotor current

i^e_{dr} = instantaneous d^e-axis rotor current

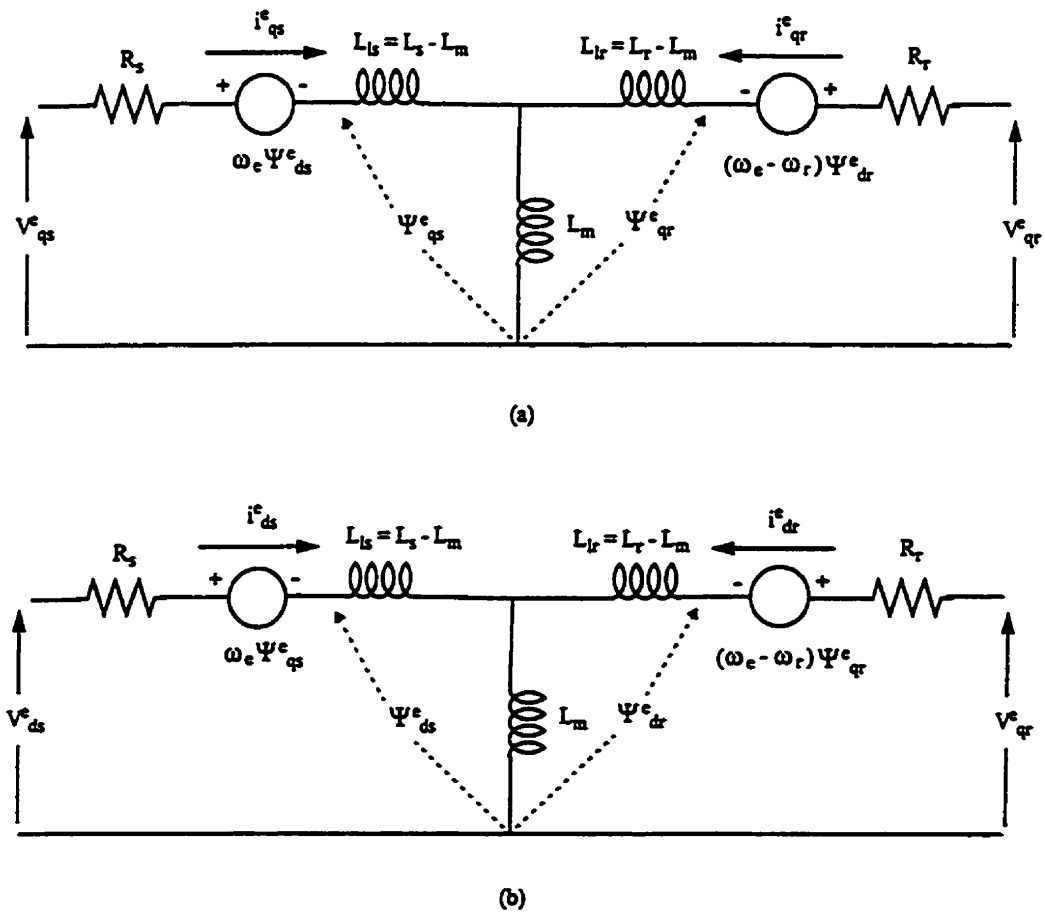


Figure 2.5. *d-q* equivalent circuits in the synchronously rotating reference frame:

(a) q^e- axis circuit, and (b) d^e- axis circuit

It should be noted that for a squirrel cage induction motor or single fed machine, v^e_{qr} and v^e_{dr} are equal to zero.

The various flux linkages are given by

$$\Psi^e_{qs} = L_{ls}\, i^e_{qs} + L_m\,(i^e_{qs} + i^e_{qr}) \qquad (2.40)$$

$$\Psi^e_{qr} = L_{lr}\, i^e_{qr} + L_m\,(i^e_{qs} + i^e_{qr}) \qquad (2.41)$$

$$\Psi^e_{ds} = L_{ls}\, i^e_{ds} + L_m\,(i^e_{ds} + i^e_{dr}) \qquad (2.42)$$

$$\Psi^e_{dr} = L_{lr}\, i^e_{dr} + L_m\,(i^e_{ds} + i^e_{dr}) \qquad (2.43)$$

where L_{ls} = stator leakage inductance

L_{lr} = rotor leakage inductance

L_m = magnetizing inductance

The model of electrical dynamics, in terms of currents and voltages, can be rewritten in matrix form as

$$\begin{bmatrix} v^e_{qs} \\ v^e_{ds} \\ v^e_{qr} \\ v^e_{dr} \end{bmatrix} = \begin{bmatrix} R_s + sL_s & \omega_e L_s & sL_m & \omega_e L_m \\ -\omega_e L_s & R_s + sL_s & -\omega_e L_m & sL_m \\ sL_m & (\omega_e - \omega_r)L_m & R_r + sL_r & (\omega_e - \omega_r)L_r \\ -(\omega_e - \omega_r)L_m & sL_m & -(\omega_e - \omega_r)L_r & R_r + sL_r \end{bmatrix} \begin{bmatrix} i^e_{qs} \\ i^e_{ds} \\ i^e_{qr} \\ i^e_{dr} \end{bmatrix} \qquad (2.44)$$

where s = Laplace operator

L_s = stator inductance

L_r = rotor inductance

During steady-state, all *s*-related terms in the above matrix become zero, and all variables in the synchronously rotating reference frame appear as DC quantities with sinusoidal excitation. Electrical rotor speed relates to torque by the following expression

$$T_e - T_L = J s \omega_m - \frac{2}{P} J s \omega_r \qquad (2.45)$$

where T_L = load torque

J = system inertia

P = number of poles

s = differential operator

The electromagnetic torque is then given by

$$T_e = \frac{3}{2}\frac{P}{2}\frac{L_m}{L_r}(\Psi^e_{dr}\, i^e_{qs} - \Psi^e_{qr}\, i^e_{ds}) \qquad (2.46)$$

$$= \frac{3}{2}\frac{P}{2} L_m (i^e_{dr}\, i^e_{qs} - i^e_{qr}\, i^e_{ds}) \qquad (2.47)$$

Eqns. (2.44), (2.45), and (2.47) together represent the complete model of the electromechanical dynamics of an induction machine.

2.1.4. Direct Field Oriented Control

Two primary methods exist in order to obtain the magnitude and position of the rotor flux; direct and indirect field oriented control.

In direct field orientation, the position of the flux to which orientation is desired is directly measured using sensing devices or estimated from terminal measurements.

Since rotor flux cannot be sensed directly, a rotor flux-oriented system requires some computation to obtain the desired information from a directly sensed signal. Two of the most widely known sensing methods are the Hall sensing method and the flux coils method.

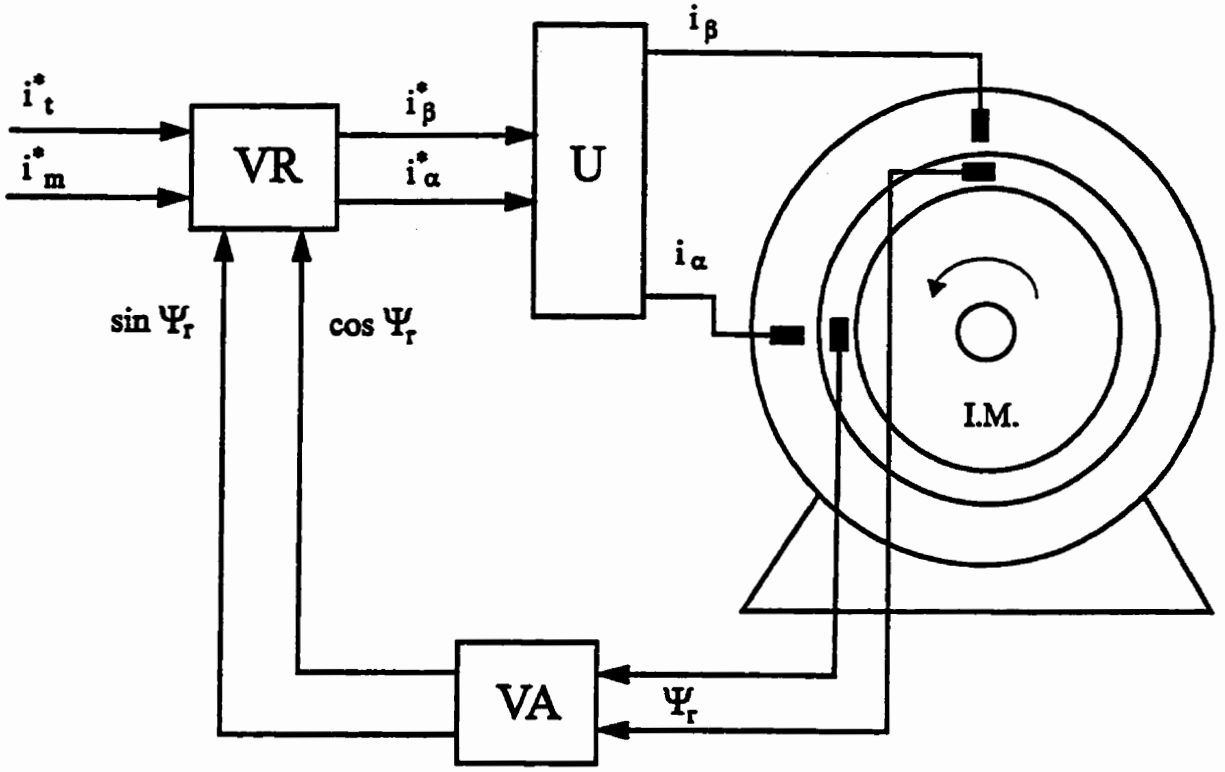


Figure 2.6. Direct field oriented control

In the Hall sensing method, shown in Fig. 2.6, two sensors (i.e. Hall sensors) are placed in the air gap of the stator and measure the angular position of the magnetic field. The signals are fed into a "vector analyzer" (VA), which filters the signal and adjusts the amplitude. The sine and cosine of the magnetic field angle come from the VA and are fed into the "vector rotator" (VR). The control signals for the magnetizing current I_m and the rotor current I_t (shown in the upper left part of the figure) together define a control

vector. The vector is fed into the VR, which rotates it to an angle where I_m and I_t are fed into the right angular positions in the motor.

Because the currents are fed into the stator windings, the motor must be controlled in stator coordinates. Since the stator reference frame is related to the stator, it is stationary. However, the coordinate system related to the magnetic field rotates (i.e. follows the rotation of the magnetic field). Following the rotation and viewing the motor from the field oriented reference frame, it behaves like a DC motor as it appears to have a stationary magnetic field vector and stationary current vectors (i.e. DC currents). Thus the motor can be controlled by DC currents, if the control signals are generated in field coordinates [8]. In Fig. 2.6, there are field coordinates to the left of the VR and stator coordinates to the right. The control signals are created in field coordinates, consequently as DC currents, and then transformed to stator coordinates.

A disadvantage with Hall sensors is that they are subject to severe thermal and mechanical stress due to their semiconductor nature, resulting in output error which is difficult to compensate.

In the flux coils method, flux coils are used to sense induced voltage (which is proportional to the flux change), followed by integrators to calculate the main flux of the motor. Flux coils avoid the problem inherent in Hall sensors by having no active semiconductor components. However, control problems can arise because most flux coils are mounted on the stator, and not the rotor. As a result, these sensors monitor the stator flux in a stationary reference frame, and not the rotor flux which is used in the decoupling scheme. The rotor flux then must be derived from the flux sensor

measurements via flux linkage equations. This leads to the introduction of detuning errors in the system dynamics [15]. As well, flux coils can only give satisfactory results at frequencies above 5 to 10 Hz. Therefore, this technique is unsuitable for position control, as the flux cannot be sensed at zero speed.

A problem with most direct orientation schemes is that they require modifications to be made to the motor (i.e. mounting of external devices) making them expensive and impractical outside the laboratory setting.

2.1.5. Indirect Field Oriented Control

An alternative to direct sensing of the flux position is presented in indirect field orientation, which can employ the slip relation to estimate the flux position to the rotor. There are no sensing devices placed inside the motor, meaning there is no direct measurement of the magnetic field. Instead, the rotor speed (i.e. rotor frequency) is measured and slip frequency is calculated. Addition of these frequencies yields an optimal stator frequency for motor control. This scheme is shown in Fig. 2.7. A sensor on the motor shaft measures the rotor angle θ_r(or measures the rotor speed ω_r, followed by an integrator for calculation of the angle). The input signals for current control are used for calculation of the desired slip frequency, ω_{sl}, which is integrated, giving a slip angle, θ_{sl}, which is added to the rotor angle. (The slip angle is required to adjust the inclination of the d-axis so that the magnetization of the motor is along this axis.) The sum of the two angles gives the instantaneous rotor flux position angle.

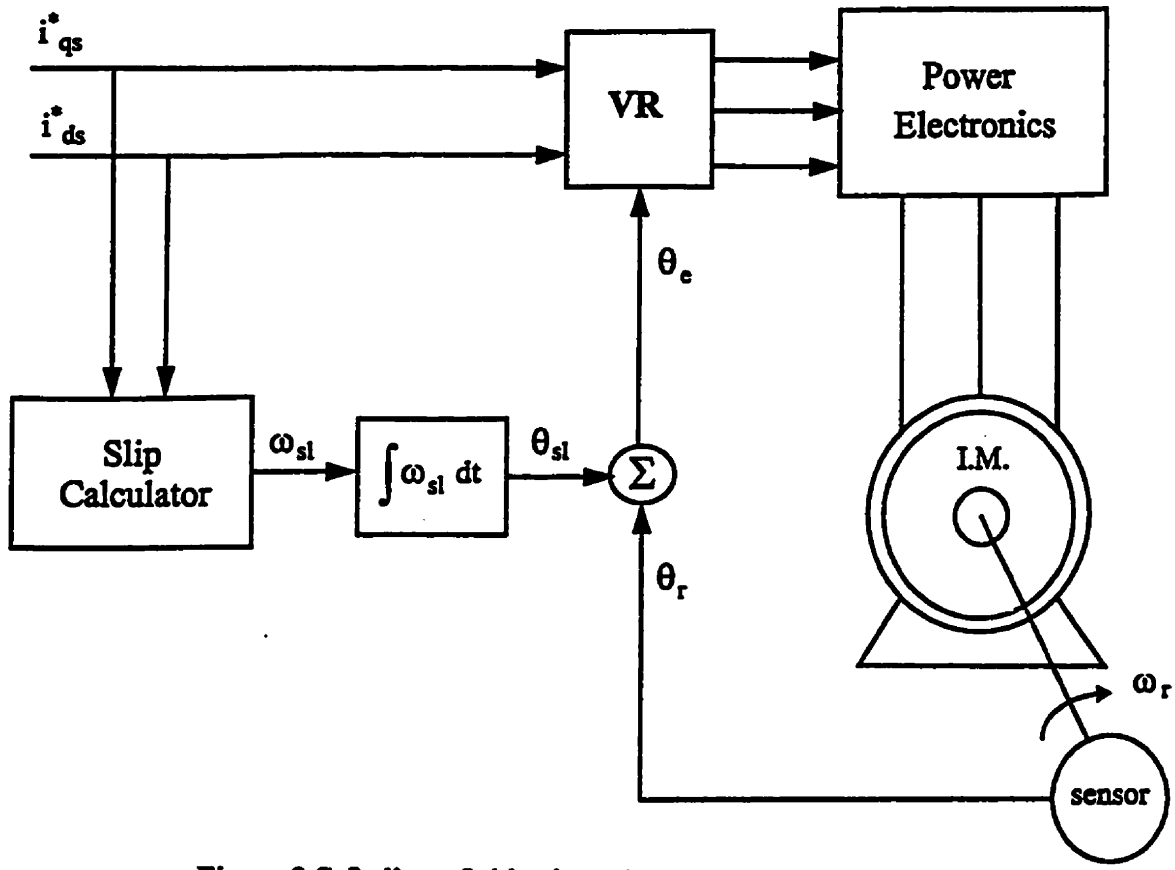


Figure 2.7. Indirect field oriented control

For determination of the rotor flux, a real time solution of the motor flux model equations is implemented using the measurable stator currents and rotor speed as driving functions. This is shown in block diagram form in Fig. 2.8.

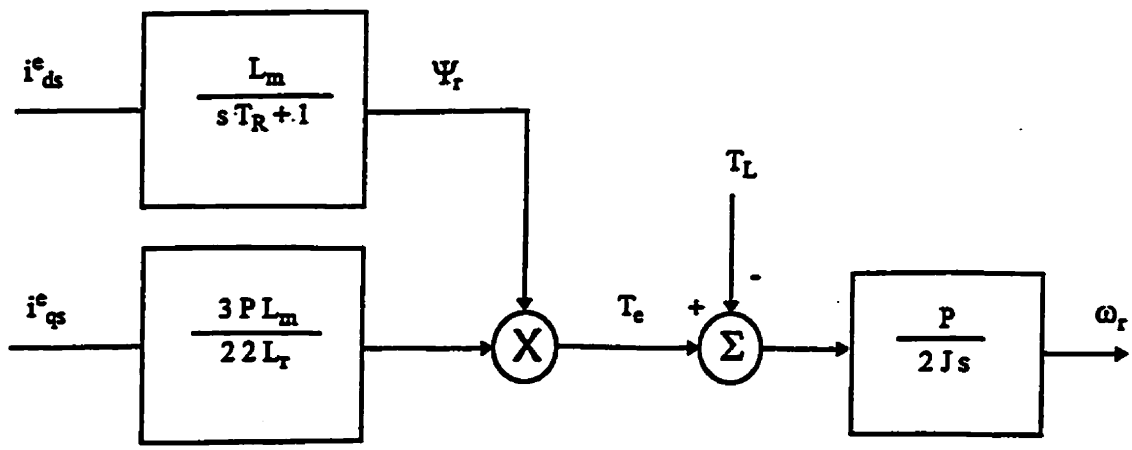


Figure 2.8. Indirect FOC induction motor model

Indirect field orientation does not have inherent low speed problems (unlike direct FOC), and is thus preferred in most systems that must operate near zero speed. As well, flux can be obtained even down to zero frequency, making it suitable for position control. A major drawback, however, is that calculation of the rotor flux depends on the rotor time constant T_R, where $T_R = L_r / R_r$. This time constant is dependent on rotor resistance, which is a function of rotor temperature and therefore tends to vary significantly due to temperature variations and the skin effect. This affects the accuracy of the flux magnitude and angle estimation, leading to a degradation in system performance and quality of control.

The functional diagram of a generalized indirect field oriented control system is presented in Fig. 2.9 [2]. The solid lines represent signals that are necessary for field orientation, while the dashed lines indicate signals that may or may not be required. As well, the stator current is divided into its direct component, i_{ds}, and its quadrature component, i_{qs}, which are expressed in the field oriented frame (i.e. the synchronously rotating reference frame fixed on the rotor magnetic field). The direct and quadrature components control the rotor flux, Ψ_r, and electromagnetic torque, T_e, respectively, of the induction motor. The inputs are the torque command (established by speed or position feedback) and the flux command (usually constant). The magnetizing reference input, v_{ds}^* or i_{ds}^*, and the torque reference, v_{qs}^* or i_{qs}^*, are produced by separate controllers. These voltages or currents are then fed into the matrix U^{-1}, which transforms them from synchronously rotating coordinates to stationary coordinates. Then, they are converted

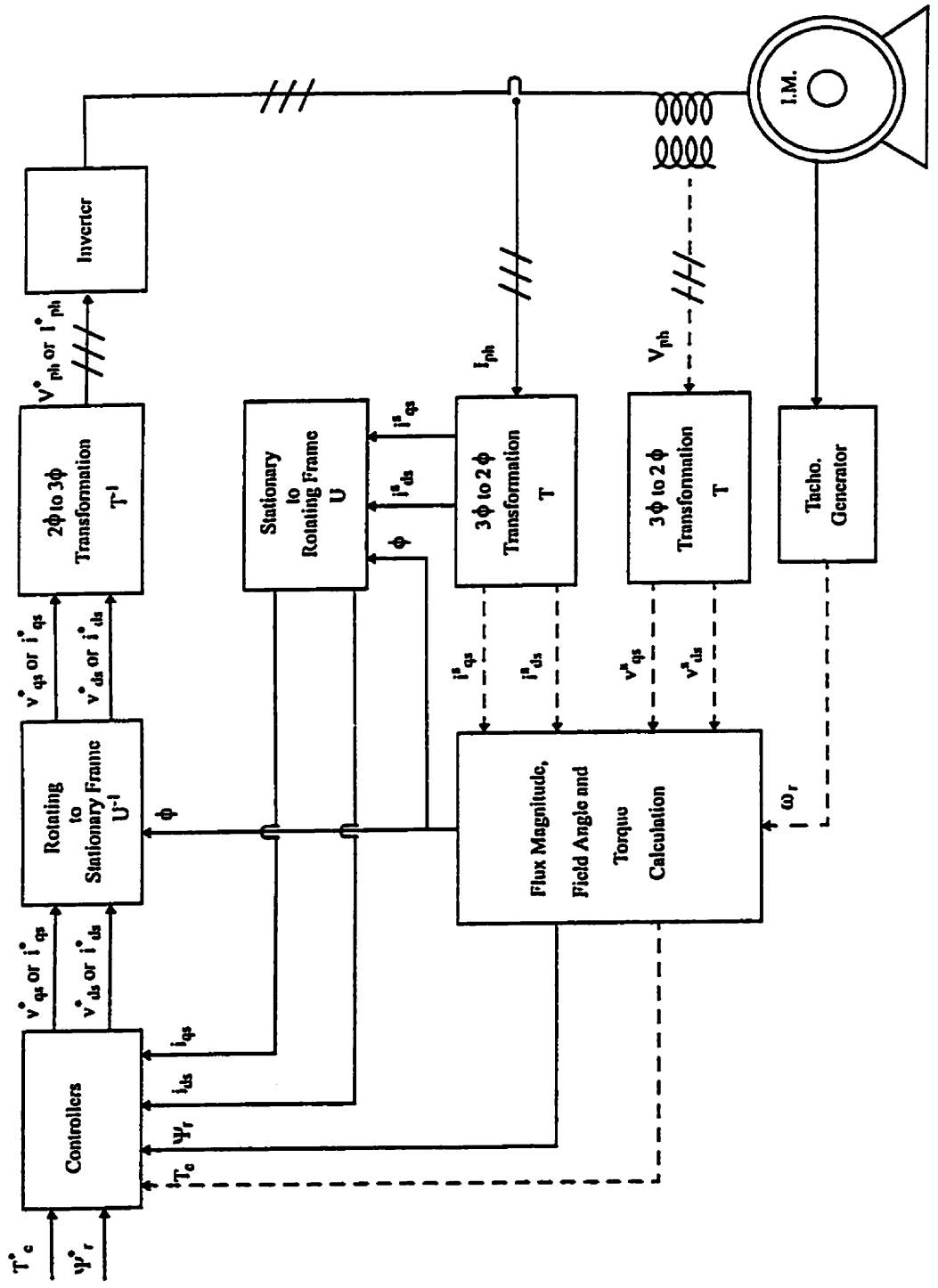


Figure 2.9. Generalized indirect FOC scheme for an induction motor

from the two-phase to the three-phase domain via the matrix T^{-1}, where they become phase commands for the inverter. Similarly, in the feedback loop, the three-phase quantities are converted back into two-phase variables from a stationary to a synchronously rotating reference frame. The flux magnitude, field angle and torque calculation block may require any of a combination of phase currents and/or phase voltages and rotor speed [18]. Even though a voltage source inverter (VSI) or a current source inverter (CSI) may be used for voltage phase commands or current phase commands, respectively, this is not essential. For example, current phase commands can be applied to a VSI with current feedback implemented for the inverter itself [19].

