2005

IMPLEMENTATION AND VALIDATION OF FAULT TOLERANT CONTROL OF A SELF-BEARING MOTOR CONSIDERING OPEN COIL FAULTS

Anand Ranganathan
University of Kentucky

ABSTRACT OF THESIS

IMPLEMENTATION AND VALIDATION OF FAULT TOLERANT CONTROL OF A SELF-BEARING MOTOR CONSIDERING OPEN COIL FAULTS

Self-bearing motor is a magnetic actuator with both bearing and motoring functionality. This work implements and validates a decoupled and fault tolerant control algorithm for the Lorentz self bearing motor containing open phase faults. The goal of the algorithm is to achieve a stable bearing force and motoring torque even with coil faults. This work simulates many non-real-time fault tolerant control models based on the algorithm using simulink. Test cases are designed in simulink and tested on these models to arrive at the best model that could be implemented in dspace for real-time control. The responses of these simulations are compared with the desired output. Simulations showed that the decoupled and fault tolerant control model does not have any cross coupling and was fault tolerant for many combinations of open phase faults. Simulink model was modified so that it was auto-complied into the dspace controller and dynamically linked with the hardware. A graphical user interface was provided for fault tolerant control in controldesk software and the motor was controlled in real-time. Many experiments are designed to test the fault tolerant control model. Experimental results validate fault tolerance in the motor with respect to open coil faults. The self-bearing motor was found to be more stable in decoupled and fault tolerant control than non-fault tolerant control.

Anand Ranganathan

May 19, 2005

IMPLEMENTATION AND VALIDATION OF FAULT TOLERANT CONTROL OF A SELF-BEARING MOTOR CONSIDERING OPEN COIL FAULTS

By

Anand Ranganathan

Dr. Lyndon Scott Stephens
_____
Director of Thesis

Dr. George Huang
_____
Director of Graduate Studies

May 19, 2005
_____
(Date)

RULES FOR THE USE OF THESES

Unpublished theses submitted for the Master's degree and deposited in the University of Kentucky Library are as a rule open for inspection, but are to be used only with due regard to the rights of the authors. Bibliographical references may be noted, but quotations or summaries of parts may be published only with the permission of the author, and with the usual scholarly acknowledgements.

Extensive copying or publication of the thesis in whole or in part also requires the consent of the dean of the Graduate School of the University of Kentucky.

A Library that borrows this thesis for use by its patrons is expected to secure the signature of each user.

Name
Date

_____

_____

_____

_____

_____

_____

_____

_____

THESIS

Anand Ranganathan

The Graduate School

University of Kentucky

2005

IMPLEMENTATION AND VALIDATION OF FAULT TOLERANT CONTROL OF SELF-BEARING MOTOR WITH RESPECT TO OPEN COIL FAULTS

THESIS

By

Anand Ranganathan

Lexington, Kentucky

Director: Dr. Lyndon Scott Stephens

Lexington, Kentucky

2005

MASTER'S THESIS RELEASE

I authorize the University of Kentucky
Libraries to reproduce this thesis
in whole or in part for purposes of research

Signed: Anand Ranganathan

Date: May 19, 2005

ACKNOWLEDGEMENTS

I am grateful to my advisor, Dr. Scott Stephens for his insightful guidance and advice. I thank him immensely for offering me this incredible opportunity to do research. Without his assistance, this thesis would not have been possible. I would like to thank him for all the knowledge that he has imparted to me during the course of study at University of Kentucky.

I would like to extend my appreciation to the members of my thesis committee Dr. Johne Parker and Dr. Keith Rouch for their support.

I am indebted to my lab members for sharing their thoughtful insights and findings during many discussions we had on this area of research. I would like to thank Dr. Zhaohui Ren, Kathy Warren, Hooi-Mei Chin and Barrett Steele from the magnetic bearing group in the Bearings and Seals lab, for advice and support. Thanks to other members of the Bearing and Seals lab for informative discussions on various topics related to my project.

Further more, I would like to express my appreciation to Dr. Stephens and Jason Payton for their valuable suggestions after careful reading and proofing of the manuscript.

I would also like to offer a special note of thanks to my parents for their unrelenting support and encouragement.

Table of contents

List of Figures

List of tables

NOMENCLATURE

$\overline{B_m}$	PM Flux Density
$B_{m,k,}$	Air gap flux due to the PM's,
$B_{w,k}$	Air gap flux due to the windings
$\overline{B_{w,k}}$	Amplitude of the sinusoidal approximation to the winding flux
I_k.	Winding current distribution
$K_{i\xi}$	Torque Current Gain
K_{ixx}	Direct Force Current Gain
K_{ixy}	Cross Coupled Force Gain
$K_{xy,w}$	Winding flux side pull
L	Motor Length
M	Number of Pole Pairs
N_{seg}	Number of Segments
N_s	Number of Winding Stations per Segment
N_w	Number of Wires per Winding Station
R	Rotor Outer Radius
i_x	Control current in x direction
i_y	Control current in y direction
i_ξ	Control current in ξ direction
i_k	Segment current
$i_{k,1}$	First phase current in the kth segment of the motor
$i_{k,2}$	Second phase current in the kth segment of the motor
$i_{k,3}$	Third phase current in the kth segment of the motor
$\overline{i_k}$	Amplitude value of segment current
γ	Phase angle of the current with respect to the permanent magnet flux
ϕ	Global angular coordinate
ψ_k	kth winding segment relative to the x-axis
θ	Local segment angle.

Chapter 1

Introduction

1.1 Magnetic Bearing

Active electromagnetic levitation is based on the attractive force of a controllable electromagnet on a ferromagnetic body. A control unit adjusts the current in an electromagnet. Hence the magnetic force acts on the ferromagnetic body, so that the body is held in suspension. A sensor continuously measures the position of the ferromagnetic body. Assume that there is a single electromagnet placed over the top of the ferromagnetic body with an air gap between them in the y-direction. If the ferromagnetic body is above the desired position, the controller reduces the current in the magnet and with it the magnetic force. If the body is below the desired position, the current in the magnet is increased. The sensor detects the position of the rotor and sends a voltage signal v_p to the controller. The controller generates perturbation voltage v_p in turn and sends it to the power amplifier, which produces the required current i. This current causes the electromagnetic force and keeps the rotor afloat against the force of gravitation. A single electromagnet is incapable of stabilizing all spatial degrees of freedom of a rotor. Two electromagnets arranged in an opposed pair are needed just to orient the position of a rotor in one direction. Two such pairs of electromagnets positioned at right angles to each other form a "radial bearing." Like a

ball bearing, this configuration is capable of holding a rotor in one position in a plane (x-y direction).

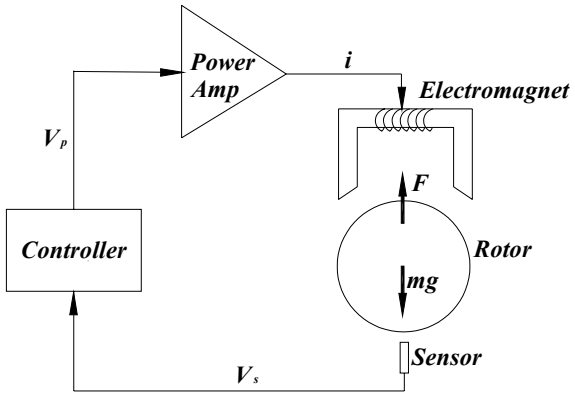


Figure 1.1: Magnetic Bearing

1.2 Disadvantages of magnetic bearings:

- Requires separate driving motor for rotation, hence the shaft is longer. This leads to unwanted vibration at low frequencies
- Excessive heat generation
- High power consumption

1.3 Lorentz Self-Bearing Motor

The deficiency of having a separate driving motor was overcome by another type of magnetic bearing. This new generation of magnetic bearing is called the Lorentz self-bearing motor. Most of the self-bearing motors have one more sensor in addition to the sensors in the magnetic bearing, which is called the optical encoder. The optical sensor senses the angular position of the rotor and sends it to the controller. There is one more PID controller which is required for the angular direction in addition to the two PID controllers for x and y directions. With the help of more electromagnets and an

appropriate control algorithm, the coils are energized for the shaft to rotate in θ direction, in addition to x and y.

The motor that is used in the Bearings and Seals Laboratory at the University of Kentucky uses Lorentz forces to generate the forces and the torque. The figure 1.4 shows how Lorentz force is developed in a motor. The Lorentz force is orthogonal to the direction of the current and magnetic field lines as shown. The direction of the force is given by Fleming's right hand rule.

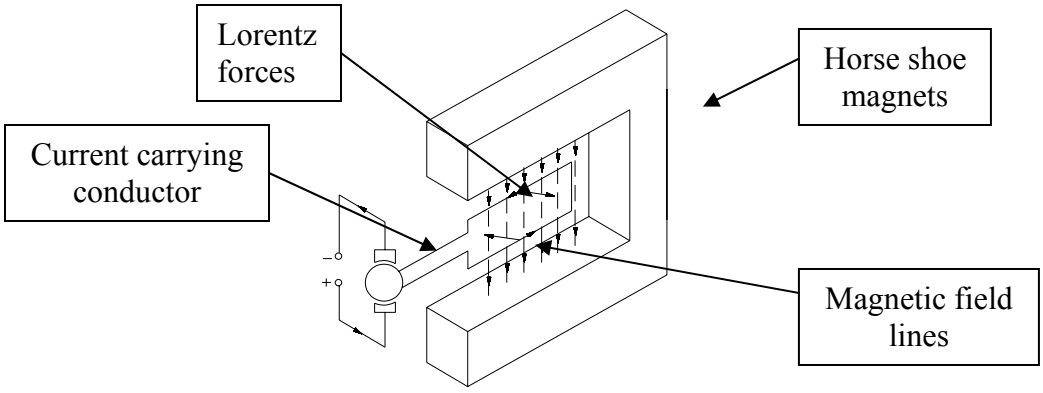


Figure 1.2: Lorentz force motor

The self-bearing motor is a magnetic bearing which can be used to levitate as well as rotate the shaft (Figures 1.3). A magnetic bearing is an electro-magnetic device used only to levitate the shaft against the gravitational pull. The ability of the self-bearing motor to provide bearing force and motoring torque renders it superior to the magnetic bearings. The figure 1.3 is a schematic of the segmented arc Lorentz type slotless self-bearing motor used in the Bearings and Seals lab at the University of Kentucky.

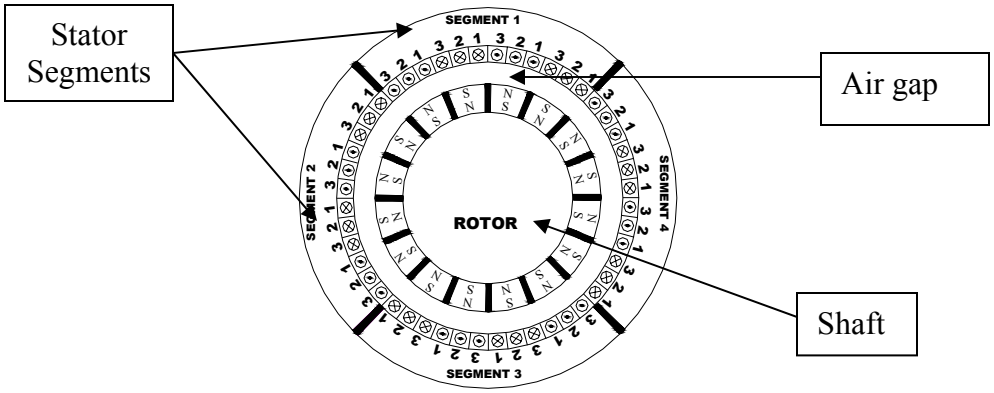


Figure 1.3: Lorentz type segmented arc self-bearing motor

1.4 Advantage of the self-bearing motor

- Less power consumption
- Eliminates the trade-off between motoring torque and radial bearing forces.
- Overall weight of the actuator and the system is reduced since there is no separate driving motor as in magnetic bearings.
- No cogging or detent torque.