2.1.6. Flux Estimation

Figure 2.10 helps to explain the procedure for estimating the rotor flux magnitude and angular position using indirect field orientation. The d^s-q^s axes are fixed on the stator while the d^e-q^e axes rotate at synchronous angular velocity ω_e. At any instant, the q^e-axis is at angular position θ_e with respect to the q^s-axis. The angle θ_e is given by

$$\theta_e = \theta_r + \theta_{sl} \qquad (2.48)$$

$$= (\omega_r + \omega_{sl})t$$

$$= \omega_e\, t$$

where θ_r = rotor angular position ($\omega_r\, t$)

θ_{sl} = slip angular position ($\omega_{sl}\, t$)

$\omega_e = \omega_r + \omega_{sl}$

The rotor flux , Ψ_r, consisting of the air gap flux and the rotor leakage flux, is aligned to the *d*-axis. Since the voltages v^e_{qr} and v^e_{dr} are zero, Eqns. (2.38) and (2.39) become

$$0 = R_r\, i^e_{qr} + s\Psi^e_{qr} + (\omega_e - \omega_r)\,\Psi^e_{dr} \quad (2.49)$$

$$0 = R_r\, i^e_{dr} + s\Psi^e_{dr} - (\omega_e - \omega_r)\,\Psi^e_{qr} \quad (2.50)$$

and from Eqns. (2.41) and (2.43),

$$\Psi^e_{qr} = L_r\, i^e_{qr} + L_m\, i^e_{qs} \quad (2.51)$$

$$\Psi^e_{dr} = L_r\, i^e_{dr} + L_m\, i^e_{ds} \quad (2.52)$$

From Eqns. (2.51) and (2.52),

$$i^e_{qr} = \frac{1}{L_r}\Psi^e_{qr} - \frac{L_m}{L_r} i^e_{qs} \quad (2.53)$$

$$i^e_{dr} = \frac{1}{L_r}\Psi^e_{dr} - \frac{L_m}{L_r} i^e_{ds} \quad (2.54)$$

The rotor currents from Eqns. (2.49) and (2.50) can be eliminated by substituting Eqns. (2.53) and (2.54) into Eqns. (2.49) and (2.50) to yield

$$0 = \frac{R_r}{L_r}\Psi^e_{qr} - \frac{L_m}{L_r} R_r\, i^e_{qs} + s\Psi^e_{qr} + (\omega_e - \omega_r)\,\Psi^e_{dr} \quad (2.55)$$

$$0 = \frac{R_r}{L_r}\Psi^e_{dr} - \frac{L_m}{L_r} R_r\, i^e_{ds} + s\Psi^e_{dr} - (\omega_e - \omega_r)\,\Psi^e_{qr} \quad (2.56)$$

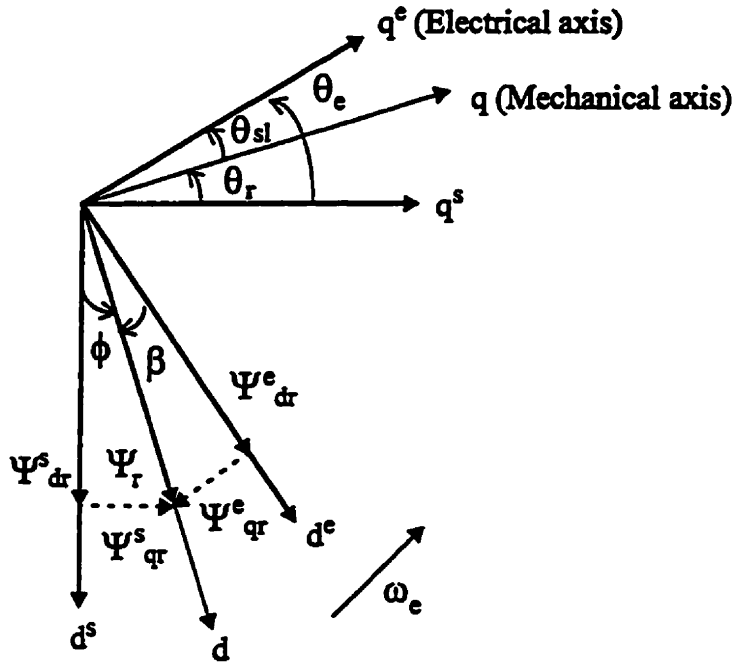


Figure 2.10. Phasor diagram for field orientation

The field orientation concept implies that the currents supplied to the machine should be oriented in phase and in quadrature to the rotor flux vector Ψ^{e}_{qdr}. This can be achieved by selecting ω_e to be the instantaneous speed of Ψ^{e}_{qdr} and locking the phase of the reference system such that the rotor flux is aligned with the d^e-axis (i.e. angle $\beta = 0$, hence the field angle ϕ equals the synchronous angle θ_e), resulting in the mathematical constraint

$$\Psi^{e}_{qr} = 0 \qquad (2.57)$$

This is the case for steady-state operation with zero load torque, neglecting stator and rotor losses. One purpose of field oriented control is to determine the field angle ϕ,

which is equal to θ_e, during this time frame. However, during steady-state, constant load torque operation, β is a small negative constant (typically $-30^o < \beta < 0^o$).

For decoupling control, it is desirable that

$$\Psi_{qr} = s\,\Psi_{qr} = 0$$

$$\Psi_{dr} = \Psi_r = \text{constant}$$

$$s\,\Psi_{dr} = 0$$

(note that no superscript denotes field oriented axes).

Hence, Eqns. (2.55) and (2.56), with decoupling control, become

$$\omega_{sl} = \frac{L_m R_r}{\Psi_r L_r} i_{qs} \qquad (2.58)$$

$$\frac{L_r}{R_r} s\Psi_r + \Psi_r = L_m i_{ds} \qquad (2.59)$$

from which it can be seen that rotor flux is a simple first order differential equation which is dependent on the d-axis current of the stator in the synchronously rotating coordinate system.

Eqn. (2.59) can be rewritten as

$$i_{ds} = \frac{1+sT_R}{L_m} \Psi_r \qquad (2.60)$$

where $s = \frac{d}{dt}$

$T_R = \frac{L_r}{R_r}$

Substituting Eqn. (2.60) into (2.58) yields

$$\omega_{sl} = \frac{1+sT_R}{T_R}\frac{i_{qs}}{i_{ds}} \qquad (2.61)$$

As well, in the field oriented frame, Eqns. (2.51) and (2.52) are expressed as

$$\Psi_{qr} = L_r\, i_{qr} + L_m\, i_{qs} \qquad (2.62)$$

$$\Psi_{dr} = L_r\, i_{dr} + L_m\, i_{ds} \qquad (2.63)$$

Eqns. (2.62) and (2.63) are quite relevant to the approach taken in this thesis to obtain field oriented control, as is discussed in Chapter 3.

2.2. Robust Control Theory and Loopshaping

No physical system can be exactly modeled by a mathematical system. Such is the case for high performance motor drive systems, which demand good command tracking and load regulating responses. Ideally, the performances should remain insensitive to the uncertainties of the drive system, which include differences in the dynamics of the model and the actual process, such as external load disturbance, unmodeled and nonlinear dynamics of the plant, and plant parameter variations. This is the definition of robust control. To determine how uncertainties affect a system, the concepts of robust stability and robust performance are applied to the model.

2.2.1. Types of Uncertainty

Modeling errors can be separated into two types. The first is parametric (or structured) uncertainty, in which these errors are differences between the numerical values of the differential equation coefficients between the actual plant and the model. It is assumed that the actual plant is of the same order as the model. The second type of uncertainty is known as unstructured uncertainty, in which the modeling errors refer to the difference in the dynamics between the finite dimensional model and the unknown, and possibly infinite, dimensional actual process [15]. This thesis considers only unstructured uncertainty, due to the fact that parametric errors and the effects of unmodeled dynamics are accounted for under its definition.

Suppose that the plant (a nominal transfer function), P, belongs to a bounded set of transfer functions, $\mathcal{P}$. The multiplicative representation of unstructured uncertainty is then defined as

$$\tilde{P}(s) = [1 + \Delta(s)\, W_2(s)]\, P(s) \qquad (2.64)$$

where $\tilde{P}(s)$ = a perturbed plant transfer function

$\Delta(s)$ = a variable stable transfer function satisfying $\| \Delta \|_\infty \leq 1$

$W_2(s)$ = a fixed, stable, and proper transfer function (also called the weight)

s = Laplace operator

Note: the infinity norm (or ∞ - norm) of a function is the least upper bound of its absolute value,

$$\| \Delta \|_\infty = \sup_\omega | \Delta(j\omega) |$$

(i.e. it is the highest gain value on a Bode magnitude plot).

The quantity $\| . \|_\infty$ is a norm, as it satisfies the following axioms (as per [14, 16]):

(i) $\| \Delta \|_\infty \geq 0$ with $\| \Delta \|_\infty = 0$ if and only if $\Delta = 0$

(ii) $\| \alpha \Delta \|_\infty = | \alpha | \cdot \| \Delta \|_\infty$ for all scalars α

(iii) $\| \Delta + \beta \|_\infty \leq \| \Delta \|_\infty + \| \beta \|_\infty$

It is assumed that no unstable or imaginary axis poles of $P(s)$ are canceled in the formation of $\tilde{P}(s)$. Thus P and $\tilde{P}$ have the same unstable poles.

Since Δ accounts for phase uncertainty and its magnitude varies between 0 and 1 at all frequencies (i.e. Δ acts as a scaling factor on the magnitude of the perturbation), the set $\mathcal{P}$ is the set of the transfer functions whose magnitude Bode plot lies in the envelope surrounding the magnitude plot of $P(s)$, as illustrated in Fig. 2.11. Thus, the size of the unstructured uncertainty is represented by the size of the envelope containing $\mathcal{P}$, and is found to increase with increasing frequency. The upper edge of the envelope conforms to the plot of $(1 + |W_2(j\omega)|)\,|P(j\omega)|$, while the lower edge of the envelope conforms to the plot of $(1 - |W_2(j\omega)|)\,|P(j\omega)|$. Typically, $|W_2(j\omega)|$ is also an increasing function of ω, as demonstrated in Fig. 2.11. Furthermore, since the phase of $\Delta(j\omega)$ is arbitrarily variable, the phase difference between any $\tilde{P}(j\omega) \in \mathcal{P}$ and $P(j\omega)$ can be arbitrarily large at high frequencies.

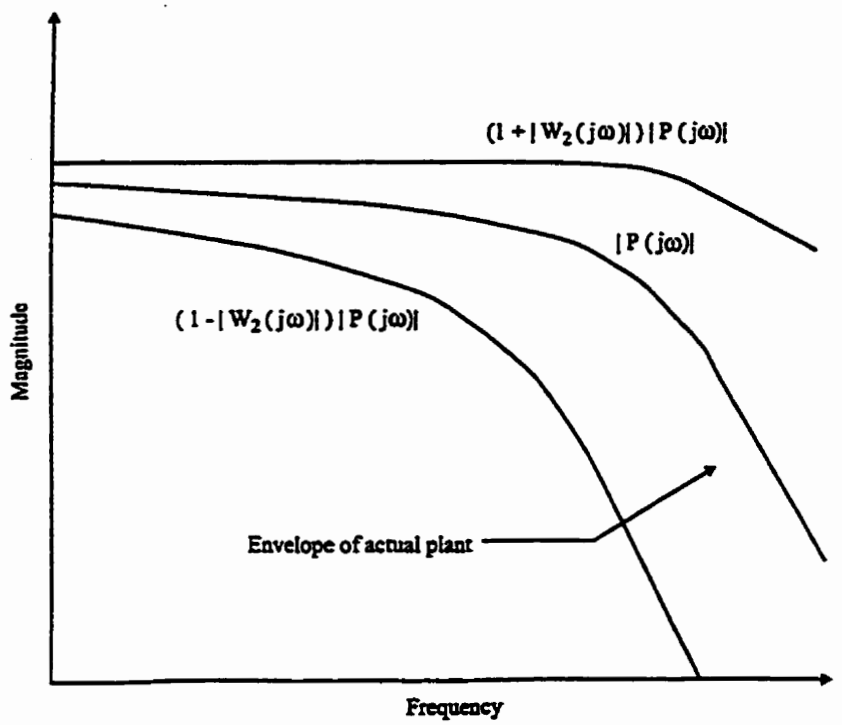


Figure 2.11. Bode plot interpretation of multiplicative uncertainty

Interpreting the uncertainty model, it can be seen that $\Delta(s)W_2(s)$ is the normalized error in the transfer function of the perturbed system with respect to the nominal model, such that

$$\frac{\tilde{P}(s)-P(s)}{P(s)} = \Delta(s)W_2(s) \qquad (2.65)$$

which reduces to

$$\frac{\tilde{P}(s)}{P(s)} - 1 = \Delta(s)W_2(s) \qquad (2.66)$$

Hence, if $\| \Delta\|_\infty \leq 1$, then

$$\left| \frac{\tilde{P}(j\omega)}{P(j\omega)} - 1 \right| \leq |W_2(j\omega)| \qquad (2.67)$$

for all frequencies, so $|W_2(j\omega)|$ provides the uncertainty profile. As shown in Fig. 2.12, this inequality describes a closed disk in the complex plane of radius $|W_2(j\omega)|$ centered at 1, which contains the point $\tilde{P}(j\omega) / P(j\omega)$, for each frequency. The unstructured uncertainty, then, is represented by the closed disk and, therefore, the direction and phase of the uncertainty is left arbitrary.

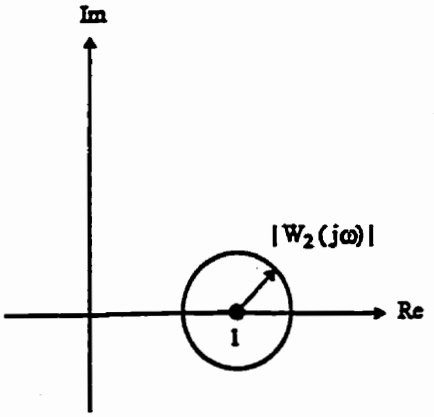


Figure 2.12. Multiplicative uncertainty in the complex plane

2.2.2. Robust Stability

The notion of robustness requires a controller, a set of plants and some characteristic of the system [14]. A controller C provides robust stability if it provides internal stability for every plant in the uncertainty set $\mathcal{P}$. (Note that the Laplace operator s has been dropped for convenience sake from this point onward. However, the transfer functions are still functions of s unless otherwise stated.) Hence, a test for robust stability involves the controller and the uncertainty set. Let L denote the open-loop transfer function (i.e. $L = P\,C$). Then, let S denote the sensitivity function (i.e. error to reference tranfer function),

$$S = \frac{1}{1+L} \qquad (2.68)$$

and the complimentary sensitivity function (i.e. output to reference transfer function) is given by

$$T = 1 - S = \frac{L}{1+L} = \frac{PC}{1+PC} \qquad (2.69)$$

Furthermore, for a multiplicative perturbation model, the robust stability condition is met if and only if $\| W_2 T \|_\infty \leq 1$ [14,15]. A graphical interpretation of this condition is shown in Fig. 2.13. Hence, the stability condition may be generalized to become

$$\| W_2 T \|_\infty \leq 1 \Leftrightarrow \left| \frac{W_2(j\omega)L(j\omega)}{1+L(j\omega)} \right| < 1, \text{ for all } \omega \qquad (2.70)$$

$$\Leftrightarrow |W_2(j\omega)L(j\omega)| < |1+L(j\omega)| \text{ , for all } \omega$$

$$\Leftrightarrow |\Delta(j\omega)W_2(j\omega)L(j\omega)| < |1+L(j\omega)| \text{ , for all } \omega, \|\Delta\|_\infty \le 1$$

Therefore, the critical point, -1, lies outside the disk, which is centered at $L(j\omega)$, radius $|W_2(j\omega)L(j\omega)|$.

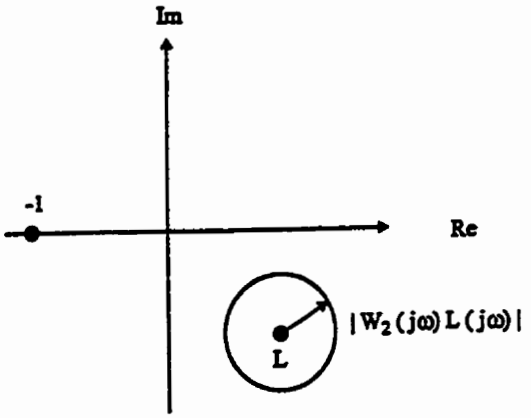


Figure 2.13. Robust stability condition in the complex plane

The relevance of the condition $\|W_2T\|_\infty \le 1$ can be seen in its relation to the small-gain theorem, which states that the feedback system is initially stable if all the transfer functions (i.e. the plant P, controller C and feedback gain F) are stable and $\|PCF\|_\infty < 1$. A block diagram of a typical perturbed system, ignoring all inputs, is shown in Fig. 2.14. The transfer function from the output of Δ to the input of Δ equals $-W_2T$. The properties of the block diagram can be reduced to those of the configuration given in Fig. 2.15. The maximum loop gain, $\|-W_2T\|_\infty$, is less than 1 for all allowable Δs if and only if the small-gain condition $\|W_2T\|_\infty \le 1$ holds.

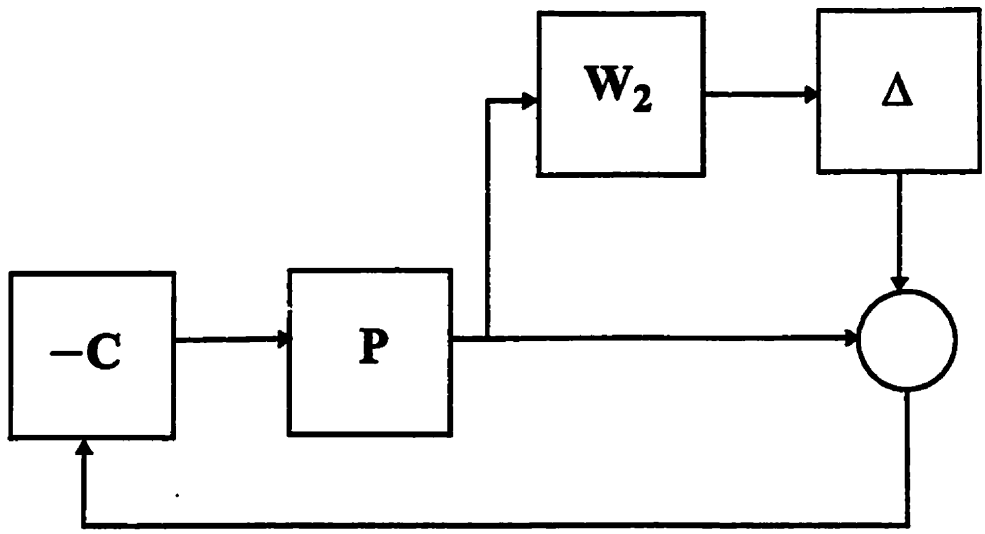


Figure 2.14. Perturbed feedback system

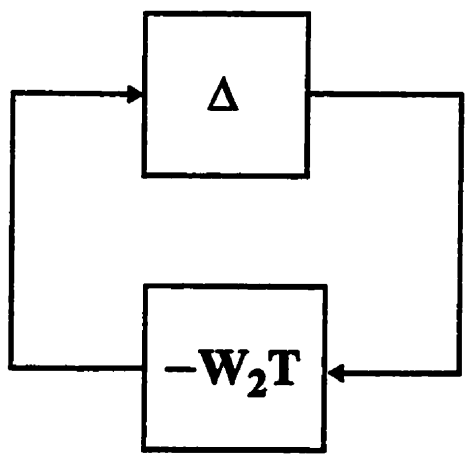


Figure 2.15. Reduced block diagram

2.2.3. Robust Performance

Internal stability and performance should hold for all plants in the uncertainty set $\mathcal{P}$, according to the generalization of robust performance. The robust stability condition for an internally stable, nominal feedback system, is $\| W_2T \|_\infty \leq 1$, and the nominal performance condition is $\| W_1S \|_\infty \leq 1$, where W_1 is a real-rational, stable, minimum-phase transfer function (also called the weighting function) such that

$$\| W_1S \|_\infty \leq 1 \Leftrightarrow \left| \frac{W_1(j\omega)}{1+L(j\omega)} \right| < 1, \text{ for all } \omega$$

$$\Leftrightarrow |W_1(j\omega)| < |1+L(j\omega)|, \text{ for all } \omega$$

If, as before, $\tilde{P} = [1 + \Delta W_2] P$, then the perturbed sensitivity function is written as

$$\tilde{S} = \frac{1}{1+\tilde{P}C} = \frac{1}{1+(1+\Delta W_2)PC} = \frac{1}{1+(1+\Delta W_2)L} = \frac{1}{(1+L)+\Delta W_2L} = \frac{S}{1+\Delta W_2T} \quad (2.71)$$

Therefore, the robust performance condition is given by

$$\| W_2T \|_\infty \leq 1 \text{ and } \left\| \frac{W_1S}{1+\Delta W_2T} \right\|_\infty \leq 1, \text{ for all allowable } \Delta \quad (2.72)$$

Since $|\Delta| \leq 1$, then $|-\Delta W_2T| \leq |W_2T|$. Thus $|1 + \Delta W_2T| \geq 1 - |W_2T|$ for a fixed frequency, and it is then implied that

$$\left\| \frac{W_1S}{1+\Delta W_2T} \right\|_\infty \leq \left\| \frac{W_1S}{1-|W_2T|} \right\|_\infty \leq 1 \quad (2.73)$$

Hence, Eqn. (2.72) can be rewritten as a necessary and sufficient condition for robust performance, which is

$$\| |W_1 S| + |W_2 T| \|_\infty \leq 1 \qquad (2.74)$$

which is a stronger constraint than nominal performance or the robust stability condition alone. A graphical interpretation of this condition is shown in Fig. 2.16, whereby

$$\| |W_1 S| + |W_2 T| \|_\infty \leq 1 \Leftrightarrow \left| \frac{W_1}{1+L} \right| + \left| \frac{W_2 L}{1+L} \right| \leq 1, \text{ for all } \omega$$

$$\Leftrightarrow |W_1| + |W_2 L| \leq |1+L|, \text{ for all } \omega \qquad (2.75)$$

At each frequency, there exist two closed disks; one disk centered at -1, radius $|W_1(j\omega)|$; the other centered at $L(j\omega)$, radius $|W_2(j\omega)L(j\omega)|$. The condition given by Eqn. (2.74) then holds if and only if the two disks have no nontrivial intersection (i.e. they can touch, but they cannot overlap).

It should be noted that since the condition for simultaneously achieving nominal performance and robust stability is

$$\| \max (|W_1 S| , |W_2 T|) \|_\infty \leq 1 \qquad (2.76)$$

and the robust performance condition is tested by Eqn. (2.74), then the conditions in Eqns. (2.74) and (2.76) differ at most by a factor of two. In other words,

$$\| \max (|W_1 S| , |W_2 T|) \|_\infty \leq \| |W_1 S| + |W_2 T| \|_\infty \leq 2 \| \max (|W_1 S| , |W_2 T|) \|_\infty \quad (2.77)$$

The choice of these norms is not crucial, even though they may vary by as much as a factor of two. The inherent tradeoffs in control problems between $|W_1 S|$ and $|W_2 T|$ allow for similar solutions to be achieved even when using different norms.

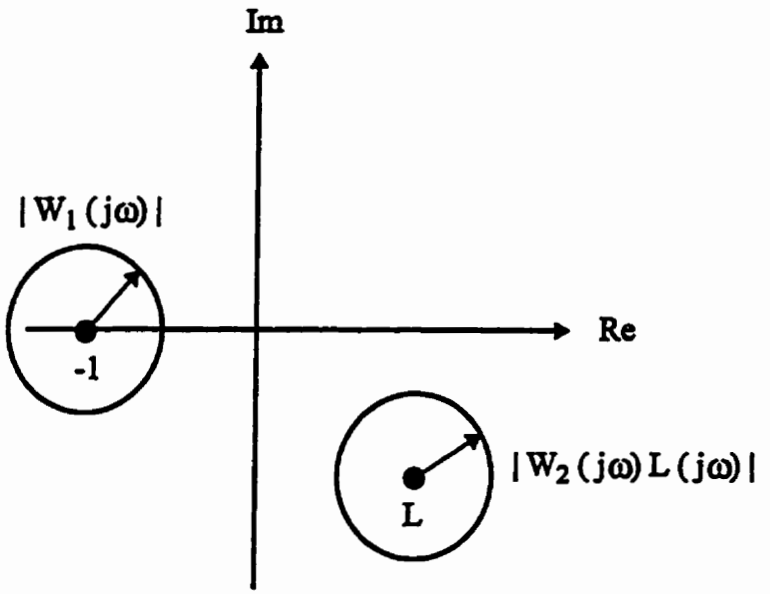


Figure 2.16. Robust performance condition in the complex plane

2.2.4. Loopshaping Technique

Loopshaping is a graphical design procedure for robust performance design, whereby P, W_1, and W_2 are the input data, and a proper controller C is designed to stabilize the plant and satisfy Eqn. (2.74) [14]. The basic idea of this method is to construct the loop transfer function L to approximately satisfy Eqn. (2.74), and then to attain C via $C = L / P$. Internal stability of the nominal feedback system and the

properness of C constitute the constraints of this method. It is assumed that P and P^{-1} are both stable, otherwise L must contain P's unstable zeros and poles. Thus, the condition $L=PC$ must have no pole-zero cancellation. In terms of W_1, W_2, and L, Eqn. (2.74) is given by

$$\Gamma(j\omega) = \left| \frac{W_1(j\omega)}{1+L(j\omega)} \right| + \left| \frac{W_2(j\omega)L(j\omega)}{1+L(j\omega)} \right| < 1 \qquad (2.78)$$

which must hold for all frequencies. (Note that the argument $j\omega$ is dropped from this point on. The transfer functions are still functions of $j\omega$ unless otherwise stated.)

A necessary condition for robust performance is that at every frequency, either $|W_1|$ or $|W_2|$ must be less than 1 [15]. Typically, $|W_1|$ is monotonically decreasing (for good tracking of low frequency signals), and $|W_2|$ is monotonically increasing (as uncertainty increases with increasing frequency). Hence, at each frequency, either $|W_1| < 1$ or $|W_2| < 1$. It is also the case when $|W_1| < 1$, $|W_2| >> 1$, and when $|W_2| < 1$, $|W_1| >> 1$. These properties can be used to determine the relationship between $|W_1|$, $|W_2|$, and $|L|$.

For the case $|W_1| >> 1 > |W_2|$, Eqn. (2.78) becomes

$$\Gamma < 1 \Leftrightarrow |W_1| + |W_2||L| < |1+L| \qquad (2.79)$$

$$\Rightarrow |W_1| + |W_2||L| < 1 + |L| \qquad (2.80)$$

$$\Rightarrow |L| > \frac{|W_1|-1}{1-|W_2|} \qquad (2.81)$$

(a necessary condition), and because $|W_1| >> 1$, it can be said that

$$|L| > \frac{|W_1|+1}{1-|W_2|} \Leftrightarrow |W_1| + |W_2||L| < |L| - 1 \quad (2.82)$$

$$\Rightarrow |W_1| + |W_2||L| < |L+1| \quad (2.83)$$

$$\Rightarrow \Gamma < 1$$

(a sufficient condition). Since $|W_1| >> 1$, the condition $\Gamma < 1$ can be approximated by

$$|L| > \frac{|W_1|}{1-|W_2|} \quad (2.84)$$

For the case $|W_1| < 1 << |W_2|$, Eqn. (2.78) becomes

$$\Gamma < 1 \Leftrightarrow |W_1| + |W_2||L| < |1+L| \quad (2.85)$$

$$\Rightarrow |W_1| + |W_2||L| < 1 + |L| \quad (2.86)$$

$$\Rightarrow |L| < \frac{1-|W_1|}{|W_2|-1} \quad (2.87)$$

(a necessary condition), and since $|W_2| >> 1$, it can be said that

$$|L| > \frac{1-|W_1|}{|W_2|+1} \Leftrightarrow |W_1| + |W_2||L| < 1 - |L| \quad (2.88)$$

$$\Rightarrow |W_1| + |W_2||L| < |1+L| \quad (2.89)$$

$$\Rightarrow \Gamma < 1$$

(a sufficient condition). Since $|W_1| >> 1$, the condition $\Gamma < 1$ can be approximated by

$$|L| < \frac{1-|W_1|}{|W_2|} \quad (2.90)$$

Therefore, because $|W_1|$ is a decreasing function of frequency, and $|W_2|$ is an increasing function of frequency, then typically at low frequencies

$$|W_1| > 1 > |W_2|$$

and at high frequencies

$$|W_1| < 1 < |W_2| .$$

The loopshaping design procedure can now be outlined (as per [14]):

1. Plot two curves on a log-log scale, magnitude versus frequency. These are graphs of

$$\frac{|W_1|}{1-|W_2|} \text{ over the low frequency range where } |W_1| > 1 > |W_2|,$$

and

$$\frac{1-|W_1|}{|W_2|} \text{ over the high frequency range where } |W_1| < 1 < |W_2| .$$

2. On this plot, fit a graph of the magnitude of the open-loop transfer function $|L|$, whereby

$$|L| > \frac{|W_1|}{1-|W_2|} \text{ at low frequencies, and } |L| >> 1;$$

and

$$|L| < \frac{1-|W_1|}{|W_2|} \text{ at high frequencies, and } |L| << 1.$$

At very high frequencies, let $|L|$ roll off at least as quickly as $|P|$ does. This ensures that the controller is proper. The general features of the open-loop transfer function, then, are that the gain in the low frequency region should be large enough, and in the high

frequency region, the gain should be attenuated as much as possible. The gain at the intermediate frequencies typically controls the gain and phase margins. Near the gain crossover frequency ω_c (where the magnitude equals 1), the slope of the log-magnitude curve in the Bode plot should be close to -20 *dB / decade* (i.e. the transition from low to high frequency should be smooth). If $|L|$ drops off too quickly through crossover, internal instability will result, so a gentle slope is crucial.

3. Obtain a stable, minimum-phase open-loop transfer function L for the gain $|L|$ already constructed, normalizing so that $L(0) > 0$. The latter condition guarantees negative feedback.

4. Recover the controller C from the condition $L = PC$.

5. Verify nominal stability and the condition of Eqn. (2.74), i.e.

$$\| |W_1 S| + |W_2 T| \|_\infty \leq 1$$

in order to validate the design. Nominal internal stability is achieved if, on a Nyquist plot of L, the angle of L at crossover is greater than 180^o (i.e. crossover occurs in the third or fourth quadrant). If either of these conditions do not hold, this entire process must be repeated.

Chapter 3

3.1. Implementation of FOC and Robust Controllers

In this chapter, the design of a field oriented control drive system for an induction motor, and subsequently, the design of robust controllers for this system, are presented. The major components to be considered are an induction motor, a PWM voltage source inverter, a robust controller for rotor flux, a robust controller for shaft speed, and two PI controllers for *d-q* field currents. Note that the field oriented control approach to be used, is based on the availability of a voltage source inverter in the power electronics laboratory at the University of Calgary, which will facilitate the future physical implementation of the proposed drive system.

3.2. Induction Motor Drive Model

The block diagram of the indirect field oriented controlled induction motor drive is shown in Fig. 3.1 (as based on the generalized FOC system presented in Fig. 2.9). It can be seen that the *d-q* voltages are generated in the field oriented reference frame from the field oriented i_{ds} and i_{qs} feedback quantities. These voltages are then rotated (via matrix U^{-1}), transformed (via matrix T^{-1}), and applied to a voltage source inverter. Utilizing a simulated induction motor model, this entire system was coded in C (see Appendix A).

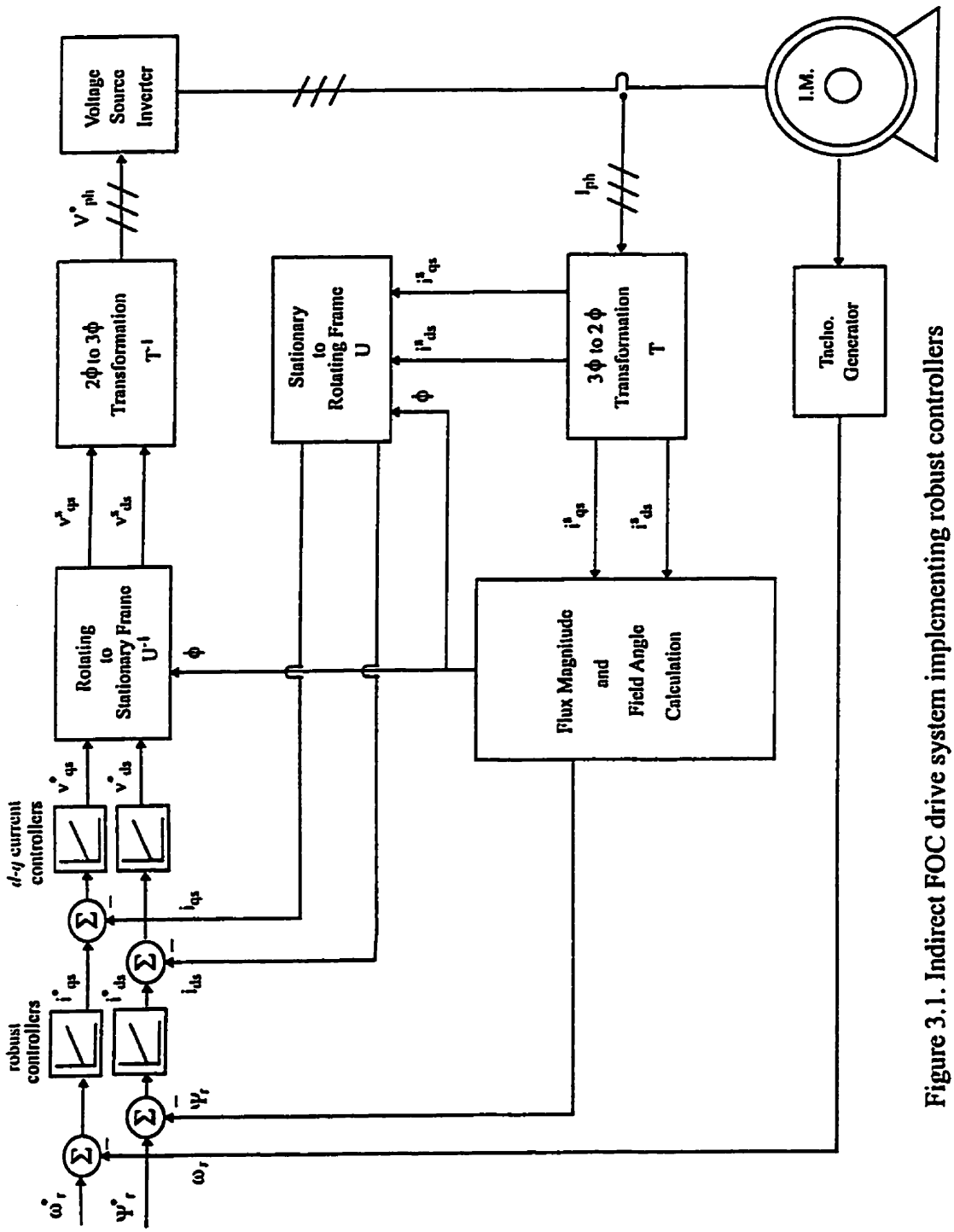


Figure 3.1. Indirect FOC drive system implementing robust controllers

Field orientation results in decoupled rotor flux and torque production, and therefore, a need is established for separate flux and speed control. This is achieved through the design of robust flux and speed controllers.

3.3. Robust Control Design

3.3.1. Plant Modeling

An accurate knowledge of the field angle, ϕ, is crucial in coordinate transformation, as is shown by Fig. 3.1. In FOC, as motor speed begins to increase from standstill, it is often desirable for the rotor flux to reach its maximum value as quickly as possible and maintain that magnitude throughout its operating range. A robust flux controller, and subsequently, a robust speed controller, is proposed to achieve this required fast speed response while allowing the system to remain relatively insensitive to the uncertainties of external disturbances and parameter variation. It should be noted that all results presented in this chapter are based on flux magnitude and field angle calculations as per Eqns. (2.44), (2.62), and (2.63).

From the simplified block diagram of the induction motor model shown in Fig. 2.8, the following relationships can be established

$$\Psi_r = \frac{L_m}{sT_R+1} i_{ds}^* \tag{3.1}$$

$$T_e = K_t\, i_{qs}^* \tag{3.2}$$

$$K_t = \left(3p/4\right)\left(L_m^2/L_r\right) i_{ds}^* \qquad (3.3)$$

where Ψ_r = rotor flux

L_m = magnetizing inductance

i_{ds}^* = flux command current

i_{qs}^* = torque command current

T_e = electromagnetic torque

K_t = torque proportionality constant

p = number of poles

T_R = rotor time constant ($= L_r/R_r$)

Note that, as in Chapter 2, the transfer functions are still functions of s unless otherwise stated.

As well, the relationship between electrical rotor speed and torque presented in Eqn. (2.45) can be rewritten as

$$T_e - T_L = \left(p/2\right) L_m = J\, d\omega_r/dt + B\, \omega_r \qquad (3.4)$$

where T_L = load torque

J = polar moment of inertia

B = coefficient of friction

ω_r = rotor angular velocity

From Eqns. (3.1) - (3.4), it is possible to create first order models of the transfer functions $i_{ds}^{*} \rightarrow \Psi_r$ and $i_{qs}^{*} \rightarrow \omega_r$, for flux and speed control, which are, respectively,

$$\frac{\Psi_r}{i_{ds}^{*}} = P_f(s) = \frac{L_m}{sT_R+1} \tag{3.5}$$

$$\frac{\omega_r}{i_{qs}^{*}} = P_s(s) = \frac{K_t}{Js+B} \tag{3.6}$$

where P_f = nominal plant transfer function for flux control

P_s = nominal plant transfer function for speed control

3.3.2. Uncertainty Modeling

There are several types of perturbations to be considered in system modeling. For both flux and speed control, one variable to be considered is the delay effect in system dynamics due to the coupling characteristics of the mechanical shaft, written as $e^{-\tau s}$, where τ is the dead-time of the drive, and $e^{-\tau s}$ is treated as a multiplicative uncertainty. A first order Pade approximation of $e^{-\tau s}$ is used to simplify the analysis [11,15], where

$$e^{-\tau s} \approx \frac{1-\tau s/2}{1+\tau s/2} \tag{3.7}$$

For the flux control system alone, the rotor resistance, R_r, is considered a perturbed variable, due to temperature variations and skin effect during the normal operation of an induction motor. For the speed control system alone, the variables to be perturbed are the inertia, J, and the torque proportionality constant, K_t, which is a

function of the flux command current, i_{ds}^{*}. This current is found to vary slightly during normal operation, rather than remain constant. The inertia constant can undergo significant change to test the effectiveness of the proposed controllers [13]. In fact, all tolerances can be made large when modeling. By overcompensating for any given perturbations in the modeling process, then, the controllers should be able to offset any unexpected changes during normal operation. In this way, the controllers are modeled for a worst-case scenario. Therefore, the rotor resistance, polar moment of inertia and the proportionality constant are assumed to be subject to variations of ΔR_r, ΔJ, and ΔK_t, whereby

$$\tilde{R}_r = R_r + \Delta R_r \tag{3.8}$$

$$\tilde{J} = J + \Delta J \tag{3.9}$$

$$\tilde{K}_t = K_t + \Delta K_t \tag{3.10}$$

where $\tilde{R}_r$ = perturbed rotor resistance

$\tilde{J}$ = perturbed inertia constant

$\tilde{K}_t$ = perturbed proportionality constant

Accordingly, the drive models are replaced by

$$\tilde{P}_f = \frac{L_m}{s\tilde{T}_R+1} e^{-\tau s} \tag{3.11}$$

$$\tilde{P}_s = \frac{\tilde{K}_t}{\tilde{J}s+B} e^{-\tau s} \tag{3.12}$$

where $\tilde{P}_f$ = perturbed plant transfer function for flux control

$\tilde{P}_s$ = perturbed plant transfer function for speed control

$\widetilde{T_r}$ = perturbed rotor time constant ($= L_r / \widetilde{R_r}$)

The coefficient of friction, B, is not varied due to its relatively small magnitude in variation. The nominal values and tolerances of these perturbed parameters are:

Table 3.1: Perturbed parameters

Parameters	nominal	limits
τ	0.025 p.u.	0.02 - 0.03 *sec*
R_r	0.0287 p.u.	up to 50%
J	0.0167 p.u.	0.00835 - 0.0835 p.u
i_{ds}^*	0.3 p.u.	0.25 - 0.35 p.u

3.3.3. Loopshaping Design

The frequency response of each perturbed plant is shown in Figs. 3.2 and 3.3, for flux and speed control, respectively. It can be seen that the tolerances listed in Table 3.1 provide the upper and lower boundaries of each perturbed plant set. A perturbed plant contained within each set can have any irregular magnitude response shape, within the known boundaries. Eqn. (2.67) provides a means whereby the weighting transfer function W_2 can be found. For convenience, this equation is rewritten here

$$\left| \frac{\tilde{P}(j\omega)}{P(j\omega)} - 1 \right| \leq |W_2(j\omega)| \qquad (3.13)$$

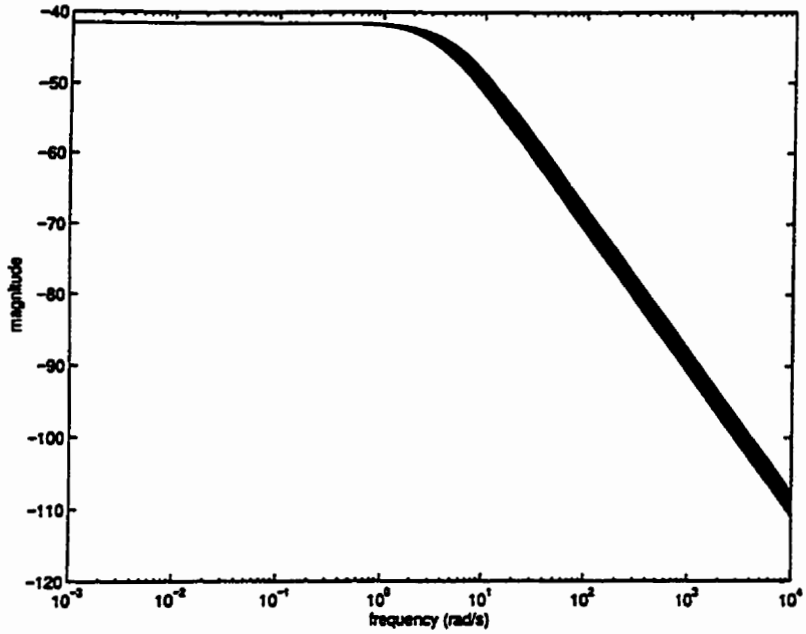


Figure 3.2. Frequency response for perturbed flux plant

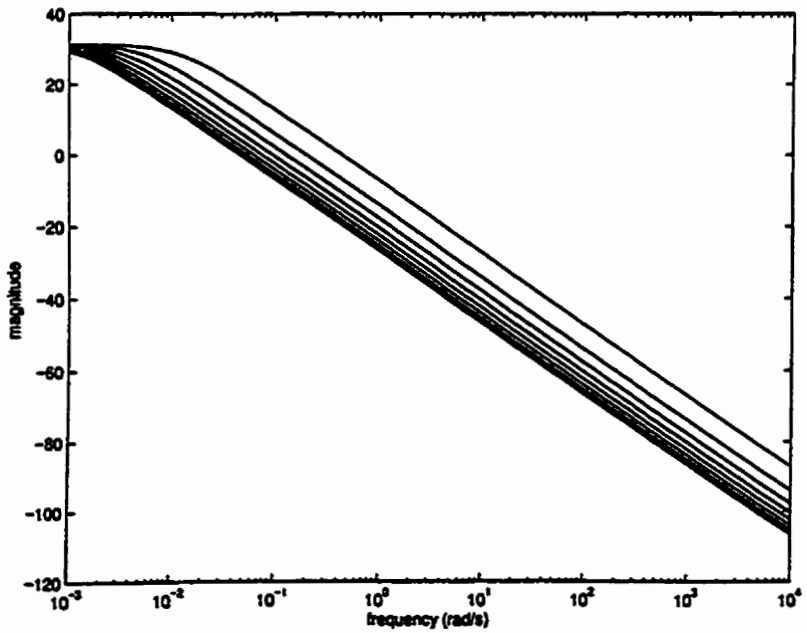


Figure 3.3. Frequency response for perturbed speed plant

Reducing Eqn. (3.13) to incorporate the various perturbed variables, a function W_2 is found to satisfy the requirements of both control systems, and is given by

$$W_2 = \frac{0.04s}{0.01s+1} \qquad (3.14)$$

The result is shown in Fig. 3.4. The frequency responses and the determination of W_2 are obtained through MATLAB [20]. As well, the entire loopshaping procedure is applied through MATLAB. These programs are listed in Appendix B.