1.5 Problem definition and research motivation

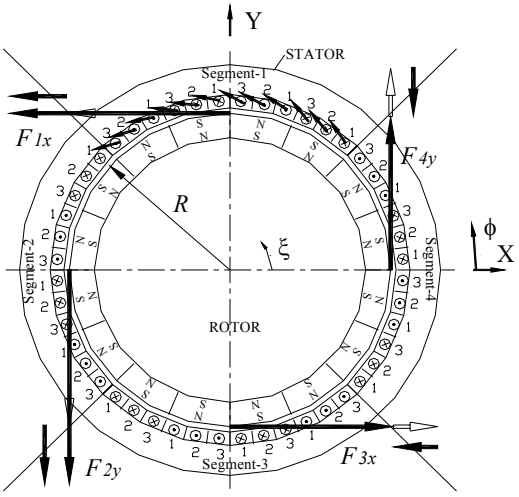


Figure 1.3: Lorentz type segmented arc self-bearing motor without faults

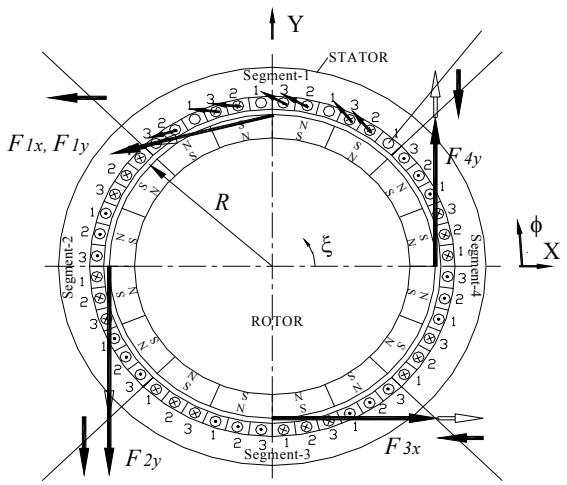


Figure 1.4: Lorentz type segmented arc self-bearing motor with phase faults

When a phase fails, the actuator does not produce the same bearing force and motoring torque as it would without the fault. With more phase faults, the actuator loses stability. The main goal of this research is to provide a stable bearing force and motoring torque even with phase faults. This thesis in particular focuses on implementing the fault tolerance of a self-bearing motor to open phase coil faults using adaptive control software without the addition of any hardware. Hence successful implementation would eliminate the hardware costs associated with the coil faults occurring in the motor. In addition, decoupling of the force-current relationship is also achieved in the motor. The problem of fault tolerance needs to be addressed and requires a thorough analysis.

There are many types of phase faults that can occur in the motor. The phase faults can be classified into the following categories

1. Open coil faults
2. Shorted faults
   a) When the coils short with themselves

b) When the coil shorts with the back iron

c) When the coil shorts with other coils

This research deals with providing fault protection for the self-bearing motor considering open coil faults. The short circuit faults are difficult to identify and fault tolerant control of such faults are a challenge to researchers to date.

1.6 Literature survey

Chiba [91] proposed a reluctance type bearingless motor and the contributed the concept of inductances, which are a function of the eccentric displacement of the rotor, to radial force. Bleuler [92] proposed a systematic way of classifying magnetic levitation methods. Lorentz force bearings and self-sensing bearings were considered to have high potential for industrial applications. Okada [Okada 96] introduced an internal permanent magnet type bearingless motor, which has the merits of strong levitation force and relatively easy control capability. Hertel [00] came up with a basic approach to designing a bearingless motor. By maintaining a constant air gap area and reducing the utilization factor by iterative processes until the rated output power and maximum force are reached, the main dimensions of bearingless AC machines was determined. Okada [00] introduced a new type of Lorentz type self-bearing motor similar to the one in Bearing and Seals lab at the University of Kentucky. The difference between these two is that different current carrying coils are used to generate radial force and motoring torque. Salazar [00] reviewed the published papers in bearingless motor types, winding types, mechanical test results and applications.

Salazar [00], Bischel [00], Chiba [91], Schoeb [94] and Okada [96] studied the permanent self bearing motor, among others for a variety of applications. In all of the designs the common conclusion was that attractive forces between the rotor and the stator (Maxwell-type forces) provide bearing function, and magnetic forces on the current carrying conductors (Lorenz type forces) produce the motoring torque. As a result of this approach, a trade-off exists between motoring torque and bearing force with respect to PM thickness. A slotless self-bearing motor was designed and proposed by Stephens and Kim [Stephens 00] for precision pointing and angular slewing applications, which could overcome the trade-off. Force and torque measurement principles in the self-bearing motor were discussed by Steele [00]. Permeance and flux models were presented by Stephens [02:1] and those were used to derive expressions for torque and force production. A linearized force-current-displacement relationship based on the permeance and flux model was derived for a general operating point. A revised set of actuator gains were derived by Chin[03:2] considering the various phase angles between winding current & permanent flux density distributions and effect of constant external loads. The design issues comprising the search for stable radial bearing function PID controllers and the various stable regions of these controller gains for different bandwidth and crosscouplings effects were analyzed by Chin[Chin 03:1]. Stephens [02:2]evaluated the robustness of the Lorenz self-bearing motor system via μ-synthesis and the utility of structured uncertainty approach for synthesizing robustly stable controllers.

Chin [Chin 03:1] explained the significant effects cross-coupling between radial and tangential direction. The cross-coupling between x and y direction was predicted

theoretically and an experimental basis was provided for future high-speed applications. The cross-coupling effect is undesirable because it limits the bearing stiffness and causes reduction in bandwidth. When the shaft rotates at the critical speed, cross-coupling gets higher creating instability in the system. These effects emphasize the importance of decoupling the model to provide a stable bearing force and motoring torque. Decoupling in simple terms is to control a variable without affecting the other variables. Aeschimann, Kummerle, Zoethout, Bleuler [Beat 00] presented a simple method to decouple an active magnetic bearing in statespace. In their work, the displacement and the velocity were chosen as the states of the model, with the former measured directly and the later obtained using a simple differentiator. A failure safe control approach to magnetic actuators can promote a broad range of applications. Most fault tolerance systems are focused on sensors and amplifiers. Kim and Stephens proposed new coil winding schemes that minimized coil failure effects and allowed the motor to levitate and rotate stably in the event of coil failure [Kim 00]. Analysis indicates that symmetric parallel winding is the most advantageous with respect to open faults in a phase. This however is at the expense of manufacturing ease and an excessive build up of end turns which increases actuator length.

Maslen [95] provided a mechanism for linearizing and decoupling the force axes in complicated magnetic actuators. A clear method has been established for achieving fault tolerance to coil failures. If one or more coils fail, a new coil current control scheme was constructed that preserves the linear relationship between required force and coil currents. Meeker [96] addressed the fault tolerant control in Maxwell type actuators and provided a general mathematical basis for such problems. Several

schemes have been proposed for achieving reliable electromagnetic devices including controller board approaches that make use of re-bias linearization [Dominick 99]. A fault tolerant magnetic bearing considering material path reluctance was proposed by Na and Palazzolo which uses a Lagrange multiplier optimization method for determining the current distribution matrices [Na 99]and makes the magnetic bearing fault tolerant to many pole and coil failures. Stephens [04] came up with a model based fault tolerant algorithm that had the potential to provide fault tolerance to open coil faults and also decouple the segmented arc, Lorenz self bearing motor.

Thesis outline

Chapter 1 introduces the basics of this research. It provides the research motivation and objectives of this thesis. A Contemporary literature survey on magnetic bearings, self-bearing motor and fault tolerance of magnetic actuators are outlined.

Chapter 2 serves as a background on the previous work by Dr. L Scott Stephens [Stephens 02:01]. This chapter describes the actuator layout and the derivation of Lorentz type forces for the self-bearing motor. In addition, this chapter also describes the fault tolerant algorithm [Stephens 04:01].

Chapter 3 compares and simulates the various fault tolerant control models. It discusses the implementation of the fault tolerant best model in the self-bearing motor.

Chapter 4 gives the experimental procedures to validate the decoupled and fault tolerant control model of a self-bearing motor to open coil faults. The results of the experiments prove that the decoupled and fault tolerant was found to be better than the non-fault tolerant model.

Chapter 5 discusses the results and conclusions from the experimental results.

Appendix A gives the C-mex file associated with the pseudo-inverse S-function.

Appendix B gives the Medit-file Function for Calculation of Forces using integral equations

Appendix C gives the Medit-file Function for the phase Distribution matrix, segment current-control current mapping matrix and fault matrix

Appendix D gives the C-mex file S-function for permanent magnet flux distribution. This S-function computes the forces from the station currents.

Appendix E shows how forces are computed for the given set of control currents in the non-fault tolerant control model.

Chapter 2

Lorentz Self-Bearing Motor

This actuator has a unique capability to produce radial bearing force as well as motoring torque independently by using Lorentz-type forces. The Lorenz self-bearing motor test-rig mainly involves electrical and mechanical design considerations of the test-rig. This chapter mainly serves as a background from the previous work of Stephens [Stephens 00:1:04]. This chapter is has four sections. Section 2.1 explains the mathematical model of the self-bearing motor test-rig. The subsections in section 2.1 explain the actuator layout, permeance flux model and Lorentz force derivations for the motor. The details about more design considerations and modeling of the motor can be obtained from the references [Steele 00, Stephens 00]. The section 2.4 describes the fault tolerant algorithm. The section 2.2 gives a systematic procedure for verification of mathematical dynamical system model. The section 2.3 gives discusses the design and implementation of the controller via matlab/simulink/dspace.

2.1 Mathematical Modeling of the Self-Bearing Motor

The physical system is best described in terms of mathematical model. An accurate mathematical model of the system is necessary in order to design a controller. However a perfect model of the system cannot be built due to unmodeled system

dynamics like unbalanced forces, external disturbances, which causes uncertainties in the system. The following section describes the system model.

2.1.1 Actuator layout and control

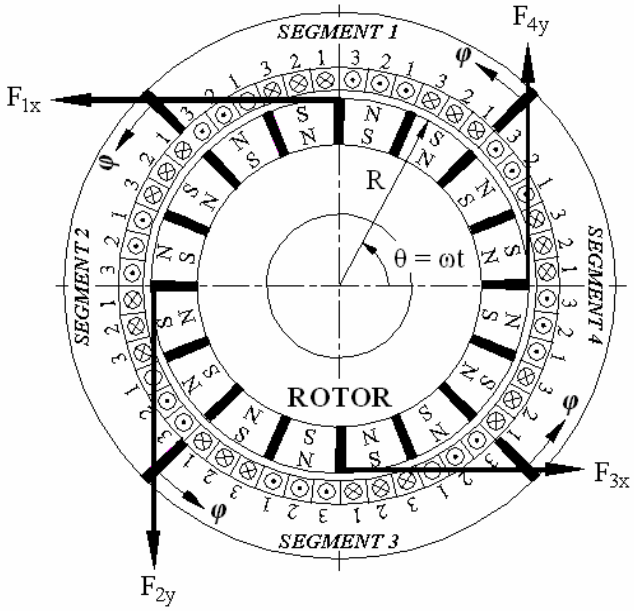


Figure 2.1: Actuator layout and force generation in the segmented arc self-bearing motor, courtesy: Stephens (00:1)

Figure 2.1 shows the layout of the actuator consisting of M=12 permanent magnet pole pairs attached to the rotor and N_{seg}=4 individually controlled winding segments attached to the stator. Each winding segment in the motor is an arc of $\pi/2$ radians and is attached to the slotless back iron. The windings occupy N_s=18 stations along each winding segment ID with N_w=96 individual wires per station. The 18 stations are divided into 6 sets of 3 phase windings.

The global angular coordinate that gives the orientation along any stator segment relative to the x coordinate axis is given by:

$$\phi = \theta + \psi_k - \frac{\pi}{4} \qquad (2.1)$$

where $\pi/4$ is half the segment angle and θ is the local segment angle. Referring to Figure 2.3, the orientation of the k^{th} winding segment relative to the x-axis is given by $\psi_k = k\,(\pi/2)$.

The control force and torque generation principle is also illustrated in Figure 2.1. Each segment generates traction on the surface of the rotor due to the PM flux linking with the segment windings (a Lorentz-type force). By precise construction of the motor, the tractions due to segments 1-4 are resolved into the forces F_{1x}, F_{2y}, F_{3x} and F_{4y}. By proper selection of the control currents in each segment, the segment forces are modulated to produce independent bearing forces and motoring torque.

2.1.2 Air gap flux and winding current distributions

Figure 2.3 shows the block diagram for servo control of the self bearing motor. Position sensors measure the x, y and ξ motions of the rotor and feed these back through signal conditioning modules and into a digital controller. Each motor segment is controlled by a separate segment current, i_k , that is proportional to the amplifier voltage, V_k, and that is sinusoidally commutated into the three phase windings using digital commutation and transconductance power amplifiers. The three phase currents in the k^{th} segment of the motor are 60^o ($\pi/3$ rad) apart in phase angle and given by:

$$i_{k,3} = i_k \cos\left[M(\xi - \gamma) + \pi/3 \right] \qquad (2.2)$$

$$i_{k,2} = i_k \cos\left[M(\xi - \gamma) \right] \qquad (2.3)$$

$$i_{k,3} = i_k \cos\left[M(\xi - \gamma) - \pi/3 \right] \tag{2.4}$$

where γ is the phase angle of the current with respect to the permanent magnet flux.