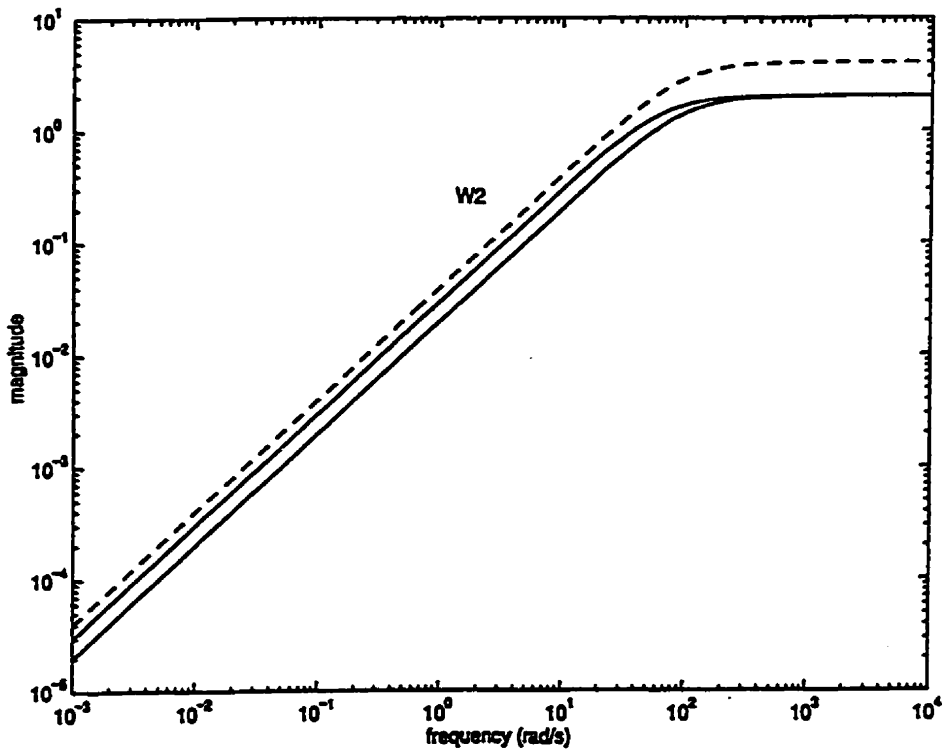


Figure 3.4. Magnitude plot of W_2

Through simulation studies, a cut-off frequency of 1 *rad/s* provides good tracking over the normalized frequency range [0, 1] for both control systems. Hence, a transfer function W_1 is chosen to be a third order Butterworth filter, given by

$$W_1 = \frac{1.05}{s^3+2s^2+2s+1} \qquad (3.15)$$

for both control systems. Using W_1 and W_2 in conjunction with Eqn. (2.78), a satisfactory $|L|$ is constructed and shown in Fig. 3.5, with $|W_1|$ and $|W_2|$. Then, a stable, minimum-phase open-loop transfer function L is obtained as

$$L = \frac{36(s+2)}{s(s^2+6s+9)} \qquad (3.16)$$

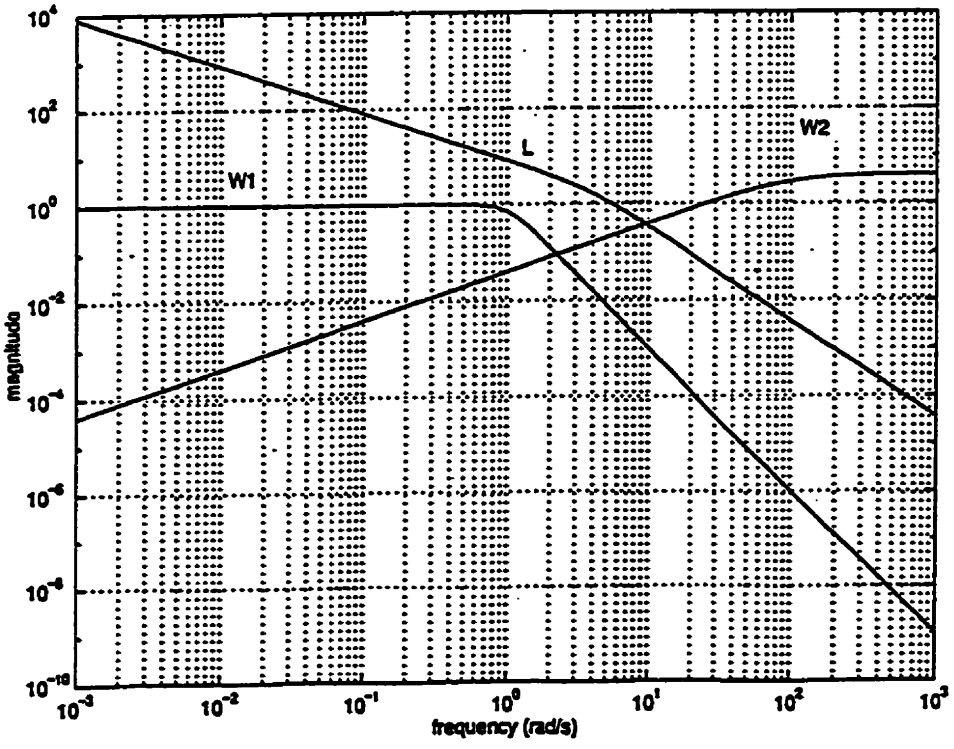


Figure 3.5. Plot of $|L|$, $|W_1|$, and $|W_2|$

It can be seen that at low frequencies $|L| > |W_1|$ (or $|W_1| > 1 > |W_2|$) , and at high frequencies $|L| < |W_2|$ (or $|W_1| < 1 < |W_2|$). Following the loopshaping procedure outlined in Chapter 2, $\frac{|W_1|}{1-|W_2|}$, $\frac{1-|W_1|}{|W_2|}$, and $|L|$ are plotted in Fig. 3.6, verifying the conditions that

$$|L| > \frac{|W_1|}{1-|W_2|} \text{ at low frequencies, and } |L| >> 1;$$

$$|L| < \frac{1-|W_1|}{|W_2|} \text{ at high frequencies, and } |L| << 1.$$

Also, the transition of $|L|$ from low to high frequency is seen to be "smooth".

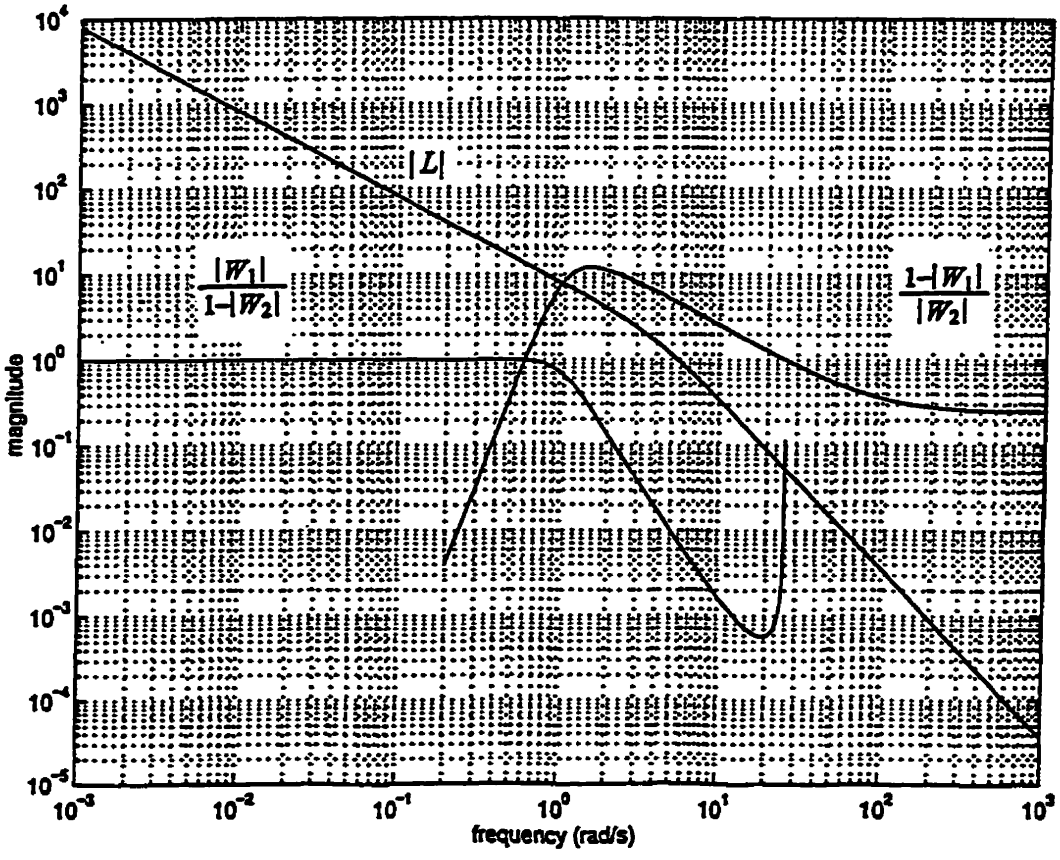


Figure 3.6. Verification of conditions on $|L|$

The corresponding controllers for flux and speed , C_f and C_s, respectively, are recovered from the condition $L = PC$, and are given by

$$C_f = \frac{12.97s^2+61.94s+72}{s(0.01s^2+0.0603s+0.0904)} \qquad (3.17)$$

$$C_s = \frac{0.6012s^2+1.206s+0.0072}{s(0.0044s^2+0.0264s+0.0395)} \qquad (3.18)$$

The robust performance condition is verified by plotting $|W_1S| + |W_2T|$, as shown in Fig. 3.7, where it can be seen that $\| |W_1S| + |W_2T| \|_\infty = 0.13 < 1$. This is the best result that can be achieved with gain manipulation (of L). The performance condition is found to depreciate towards 1 if the gain is manipulated in either the positive or negative direction.

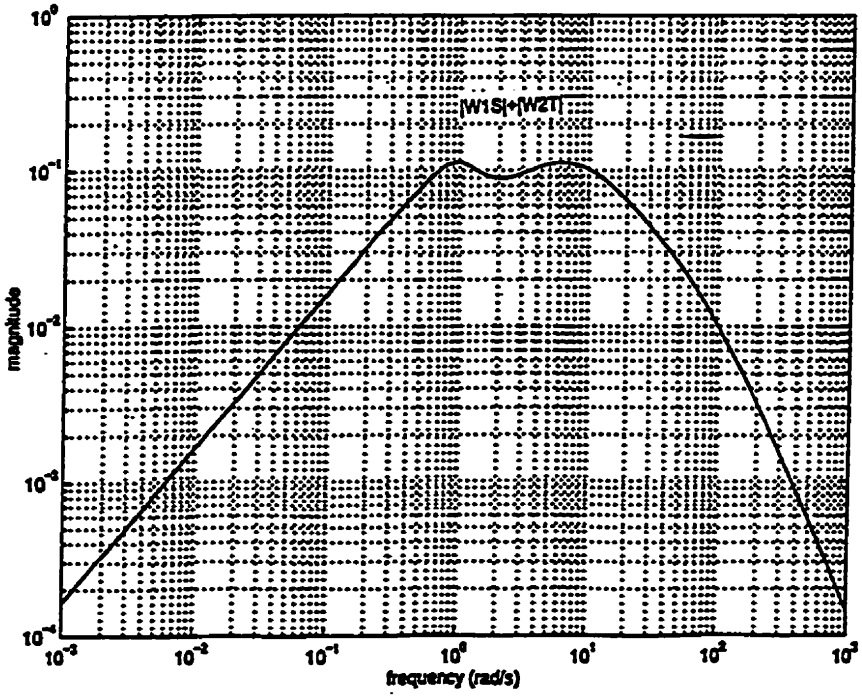


Figure 3.7. Robust performance verification

Also, it can be shown that $L(0) > 0$ and that the roots of the closed loop system $1 + L = 0$ are

$$-2.0303 + j5.7448$$

$$-2.0303 - j5.7448$$

$$-1.9394$$

verifying that this L provides nominal internal stability. This is correlated by Fig. 3.8, which shows the Nyquist plot of L, and Fig. 3.9, which shows gain and phase margins of L. Gain margin (GM) and phase margin (PM) are used to measure the system's relative stability [21]. The gain margin is the amount of gain in *dB* that can be inserted into the closed loop before the system reaches instability. The phase margin is the change in open loop phase shift required at unity gain to make the closed loop system unstable. The gain crossover frequency, ω_c, is the frequency at which the loop gain crosses the 0 *dB* line; and the phase crossover frequency, ω_p, is

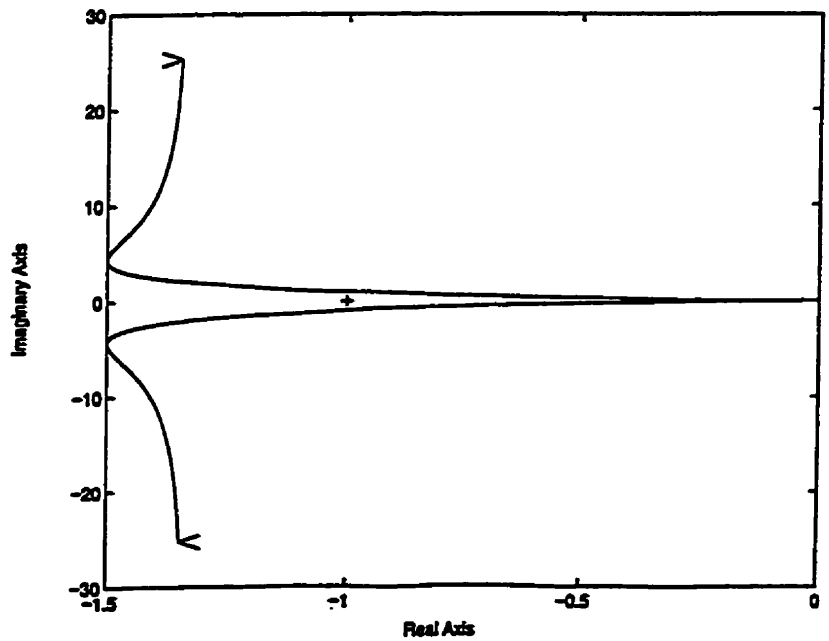


Figure 3.8. Nyquist plot of L

the frequency at which the angle of $L(j\omega_p)$ crosses the - 180° line. If $|L(j\omega_p)|$ lies below the 0 *dB* line, GM is positive; otherwise it is negative. If the angle of $L(j\omega_c)$ lies above the - 180° line, PM is positive, and if it lies below, PM is negative. A minimum-phase, proper transfer function has all its poles and zeros lying in the open left half *s*-plane. Generally, for a minimum-phase L, the system is stable if GM > 0 and it is unstable if GM < 0; and generally, a minimum-phase system has a PM > 0 and becomes unstable if PM < 0. Thus, in Fig. 3.9, it can be seen that the gain margin is positive and the phase margin is positive, also verifying that the system is stable.

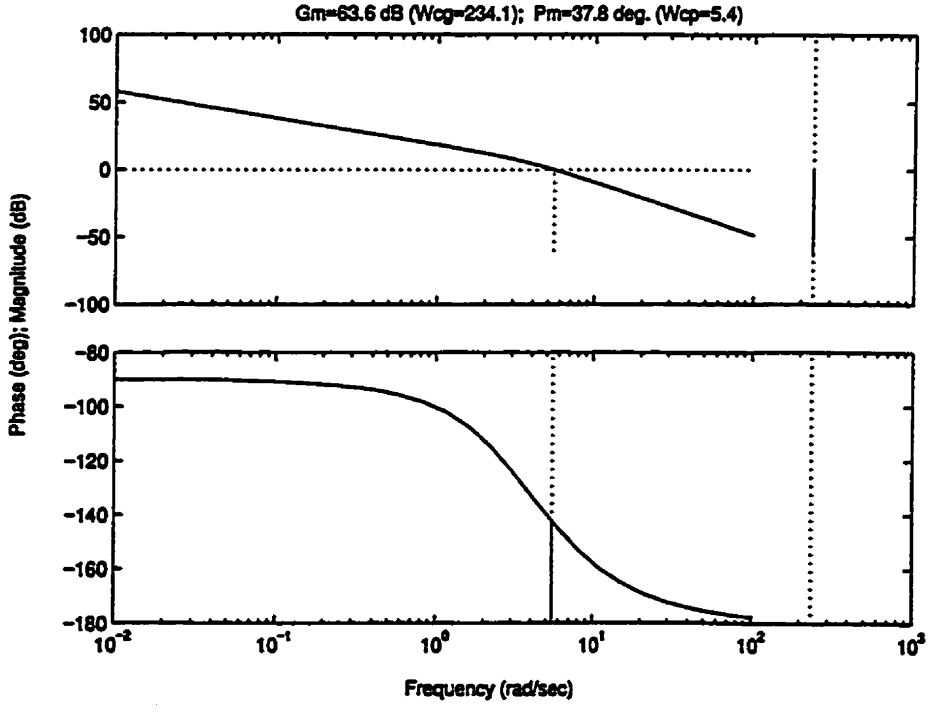


Figure 3.9. Gain and phase margins

Next in the design procedure, a unit step input is applied to the closed loop system as there is a close correlation between a system response to a unit step input and the system's ability to perform under normal operating conditions [15, 21]. The response is given in Fig. 3.10, from which the following quantities are determined:

(a) peak time (t_p) = the time for the step response to reach its first peak

(b) percent overshoot (PO) = the amount that the response overshoots its steady-state (or final) value at the peak time; expressed as a percentage of the final value,

i.e. $100\left(\frac{peak\_value - final\_value}{final\_value}\right)$

(c) rise time (t_r) = the time at which the step response first reaches 90% of its final value minus the time at which it first reaches 10% of its final value

(d) settling time (t_s) = the time required for the response to settle to within ±2% of its final value.

For this system,

$$t_p = 0.5439\ sec$$

$$PO = 30.6255\ \%$$

$$t_r = 0.2448\ sec$$

$$t_s = 1.7949\ sec$$

It is found that if the gain of L is increased, the settling time and the percent overshoot increase, while the rise time decreases. The opposite effect occurs for each

quantity if the gain is decreased. Ideally, a low settling time, low percent overshoot and quick rise time are desired (i.e. as close to a unit step as possible). Therefore, an optimum case should be found which best satisfies all three criteria. In this thesis, the design focuses on the best robust performance for the controllers, thereby finding a middle ground in terms of PO, t_r, and t_s.

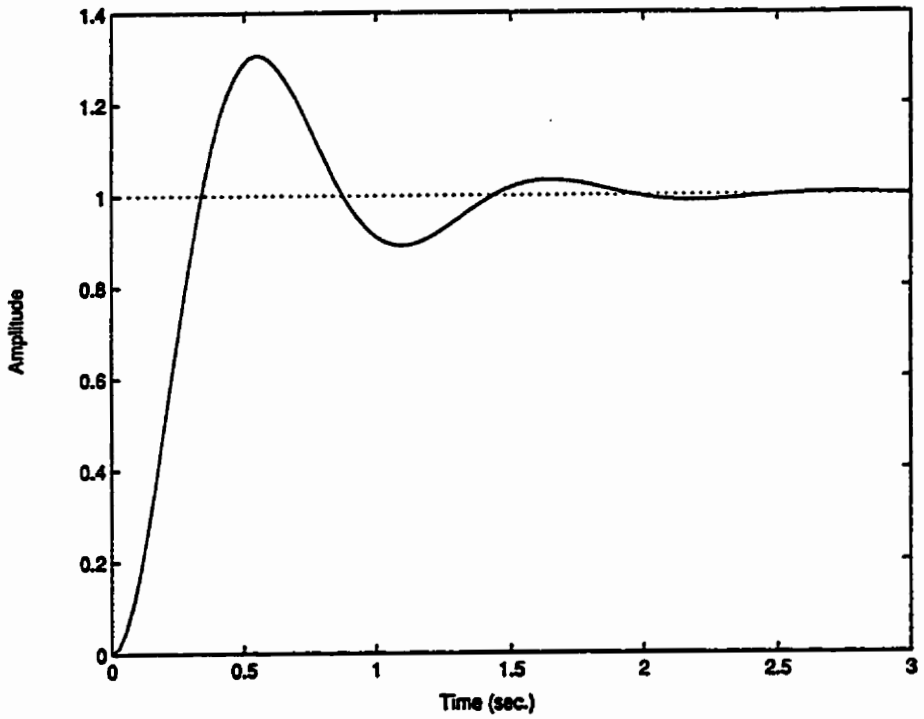


Figure 3.10. Unit step response

As another test of controller performance, the unit step response of each perturbed system is found for the varying values of $\tau, R_r, J,$ and K_t . The results are shown in Figs. 3.11 and 3.12 for flux and speed, respectively. It can be seen from these figures that, in each case, as the uncertainties increase, the response percent overshoot decreases, but the rise time increases.

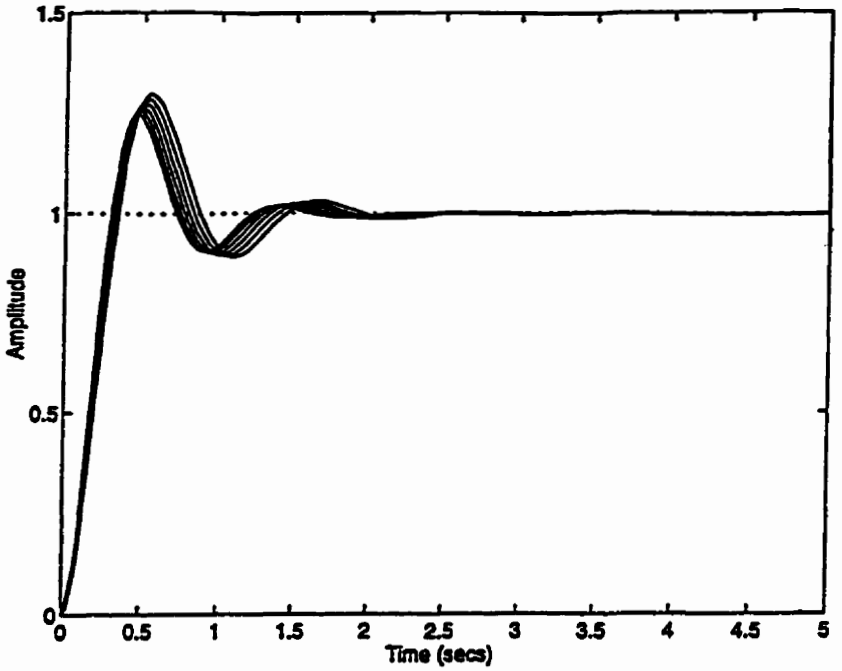


Figure 3.11. Unit step response for perturbed flux control system

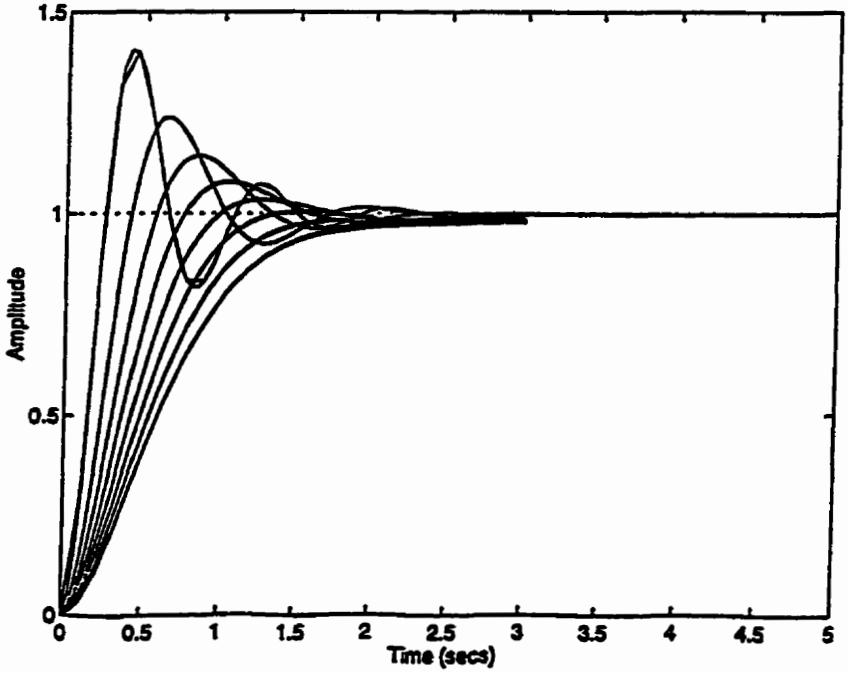


Figure 3.12. Unit step response for perturbed speed control system

3.3.4. Implementation

The robust controllers are placed into the drive system (as per Fig. 3.1) for simulation by converting them from the continuous time system of their MATLAB environment to the discrete time system employed in the C program for the FOC drive system. This is accomplished by using Tustin's method (also known as bilinear transformation) [20], which directly converts the s-domain controller $C(s)$ into the z-domain controller $C_d(z)$ by the following substitution:

$$\text{let} \qquad s = \frac{2}{T_s}\left(\frac{z-1}{z+1}\right) \tag{3.19}$$

$$\text{then} \qquad C_d(z) = C\left(\frac{2}{T_s}\frac{(z-1)}{(z+1)}\right) \tag{3.20}$$

where T_s = sampling interval (chosen to be 0.3 *sec* for these systems)

The resulting z-domain controllers for flux and speed, respectively, are then

$$C_{df}(z) = \frac{169.9393z^3+8.4041z^2-123.827z+37.708}{z^3-1.7558z^2+0.89762z-0.1418} \tag{3.21}$$

$$C_{ds}(z) = \frac{12.687z^3-6.8087z^2-12.6765z+6.819}{z^3-1.7593z^2+0.90296z-0.14367} \tag{3.22}$$

The controllers are then converted to the discrete time domain as per [22, 23], to become difference equations in delay-operator form, defined by the controllers' inputs and outputs. That is, for flux control, the input to the controller C_f is given by $x_f(k)$ and the output is given by $y_f(k)$, where

$$x_f(k) = \Psi_r^* - \Psi_r \tag{3.23}$$

$$y_f(k) = i_{ds}^* \tag{3.24}$$

and for speed control, the input to the controller C_s is given by $x_s(k)$ and the output is given by $y_s(k)$, where

$$x_s(k) = \omega_r^* - \omega_r \tag{3.25}$$

$$y_s(k) = i_{qs}^* \tag{3.26}$$

Therefore, for this drive system, the difference equations are

$$\begin{aligned} y_f(k) = {} & 169.9x_f(k) + 8.404x_f(k-1) - 123.8x_f(k-2) + 37.71x_f(k-3) \\ & + 1.756y_f(k-1) - 0.8976y_f(k-2) + 0.1418y_f(k-3) \end{aligned} \tag{3.27}$$

$$\begin{aligned} y_s(k) = {} & 12.687x_s(k) - 6.8087x_s(k-1) - 12.6765x_s(k-2) + 6.8192x_s(k-3) \\ & + 1.7593y_s(k-1) - 0.90296y_s(k-2) + 0.14367y_s(k-3) \end{aligned} \tag{3.28}$$

The initial conditions, for $k = 0$, are chosen to be 0.0 p.u. for x_f and x_s at (0), (-1), (-2), etc., and 6.0 p.u. for y_f and y_s at (0), (-1), (-2), etc., as this corresponds to machine operation. In fact, the choice of values for y_f and y_s is not crucial as the FOC program converges quickly (i.e. there is no difference in the output if the initial values for y_f and y_s are 0.0 p.u. or 6.0 p.u.).

3.4. Current Controllers

In successful field orientation, the *d-q* axes are decoupled, yielding the correct stator currents. Thus, current controllers are needed to regulate the stator voltages, for a system using a voltage source inverter, in order to achieve the desired currents. In the FOC scheme of Fig. 3.1, current control is performed in the field oriented reference frame, whereby i_{ds}^* is obtained from the rotor flux controller and i_{qs}^* is obtained from the speed controller. These controllers are chosen to be PI controllers, and as such, they must be tuned. The proportional control coefficient is tuned first until the drive system just begins to become unstable, in order to determine a conservative value. Then, the integral coefficient is tuned to have the best steady-state error.

3.5. Results

The results of implementing the robust controllers into the drive system, and of optimal tuning of the PI controllers, can be seen in Figs. 3.13 - 3.15.

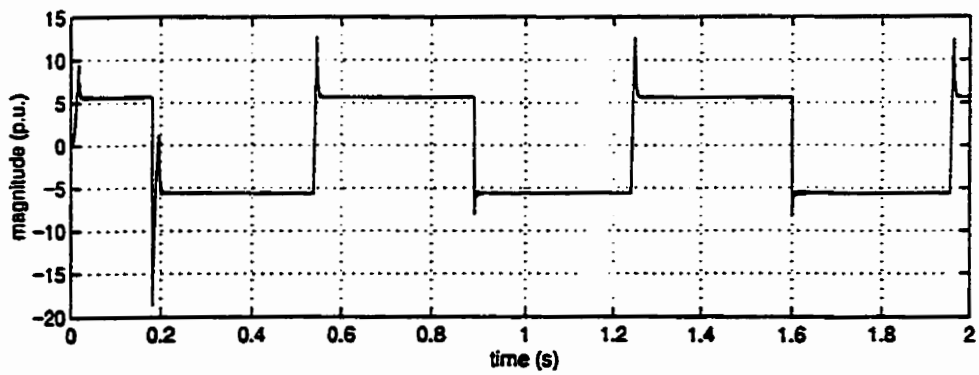


Figure 3.13. Electromagnetic torque

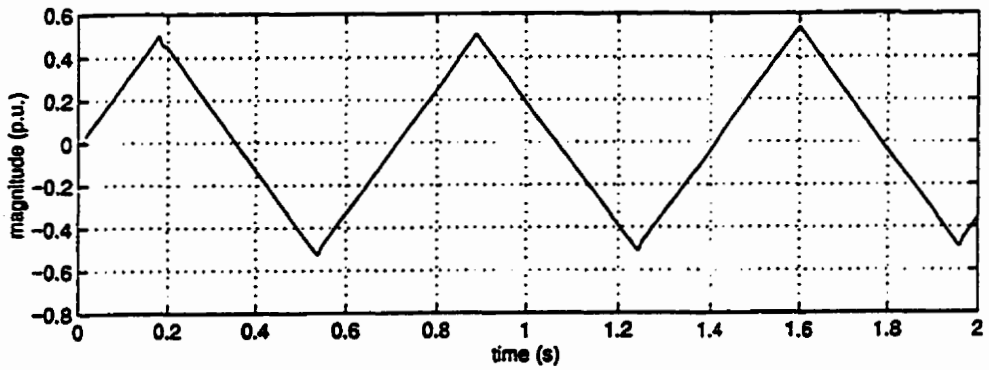


Figure 3.14. Rotor speed

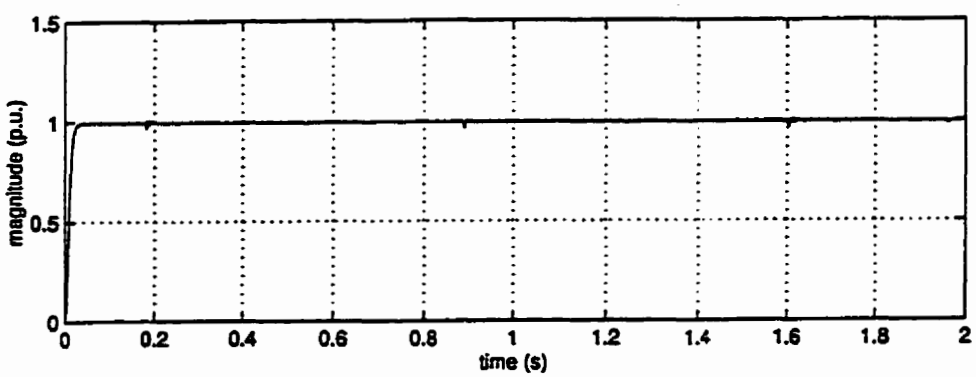


Figure 3.15. Flux magnitude

The electromagnetic torque, T_e, shown in Fig. 3.13, reaches steady-state in a very short amount of time (less than 0.03 *sec*). However, there appear to be oscillatory transients during switching. In Fig. 3.14, the rotor speed curve is produced by toggling i_{qs}^* between 6.0 p.u. and -6.0 p.u. every 0.2 *sec*. As can be seen from Fig. 3.15, the rotor flux

magnitude reaches 1.0 p.u. in less than 0.02 *sec* with FOC (and the robust controllers) implemented, whereas without FOC it takes this motor drive 1.5 *sec* to reach 1.0 p.u. [2].

In FOC, the field angle, ϕ, is responsible for the precise location of the rotor flux field. Since it is required in coordinate transformation of the system variables, from the synchronously rotating reference frame to the stationary reference frame, and vice versa, the field angle is essential to the goal of field orientation. The sine and cosine of the field angle (i.e. $\sin(\phi)$ and $\cos(\phi)$), for the induction motor, are shown in Figs. 3.16 and 3.17, respectively. Since ϕ is an unbounded function, and $\sin(\phi)$ is a bounded function, both $\sin(\phi)$ and $\cos(\phi)$ are needed in order to work backwards to define the field angle.

The *d-q* axes currents in the stationary reference frame are shown in Figs. 3.18 and 3.19.

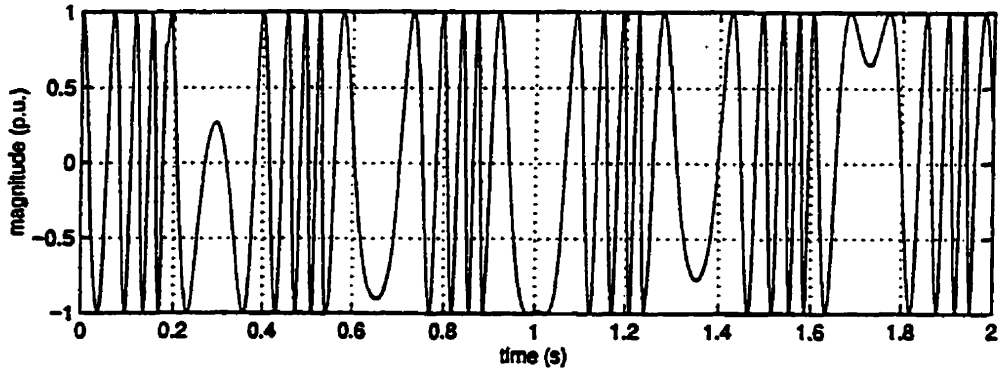


Figure 3.16. Sine of field angle (ϕ)

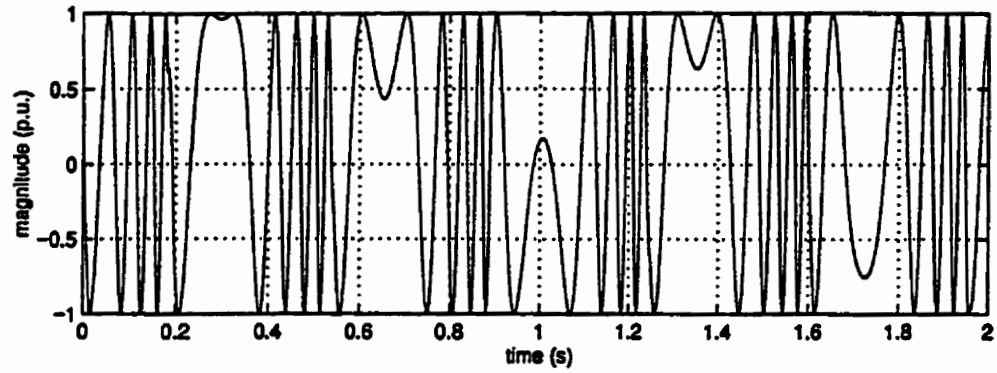


Figure 3.17. Cosine of field angle (ϕ)

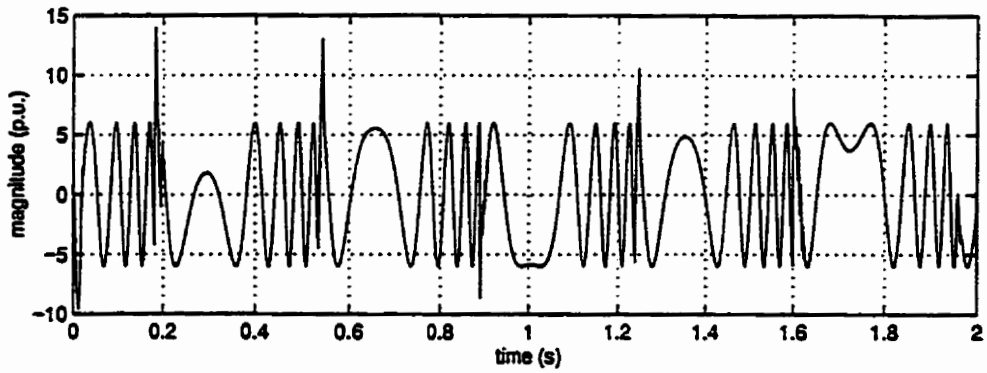


Figure 3.18. i_{ds} in stationary reference frame

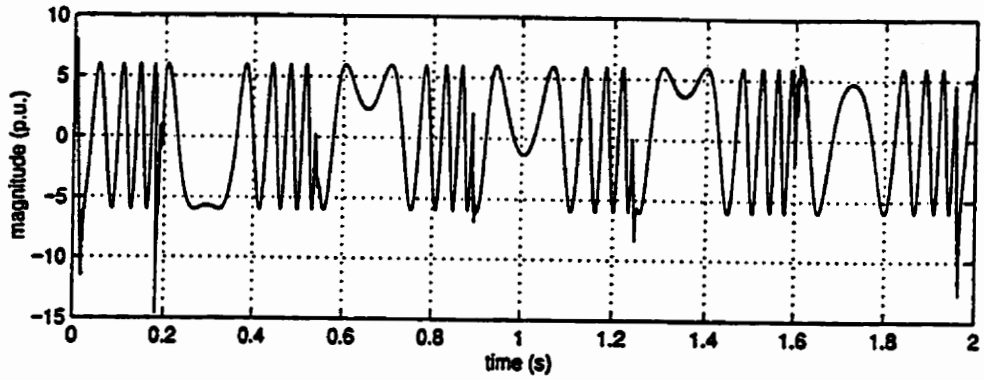


Figure 3.19. i_{qs} in stationary reference frame

Chapter 4

4.1. Parameter Sensitivity

The estimation of the rotor flux magnitude and field angle relies on the value of the motor parameters. The variation in rotor resistance, in particular, is the dominant factor affecting flux estimation, due to the temperature variations inside the motor. To provide some indication of estimation sensitivity, the value of the rotor resistance is changed in the FOC program, without altering the robust controllers (or any other component of the simulated drive system).

The following results verify that the flux estimator is capable of achieving accurate results irrespective of the presence of rotor resistance variations. The rotor resistance is increased by 50% and 100% above the nominal value and the resulting flux magnitude and sin ϕ are shown in Figs. 4.1 - 4.4, for each case. The results of cos ϕ are similar to those of sin ϕ and consequently are not shown here.

In achieving these results, there is no adjustment of robust controller coefficients, verifying their effectiveness. The only adjustment to be made is in recalibrating the gain-limiting condition imposed on the command current i_{ds}^* (i.e. necesarry to maintain the command current within reasonable physical limits of the induction motor). This value decreases by 0.135, from the nominal 0.315 value, for a 100% change in R_r. The gain-limiting condition imposed on i_{qs}^* need not be altered from the nominal value of 0.1.

As can be seen, the maximum absolute error for flux magnitude for both cases is 0.01 p.u.

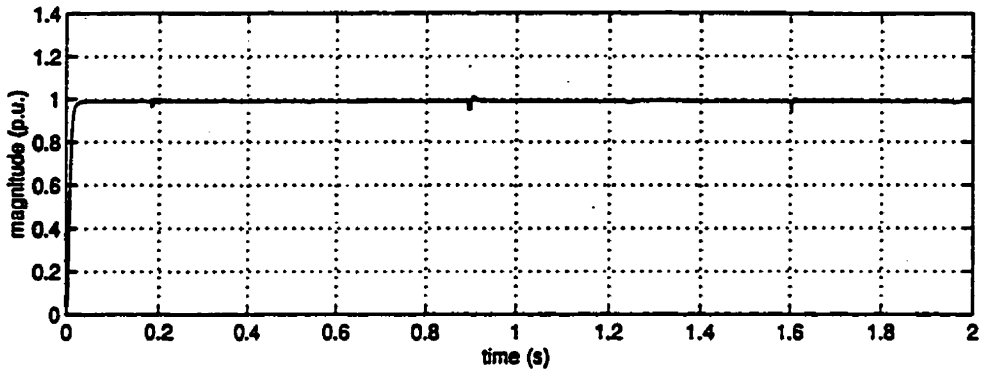


Figure 4.1. Flux magnitude for R_r increased by 50%

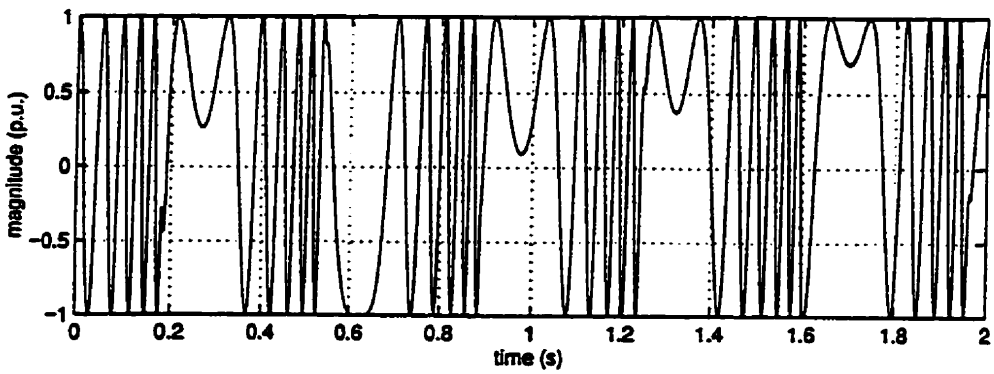


Figure 4.2. Sine of field angle (ϕ) for R_r increased by 50%

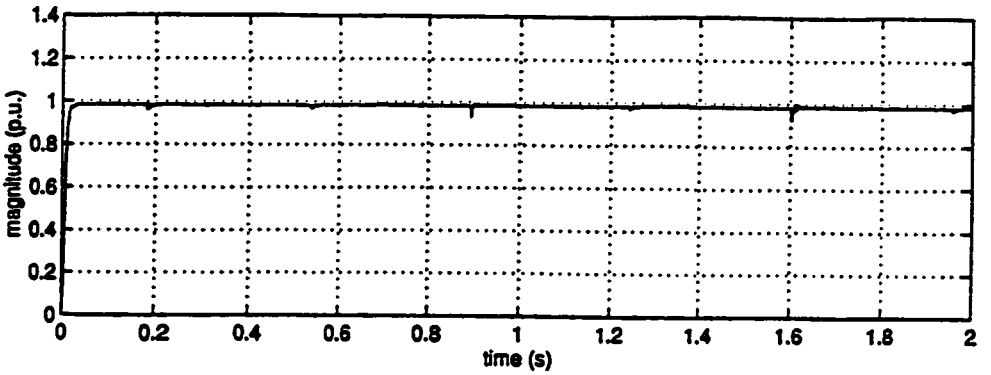


Figure 4.3. Flux magnitude for R_r increased by 100%

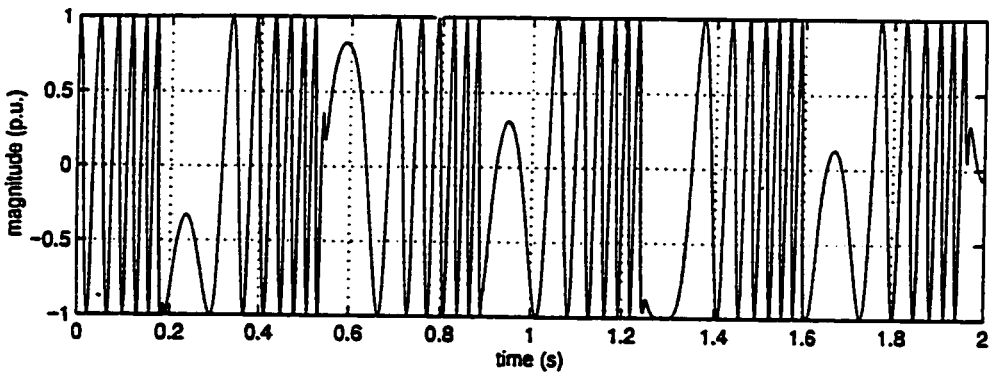


Figure 4.4. Sine of field angle (ϕ) for R_r increased by 100%

4.2. PI Controller Comparison

To further demonstrate the adaptive capabilities of the robust controllers, their performance is compared with that of simple PI controllers for rotor flux and speed control. This is a worthwhile comparison, as PI control is still used to a great extent in industry due to the simplicity of design and ease of implementation. Under nominal conditions, the PI controllers provide results comparable to those of robust control. However, it can be seen in Figs. 4.5 - 4.8 that, without PI controller coefficient retuning, the controllers cannot regulate the flux magnitude for rotor resistance variations greater than 20%.

The PI controller coefficients are not retuned specifically to reflect this as a time-consuming drawback in practice. It also further emphasizes the advantages of using robust control.

The results presented in this chapter indicate that the robust controlled flux estimator should work quite well under all normal conditions of induction motor operation.