Control of the actuator such that independent torque and force generation is achieved as depicted in Figure 2.3, depends upon proper selection of the segment winding currents This selection is done with respect to three rotor position control currents, i_x, i_y and i_ξ that correspond to the x and y direction bearing forces and the ξ direction motoring torque:

$$\begin{aligned} i_1 &= (i_\xi - i_x) \\ i_2 &= (i_\xi - i_y) \\ i_3 &= (i_\xi + i_x) \\ i_4 &= (i_\xi + i_y) \end{aligned} \tag{2.5}$$

The air gap flux due to the permanent magnets, $B_{m,k,}$ the air gap flux due to the windings, $B_{w,k,}$ and the winding current distribution, I_k. are approximated sinusoidally as:

$$B_{m,k}(\theta,\xi) = \overline{B_{m,k}} \sin\left[M(\xi - \theta) \right] \tag{2.6}$$

$$I_k(\theta,\xi) = \overline{i_k} \sin\left[M(\xi - \theta - \gamma) \right] \tag{2.7}$$

$$B_{w,k}(\theta,\xi) = \overline{B_{w,k}} \sin\left[M(\xi - \theta - \gamma + \pi/2M \right] \tag{2.8}$$

where $\overline{B_{m,k}}$, $\overline{i_k}$, and $\overline{B_{w,k}}$ are the amplitudes of the sine wave approximations, and π/2M is the phase shift of the winding flux with respect to the winding current. Both $\overline{B_{m,k}}$ and $\overline{B_{w,k}}$ are functions of ϕ (and therefore θ, by equation 1) when the rotor is in

an eccentric position. The PM flux amplitude, $\overline{B_{m,k}}$, is computed using the remnance flux density.

2.1.3 Force and torque generation

Given the air gap flux and winding distributions in the previous section, the forces acting on the rotor are divided into three groups: (1) Lorentz-type due to PM flux and winding current interaction, (2) Maxwell-type due to the PM flux interaction with the rotor and (3) Maxwell-type due to the winding flux interaction with the rotor. Those used for bearing force and torque controls are the Lorentz-type and computed as:

$$F_{x,L} = ML \sum_{k=1}^{N_{seg}=4} \int_0^{\frac{\pi}{2}} B_{m,k}(\theta) I_k(\theta) \cos\left[\theta + \psi_k + \frac{\pi}{4}\right] d\theta \qquad (2.9)$$

$$F_{y,L} = ML \sum_{k=1}^{N_{seg}=4} \int_0^{\frac{\pi}{2}} B_{m,k}(\theta) I_k(\theta) \sin\left[\theta + \psi_k + \frac{\pi}{4}\right] d\theta \qquad (2.10)$$

$$T_{\xi} = MRL \sum_{k=1}^{N_{seg}=4} \int_0^{\frac{\pi}{2}} B_{m,k}(\theta) I_{\ k}(\theta) d\theta \qquad (2.11)$$

where R is the outside radius of the rotor. The Maxwell type forces on the rotor due to the PM flux are computed using:

$$F_{x,M} = \frac{RL}{2\mu_o} \sum_{k=1}^{N_{seg}=4} \int_0^{\frac{\pi}{2}} \left[B_{m,k}(\theta)\right]^2 \cos\left[\phi(\theta)\right] d\theta \qquad (2.12)$$

$$F_{y,M} = \frac{RL}{2\mu_o} \sum_{k=1}^{N_{seg}=4} \int_0^{\frac{\pi}{2}} \left[B_{m,k}(\theta)\right]^2 \sin\left[\phi(\theta)\right] d\theta \qquad (2.13)$$

The Maxwell type forces on the rotor due to the winding flux are computed using:

$$F_{x,w} = \frac{RL}{2\mu_o} \sum_{k=1}^{N_{seg}=4} \int_0^{\frac{\pi}{2}} \left[B_{w,k}(\theta) \right]^2 \cos\left[\phi(\theta)\right] d\theta \qquad (2.14)$$

$$F_{y,w} = \frac{RL}{2\mu_o} \sum_{k=1}^{N_{seg}=4} \int_0^{\frac{\pi}{2}} \left[B_{w,k}(\theta) \right]^2 \sin\left[\phi(\theta)\right] d\theta \qquad (2.15)$$

The net force and torque on the self bearing motor rotor is then the sum of these components:

$$\begin{aligned} F_x &= F_{x,L} + F_{x,m} + F_{x,w} \\ F_y &= F_{y,L} + F_{y,m} + F_{y,w} \\ T_\xi &= T_\xi \end{aligned} \qquad (2.16)$$

Performing integrals (2.9)-(2.15), results in net forces and torque that are a function of the control currents, i_x, i_y, and i_ξ, and rotor motion x, y, and ξ.

2.2 Verification of mathematical system model

Figure 2.2 describes the procedure for both the verification of mathematical system model and the implementation of the controller in dspace. The physical system is rendered as a modeled into mathematical model based on assumptions about the difficulty and cost criteria. The mathematical model is simulated in software like simulink and checked for favorable results. It is then compared with the actual dynamic response. If the response does not match the mathematical model is modified and the simulation is performed again. This process is continued until the theoretical response matches with the actual dynamic response. The controller is designed and implemented using simulink, dspace and real time workshop.

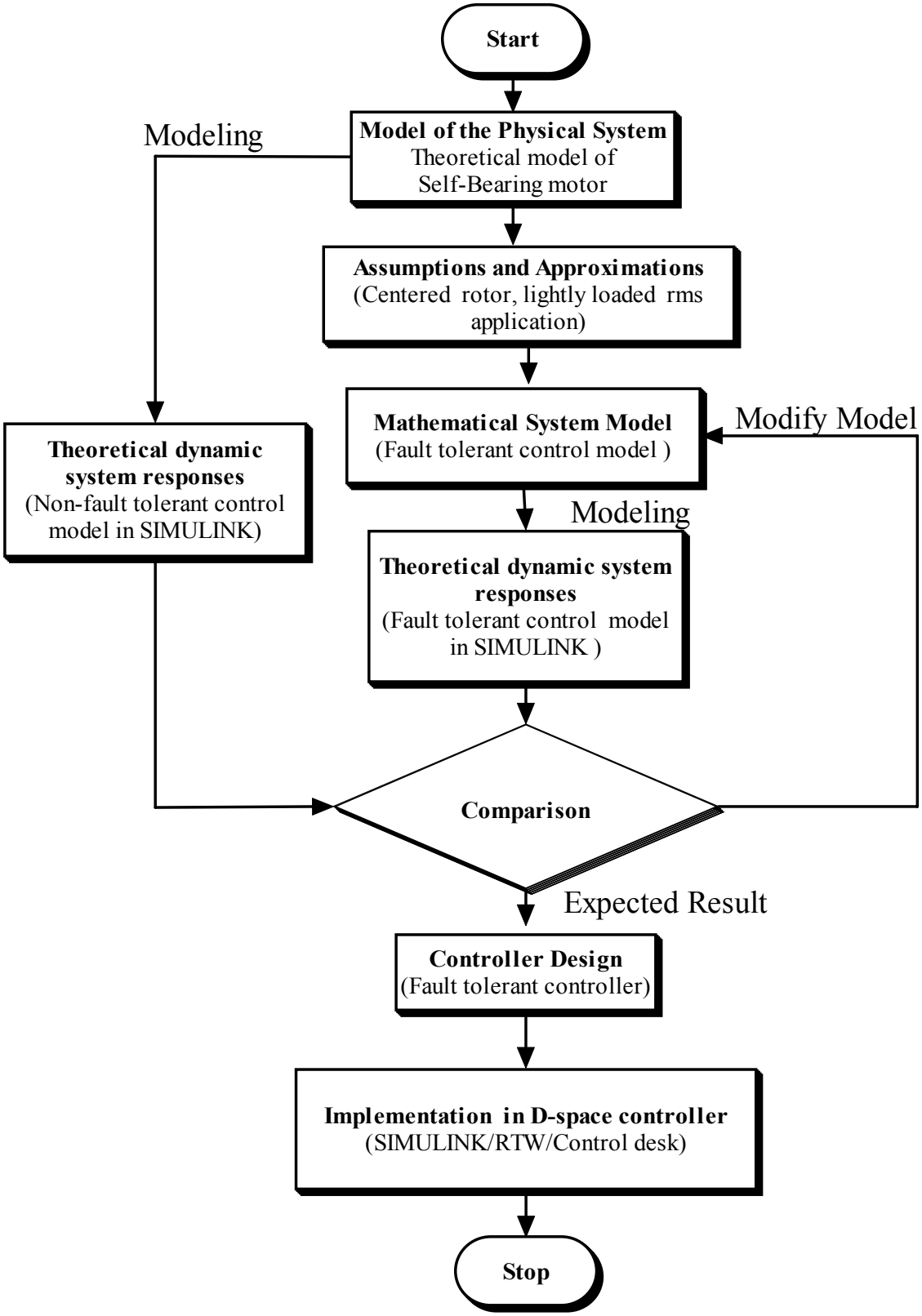


Figure 2.2: Flowchart for verification of mathematical system model and implementation of the controller in dspace

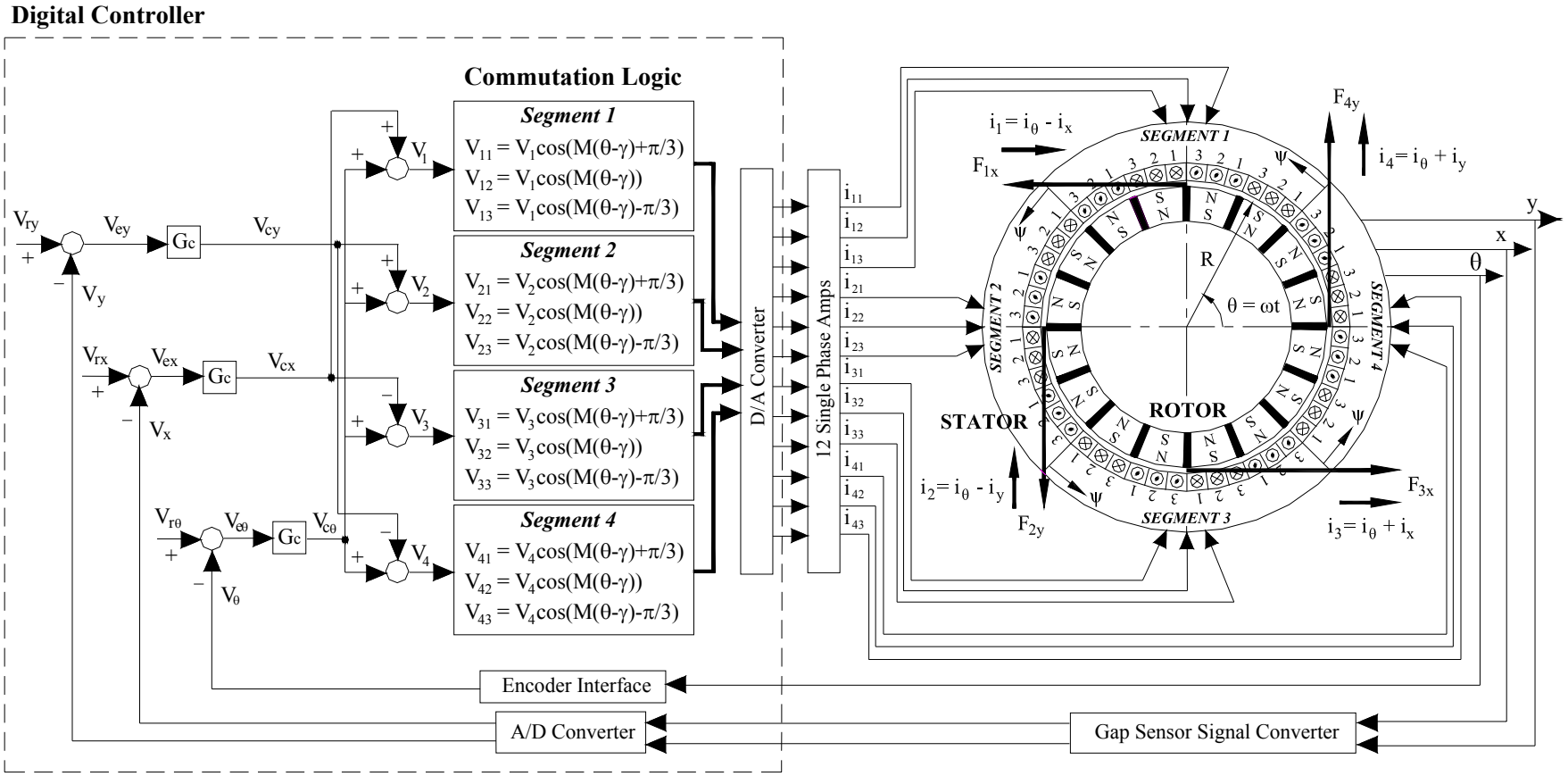


Figure 2.3: Non-Fault Tolerant Control Approach, Courtesy: Stephens 04

2.3 Controller design and implementation of controllers

The figure 2.3 shows the schematic of the self-bearing motor with non-fault tolerant approach. The radial displacements x and y, and angular displacements, θ of the rotor are measured using the proximity probes and encoder respectively. These signals are sent to the analog-to-digital converter (ADC) after signal conditioning. The sensor voltages V_x, V_y and V_θ are added to reference voltages V_{rx}, V_{ry} and $V_{r\theta}$ to obtain the error voltages V_{ex}, V_{ey} and $V_{e\theta}$. The error voltages serve as the input to controllers. The control voltages V_{cx}, V_{cy} and $V_{c\theta}$ are converted to segment voltages V_1, V_2 , V_3 and V_4 using appropriate mapping. The segment voltages are commuted digitally to obtain 3 phase voltages per segment, i.e, 12 in all. These voltages are sent to the digital-to-analog converter (DAC) and the analog signals are amplified to 12 phase currents using a tranconductance amplifier.