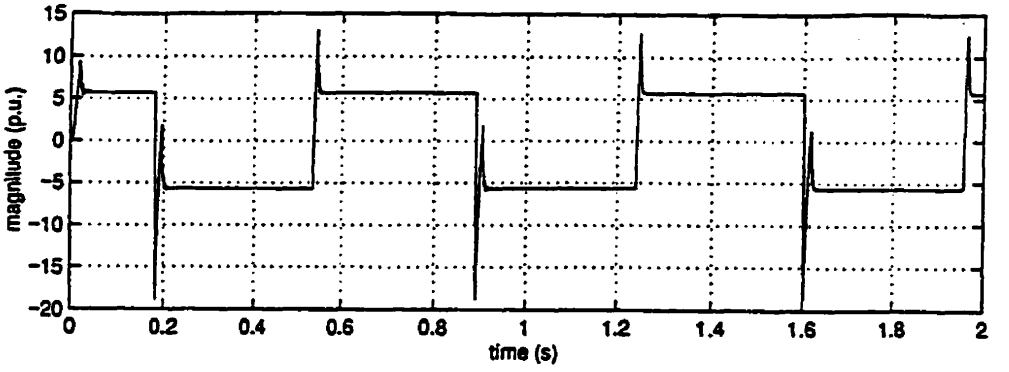


Figure 4.5. Electromagnetic torque with PI controllers

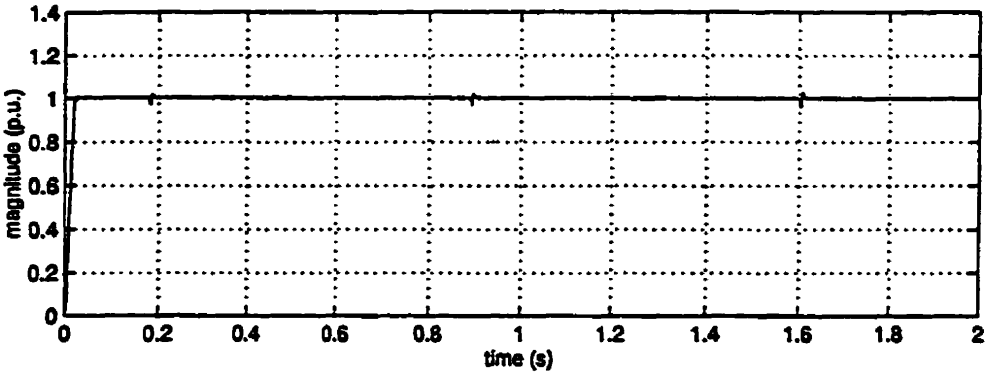


Figure 4.6. Flux magnitude with PI controllers

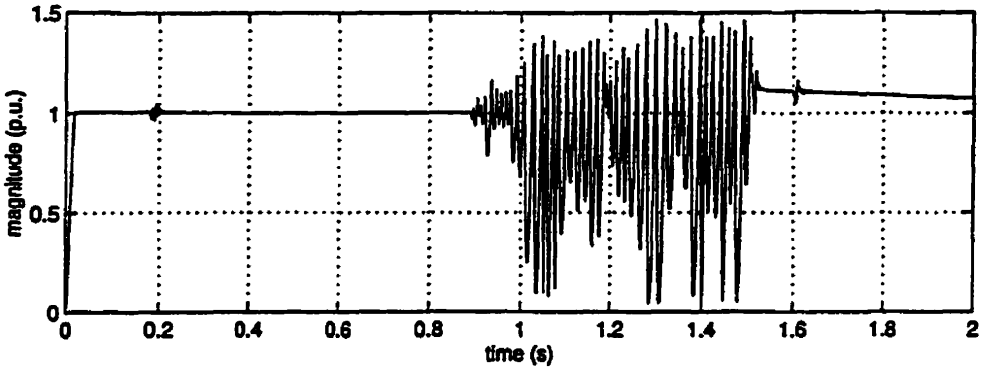


Figure 4.7. Flux magnitude with PI controllers (R_r increased by 10%)

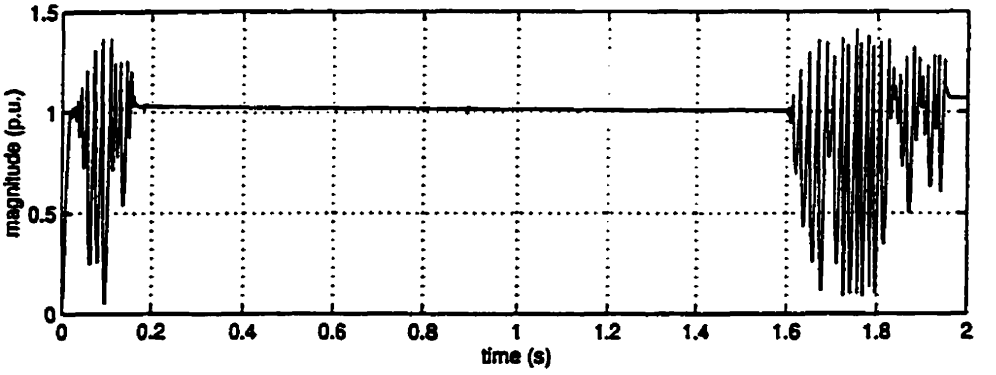


Figure 4.8. Flux magnitude with PI controllers (R_r increased by 20%)

Chapter 5

Discussion

5.1. Command current i_{ds}^*

When modeling the perturbed system for robust control, the torque proportionality constant is found to be a perturbed variable due to the fact that the command current i_{ds}^* changes during the normal operation of the induction motor. This current may therefore be considered as a perturbed value itself in the design of the speed controller. However, a more accurate model would be to treat i_{ds}^* as a feedback quantity. That way, in designing the speed controller, the system can work with the actual, changing value of the command current, rather than with an estimation. As can be seen in Fig. 5.1, the current oscillates around 0.3 p.u.

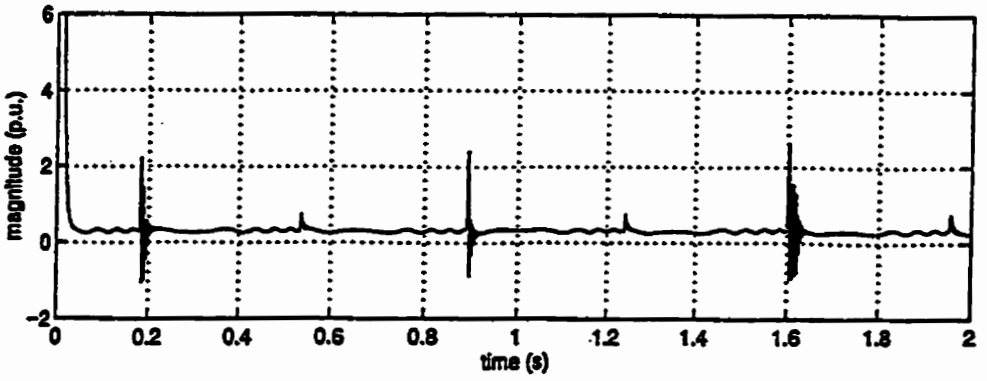


Figure 5.1. i_{ds} command signal in field oriented reference frame

The origin of the spikes in the figure is unknown. It may be because the phase margin is not large enough, resulting in instability during switching, or it may be a problem inherent in the FOC program as these spikes are present even when using PI controllers for flux and speed. However, as seen in Chapter 4, the size of the spikes decreases dramatically with the implementation of robust control in place of PI control. Also, it is noticed that the spikes deteriorate, that is, Fig. 5.1 becomes "cleaner", when the gain-limiting condition for rotor flux is decreased from its nominal value. The setback to this, though, is that the flux magnitude itself deteriorates; it becomes slightly oscillatory in nature, and the magnitude is found to be just below 1 p.u. (which is not desirable).

5.2. Gain-limiting conditions

In Chapter 4, system performance is examined given variations in the rotor resistance. For such cases, the gain-limiting condition imposed on i_{qs}^{*} need not be altered from its nominal value, while the gain-limiting condition for the command current i_{ds}^{*} should be recalibrated slightly. It is found that, as rotor resistance increases, the gain-limiting condition decreases. For nominal conditions, it is 0.315. For a 50% change in R_r, this value is becomes 0.25, a change of approximately 20%. For a 100% change in R_r, this value decreases to 0.18, a change of approximately 40% from the nominal. Normally, this may be significant enough to be considered a drawback in controller design, as this value would have to be slightly altered for rotor resistance variations.

However, it is found that for R_r variances up to 20%, the decrease in the gain-limiting condition is so small (i.e. less than 5%), that it can be considered negligible. It is only for great increases in R_r that this value requires slight modification.

A more dynamic system model is to have R_r as a feedback quantity that is continuously measured during normal operation of the induction motor, and through the FOC program, the gain-limiting condition is continuously adapted to these changes. In this way, the correct flux is always generated.

5.3. Application of an artificial neural network

A neural network "learns" to perform a task after it undergoes a period of training, so that the network is able to generate the desired output when given a valid input [24, 25]. It is this ability, along with the other advantages of multi-layer neural networks, that make them ideally suited for signal processing as well as control applications [26-28]. Theoretically, a three layer artificial neural network can approximate arbitrarily closely any nonlinear, nonsingular function [29]. This has led to research in areas of induction motor control, speed estimation [30], and flux estimation [19].

The potential benefits of neural networks are well known [24 - 26], and are as follows:

(i) Neural network models have many neurons arranged in a massive parallel configuration. Due to this high parallelism, failures of several neurons do not significantly affect overall system performance. This is also known as fault-tolerance.

(ii) Neural networks have a characteristic ability to approximate any nonlinear continuous function to a desired degree of accuracy.

(iii) Neural networks can have many inputs and outputs, making them easily applicable to multivariable systems.

(iv) VLSI hardware implementation of neural networks is possible, resulting in additional speed in neural computing.

(v) Neural networks have inherent learning and adaptive abilities, meaning they can deal with imprecise data and ill-defined situations.

(vi) A suitably trained network has the ability to generalize when presented with inputs not appearing in the trained data.

The addition of a neural network to this flux estimator, then, seems to be a positive step in improving overall system performance. It is hoped that it may be advantageous to combine robust control and neural network rotor flux estimation in a practical drive system. It is expected that the resulting system would be relatively insensitive to parameter variations.

Chapter 6

Conclusions and Future Work

6.1. Conclusions

In this thesis, a novel robust flux controller and a robust speed controller, for the purpose of indirect field oriented control flux estimation, are presented. The research work underlying this thesis is basically divided into two components, field orientation and robust control.

The complex matrix transformations required to perform coordinate conversions from AC quantities of the induction motor to DC quantities of the *d-q* axis model, and vice-versa, used in field oriented control, are detailed in Chapter 2. Robust control theory and the techniques used to design the rotor flux and shaft speed controllers are also outlined in Chapter 2. The proposed FOC drive system, implementing the robust controllers, is discussed in Chapter 3. This drive system is intended for physical implementation with a pulse width modulation (PWM) voltage source inverter. The inputs to the flux estimator are the direct and quadrature stator currents, d^s- q^s, respectively, and the outputs are the flux magnitude, Ψ_r, and the sine and cosine of the field angle ϕ. The system's ability to accurately estimate the flux magnitude and field angle despite parameter variations is verified by test cases and comparisons presented in Chapter 4.

The advantage of the proposed robust controllers over their conventional counterparts is parameter insensitivity in the face of uncertainties. Also, the design of the controllers, based on the loopshaping technique, is a well defined straight forward procedure. In particular, it is shown that the effectiveness of the robust controllers is verified by rotor resistance variation testing. For a doubling of the rotor resistance, the maximum absolute error in the flux magnitude is approximately 0.01 p.u. These factors suggest that it would be practical to physically implement a robust controlled FOC flux estimator.

6.2. Future Work

Some suggestions for future work are:

1. incorporation of a subroutine in the FOC program treating rotor resistance as a feedback variable rather than as a constant, in order to adapt the i_{ds}^{*} gain-limiting condition to meet the changes in R_r.
2. further, to investigate the practicality of such an approach.
3. incorporation of motor non-linearities in the induction motor model.
4. addition of an artificial neural network (ANN) to further improve the performance of the flux estimator.
5. physical implementation of the robust flux and speed controllers by means of a microprocessor or a digital signal processor.

References

1. P. C. Krause, "Analysis of Electric Machinery", McGraw-Hill, 1986, pp. 133 - 179.

2. A. K. P. Toh, "Artificial Neural Network Flux Estimator for Field Oriented Control," *MSc Thesis*, Dept. of Electrical and Computer Engg., University of Calgary, AB, Canada, July 1994.

3. B. K. Bose, "Power Electronics and Variable Frequency Drives," IEEE Press, 1997.

4. D. W. Nowotny, T. A. Lipo, "Introduction to Field Orientation and High Performance AC Drives; Section 2: Principles of Vector Control and Field Orientation," Tutorial course, Industry Application Society Annual Meeting, October 1985.

5. F. Blaschke, "Das Verfahren der Feldorientierung zur Regelung der Asynchronmaschine," Siemens Forsch. -u. Entwickl. -Ber. Bd. 1, Nr. 1, 1972, pp. 184 - 193.

6. F. Blaschke, "The principle of field orientation as applied to the new TRANSVECTOR closed loop control system for rotating field machines," Siemens Review, Vol. 34, May 1972, pp. 217 - 220.

7. K. Hasse, "On the dynamic behavior of induction machines driven by variable frequency and voltage sources," ETZ - A Bd 89 H. 4, 1968, pp. 77 - 81.

8. R. Jonsson, "Natural Field Orientation (NFO) Provides Sensorless Control of AC Induction Servo Motors," PCIM, June 1995, pp. 44 - 51.

9. J. Zhang, "Field Oriented Control of Induction Motor Speed," *MSc Thesis*, Dept. of Electrical Engg., University of Calgary, AB, Canada, 1985.

10. A. Hughes, J. Corda, D. A. Andrade, "Vector Control of cage induction motors: a physical insight," IEE Proc. -Electr. Power Appl., Vol. 143, No. 1, January 1996, pp. 59 - 68.

11. G. M. Liaw, F. J. Lin, "A Robust Speed Controller for Induction Motor Drives," IEEE Transactions on Industrial Electronics, Vol. 41, No. 3, June 1994, pp. 308 - 317.

12. Y. Y. Tzou, "DSP-Based Robust Control of an AC Induction Servo Drive for Motion Control," IEEE Transactions on Control System Technology, Vol. 4, No. 6, November 1996, pp. 614 - 626.

13. F. J. Lin, "Robust speed-controlled induction motor drive using EKF and RLS estimators," IEE Proc. -Electr. Power Appl., Vol. 143, No. 3, May 1996, pp. 186 - 192.

14. J. C. Doyle, B. A. Francis, A. R. Tannenbaum, "Feedback Control Theory," Macmillan Publishing Co., 1992.

15. W. S. Levine, "The Control Handbook," CRC Press & IEEE Press, 1996.

16. M. Green, D. J. N. Limebeer, "Linear Robust Control," Prentice Hall, 1995.

17. U. Dirker, "AC-Flux-Vector Control Improves Induction Motor Performance," PCIM, June 1995, pp. 52 - 59.

18. P. L. Jansen, R. D. Lorenz, "A Physically Insightful Approach to the Design and Accuracy Assessment of Flux Observers for Field Oriented Induction Machine Drives," IEEE Transactions on Industry Applications, Vol. 30, No. 1, January/February 1994, pp. 101 - 110.

19. A. K. P. Toh, E. P. Nowicki, F. Ashrafzadeh, "A Flux Estimator for Field Oriented Control of an Induction Motor using an Artificial Neural Network," Conf. Rec. IEEE Ind. Applicat. Soc. Ann. Meeting, Vol. 1, October 1994, pp. 585 - 592.

20. N. E. Leonard, W. S. Levine, "Using MATLAB to Analyze and Design Control Systems," Benjamin/Cummings Publishing Co., 1992.

21. K. Ogata, "Modern Control Engineering," Prentice Hall, 1990.

22. M. S. Santina, A. R. Stubberud, G. H. Hostetter, "Digital Control System Design," Saunders College Publishing, 1994.

23. B. P. Lathi, "Linear Systems and Signals," Berkely-Cambridge Press, 1992.

24. S. I. Gallant, "Neural Network Learning and Expert Systems," Massachusetts Institute of Technology Press, 1993.

25. W. F. Allman, "Apprentices of Wonder: Inside the Neural Network Revolution," Bantam Books, 1989.

26. P. K. Simpson, "Neural Networks Theory, Technology, and Applications," IEEE Technical Activities Board, 1996.

27. T. Fukuda, T. Shibata, "Theory and application of neural networks for industrial control systems," IEEE Transactions on Industrial Electronics, Vol. 39, No. 6, 1992, pp. 472 - 489.

28. Cybenko, "Approximations by Superpositions of a Sigmoidal Function," Mathematics of Control, Signals and Syst., Vol. 2, 1989, pp. 303 - 314.

29. M. Mohamadian, E. P. Nowicki, F. Ashrafzadeh, J. C. Salmon, "Training of a Neural Network Controller for Indirect Field Oriented Control," 1996 Canadian Conf. on Elec. and Comp. Engg. (CCECE), 1996, pp. 615 - 618.

30. P. Mehotra, J. E. Quaicoe, R. Venkatesan, "Induction Motor Speed Estimation Using Artificial Neural Networks," 1996 Canadian Conf. on Elec. and Comp. Engg. (CCECE), 1996, pp. 607 - 610.

31. J. R. Smith, A. J. Tait, "Electrical drive simulator for teaching purposes," IEE Proceedings, Vol. 135, Pt. A, No. 1, January 1988, pp. 29 - 32.

32. J. R. Smith, M. J. Chen, "Three-phase Electrical Machine Systems: computer simulation," Research Studies Press, 1993.

33. J. R. Smith, "Response Analysis of A.C. Electrical Machines," Research Studies Press, 1990.

Appendix A

A.1. Induction motor model

The induction motor model implemented in the field oriented control scheme is outlined in this appendix. In order to model the dynamic behavior of the induction motor, a computer simulation of the FOC machine is developed. The simulation is based on [2, 31, 32]. The program is written in C code. The induction motor simulator can perform a detailed analysis of different loading conditions and/or voltage disturbances, as outlined in [2] and [33]. It is capable of doing this in the rotor reference frame, stator reference frame or the synchronously rotating reference frame.

The induction motor model is based on the two-axis variable equations detailed in Chapter 2. The program is structured as a main segment and three principal subroutines. The subroutine ME apportions parameter values for matrices and performs matrix inversions. The subroutine RK performs a numerical integration as based on the Runge-Kutta algorithm. The integration step-length is based on achieving an accurate solution during the run-up period, without incurring numerical instability. In this case, a value of 0.001 *sec* is chosen. The subroutine AUX is used by the Runge-Kutta algorithm to recalculate, at each time step, the vector containing i_{qs}, i_{ds}, i_{qr}, and i_{dr}.

A.2. Motor parameters

The induction motor for this drive system is a 30 hp, 415 V, 2 pole, 50 Hz machine [33], with the following parameters:

Table A.1. 30 hp Induction Motor Parameters

Parameters	30 hp
Line Frequency	50 Hz
Stator Resistance, R_s	0.0147 p.u.
Rotor Resistance, R_r	0.0287 p.u..
Stator Reactance, X_s	3.2340 p.u.
Rotor Reactance, X_r	3.2484 p.u.
Leakage Reactance, X_m	3.1568 p.u.
Inertia Constant, H	1.0167 p.u.

A.3. FOC induction motor model program

The following is a C program of the FOC drive system employing robust controllers (using the machine model of Appendix A.2).

```
/*************************************************************
 *                                                           *
 *      The Following program is to give an Estimate         *
 *      of the Rotor Flux Magnitude and Angle in an          *
 *      Induction Machine with the D-Q axis on               *
 *      Synchronous Rotating Ref. frame.                     *
 *                                                           *
 *      The induction motor model is based on the            *
 *      stationary (stator) ref.frame, the numerical         *
 *      method used to accomplish this is Runge-Kutta        *
 *      5th order method (modified).                         *
 *                                                           *
 *      Using motor model fixed on STATOR (stationary        *
 *      reference frame) the sine and cosine of angle        *
 *      phi (i.e the field angle) will be calculated.        *
 *      This program incorporates Field Oriented Control     *
 *      for the induction motor.                             *
 *                                                           *
 *      This program also incorporates a robust flux         *
 *      controller and a robust speed controller in          *
 *      order to compensate for various perturbations        *
 *      (i.e. rotor resistance) and improve performance.     *
 *                                                           *
 *      Author (robust controllers):   Cumbria, Neil         *
 *      Author (FOC):                  Toh, Allan            *
 *      Date:                          April 7th 1997        *
 *      Revised:                       January 17th 1998     *
 *                                                           *
 *************************************************************/


#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define pi 3.14159265359

void ME(double ALI_me[5][5], double R_me[5][5],
    double G_me[5][5], double H_me[5][5], double WE_me,
    double XS_me, double XM_me, double XR_me, double RS_me,
    double RR_me, double XRR_me, double RRR_me, int N_me);
```

```
    int inrun, ndis, itype, iter, ir, tq, k, N, i, j, nes;

    double CHI_r_ERR, CHI_r_ERR1, CHI_r_ERR2, CHI_r_ERR3;
    double Ids_foc_REF, Ids_foc_REF1, Ids_foc_REF2,
    Ids_foc_REF3;
    double WR_ERR, WR_ERR1, WR_ERR2, WR_ERR3;
    double Iqs_foc_REF, Iqs_foc_REF1, Iqs_foc_REF2,
    Iqs_foc_REF3;

    char infile[10];

    FILE *outptr1, *inptr, *outptr2;

    printf("Enter DATA FILENAME please :\n");
    scanf("%s", infile);
    printf("Reading & Processing, Please Wait...\n");

    inptr = fopen(infile, "r");

    outptr1 = fopen("Results.m", "w");
    outptr2 = fopen("VI.m", "w");

/*************************************************************

            Input Data Stream Sequence

   INRUN : No. of iterations before application of
           disturbance
   NDIS  : The ending iteration number of the
           disturbance being applied
   ITYPE : Type of Disturbance,
                    1= Voltage Disturbance
                    2= Load Disturbance
                    3= No Disturbance
   ITER  : Total No. of iterations
   IR    : Type of starting/initial conditions,
                    0= Motor Start
                    1= Read Initial States
   TQ    : Load type,
                    0= Free Acceleration
                    1= Constant Load Torque
                    2= Load Proportional to square
                       of speed
    XS    : Total stator reactance
    XM    : Mutual reactance
    XR    : Total rotor reactance
    RS    : Stator resistance
    RR    : Rotor resistance
    AH    : Inertia constant
    XRR   : Rotor reactance (run mode)
    RRR   : Rotor resistance (run mode)
    DX    : Integration step interval
    FREQ  : Supply frequency
    VT    : Inverter DC bus Voltage
```

```
      TM     : Load torque
      VT1    : Voltage disturbance value
      TM1    : Torque disturbance value
      VT2    : Removal of disturbance voltage value
               (i.e resume value)
      TM2    : Removal of torque disturbance value
               (i.e resume value)

               Variable Dictionary
               -------------------
      c[1] = ids, c[2] = iqs, c[3] = idr, c[4] = iqr
      c[5] = theta slip, c[6] = w_slip
      v[1] = vds, v[2] = vqs, v[6] = load torque(TM)
      CHI_r = rotor flux magnitude

*************************************************************/
      fscanf(inptr,"%d", &inrun);
      fscanf(inptr,"%d", &ndis);
      fscanf(inptr,"%d", &itype);
      fscanf(inptr,"%d", &iter);
      fscanf(inptr,"%d", &ir);
      fscanf(inptr,"%d", &tq);
      fscanf(inptr,"%lf", &XS);
      fscanf(inptr,"%lf", &XM);
      fscanf(inptr,"%lf", &XR);
      fscanf(inptr,"%lf", &RS);
      fscanf(inptr,"%lf", &RR);
      fscanf(inptr,"%lf", &ah);
      fscanf(inptr,"%lf", &XRR);
      fscanf(inptr,"%lf", &RRR);
      fscanf(inptr,"%lf", &dx);
      fscanf(inptr,"%lf", &freq);
      fscanf(inptr,"%lf", &VT);
      fscanf(inptr,"%lf", &TM);

      /*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*/

      /**** CHI_base is different for differnt m/c ****/

/*    CHI_base = 393.7007874; */     /* 10 hp */
      CHI_base = 323.2062055;        /* 30 hp */

      /**** Initial States ****/

      am = ah/(pi*freq);
      WE = WE_base = 2.0*pi*freq;
      WRR = 0.0;
      CHI_r = 0.0;
      CHI_REF = 1.0;
      WR_ERR1 = 0.0;
      WR_ERR2 = 0.0;
      WR_ERR3 = 0.0;
      CHI_r_ERR1 = 0.0;
      CHI_r_ERR2 = 0.0;
```

```
void RK(int N_rk, double dx_rk, int nes_rk, double Y_rk[5],
    double E_rk[5], double v_rk[5], double ALI_rk[5][5],
    double G_rk[5][5], double H_rk[5][5], double R_rk[5][5],
    double am_rk, double *TM_rk, double VT_rk, double *TE_rk,
    double WE_rk, int tq_rk, int inrun_rk, int k_rk,
    double XS_rk, double XM_rk, double XR_rk,
    double *Vds_sta_rk, double *Vqs_sta_rk, double t_rk,
    double *WRR_rk, double *Ids_syn_rk, double *Iqs_syn_rk,
    double *Idr_syn_rk, double *Iqr_syn_rk, double *CHI_r_rk,
    double *CHI_dr_sta_rk, double *CHI_qr_sta_rk,
    double *Sin_Phi_rk, double *Cos_Phi_rk);

void AUX(double c_aux[5], double PC_aux[5], double v_aux[5],
    double ALI_aux[5][5], double G_aux[5][5],
    double H_aux[5][5], double R_aux[5][5], double am_aux,
    double *TM_aux, double VT_aux, double *TE_aux,
    double WE_aux, int tq_aux, int inrun_aux, int k_aux,
    double XS_aux, double XM_aux, double XR_aux,
    double *Vds_sta_aux, double *Vqs_sta_aux, double t_aux,
    double *WRR_aux, double *Ids_syn_aux,
    double *Iqs_syn_aux, double *Idr_syn_aux,
    double *Iqr_syn_aux, double *CHI_r_aux,
    double *CHI_dr_sta_aux, double *CHI_qr_sta_aux,
    double *Sin_Phi_aux, double *Cos_Phi_aux);

void MM(double A_mm[5][5], double B_mm[5][5],
    double C_mm[5][5], int N_mm);


void main()
{
    double c[5], PC[5], v[5], R[5][5], G[5][5], H[5][5],
    ALI[5][5];
    double XS, XM, XR, RS, RR, XRR, RRR, ah, freq, dx, WRR;
    double am, TM, VT, TE, WE, WR, VT1, VT2, TM1, TM2, VTT,
    AIM;
    double Ls, Lm, Lr, t, CHI_r, KP, KPP, KI, WE_base;
    double Iqs_syn_REF, Ids_syn_REF, WSL_REF;
    double Iqs_sta_REF, Ids_sta_REF;
    double VAO, VBO, VCO, Vds_sta, Vqs_sta, Ids_sta,
    Iqs_sta, W_r;
    double Ids_syn, Iqs_syn, Idr_syn, Iqr_syn, Vds_foc,
    Vqs_foc, PHI;
    double Sin_Phi, Cos_Phi, Ids_foc, Iqs_foc, CHI_dr_sta,
    CHI_qr_sta;
    double Iqr_sta, sin_err, cos_err, Sin_Phi_AUX,
    Cos_Phi_AUX, Ids_ERR;
    double Ids_ERR_INTEG, Iqs_ERR_INTEG;
    double Vqs_foc_lim, vds_foc, vqs_foc, CHI_r_ERR_INTEG,
    CHI_REF;
    double KP_CHI, KI_CHI, CHI_base, Idr_sta, Iqs_ERR;
    double Ids_foc_lim, Vds_foc_lim, WR_ERR_INTEG;
    double WR_REF, KP_WR, KI_WR, Iqs_foc_lim;
```

```
CHI_r_ERR3 = 0.0;
WR_REF = 310.0;
Vds_foc_lim = 1.0;
Vqs_foc_lim = 1.0;
Ids_foc_lim = 6.0;
Iqs_foc_REF = 6.0;
Iqs_foc_REF1 = 6.0;
Iqs_foc_REF2 = 6.0;
Iqs_foc_REF3 = 6.0;
Ids_foc_REF1 = 6.0;
Ids_foc_REF2 = 6.0;
Ids_foc_REF3 = 6.0;
Vds_sta = 0.0;
Vqs_sta = 0.0;


/**** P.I. controls coefficients ****/

KP = 0.8;
KI = 400.0;
KP_WR = 0.1;
KI_WR  = 0.05;
KP_CHI = 100.0;
KI_CHI = 100.0;

/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*/

/*********************************************************
        Initial States Read in
        Order in which the states are read in:
           d-axis stator current
           q-axis stator current
           d-axis rotor current
           q-axis rotor current
*********************************************************/
if (ir == 1) {
   for(i=1; i<=4; i++)
      fscanf(inptr,"%lf", &c[i]);
   for(j=1; j<=4; j++)
      fscanf(inptr,"%lf", &v[j]);
}


/***** INITIAL CONDITIONS *****/

if (ir == 0) {
   for(i=1; i<=4; i++){
      c[i] = 0.001;
      v[i] = 0.001;
   }
}
```

```
/***** Dynamic Cycle *****/

fscanf(inptr,"%lf", &VT1);
fscanf(inptr,"%lf", &TM1);
fscanf(inptr,"%lf", &VT2);
fscanf(inptr,"%lf", &TM2);


for(k = 1; k <= iter; k++) {

/***** Perform Matrix inversion *****/

ME(ALI, R, G, H, WE, XS, XM, XR, RS, RR, XRR, RRR, 1);


/**** Disturbance Application *****/

if (k == inrun)
   if (itype == 1) VT = VT1;
   else if (itype == 2) TM = TM1;

/**** Disturbance Removal *****/

if (k == ndis)
   if (itype == 1) VT = VT2;
   else if (itype == 2) TM = TM2;

/**** Perform (modified) Runge-Kutta *****/

t = dx*k;

RK(4, dx, 1, c, PC, v, ALI, G, H, R, am, &TM, VT, &TE,
   WE, tq, inrun, k, XS, XM, XR, &Vds_sta, &Vqs_sta,
   t, &WRR, &Ids_syn, &Iqs_syn, &Idr_syn, &Iqr_syn,
   &CHI_r, &CHI_dr_sta, &CHI_qr_sta, &Sin_Phi_AUX,
   &Cos_Phi_AUX);

Ls = XS/WE;
Lm = XM/WE;
Lr = XR/WE;

/**** Change variable name ****/

Ids_sta = c[1];
Iqs_sta = c[2];
Idr_sta = c[3];
Iqr_sta = c[4];

/***** Calc. sine & cosine of field angle (phi)
       & Calc. Terminal Voltage and Current *********/

W_r = WRR*pi/15.0; /*  Change from RPM to rad/sec */
VTT = sqrt((v[2]*v[2] + v[1]*v[1]));
AIM = sqrt((c[2]*c[2] + c[1]*c[1]));
```

```
WR = W_r;

Sin_Phi = CHI_qr_sta/(CHI_r);
Cos_Phi = CHI_dr_sta/(CHI_r);


/**** Perform Stationary to FOC Co-ordinate change ****/

Ids_foc = Iqs_sta*Sin_Phi + Ids_sta*Cos_Phi;
Iqs_foc = Iqs_sta*Cos_Phi - Ids_sta*Sin_Phi;


/**** Calc. of Controller Command Signals for CHI_r ****/

CHI_r_ERR = (CHI_REF - CHI_r*CHI_base);

Ids_foc_REF = 169.9*CHI_r_ERR + 8.404*CHI_r_ERR1 -
              123.8*CHI_r_ERR2 + 37.71*CHI_r_ERR3 +
              1.756*Ids_foc_REF1 - 0.8976*Ids_foc_REF2 +
              0.1418*Ids_foc_REF3;

Ids_foc_REF = Ids_foc_REF*0.315;

Ids_foc_REF3 = Ids_foc_REF2;
Ids_foc_REF2 = Ids_foc_REF1;
Ids_foc_REF1 = Ids_foc_REF;
CHI_r_ERR3 = CHI_r_ERR2;
CHI_r_ERR2 = CHI_r_ERR1;
CHI_r_ERR1 = CHI_r_ERR;

/* Calc. of P.I. Controller Command Signals for CHI_r */

/*  CHI_r_ERR = (CHI_REF - CHI_r*CHI_base);
    CHI_r_ERR_INTEG = CHI_r_ERR_INTEG + CHI_r_ERR*dx;

    Ids_foc_REF = KP_CHI*CHI_r_ERR + KI_CHI*CHI_r_ERR_INTEG;
*/

/**** Adding Current limiter for Ids foc ref. ****/

if (Ids_foc_REF > Ids_foc_lim) Ids_foc_REF = Ids_foc_lim;


/***** Calc. of Controller Command Signals for Wr *****/

WR_ERR = (WR_REF - WR);

/**** With Iqs_foc_REF as a constant value of 0.25 ****/

/*  Iqs_foc_REF = 15.1552*WR_ERR - 8.1333*WR_ERR1 -
                  15.1427*WR_ERR2 + 8.1459*WR_ERR3 +
                  1.764*Iqs_foc_REF1 - 0.91153*Iqs_foc_REF2
                  + 0.14756*Iqs_foc_REF3;
*/
```

```
/*************************************************************
 *                                                           *
 *    Subroutine : RK                                        *
 *    This subroutine RK uses the Runge-Kutta algorithm      *
 *    to calculate [x]t vector  (Merson Modified)            *
 *                                                           *
 *************************************************************/

void RK(int N_rk, double dx_rk, int nes_rk, double Y_rk[5],
    double E_rk[5], double v_rk[5], double ALI_rk[5][5],
    double G_rk[5][5], double H_rk[5][5], double R_rk[5][5],
    double am_rk, double *TM_rk, double VT_rk,
    double *TE_rk, double WE_rk, int tq_rk, int inrun_rk,
    int k_rk, double XS_rk, double XM_rk, double XR_rk,
    double *Vds_sta_rk, double *Vqs_sta_rk, double t_rk,
    double *WRR_rk, double *Ids_syn_rk, double *Iqs_syn_rk,
    double *Idr_syn_rk, double *Iqr_syn_rk,
    double *CHI_r_rk, double *CHI_dr_sta_rk,
    double *CHI_qr_sta_rk, double *Sin_Phi_rk,
    double *Cos_Phi_rk)

{
    double A[5], B[5], C[5], D[5], h, Z;
    int i;

    h = dx_rk/3.0;

    for(i = nes_rk; i <= N_rk; i++)  D[i] = Y_rk[i];

    AUX(Y_rk, E_rk, v_rk, ALI_rk, G_rk, H_rk, R_rk, am_rk,
        TM_rk, VT_rk, TE_rk, WE_rk, tq_rk, inrun_rk, k_rk,
        XS_rk, XM_rk, XR_rk, Vds_sta_rk, Vqs_sta_rk, t_rk,
        WRR_rk, Ids_syn_rk, Iqs_syn_rk, Idr_syn_rk,
        Iqr_syn_rk, CHI_r_rk, CHI_dr_sta_rk, CHI_qr_sta_rk,
        Sin_Phi_rk, Cos_Phi_rk);

    for(i = nes_rk; i <= N_rk; i++) {
       A[i] = h*E_rk[i];
       Y_rk[i] = D[i] + A[i];
    }

    AUX(Y_rk, E_rk, v_rk, ALI_rk, G_rk, H_rk, R_rk, am_rk,
        TM_rk, VT_rk, TE_rk, WE_rk, tq_rk, inrun_rk, k_rk,
        XS_rk, XM_rk, XR_rk, Vds_sta_rk, Vqs_sta_rk, t_rk,
        WRR_rk, Ids_syn_rk, Iqs_syn_rk, Idr_syn_rk,
        Iqr_syn_rk, CHI_r_rk, CHI_dr_sta_rk, CHI_qr_sta_rk,
        Sin_Phi_rk, Cos_Phi_rk);

    for(i = nes_rk; i <= N_rk; i++) {
       B[i] = h*E_rk[i];
       Y_rk[i] = D[i] + (A[i] + B[i])*0.5;
    }

    AUX(Y_rk, E_rk, v_rk, ALI_rk, G_rk, H_rk, R_rk, am_rk,
```

```
    /***** Input [R] matrix *****/

    R_me[1][1] = RS_me;
    R_me[2][2] = RS_me;
    R_me[3][3] = RR_me;
    R_me[4][4] = RR_me;

    /***** Input [G] matrix in reactance *****/

    G_me[3][2] = XM_me;
    G_me[3][4] = XR_me;
    G_me[4][1] = -XM_me;
    G_me[4][3] = -XR_me;

    /***** Input [H] matrix in inductance *****/

    H_me[3][2] = XM_me/WE_me;
    H_me[3][4] = XR_me/WE_me;
    H_me[4][1] = -XM_me/WE_me;
    H_me[4][3] = -XR_me/WE_me;


    /***** Running Parameters *****/

    if (N_me == 2) {
       ALR = XRR_me/WE_me;
       R_me[3][3] = RRR_me;
       R_me[4][4] = RRR_me;
    }

    /***** Input [L] matrice *****/

    ALI_me[1][1] = ALR;
    ALI_me[1][3] = -ALM;
    ALI_me[2][2] = ALR;
    ALI_me[2][4] = -ALM;
    ALI_me[3][1] = -ALM;
    ALI_me[3][3] = ALS;
    ALI_me[4][2] = -ALM;
    ALI_me[4][4] = ALS;

    /***** CALCULATE [L]^-1 matrice *****/

    u = ALS*ALR - ALM*ALM;
    for(i = 1; i <= 4; i++) {
       for(j = 1; j <= 4; j++) {
          ALI_me[i][j] = ALI_me[i][j]/u;
       }
    }
    return;
}   /* end of subroutine ME */
```

```
    /* Perform FOC. Ref frame to Stat. Ref frame transfm */

    Vds_sta = Vds_foc*Cos_Phi - Vqs_foc*Sin_Phi;
    Vqs_sta = Vqs_foc*Cos_Phi + Vds_foc*Sin_Phi;


    /*** Print out Time, Terminal Voltage, sin (phi),
         Torque, Current, Rotor speed and Flux Mag. ***/

    fprintf(outptr1, "%lf\t %lf\t %lf\t %lf\t %lf\t %lf\t
            %lf\t %lf\n", t, VTT, Sin_Phi, Cos_Phi, TE,
            AIM, W_r/WE_base, CHI_r*CHI_base);

    fprintf(outptr2, "%lf\t %lf\t %lf\t %lf\t %lf\t %lf\t
            %lf\t %lf\n", t, Vds_sta, Vqs_sta, Ids_foc_REF,
            Ids_sta, Iqs_sta, Idr_sta, Iqr_sta);

    } /* k-loop */

} /*End of main() */


/*************************************************************
 *                                                           *
 *          Subroutine : ME                                  *
 *          This subroutine performs matrix inversion        *
 *                                                           *
 *************************************************************/

void ME(double ALI_me[5][5], double R_me[5][5],
    double G_me[5][5], double H_me[5][5], double WE_me,
    double XS_me, double XM_me, double XR_me, double RS_me,
    double RR_me, double XRR_me, double RRR_me, int N_me)

{

    double ALS, ALM, ALR, u;
    int i, j;

    /***** Convert Reactances to Inductances *****/

    ALS = XS_me/WE_me;
    ALM = XM_me/WE_me;
    ALR = XR_me/WE_me;

    for(i = 1; i <= 4; i++){
       for(j = 1; j <= 4; j++){
          ALI_me[i][j] = 0.0;
          R_me[i][j] = 0.0;
          G_me[i][j] = 0.0;
          H_me[i][j] = 0.0;
       }
    }
```

```
/* With Iqs_foc_REF as a perturbed value of 0.3 +- 0.05 */

    Iqs_foc_REF = 12.687*WR_ERR - 6.8087*WR_ERR1 -
                  12.6765*WR_ERR2 + 6.8192*WR_ERR3 +
                  1.7593*Iqs_foc_REF1 - 0.90296*Iqs_foc_REF2
                  + 0.14367*Iqs_foc_REF3;

    Iqs_foc_REF = Iqs_foc_REF*0.01;

    Iqs_foc_REF3 = Iqs_foc_REF2;
    Iqs_foc_REF2 = Iqs_foc_REF1;
    Iqs_foc_REF1 = Iqs_foc_REF;
    WR_ERR3 = WR_ERR2;
    WR_ERR2 = WR_ERR1;
    WR_ERR1 = WR_ERR;

    /** Calc. of P.I. Controller Command Signals for Wr **/

/*  WR_ERR = (WR_REF - WR);
    WR_ERR_INTEG = WR_ERR_INTEG + WR_ERR*dx;

    Iqs_foc_REF = KP_WR*WR_ERR + KI_WR*WR_ERR_INTEG;
*/

    /**** Adding Current limiter for Iqs foc ref. ****/

    if (Iqs_foc_REF>Iqs_foc_lim) Iqs_foc_REF = Ids_foc_lim;


    if (t <= 0.18) Iqs_foc_REF = 6.0;
    if ((0.18 < t) && (t < 0.5350)) Iqs_foc_REF = -6.0;
    if ((0.5350 < t) && (t < 0.89)) Iqs_foc_REF = 6.0;
    if ((0.89 < t) && (t < 1.24)) Iqs_foc_REF = -6.0;
    if ((1.24 < t) && (t < 1.6)) Iqs_foc_REF = 6.0;
    if ((1.6 < t) && (t < 1.955)) Iqs_foc_REF = -6.0;
    if (t >= 1.955) Iqs_foc_REF = 6.0;


 /* Calc. of P.I. Controller Cmd Sig. for Vds & Vqs FOC */

    Ids_ERR = (Ids_foc_REF - Ids_foc);
    Iqs_ERR = (Iqs_foc_REF - Iqs_foc);
    Ids_ERR_INTEG = Ids_ERR_INTEG + Ids_ERR*dx;
    Iqs_ERR_INTEG = Iqs_ERR_INTEG + Iqs_ERR*dx;

    Vds_foc = KP*Ids_ERR + KI*Ids_ERR_INTEG;
    Vqs_foc = KP*Iqs_ERR + KI*Iqs_ERR_INTEG;


    /**** Adding Voltage limiter ****/

    if (Vds_foc > Vds_foc_lim) Vds_foc = Vds_foc_lim;
    if (Vqs_foc > Vqs_foc_lim) Vqs_foc = Vqs_foc_lim;
```

```
        double *Ids_syn_aux, double *Iqs_syn_aux,
        double *Idr_syn_aux, double *Iqr_syn_aux,
        double *CHI_r_aux, double *CHI_dr_sta_aux,
        double *CHI_qr_sta_aux, double *Sin_Phi_aux,
        double *Cos_Phi_aux)
{

        double A[5][5], B[5][5], F[5][5], Z[5][5], it[2][5],
               it_G[2][5];
        double ALS, ALM, ALR, W, WR, WD, it_G_I, T_accel;
        int i, j, l;
        double CHI_dr_syn, CHI_qr_syn, CHI_dr_sta, CHI_qr_sta,
               I[5][2];

        /***** Convert Reactances to Inductances *****/

        ALS = XS_aux/WE_aux;
        ALM = XM_aux/WE_aux;
        ALR = XR_aux/WE_aux;

        /***** Voltages in STATOR Ref. frame******/

        v_aux[1] = *Vds_sta_aux;
        v_aux[2] = *Vqs_sta_aux;

        for(i = 1; i <= 4; i++) {
           for(j = 1; j <= 4; j++) {
              F[i][j] =  *WRR_aux*pi/15.0*H_aux[i][j] +
                         R_aux[i][j];
           }
        }

        MM(ALI_aux, F, Z, 4);

        for(i = 1; i <= 4; i++) {
           for(j = 1; j <= 4; j++) {
                A[i][j] = -Z[i][j];
                B[i][j] = ALI_aux[i][j];
           }
        }

        for(i = 1; i <= 4; i++) {
           PC_aux[i] = 0.0;
           for(j = 1; j <= 4; j++)
              PC_aux[i] = PC_aux[i] + A[i][j]*c_aux[j] +
                          B[i][j]*v_aux[j];
        }


        /***** Converting frame fixed on STATIONARY (Stator)
               frame to frame fixed on SYNCHRONOUS ROTATING
               frame ******/
```

```
        TM_rk, VT_rk, TE_rk, WE_rk, tq_rk, inrun_rk, k_rk,
        XS_rk, XM_rk, XR_rk, Vds_sta_rk, Vqs_sta_rk, t_rk,
        WRR_rk, Ids_syn_rk, Iqs_syn_rk, Idr_syn_rk,
        Iqr_syn_rk, CHI_r_rk, CHI_dr_sta_rk, CHI_qr_sta_rk,
        Sin_Phi_rk, Cos_Phi_rk);

    for(i = nes_rk; i <= N_rk; i++) {
       B[i] = h*E_rk[i];
       Y_rk[i] = D[i] + (A[i] + B[i]*3.0)*0.375;
    }

    AUX(Y_rk, E_rk, v_rk, ALI_rk, G_rk, H_rk, R_rk, am_rk,
        TM_rk, VT_rk, TE_rk, WE_rk, tq_rk, inrun_rk, k_rk,
        XS_rk, XM_rk, XR_rk, Vds_sta_rk, Vqs_sta_rk, t_rk,
        WRR_rk, Ids_syn_rk, Iqs_syn_rk, Idr_syn_rk,
        Iqr_syn_rk, CHI_r_rk, CHI_dr_sta_rk, CHI_qr_sta_rk,
        Sin_Phi_rk, Cos_Phi_rk);

    for(i = nes_rk; i <= N_rk; i++) {
       C[i] = h*E_rk[i];
       Y_rk[i] = D[i] + (A[i] - B[i]*3.0 + C[i]*4.0)*1.5;
    }

    AUX(Y_rk, E_rk, v_rk, ALI_rk, G_rk, H_rk, R_rk, am_rk,
        TM_rk, VT_rk, TE_rk, WE_rk, tq_rk, inrun_rk, k_rk,
        XS_rk, XM_rk, XR_rk, Vds_sta_rk, Vqs_sta_rk, t_rk,
        WRR_rk, Ids_syn_rk, Iqs_syn_rk, Idr_syn_rk,
        Iqr_syn_rk, CHI_r_rk, CHI_dr_sta_rk, CHI_qr_sta_rk,
        Sin_Phi_rk, Cos_Phi_rk);


    for(i = nes_rk; i <= N_rk; i++) {
       Z = D[i];
       D[i] = h*E_rk[i];
       Y_rk[i] = Z + (A[i] + C[i]*4.0 + D[i])*0.5;
    }
    return;
}  /* end of subroutine RK */


/*************************************************************
 *  This subroutine AUX is used by the Runge-Kutta          *
    algorithm to calc. [x]t vector, i.e. iqs, ids,          *
 *  iqr, idr                                                *
 *************************************************************/

void AUX(double c_aux[5], double PC_aux[5],
    double v_aux[5], double ALI_aux[5][5],
    double G_aux[5][5], double H_aux[5][5],
    double R_aux[5][5], double am_aux, double *TM_aux,
    double VT_aux, double *TE_aux, double WE_aux,
    int tq_aux, int inrun_aux, int k_aux, double XS_aux,
    double XM_aux, double XR_aux, double *Vds_sta_aux,
    double *Vqs_sta_aux, double t_aux, double *WRR_aux,
```

```
        *Ids_syn_aux = c_aux[1]*(double)cos((double)
                       WE_aux*t_aux) + c_aux[2]*(double)
                       sin((double)WE_aux*t_aux);
        *Iqs_syn_aux = -c_aux[1]*(double)sin((double)
                       WE_aux*t_aux) + c_aux[2]*(double)
                       cos((double)WE_aux*t_aux);

        *Idr_syn_aux = c_aux[3]*(double)cos((double)
                       WE_aux*t_aux) + c_aux[4]*(double)
                       sin((double)WE_aux*t_aux);
        *Iqr_syn_aux = -c_aux[3]*(double)sin((double)
                       WE_aux*t_aux) + c_aux[4]*(double)
                       cos((double)WE_aux*t_aux);


        /**** Calc. TE = it*[G]*I, it = I-transpose ****/

        for(l = 1; l <= 4; l++) {
           it_G[1][l] = 0.0;
           for(j = 1; j <= 4; j++) {
                it[1][j] = c_aux[j];
                it_G[1][l] = it_G[1][l] +
                             it[1][j]*G_aux[j][l];
           }
        }

        it_G_I = 0.0;
        for(j = 1; j <= 4; j++) {
           I[j][1] = c_aux[j];
           it_G_I = it_G_I + it_G[1][j] * I[j][1];
        }

        *TE_aux = it_G_I;

        if (tq_aux == 0) T_accel = *TE_aux - *TM_aux;

        if (tq_aux == 2) {
           WD = (*WRR_aux*pi/15.0)/(WE_aux);
           *TM_aux = 1.1*WD*WD;
        }

        if (tq_aux == 1) {
*TM_aux = 1.0;
        }

        T_accel = *TE_aux - *TM_aux;


        /*** Calc. of WR using Euler's approx.
           (step size = 0.001) ***/

        *WRR_aux = *WRR_aux + ((T_accel)/am_aux)*0.001;
        /* in RPM */
```

```
    /***  Calc. d-q Flux Magnitude in Syn. frame ***/

    CHI_dr_syn=ALR*(*Idr_syn_aux)+ALM*(*Ids_syn_aux);
    CHI_qr_syn=ALR*(*Iqr_syn_aux)+ALM*(*Iqs_syn_aux);


    /* Calc. d-q Flux Magnitude in Stationary frame */

    *CHI_dr_sta_aux = ALR*(c_aux[3]) + ALM*(c_aux[1]);
    *CHI_qr_sta_aux = ALR*(c_aux[4]) + ALM*(c_aux[2]);


    /****  Rotor Flux Magnitude *****/

    *CHI_r_aux = sqrt((*CHI_dr_sta_aux*
                 (*CHI_dr_sta_aux)) +
                 (*CHI_qr_sta_aux*(*CHI_qr_sta_aux)));


    /****  Rotor Flux Angle (Field Angle, Phi) *****/

    *Sin_Phi_aux = *CHI_qr_sta_aux/(*CHI_r_aux);
    *Cos_Phi_aux = *CHI_dr_sta_aux/(*CHI_r_aux);


    return;
} /* end of subroutine AUX  */


/*************************************************************
 *  This subroutine MM is used by AUX subroutine             *
 *  to multiply matrices to obtain [L]^-1(Wr[G] + [R])       *
 *************************************************************/
void MM(double A_mm[5][5], double B_mm[5][5],
    double C_mm[5][5], int N_mm)

{
    int i, j, l;

    for(i = 1; i <= 4; i++) {
       for(l = 1; l <= 4; l++) {
          C_mm[i][l] = 0.0;
          for(j = 1; j <= 4; j++) {
             C_mm[i][l] = C_mm[i][l] + A_mm[i][j]*B_mm[j][l];
          }
       }
    }
    return;
}  /* end of subroutine MM  */
```