The control system is implemented in simulink so that it can be implemented in the dspace system. The simulink block diagram is downloaded in the dspace DSP boards of the dspace system. Using the dspace controldesk interface, the motor is controlled in real-time. Experiments are designed to validate the control algorithm and performance of the motor is evaluated.

2.4 Fault tolerant algorithm

The decoupled and fault tolerant algorithm serves as a background for implementation of fault tolerance of self-bearing motor [Stephens 04]. The model based algorithm presented was used to decouple the segmented arc, Lorenz self bearing motor. Simulations showed that the algorithm gracefully degrades the performance of the motor under open coil faults. The redundancy in the actuator

phases is taken advantage of in providing protection to phase faults. The cost of the fault protection was found to be a lower limit on peak actuator force and torque, and a high power loss.

2.5 Fault tolerant control approach

Fault tolerant control is achieved by constructing a detailed model of the force-current relationship at each rotor angle, θ, and simply inverting that model onto itself to decouple the system. Referring to Figure 3.2, the inverted model is inserted in the "fault tolerant mapping" block. The appropriate mapping depends upon the relationship between the actuator force vector, $\mathbf{F_c}$, and the actuator segment current vector, $\mathbf{i_s}$, which is given by:

$$\overbrace{\begin{bmatrix} F_{cx} \\ F_{cy} \\ T_{c\theta} \end{bmatrix}}^{3\,x\,1}_{\mathbf{F}_c} = \overbrace{\underbrace{\Phi(\theta)}_{3\,x\,48}\overbrace{\Lambda}^{48\,x\,12}}^{A}\underbrace{F}_{12\,x\,12}\underbrace{Y(\theta)}_{12\,x\,4}\overbrace{\begin{bmatrix} i_1 \\ i_2 \\ i_3 \\ i_4 \end{bmatrix}}^{4\,x\,1}_{\mathbf{i}_s} \tag{2.17}$$

where $\Phi(\theta)$ is a matrix that describes how the permanent magnets are distributed about the rotor, Λ is a matrix that describes how the phases are wound into the stator, F is a matrix that encodes the faults (an open circuit on any given phase) and $Y(\theta)$ is the commutation matrix.

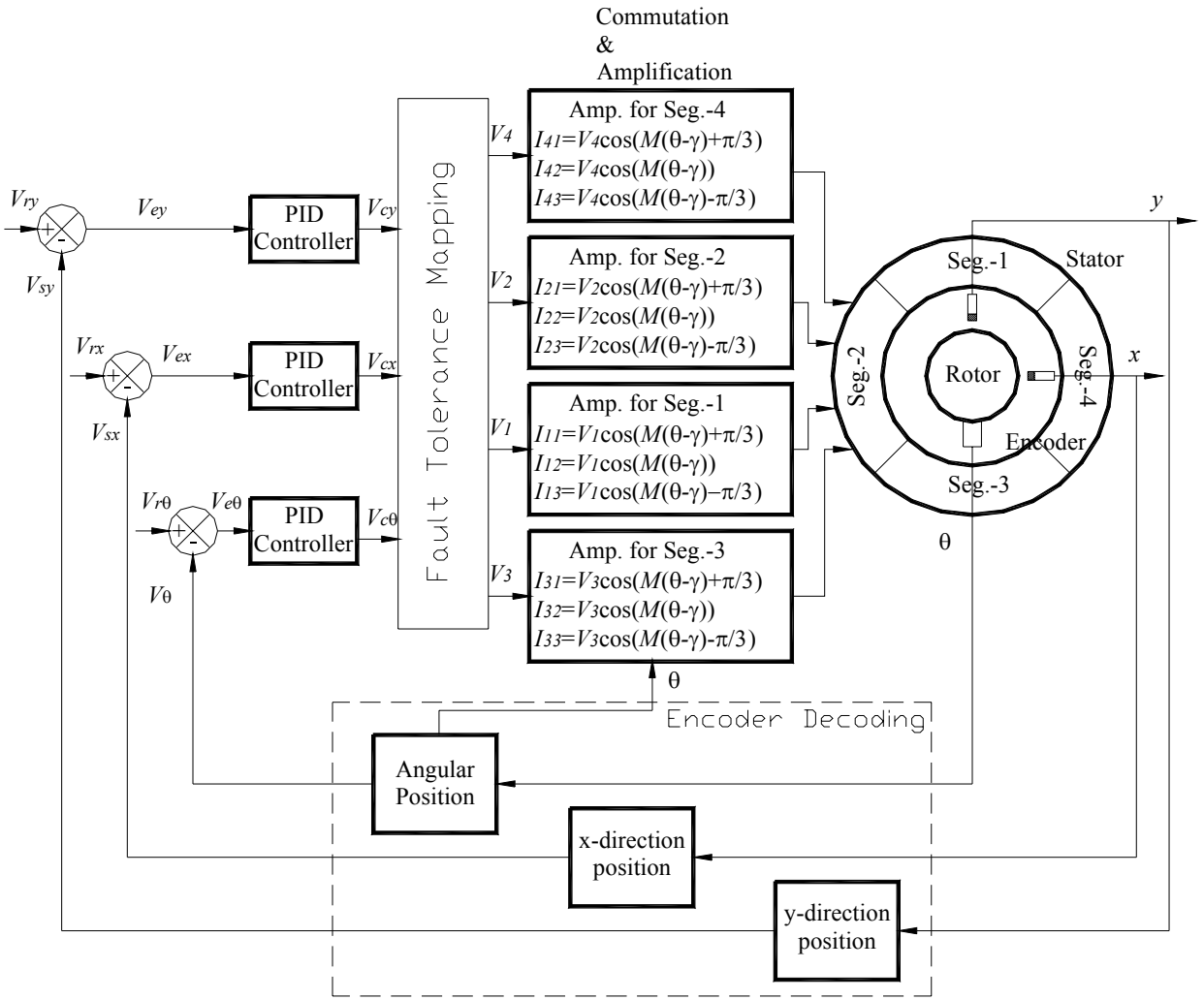


Figure 2.4: Fault tolerant control approach, courtesy: Stephens 04

The model is completely decoupled by defining the fault tolerant mapping between the segment currents and the control currents as:

$$\mathbf{i_s} = A^{+}\bar{K}_i\mathbf{i_c} \tag{2.18}$$

where $A^{+}=A^{T}(AA^{T})^{-1}$ and is the Moore-Penrose pseudo inverse of the underdetermined model, A, and $\bar{K}_i$ is any desired current gain matrix as defined by the designer. Of course the desired current gain matrix is of the completely decoupled variety and may be as simple as the identity matrix. Combining equations (2.17) and (2.18) illustrates how the method essentially cancels the original system, whether it has a fault or not, and replaces it with the desired current gain matrix:

$$\mathbf{F_c} = \underbrace{AA^{+}}_{I}\bar{K}_i\mathbf{i_c} \tag{2.19}$$

Note that this mapping solves the problem of cross-coupling and current gain variation that exists in this actuator even for the case of no fault, as well as provides a current gain matrix that remains invariant under open coil faults.

Chapter 3

Fault Tolerant Model: Development, Verification and Implementation

A fail-safe control approach to self-bearing motors can promote a broad range of applications. The approach to fault tolerance for the self bearing motors on the optical tracking test rig is a combination of the redundant hardware and adaptive control software. The hardware redundancy on each self-bearing motor consists of 12 phases that comprise four segments to generate only 2 radial bearing forces and 1 motoring torque. These phases (windings) can be driven in a variety of ways by the power amplifiers. There is adequate redundancy in the self-bearing motor to achieve fault tolerance in the radial and angular pointing direction. The hardware redundancy in a self-bearing motor leads to increase in weight and cost.

The software algorithm gives the flexibility of implementing different kinds of control algorithms without the addition of much hardware. The fault tolerance was achieved in the self-bearing motor by manipulating the with software part of the system. This model is split into four matrices, two of them in software and two of them in hardware, so that control current times the matrices provide the forces. The commutation matrix and segment-control current mapping matrix form a part of software. The permanent magnet distribution matrix and coiling winding distribution matrix forms a part of hardware. This study achieved fault tolerance using the software since altering with the hardware was difficult, weighty and costly.

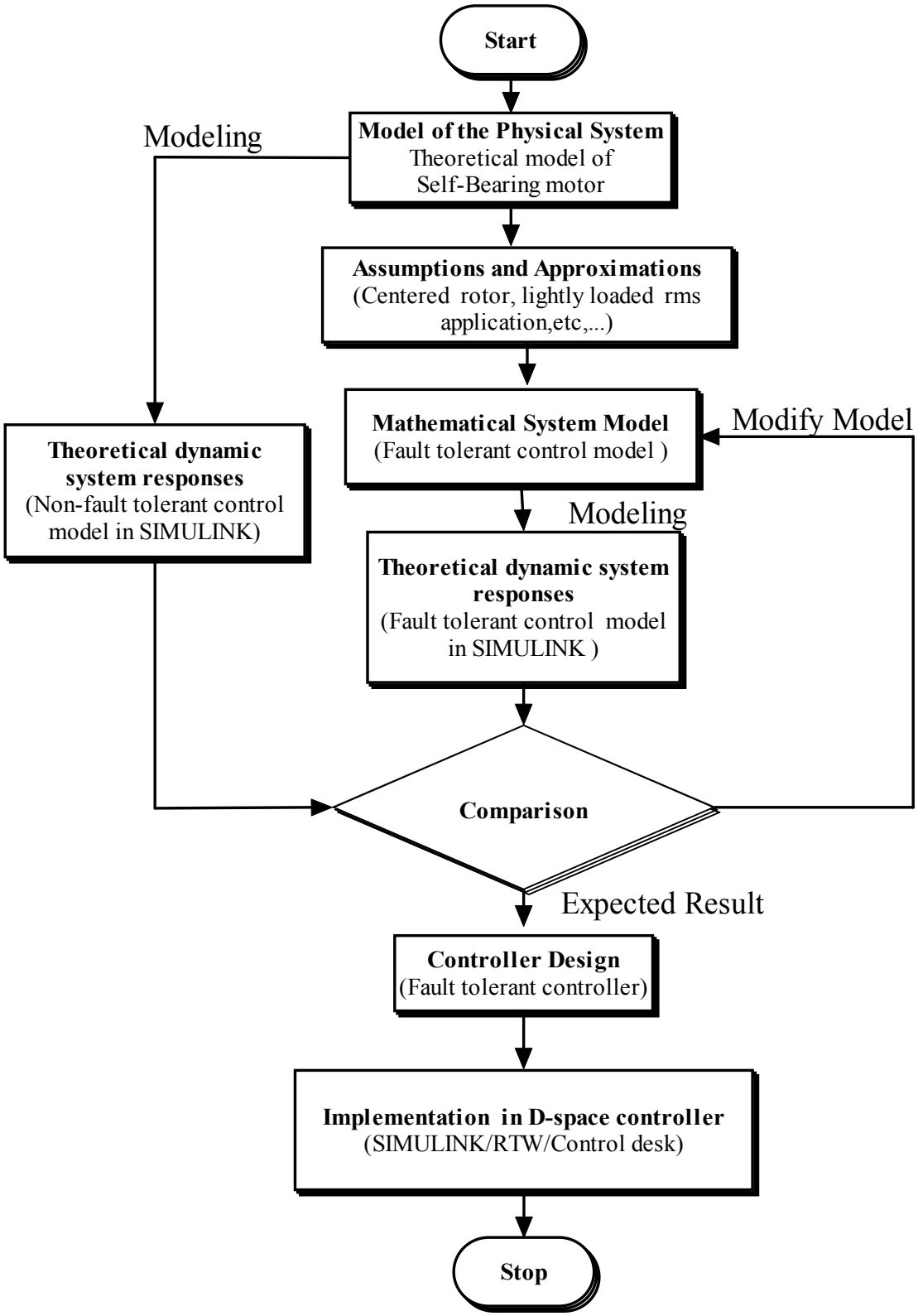


Figure 3.1: Flowchart for verification of Fault tolerant model and implementation of the fault tolerant controller in dSPACE

3.1 Verification of fault-tolerant model and model implementation

The mathematical model of the non-fault tolerant control approach was available. This model was advantageously used for constructing a fault tolerant control model. The dynamic response of the non-fault tolerant control model was produced by simulation in simulink. Assumptions were made based on difficulty, applications, unmodeled dynamics, feasibility and cost, before constructing the mathematical model for the fault tolerant control model. The dynamic response of the fault-tolerant model was produced in simulations in simulink. The responses of both the fault tolerant and non-fault tolerant model are compared. On obtaining the expected result, the model was designed and implemented in dspace for performing real-time control of the motor. Experiments are designed for gathering useful results from the system. Figure 3.1 gives the procedure for the verification of fault tolerant control model and the implementation of the fault tolerant controller in dspace.

3.2 Simulation and Testing Procedures

The previous chapter put forth a mathematical model for achieving fault tolerance in a self-bearing motor. The model was deciphered into simulink codes. The mathematical model essentially builds a force-current relationship using different methods. There are as many simulink models as the number of methods. Every simulink model presents results which are evaluated and verified with the expected trend from the mathematical model. Upon the successful performance of the model in simulation, it was implemented in real-time with some modification. The model was then downloaded in dspace controller and tested in the dspace environment. This testing was performed without levitating the test-rig directly using

the model instead and thereby ensuring that the testing would not harm the test-rig. A risk-free testing was executed by keeping the power amplifiers switched off so that the power that gets out into the motor was meager. Next, the values in the virtual indicators at different significant locations in the model are read. The read out was checked for expected results. This testing was a hybrid test of software and hardware called a hardware-in the loop simulation. The successful completion completes the final phase of testing. After obtaining convincing results in the simulation, the power amplifier was switched on and the motor was levitated using the controls in the dspace software.

3.3 Comparison of various fault tolerant control model

This chapter focuses on a comparison of the following fault tolerant models.

1. **Integral equations model (Non-fault tolerant control model):** This model gives the force-current relationship based on integral equations, when there are no faults in the system.
2. **Lumped parameter model (Non-fault tolerant control model):** This model gives the force-current relationship based on lumped parameter matrix equstions, when there are no faults in the system.
3. **Fault tolerant control model based on integral equations:** This model gives the force-current relationship based on integral equations, when there are faults in the system.
4. **Fault tolerant control model based on lumped parameter model:** This model gives the force-current relationship based on lumped parameter matrix equations, when there faults in the system.

5. Decoupled and fault tolerant control model: This model gives the force-current relationship based on lumped parameter matrix equations, when there are faults in the system. The key difference is that a decoupled K_i is implemented here. In all of the previous models, the K_i matrix implemented is the coupled one, which corresponds to the non-faulted motor

All these models are developed and simulated and the best model was chosen for implementation in the dspace controller. The commonality in these control models is that, all of them bear a force-current relationship.

$$\overbrace{\underbrace{\begin{bmatrix} F_{cx} \\ F_{cy} \\ T_{c\theta} \end{bmatrix}}_{\mathbf{F}_c}}^{3\,x\,1} = function(\, i_{cx},\, i_{cy}, i_{c\theta}, \theta, x, y)$$

3.3.1 Integral equations model

This model gives a force-current relationship based on integral equations. This model was the most accurate of the models available and includes most of the effects. The control currents $i_{cx}, i_{cy}, i_{c\theta}$ were sent in as inputs to the integral equations model and forces F_x F_y and torque T are obtained as the output (Figure 3.2). The force produced on the shaft by a set of control currents using integral equations, was determined in the simulink model (Figures 3.3)

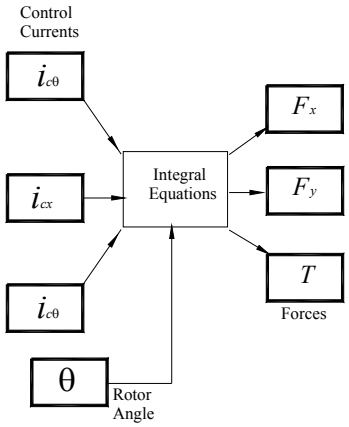


Figure 3.2: Integral equations model

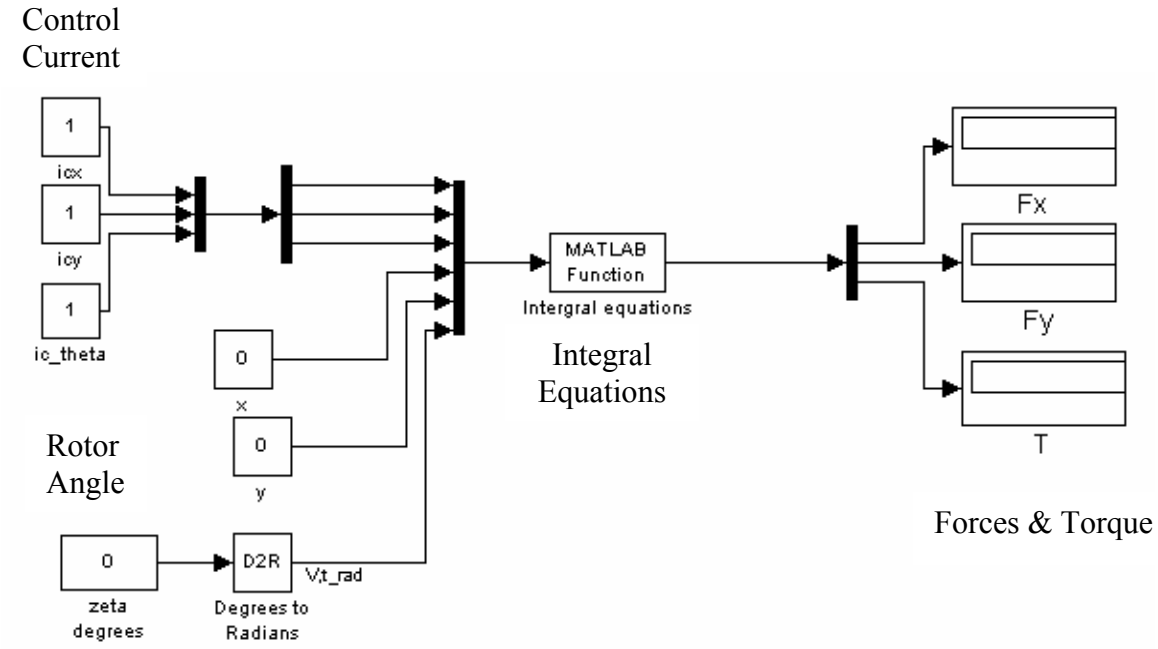


Figure 3.3: Simulink model for using integral equations model

3.3.2 Lumped parameter model

Achieving fault tolerance implies achieving the desired forces in the shaft for a given set of control currents, even with some phase faults. The lumped parameter model serves as a basis of accomplishing fault tolerance later. This model was less accurate than the integral equations models and breaks up into the following components

$\Phi(\theta)$-Flux linkage matrix

Λ -Phase Distribution matrix

Y -Commutation matrix

$T3$ -Segment Current-Control current mapping

The entire system is a mixture of hardware and software components. The hardware components are replaced by blocks in simulink and integrated with the existing software blocks in simulink for the purpose of simulation. The force computed in terms of the components of the system is given by the following eqn.

$$\underbrace{\begin{bmatrix} F_{cx} \\ F_{cy} \\ T_{c\theta} \end{bmatrix}}_{\mathbf{F}_c} = \overbrace{\Phi(\theta)}^{3x48} \overbrace{\Lambda}^{48x12} \overbrace{Y(\theta)}^{12x4} \overbrace{T3}^{4x3} \underbrace{\begin{bmatrix} i_{cx} \\ i_{cy} \\ i_{c\theta} \end{bmatrix}}_{\mathbf{i}_c} \qquad (3.1)$$

The control currents $i_{cx}, i_{cy}, i_{c\theta}$ were sent in as inputs to the lumped parameter model and forces F_x F_y and torque T are obtained as the output. The result of the lumped parameter model, simulated in simulink as shown in figure 3.4, was checked for agreement with the integral equation result. The advantage of the lumped parameter model over the integral equations model is that it can be inverted so that it is used in the fault tolerant models.

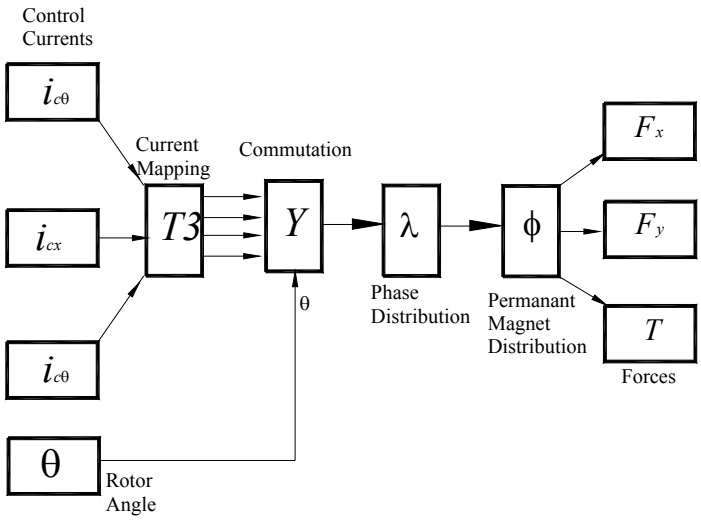


Figure 3.4: Lumped parameter model

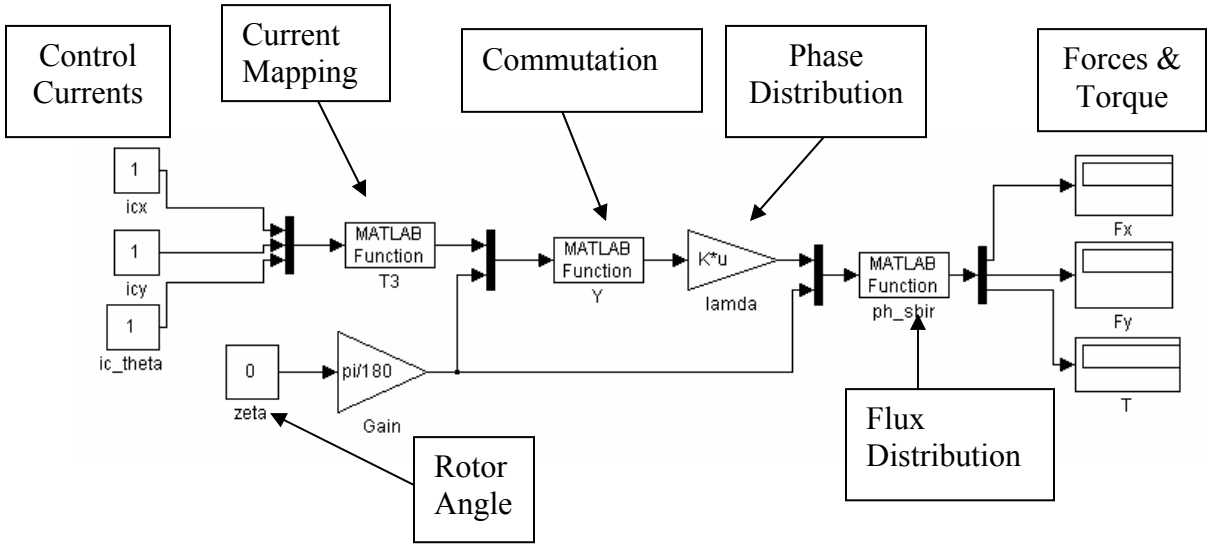


Figure 3.5: Simulink model for Lumped parameter model

3.3.3 Crosscoupled K_i model:

The theoretical force-current relationship based on integral equations and the lumped parameter model are crosscoupled and are represented by a single 3 x 3 matrix as follows:

$$\underbrace{\begin{bmatrix} F_x \\ F_y \\ T_\theta \end{bmatrix}}_{\mathbf{F_c}} \approx \underbrace{\begin{bmatrix} K_{ixx}(\theta) & \left[K_{xy,w}\bar{i}_\theta \pm K_{ixy}(\theta)\right] & K_{xy,w}\bar{i}_y \\ -\left[K_{xy,w}\bar{i}_\theta \pm K_{ixy}(\theta)\right] & K_{ixx}(\theta) & K_{xy,w}\bar{i}_x \\ 0 & 0 & K_{i\theta} \end{bmatrix}}_{[K_i]} \underbrace{\begin{bmatrix} i_x \\ i_y \\ i_\theta \end{bmatrix}}_{\mathbf{i_c}} \tag{3.2}$$