Appendix B

B.1. W_2 for flux controller

```
% Program for determination of W2 for Flux Controller
% with Rotor Resistance
% by Neil Cumbria

clear;

w = logspace(-3,4,500);
mag =zeros(1,500);
i = 1;                          % index for mag matrix
                                % first row are all zeros

%       The nominal plant conditions are given by:
Lm = 3.1568/(2*pi*60);
Lr = 3.2484/(2*pi*60);
Rr_nom = 0.0287;                % up to 50% perturbation

%       The variables to be perturbed are tau, the
%       drive dead-time, and Rr, the rotor resistance.
%       The frequency response of the perturbed plant
%       is derived as follows:

clf;
figure(1)
        for tau = 0.02:0.001:0.03
          num = [-Lm*tau/2 Lm];
            for Rr = Rr_nom:0.001:1.5*Rr_nom
              den = [(Lr/Rr)*tau/2 (Lr/Rr)+(tau/2) 1];
              i = i+1;
              [magi phase w] = bode(num,den,w);
              mag = [mag; (20*log10(magi))'];
              %mag = [mag; (magi)'];
              semilogx(w,mag(i,:));
            hold on;
            end;
        end;

title('Frequency Response of Perturbed Plant');
xlabel('frequency (rad/s)');
ylabel('magnitude');

%       W2 is determined by
%       |P_perturbed(jw)/P_nom(jw) - 1|
%       whereby this equation is reduced to
%       incorporate the perturbed variables.
```

```
figure(2)
for tau = 0.02:0.01:0.03
        for Rr = Rr_nom:0.001:1.5*Rr_nom
          num = [-(Lr/Rr)*tau -tau 0];
          den = [(Lr/Rr)*tau/2 (Lr/Rr)+(tau/2) 1];
          i = i+1;
          [magi phase w] = bode(num,den,w);
          %mag = [mag; (20*log10(magi))'];
          mag = [mag; (magi)'];
          loglog(w,mag(i,:));
          hold on;
          end;
        end;

num_W2 = [0.04 0];
den_W2 = [0.01 1];

%       W2 = (0.04s)/(0.01s+1)

[mag_W2, phase_W2, w] = bode(num_W2, den_W2, w);
loglog(w, mag_W2, '--');
title('Magnitude plot of W2');
xlabel('frequency (rad/s)');
ylabel('magnitude');
```

B.2. W_2 for speed controller

```
% Program for determination of W2
% for the Speed Controller
% by Neil Cumbria

clear;

w = logspace(-3,4,500);
mag =zeros(1,500);
i = 1;                      % index for mag matrix
                            % first row are all zeros

%       The nominal plant conditions are given by:

Lm = 3.1568/(2*pi*50);
Lr = 3.2484/(2*pi*50);
Rr_nom = 0.0287;
p = 2;
Ids = 0.3;

Kt_nom = Ids*(3/4)*p*Lm*Lm/Lr % without rotor
                              % resistance
%Kt_num = Ids*(3/4)*p*Lm*Lm;  % with rotor
                              % resistance
%Kt_den = Lr*[Lr/Rr_nom 1];   % with rotor
                              % resitance
B = 0.0001;  % can't be zero (linear freq response)
J_nom = 0.0167;

%       The variables to be perturbed are tau,
%       the drive dead-time, and J, the inertia.
%       The frequency response of the perturbed plant
%       is derived as follows:

clf;
figure(1)
        for tau = 0.02:0.001:0.03
          for Kt = 0.8333*Kt_nom:0.1:1.1667*Kt_nom
                num = [-Kt*tau/2 Kt];
                  for J = 0.5*J_nom:0.01:5*J_nom
                        den = [J*tau/2 (B*tau/2)+J B];
                        i = i+1;
                        [magi phase w]=bode(num,den,w);
                        mag = [mag; (20*log10(magi))'];
                        %mag = [mag; (magi)'];
                        semilogx(w,mag(i,:));
                   hold on;
                   end;
          end;
        end;

title('Frequency Response of Perturbed Plant');
xlabel('frequency (rad/s)');
ylabel('magnitude');
```

```
%       W2 is determined by
%       |P_perturbed(jw)/P_nom(jw) - 1|
%       whereby this equation is reduced to
%       incorporate the perturbed variables.

figure(2)
for tau = 0.02:0.01:0.03
        for J = 0.5*J_nom:0.1:5*J_nom
          num = [-J*tau -B*tau 0];
          den = [J*tau/2 (B*tau/2)+J B];
          i = i+1;
          [magi phase w] = bode(num,den,w);
          %mag = [mag; (20*log10(magi))'];
          mag = [mag; (magi)'];
          loglog(w,mag(i,:));
          hold on;
        end;
end;

num_W2 = [0.04 0];
den_W2 = [0.01 1];

%       W2 = (0.04s)/(0.01s+1)

[mag_W2, phase_W2, w] = bode(num_W2, den_W2, w);
loglog(w, mag_W2, '--');
title('Magnitude plot of W2');
xlabel('frequency (rad/s)');
ylabel('magnitude');
```

B.3. Loopshaping program for design of flux controller

```
% Loopshaping Program for the determination
% of the Flux Controller with Rotor Resistance
% by Neil Cumbria

% The nominal plant conditions are given by:

Lm = 3.1568/(2*pi*50);              % ( X = 2*pi*f*L)
Lr = 3.2484/(2*pi*50);
Rr_nom = 0.0287;

num_P = Lm;
den_P = [Lr/Rr_nom 1];

%       Determination of W1:
%       For mechanical systems with relatively
%       large inertia a cut off frequency of
%       1 rad/s may be large enough.
%       A third order Butterworth filter for W1
%       is derived from the Matlab 4.2c function:
%       butter(3,1,'s');
%       W1 = 1.05/(s^3 + 2s^2 + 2s + 1)    max = 1
%                                  f_corner = 1 rad/s

num_W1 = 1.05;          den_W1 = [1 2 2 1];
num_W2 = [0.04 0];      den_W2 = [0.01 1];
w = linspace(0.001,1000,5000);

%       Determination of W2:
%       W2 = (0.04s)/(0.01s+1)
%                                  max = 4
%                                  f_cross = 20 rad/s

[mag_P phase_P] = bode(num_P,den_P,w);
[mag_W1 phase_W1] = bode(num_W1,den_W1,w);
[mag_W2 phase_W2] = bode(num_W2,den_W2,w);

%       Relative degree of L >= relative degree
%       of P in order to roll off as fast as P
%       so that C is proper.

%       L incorporates an integrator to yiels a
%       better unit-step response, and to decrease
%       the tracking error.
%       L(s) = 36(s+2)/s(s^2+6s+9)

num_L = 36*[1 2];
den_L = poly([0 -3 -3]);

%       L(0) > 0;
%       The roots of 1+L=0 are: -2.0303 + 5.7448i
%                               -2.0303 - 5.7448i
%                               -1.9394
%       verifying that this L(s) gives nominal
%       stability.
```

```
[mag_L phase_L] = bode(num_L,den_L,w);
mag_S = 1./(1.+mag_L);
mag_T = mag_L./(1.+mag_L);

%       |W1S| + |W2T| calculation

mag_W3 = (mag_W1.*mag_S) + (mag_W2.*mag_T);

figure(1);
loglog(w, mag_W1, w, mag_W2, w, mag_L);
title('Plot of L, W1, and W2');
xlabel('frequency (rad/s)');
ylabel('magnitude');
grid;

mag_W4 = mag_W1./(1.-mag_W2);
mag_W5 = (1.-mag_W1)./mag_W2;

figure(2);
loglog(w, mag_W4, w, mag_L, w, mag_W5);
title('Magnitude Plot of L');
xlabel('frequency (rad/s)');
ylabel('magnitude');
grid;

figure(3);
loglog(w, mag_W3);
title('Magnitude Plot of |W1S|+|W2T|');
xlabel('frequency (rad/s)');
ylabel('magnitude');
grid;

%       check error in output sinusoid when w <= 1

a = mag_L.*(1.-mag_W2);
index = max(find(w < 1));
b = a(1:index);
sine_error = 1/min(b)

%       error in the output sinusoid = 0.1029,
%       which decreases as the gain is increased,
%       but performance condition (fig. 6) tends
%       toward 1

figure(4);
plot(w(1:index),mag_S(1:index));
title('Output Sinusoid Error for w <= 1 rad/s');
grid;

%       NOTE:
%       The gain is constrained by the condition
%       that at frequencies higher than the
%       cross over frequency at W2, the magnitude
%       of L < (1-|W1|)/|W2|
```

```
%       Check
%       syst

figure(5);
freq = logsp
nyquist(num_L
title('Nyqui

figure(6);
margin(num_L,

%       Unit

num_L = [0 0
num_cl = num
den_cl = num

figure(7);
step(num_cl,d
title('Unit S

%       Param

final_value =
[y,x,t] = ste
[Y,k] = max(y
time_to_peak
percent_overs

%       Deter

n = 1;
        while
        n=n+1
        end
m = 1;
        while
        m=m+1
        end
rise_time = t

%       Deter

l = length(t)
        while

        l = l
        end
settling_time

%       Deter
%       C(s)
%       num_C
%       den_C
```

```
num_C = conv(num_L, den_P)
den_C = conv(den_L, num_P)

num_C = [12.9700 61.9400 72.0000];
den_C= [0.0100 0.0603 0.0904 0];

%       Convert controller from continuous to
%       discrete time system.

Ts = 0.3;
[num_Cd,den_Cd] = c2dm(num_C,den_C,Ts,'tustin')
printsys(num_C,den_C,'s')
printsys(num_Cd,den_Cd,'z')

%       Determination of unit step response for
%       the perturbed system.

for tau = 0.02:0.01:0.03
  for Rr = Rr_nom:0.001:1.5*Rr_nom
        num_P = Lm;
        den_P = [Lr/Rr 1];
        num_tau = [1 -tau/2];
        den_tau = [1 tau/2];
        [num_ol den_ol]=series(num_C,den_C,
                        num_P,den_P);
        [num_ol den_ol]=series(num_ol,den_ol,
                        num_tau,den_tau);
        [num_cl den_cl]=feedback(num_ol,den_ol,
                        -1);
        figure(8);
        step(num_cl,den_cl);
        title('Step Response of Perturbed
              System');
        hold on;
  end;
end;
axis([0 5 0 1.5]);
```

B.4. Loopshaping program for design of speed controller

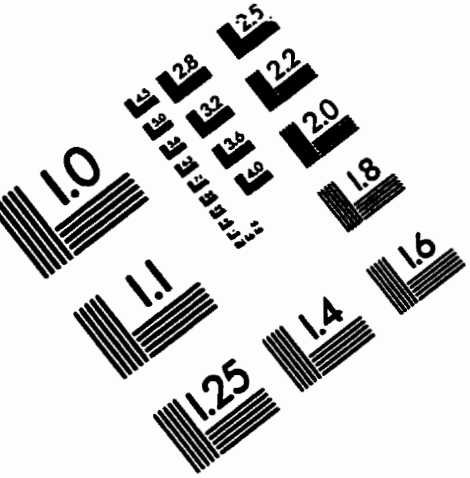
```
% Loopshaping Program for the determination
% of the Speed Controller
% by Neil Cumbria

% The nominal plant conditions are given by:

Lm = 3.1568/(2*pi*50);
Lr = 3.2484/(2*pi*50);
Rr_nom = 0.0287;
p = 2;
Ids = 0.3;

Kt_nom = Ids*(3/4)*p*Lm*Lm/Lr    % without rotor
                                 % resistance
%Kt_num = Ids*(3/4)*p*Lm*Lm;     % with rotor
                                 % resistance
%Kt_den = Lr*[Lr/Rr_nom 1];      % with rotor
                                 % resitance
B = 0.0001; % can't be zero (linear freq response)
J_nom = 0.0167;

num_P = Kt_nom;
den_P = [J_nom B];

%       Determination of W1:
%       For mechanical systems with relatively
%       large inertia a cut off frequency of
%       1 rad/s may be large enough.
%       A third order Butterworth filter for W1
%       is derived from the Matlab 4.2c function:
%       butter(3,1,'s');
%       W1 = 1.05/(s^3 + 2s^2 + 2s + 1)   max = 1
%                                 f_corner = 1 rad/s

num_W1 = 1.05;           den_W1 = [1 2 2 1];
num_W2 = [0.04 0];       den_W2 = [0.01 1];
w = linspace(0.001,1000,5000);

%       Determination of W2:
%       W2 = (0.04s)/(0.01s+1)
%                                 max = 4
%                                 f_cross = 20 rad/s

[mag_P phase_P] = bode(num_P,den_P,w);
[mag_W1 phase_W1] = bode(num_W1,den_W1,w);
[mag_W2 phase_W2] = bode(num_W2,den_W2,w);

%       Relative degree of L >= relative degree
%       of P in order to roll off as fast as P
%       so that C is proper.
```

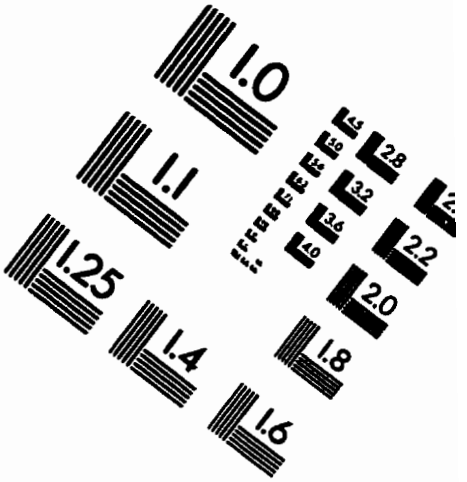
```
%       L incorporates an integrator to yiels a
%       better unit-step response, and to decrease
%       the tracking error.
%       L(s) = 36(s+2)/s(s^2+6s+9)

num_L = 36*[1 2];
den_L = poly([0 -3 -3]);
%       L(0) > 0;
%       The roots of 1+L=0 are: -2.0303 + 5.7448i
%                               -2.0303 - 5.7448i
%                               -1.9394
%       verifying that this L(s) gives nominal
%       stability.

[mag_L phase_L] = bode(num_L,den_L,w);
mag_S = 1./(1.+mag_L);
mag_T = mag_L./(1.+mag_L);

%       |W1S| + |W2T| calculation

mag_W3 = (mag_W1.*mag_S) + (mag_W2.*mag_T);

figure(1);
loglog(w, mag_W1, w, mag_W2, w, mag_L);
title('Plot of L, W1, and W2');
xlabel('frequency (rad/s)');
ylabel('magnitude');
grid;

mag_W4 = mag_W1./(1.-mag_W2);
mag_W5 = (1.-mag_W1)./mag_W2;

figure(2);
loglog(w, mag_W4, w, mag_L, w, mag_W5);
title('Magnitude Plot of L');
xlabel('frequency (rad/s)');
ylabel('magnitude');
grid;

figure(3);
loglog(w, mag_W3);
title('Magnitude Plot of |W1S|+|W2T|');
xlabel('frequency (rad/s)');
ylabel('magnitude');
grid;

%       check error in output sinusoid when w <= 1

a = mag_L.*(1.-mag_W2);
index = max(find(w < 1));
b = a(1:index);
sine_error = 1/min(b)

%       error in the output sinusoid = 0.1029,
%       which decreases as the gain is increased,
%       but performance condition (fig. 6) tends
%       toward 1
```

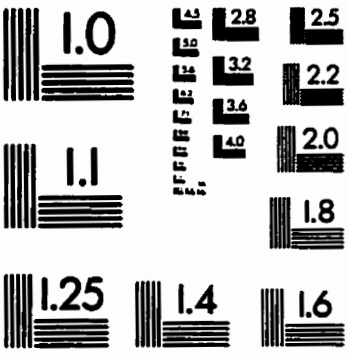
```
figure(4);
plot(w(1:index),mag_S(1:index));
title('Output Sinusoid Error for w <= 1 rad/s');
grid;

%       NOTE:
%       The gain is constrained by the condition
%       that at frequencies higher than the
%       cross over frequency at W2, the magnitude
%       of L < (1-|W1|)/|W2|

%       Check stability of nominal closed loop
%       system

figure(5);
freq = logspace(-0.5,4,500);
nyquist(num_L,den_L,freq);
title('Nyquist Plot of L');

figure(6);
margin(num_L,den_L);

%       Unit step response of the system

num_L = [0 0 num_L];
num_cl = num_L;          % closed loop numerator
den_cl = num_L+den_L;    % closed loop denominator

figure(7);
step(num_cl,den_cl);
title('Unit Step Response');

%       Parameters of Step Response

final_value = polyval(num_cl,0)/polyval(den_cl,0)
[y,x,t] = step(num_cl,den_cl);
[Y,k] = max(y);
time_to_peak = t(k)
percent_overshoot = 100*(Y-final_value)/final_value

%       Determination of Rise Time

n = 1;
        while y(n)<0.1*final_value
        n=n+1;
        end
m = 1;
        while y(m)<0.9*final_value
        m=m+1;
        end
rise_time = t(m) - t(n)
```

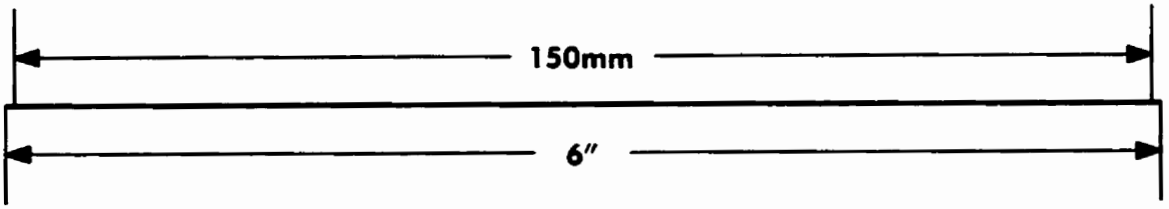
```
%       Determination of Settling Time

l = length(t);
        while (y(l)>0.98*final_value)&
              (y(l)<1.02*final_value)
        l = l - 1;
        end
settling_time = t(l)

%       Determination of C:
%       C(s) = L(s)/P(s);
%       num_C = 0.0252s^2 + 0.0540s + 0.0072
%       den_C = s(0.8779s^2 + 5.2674s + 7.9011)

num_C = conv(num_L, den_P)
den_C = conv(den_L, num_P)

num_C = [0.6012 1.2060 0.0072];
den_C= [0.0044 0.0264 0.0395 0];

%       Convert controller from continuous to
%       discrete time system.

Ts = 0.3;
[num_Cd,den_Cd] = c2dm(num_C,den_C,Ts,'tustin')
printsys(num_C,den_C,'s')
printsys(num_Cd,den_Cd,'z')

%       Determination of unit step response for the
%       perturbed system.

for tau = 0.02:0.01:0.03
  for J = 0.5*J_nom:0.01:5*J_nom
     for Kt = 0.8333*Kt_nom:0.1:1.1667*Kt_nom
         num_P = Kt;
        den_P = [J B];
        num_tau = [1 -tau/2];
        den_tau = [1 tau/2];
        num_gain = 1;
         den_gain = 1;
        [num_ol den_ol]=series(num_C,den_C,num_P,
                        den_P);
        [num_ol den_ol]=series(num_ol,den_ol,
                        num_tau,den_tau);
        [num_cl den_cl]=feedback(num_ol,den_ol,
                        num_gain,den_gain,-1);
         figure(8);
         step(num_cl,den_cl);
         title('Step Response of Perturbed System');
         hold on;
     end;
  end;
end;
axis([0 5 0 1.5]);
```

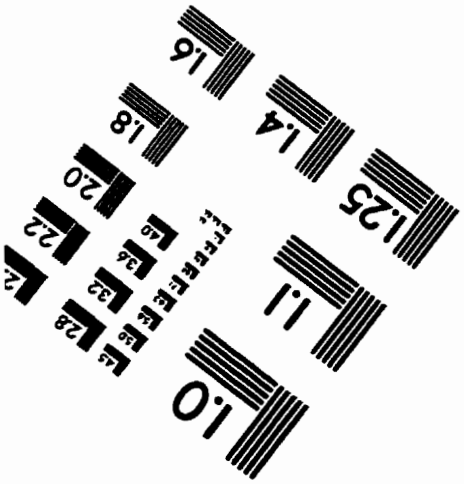
IMAGE EVALUATION
TEST TARGET (QA-3)

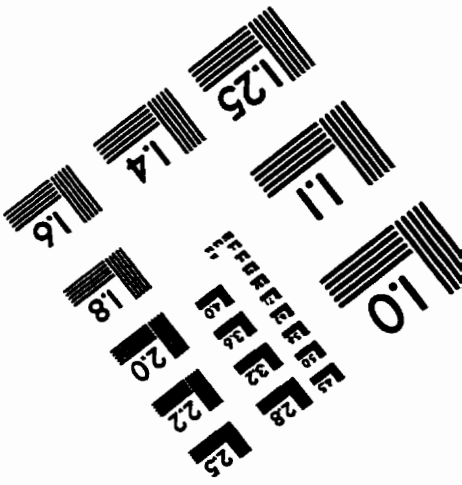












APPLIED IMAGE, Inc
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989