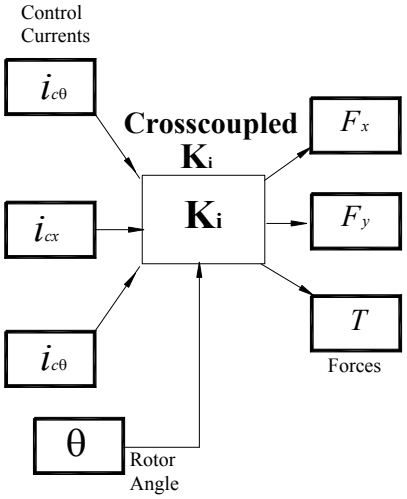


Figure 3.6: Crosscoupled K_i model

3.3.4 Desired K_i model

The crosscoupled matrix can be simplified further into a decoupled matrix by making all the elements except the leading diagonal to be zero. This model is the desired model and uses decoupled K_i . The decoupled K_i has no crosscoupled

terms and no θ dependence. It gives the desired force-current relationship and this helps to decouple the forces in the motor

$$\left[K_i\right] = \begin{bmatrix} K_{ixx}(\theta) & 0 & 0 \\ 0 & K_{ixx}(\theta) & 0 \\ 0 & 0 & K_{i\theta} \end{bmatrix} \tag{3.8}$$

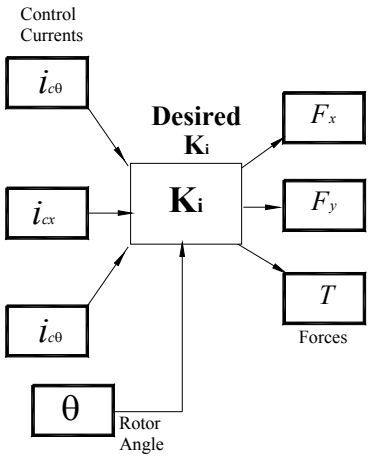


Figure 3.7: Desired K_i model

3.3.5 Lumped parameter model with the addition of fault matrix

If there are faults in the matrix, they can be represented by a fault matrix F and the equation for which is represented by the following formula and modeled as shown in simulink. With the faults in the fault matrix, the force and torque produced was different from those found in a faultless system. With more faults the forces and torque produced decrease drastically and causes instability to the system.

$$\underbrace{\begin{bmatrix} F_{cx} \\ F_{cy} \\ T_{c\theta} \end{bmatrix}}_{\mathbf{F}_c}^{3x1} = \overbrace{\Phi(\theta)}^{3x48}\ \overbrace{\Lambda}^{48x12}\ \overbrace{F}^{12x12}\ \overbrace{Y(\theta)}^{12x4}\ \overbrace{\mathrm{T}3}^{4x3}\ \underbrace{\begin{bmatrix} i_{cx} \\ i_{cy} \\ i_{c\theta} \end{bmatrix}}_{\mathbf{i}_c}^{3x1} \tag{3.5}$$

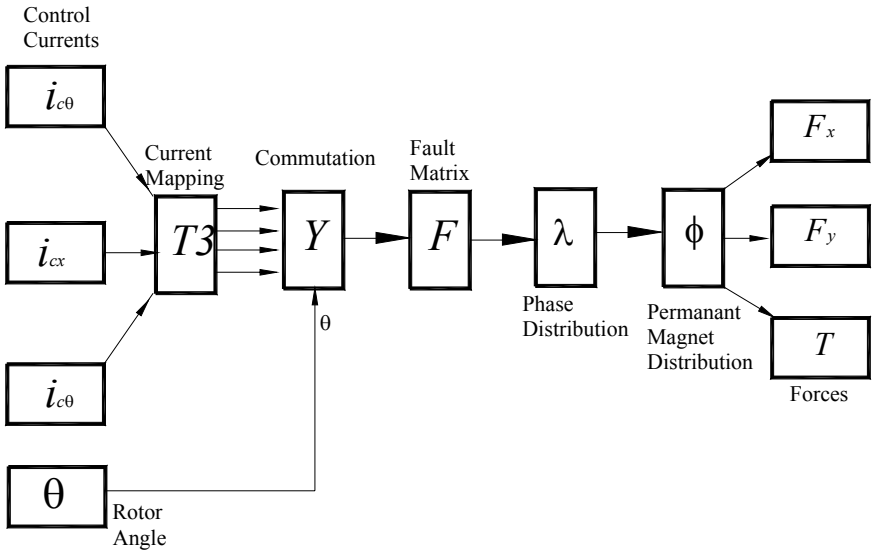


Figure 3.8: Lumped parameter model with the addition of fault matrix

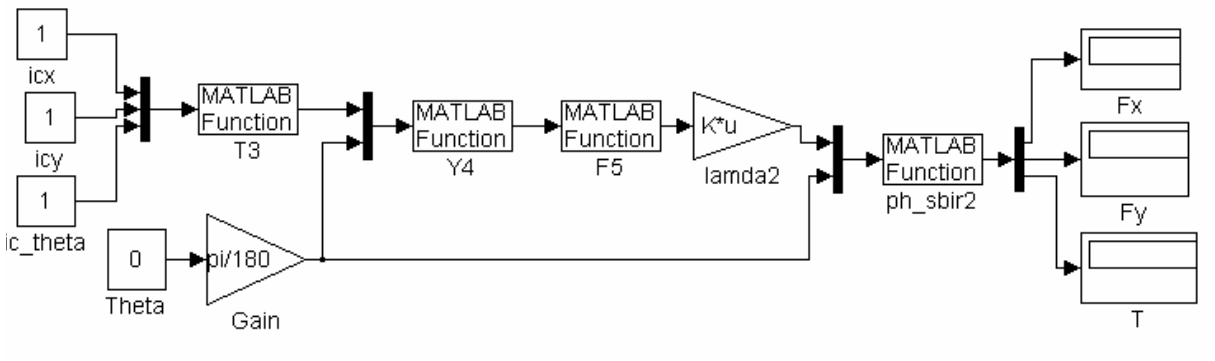


Figure 3.9: Simulink model for lumped parameter model with the addition of fault matrix

3.3.6 Fault tolerant model based on lumped parameter model

A fault tolerant model based on the lumped parameter model was developed from the lumped parameter model. The fault tolerant based on the lumped parameter model is based on the following equation:

$$\mathbf{F_c} = \underbrace{AA^{+}}_{I} K_i \mathbf{i_c} \tag{3.6}$$

$$\underbrace{\begin{bmatrix} F_{cx} \\ F_{cy} \\ T_{c\theta} \end{bmatrix}}_{\mathbf{F}_c} = \overbrace{\begin{bmatrix} \underbrace{\Phi(\theta)}_{3x48} & \underbrace{\Lambda}_{48x12} & \underbrace{F}_{12x12} & \underbrace{Y(\theta)}_{12x4} \end{bmatrix}}^{A} \overbrace{\begin{bmatrix} \underbrace{\Phi(\theta)}_{3x48} & \underbrace{\Lambda}_{48x12} & \underbrace{F}_{12x12} & \underbrace{Y(\theta)}_{12x4} \end{bmatrix}^{-1}}^{A^{-1}} \overbrace{\underbrace{\Phi(\theta)}_{3x48} \; \underbrace{\Lambda}_{48x12} \; \underbrace{Y(\theta)}_{12x4} \underbrace{\mathrm{T}}_{4x3} 3}^{K_i} \underbrace{\begin{bmatrix} i_{cx} \\ i_{cy} \\ i_{c\theta} \end{bmatrix}}_{\mathbf{i}_c}^{3x1} \tag{3.7}$$

Assuming that there is no fault, A and A^{-1} cancel out each other and leave out the original model. Next, if there are faults, A and A^{-1} cancel out each other and would still leave out the original model if $\left|AA^{-1}\right| \neq 0$. The existence or non-existence of the fault tolerant model was elaborately discussed by Stephens [04]. The above equation gives a crosscoupled fault tolerant model and was modeled in simulink as shown figure 3.11. In case of faults, a fault tolerant model based on the lumped parameter model would return the same amount of forces that a lumped parameter model would, without the faults. This model was tolerant to many different fault configurations, but does not decouple the forces.

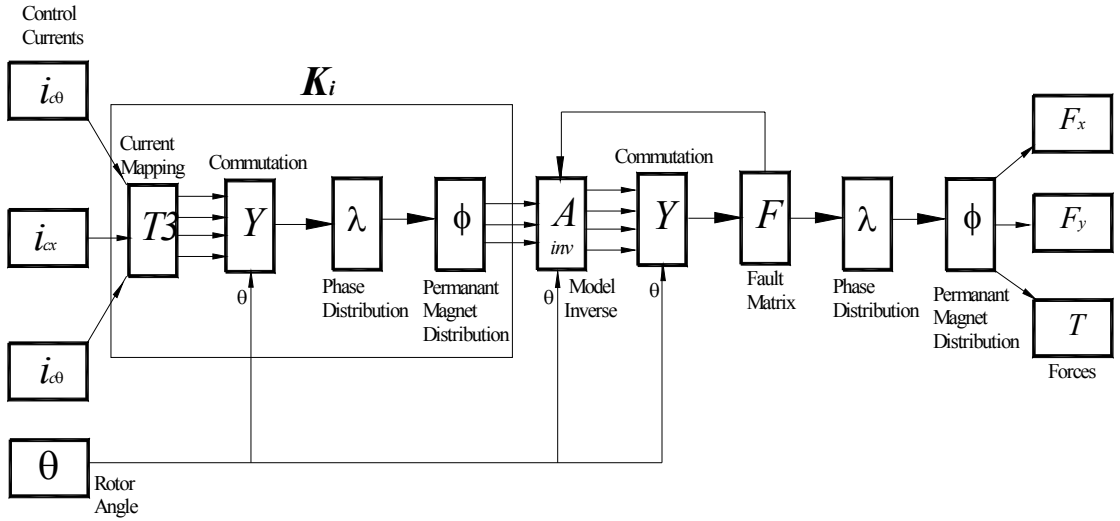


Figure 3.10: Fault tolerant model using lumped parameter model

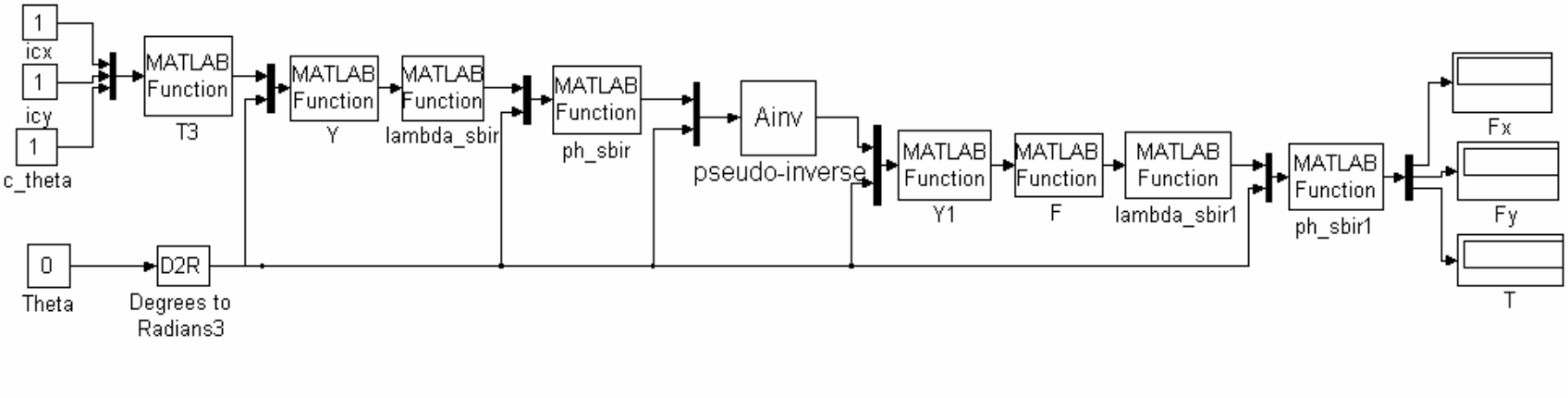


Figure 3.11: Simulink model for fault tolerant model using lumped parameter model

3.3.7 Fault tolerant model based on integral equations

The fault tolerant model based on integral equations was developed from the integral equations model. The fault tolerant control based on the lumped parameter model is governed by the following equation.

$$\underbrace{\begin{bmatrix} F_{cx} \\ F_{cy} \\ T_{c\theta} \end{bmatrix}}_{\mathbf{F}_c} = \underbrace{\begin{bmatrix} \overbrace{\Phi(\theta)}^{3x48} & \overbrace{\Lambda}^{48x12} & \overbrace{F}^{12x12} & \overbrace{Y(\theta)}^{12x4} \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} \overbrace{\Phi(\theta)}^{3x48} & \overbrace{\Lambda}^{48x12} & \overbrace{F}^{12x12} & \overbrace{Y(\theta)}^{12x4} \end{bmatrix}^{-1}}_{A^{-1}} \underbrace{K_i}_{INT} \underbrace{\begin{bmatrix} i_{cx} \\ i_{cy} \\ i_{c\theta} \end{bmatrix}}_{\mathbf{i}_c}$$

This model uses the inverse of the lumped parameter model and the integral equations model to provide the fault tolerance. The inverse of a lumped parameter model must be used because integral equations cannot be inversed. Since the lumped parameter model and integral equations model do not give the same results, hence inverse of the lumped parameter model would not cancel perfectly with the integral equations model. This model was tolerant to multiple faults, but does not decouple the forces.

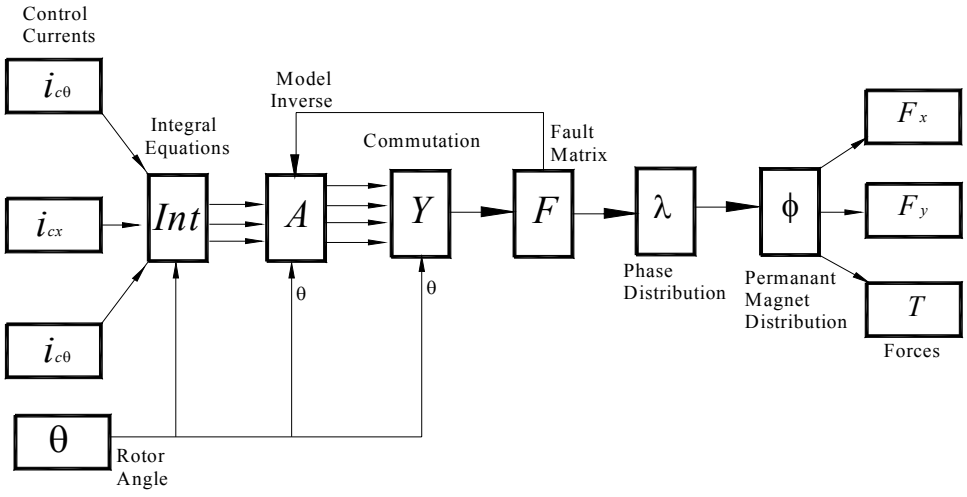


Figure 3.12: Fault tolerant model using integral equations

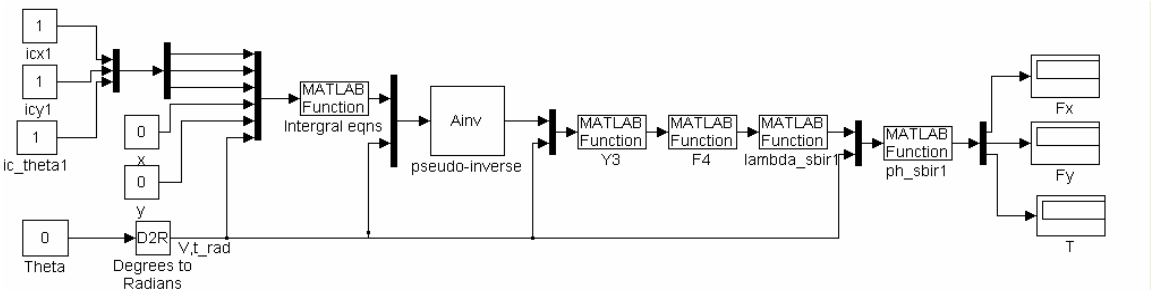


Figure 3.13: Simulink model for fault tolerant model using integral equations

3.3.8 Decoupled & Fault tolerant model

The decoupled and fault tolerant model is a model which uses the inverse of the lumped parameter model and a decoupled K_i to remove the crosscoupling. The force current relationship based on the decoupled and fault tolerant algorithm is given by the following equation:

$$\mathbf{F_c} = \underbrace{AA^{+}}_{I}\bar{K}_i\mathbf{i_c}$$

$$\mathbf{F_c} = \underbrace{\overbrace{\begin{bmatrix} \overbrace{\Phi(\theta)}^{3x48} & \overbrace{\Lambda}^{48x12} & \overbrace{F}^{12x12} & \overbrace{Y(\theta)}^{12x4} \end{bmatrix}}^{A} \overbrace{\begin{bmatrix} \overbrace{\Phi(\theta)}^{3x48} & \overbrace{\Lambda}^{48x12} & \overbrace{F}^{12x12} & \overbrace{Y(\theta)}^{12x4} \end{bmatrix}^{-1}}^{A^{-1}}}_{I} \bar{K}_i\mathbf{i_c}$$

This model uses the lumped parameter and the inverse of the lumped parameter to provide fault tolerance. Incase of faults, the model would cancel with the model inverse and return the forces depending on the $\bar{K}_i$. The $\bar{K}_i$ is a matrix that has entries only in the leading diagonal. This would return the decoupled forces for any set of control currents. This model thus removes the crosscoupling as well as θ dependence. The simulations of all the three fault tolerant models performed later, revealed that this model was the better than the other fault tolerant models.

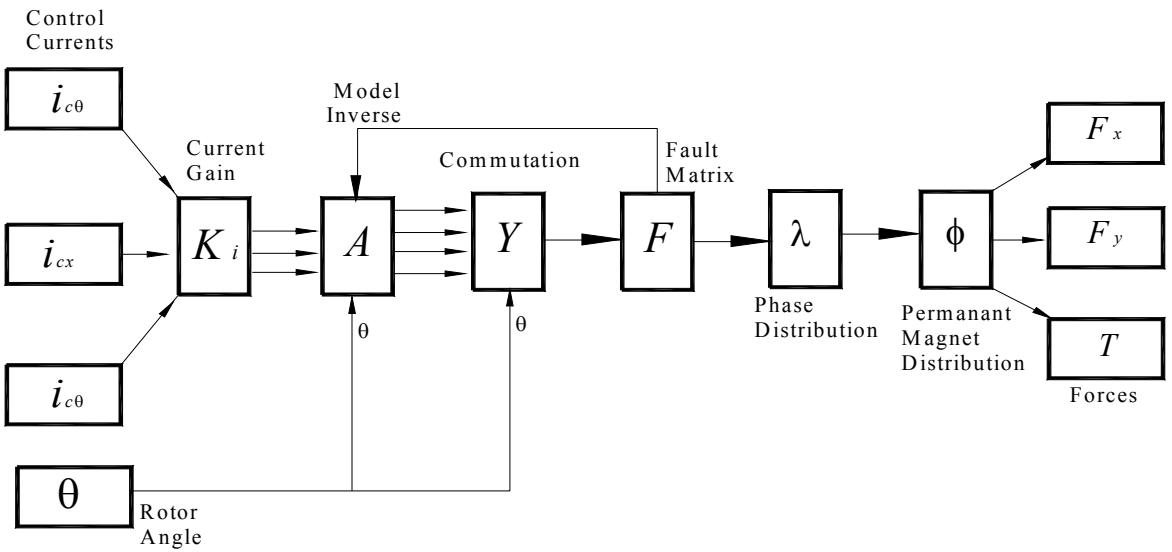


Figure 3.14: Decoupled and fault tolerant model

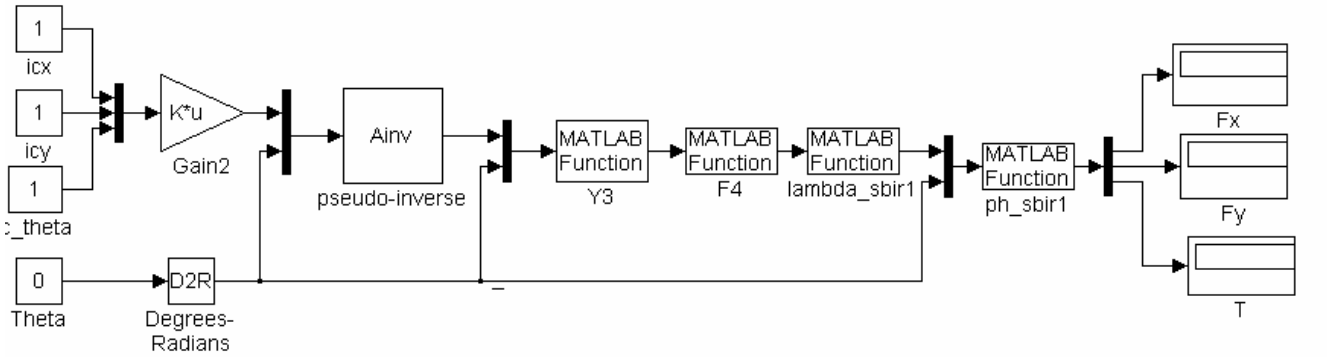


Figure 3.15: Simulink model for decoupled and fault tolerant model

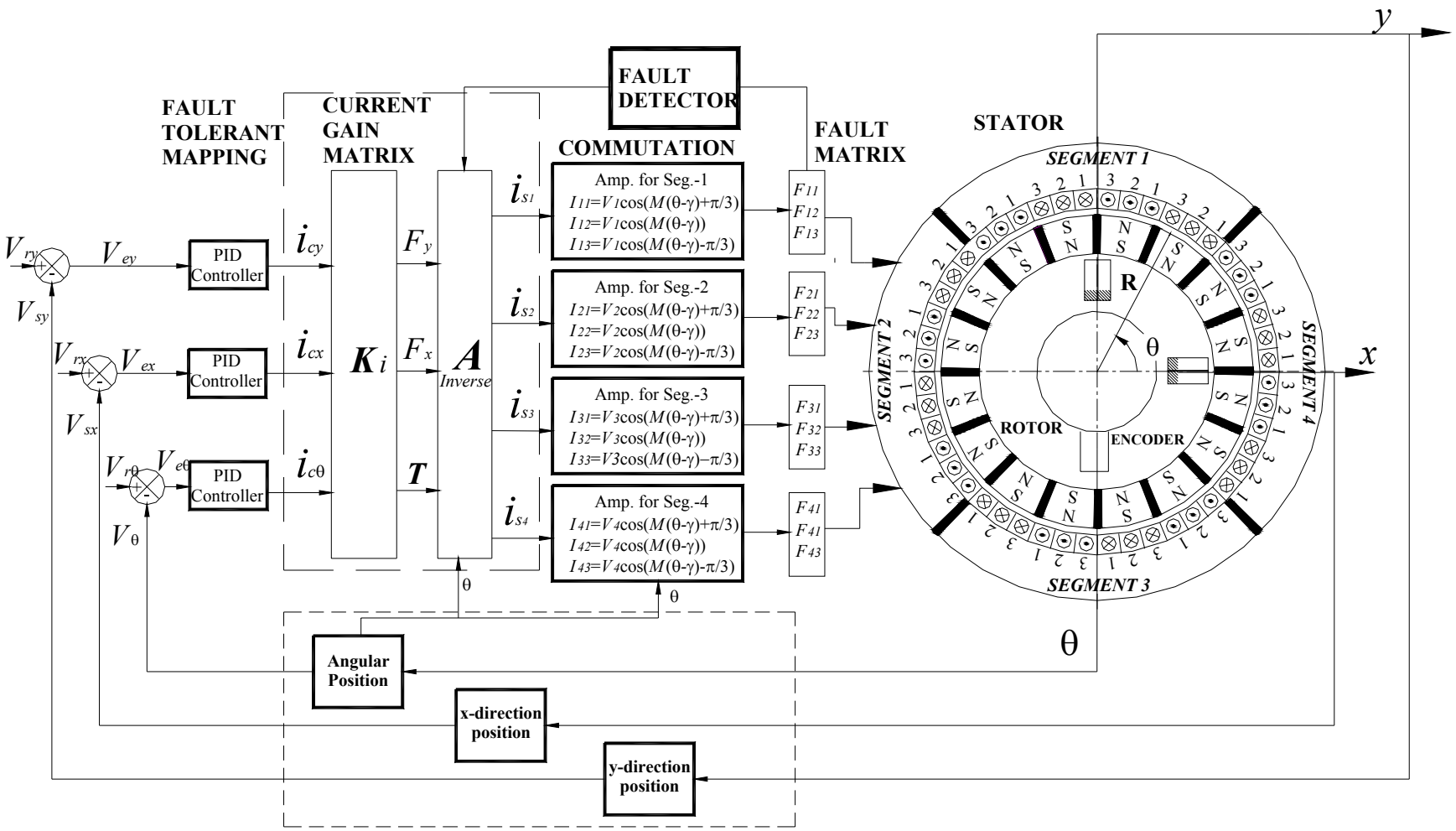


Figure 3.16: Decoupled and fault tolerant control with the actuator

Rotor Angle	Control Currents			Forces																			
				Model 1 Desired Model $\overline{K_i}$			Model 2 Crosscoupled Model $\overline{K_i}$			Model 4 Lumped Parameter Model			Model 4 Integral Equations Model			Model 4 Fault tolerant Model Using Integral Equations			Model 6 Chapter 2 Fault tolerant Model Using Lumped Parameter Model			Model 7 Decoupled and Fault Tolerant Model		
θ	i_x	i_y	i_θ	F_x	F_y	T	F_x	F_y	T	F_x	F_y	T	F_x	F_y	T	F_x	F_y	T	F_x	F_y	T	F_x	F_y	T
0	1	0	0	156.5	0	0	164.6	-9.88	0	156.5	0	0	156.5	0	0	156.5	0	0	156.5	0	0	156.5	0	0
	0	1	0	0	156.5	0	-9.88	164.6	0	0	156.5	0	0	156.5	0	0	156.5	0	0	156.5	0	0	156.5	0
	0	0	1	0	0	38.2	0	0	40.25	0	0	38.2	0	0	38.2	0	0	38.18	0	0	38.2	0	0	38.2
15*(1/4)	1	0	0	156.5	0	0	164.6	-9.88	0	156.3	7.89	0	156.3	6.55	0	156.2	6.53	0	156.3	7.89	0	156.5	0	0
	0	1	0	0	156.5	0	-9.88	164.6	0	-7.89	156.3	0	-6.55	156.3	0	-6.53	156.2	0	-7.89	156.3	0	0	156.5	0
	0	0	1	0	0	38.2	0	0	40.25	0	0	38.2	0	0	38.2	0	0	38.18	0	0	38.2	0	0	38.2
15*(1/2)	1	0	0	156.5	0	0	164.6	-9.88	0	156.1	0	0	156	0	0	155.9	0	0	156.1	0	0	156.5	0	0
	0	1	0	0	156.5	0	-9.88	164.6	0	0	156.1	0	0	156	0	0	155.9	0	0	156.1	0	0	156.5	0
	0	0	1	0	0	38.2	0	0	40.25	0	0	38.2	0	0	38.2	0	0	38.18	0	0	38.2	0	0	38.2
15*(3/4)	1	0	0	156.5	0	0	164.6	-9.88	0	156.3	-7.89	0	156.3	-6.55	0	156.2	-6.53	0	156.3	-7.89	0	156.5	0	0
	0	1	0	0	156.5	0	-9.88	164.6	0	7.885	156.3	0	6.55	156.3	0	6.53	156.2	0	7.885	156.3	0	0	156.5	0
	0	0	1	0	0	38.2	0	0	40.25	0	0	38.2	0	0	38.2	0	0	38.18	0	0	38.2	0	0	38.2
15	1	0	0	156.5	0	0	164.6	-9.88	0	156.5	0	0	156.5	0	0	156.5	0	0	156.5	0	0	156.5	0	0
	0	1	0	0	156.5	0	-9.88	164.6	0	0	156.5	0	0	156.5	0	0	156.5	0	0	156.5	0	0	156.5	0
	0	0	1	0	0	38.2	0	0	40.25	0	0	38.2	0	0	38.2	0	0	38.18	0	0	38.2	0	0	38.2
0	1	1	1	156.5	156.5	38.2	154.7	154.7	40.25	156.5	156.5	38.2	156.5	156.5	38.2	156.5	156.5	38.18	156.5	156.5	38.2	156.5	156.5	38.2
15*(1/4)	1	1	1	156.5	156.5	38.2	154.7	154.7	40.25	148.4	164.2	38.2	149.7	162.8	38.2	149.7	162.7	38.18	148.4	164.2	38.2	156.5	156.5	38.2
15*(1/2)	1	1	1	156.5	156.5	38.2	154.7	154.7	40.25	156.1	156.1	38.2	156	156	38.2	155.9	155.9	38.18	156.1	156.1	38.2	156.1	156.1	38.2

Table 3.1: Forces for different fault configurations in different motor models

3.4 Comparison of forces in different models

The Table 3.1 describes the amount of crosscoupling in the different models described above. There are predominantly 7 models discussed in this table. The data analysis in the table was limited a change in rotor angle through a pole pitch, which is a good representation of the entire rotor angle. This is due to the fact that after the rotation of the magnet through a pole pitch, the same north-south pole configuration will repeat itself. Model 1 is the desired model and the goal of decoupled and fault tolerant model. In this case, the forces produced are not dependent on the rotor angle and there is no crosscoupling of forces. Model 2 is the crosscoupled model and is a very approximate way of representing the force-current relationship in the existing model. Model 3 is the lumped parameter model and the forces produced by it are dependent on rotor angle and crosscoupled. Model 4 is the integral equations model and is closer to the actual model of the motor. It was found from the simulation that the crosscoupling of both the lumped parameter model and the integral equations model were highest, when the rotor angle was one-forth of a pole pitch. At this rotor angle and [i_x=1, i_y=0, i_θ=0], the force produced in y-direction is F_{xy} = -7.89 in the lumped parameter model and is F_{xy} = -6.55 in the integral equations model. The forces produced are dependent on rotor angle and are less crosscoupled than the lumped parameter model. Model 5 is the fault tolerant model based on the integral equations model. In no fault condition, this model is as good the integral equations model. Forces produced in this model are also dependent on rotor angle and exhibit the same crosscoupling as the integral equations model in no fault condition. Model 6 is the fault tolerant model based on

the lumped parameter model. In “no fault” condition, this model is as good the lumped parameter model. Forces produced in this model are also dependent on rotor angle and exhibit the same crosscoupling as the lumped parameter model in “no fault” condition. In both models 5 and 6, A and A^+ cancel out each other leaving out the base model (integral equations and lumped parameter model). In both models 5 and 6 , A and A^+ cancels even if there are faulted phases thus allowing the base model to define the force-current relationship. Both the models are fault tolerant, but are as crosscoupled as the base model. Model 7 is the decoupled & fault tolerant model and is as good as the desired model. The forces produced are not dependent on rotor angle and are not crosscoupled. For any rotor angle, the control current [i_x=1, i_y=0, i_θ=0] produces the desired force (F_{xx}=156.5) only in that direction and no force in the other directions (F_{xy}=0). The trend is seen when a control current [i_x=0, i_y=1, i_θ=0] is sent in y-direction. The advantage of this model is that A and A^+ cancel out each other leaving out the desired K_i. The desired K_i does not have any crosscoupled terms in it, causing the forces to be decoupled. In this model, A and A^+ would cancel out even when the phases are faulted, letting the desired K_i to define the force-current relationship.

1-Integral Equations, 2-Lumped Parameter model, 3-Fault tolerant control model based on integral equations, 4-Fault tolerant control model based on lumped parameter model, 5-Decopled and fault tolerant control

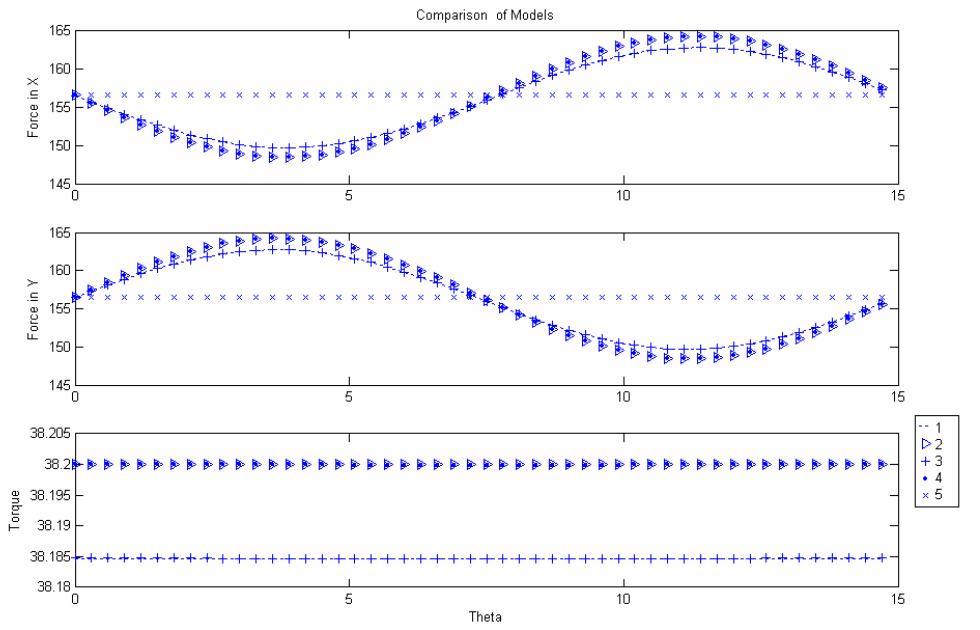


Figure 3.17: Desired forces when rotor angle turns through a pole pitch in no fault conditions, $\mathbf{i}_c = [1\ 1\ 1]$

Figure 3.17 plots the forces produced for different models when the rotor angle turns through a pole pitch in no fault conditions, $\mathbf{i}_c = [1\ 1\ 1]$. In the plots, K_i desired model and the decoupled fault tolerant model are a straight line running from left to right as the rotor angle increases through one pole pitch. The decoupled fault tolerant model is the only model which would meet the desired model specifications. The forces produced in the lumped parameter model and the fault tolerant model based on the lumped parameter model is a sinusoidal wave as shown in the plot and overlap. The plot shows that both the lumped parameter model and fault tolerant model based on the lumped parameter model are equally crosscoupled at every rotor angle. The forces produced in the integral equations model and the fault tolerant model based on the integral equations model is a sinusoidal wave as shown in the plot and overlap. The plot shows that both the integral equations model and fault tolerant model based on integral equations model are equally crosscoupled at

every rotor angle. From the plot, it was inferred that the decoupled fault tolerant model is better than any of the other models. The plot also shows that the fault tolerant model based on the integral equations model is better than the fault tolerant model based in the lumped parameter model because it is less crosscoupled.

Models	K_i Desired	Integral Equations Model	Lumped Parameter Model	Fault Tolerant Model using Integral Equations	Fault Tolerant Model	Decoupled and Fault Tolerant Model
Percentage Maximum Difference in FX	0	4.1762	5.0365	4.1762	5.0365	0
Percentage Maximum Difference in FY	0	4.1762	5.0365	4.1762	5.0365	0
Percentage Maximum Difference in T	0	0	0	0	0	0

Table 3.2: Percentage maximum variation in FX & FY from the desired forces when rotor angle turns through a pole pitch, $\mathbf{i}_c = [1\ 1\ 1]$, No fault condition

Table 3.2 shows the percentage of maximum variation in FX & FY from the desired forces when rotor angle turns through a pole pitch, $\mathbf{i}_c = [1\ 1\ 1]$ in no fault conditions. The percentage variation of FX and FY in the lumped parameter model (5.04%) is more than the integral equations model (4.17%). Percentage variations in the fault tolerant models based on the respective base models are also the same. Hence the fault tolerant model based on the integral equations model is better than the fault tolerant model based on the lumped parameter model. The table shows that the decoupled and fault tolerant model has 0 percentage variation in FX and FY

from the desired, when the rotor angles turns through a pole pitch. Thus the decoupled and fault tolerant model is better than the other models with respect to crosscoupling. There is no crosscouping in θ direction in any of the models.

All of the above discussions have been restricted to no fault conditions. Now the faults were introduced and the models were tested for fault tolerance. There are three models discussed: The non-fault tolerant model (lumped parameter model), the fault Tolerant based on the lumped parameter model and the decoupled fault tolerant control model. Table 3.3 compares the forces between these three models. In the Non-fault tolerant model, for an input control current $\mathbf{i}_c = [1\ 0\ 0]$ and rotor angle of θ=0, the force produced in x direction is F_x=156.5 in "no fault" condition. When the segment 1 phase 1 is faulted, the force produced decreases to F_x=143.5. With more faults, the force produced decreases further. In addition to decrease in force in the required direction, there is an increase in crosscoupling of the forces. As the segment 1 phase1 is faulted, the crosscoupled forces increase from F_{xy}=0 to F_{xy}=1.138. In the fault tolerant model based on the lumped parameter model and the decoupled fault tolerant models, force production does not decrease with faults making them superior to the non-fault tolerant model.

	Non-Fault Tolerant system			Fault tolerant system			Decoupled & Fault tolerant system		
Failures	**FX**	**FY**	**T**	**FX**	**FY**	**T**	**FX**	**FY**	**T**
				i_x=1 i_y=0 i_θ=0 θ=0					
No fault	156.5	0	0	156.5	0	0	156.5	0	0
Segment 1									
F11	143.5	1.138	1.592	156.5	0	0	156.5	0	0
F11,F12	91.26	1.138	7.958	156.5	0	0	156.5	0	0
F11,F12,F13	78.25	0	9.55	156.5	0	0	156.5	0	0
Segment 3									
F31	143.5	1.138	-1.592	156.5	0	0	156.5	0	0
F31,F32	91.26	1.138	-7.958	156.5	0	0	156.5	0	0
F31,F32,F33	78.25	0	-9.55	156.5	0	0	156.5	0	0
				i_x=0 i_y=1 i_θ=0 θ=0					
No fault	0	156.5	0	0	156.5	0	0	156.5	0
Segment 2									
F21	-1.138	143.5	1.592	0	156.5	0	0	156.5	0
F21,F22	-1.138	91.26	7.958	0	156.5	0	0	156.5	0
F21,F22,F23	0	78.25	9.55	0	156.5	0	0	156.5	0
Segment 4									
F41	-1.138	143.5	-1.592	0	156.5	0	0	156.5	0
F41,F42	-1.138	91.26	-7.958	0	156.5	0	0	156.5	0
F41,F42,F43	0	78.25	-9.55	0	156.5	0	0	156.5	0
				i_x=0 i_y=0 i_θ=1 θ=0					
No fault	0	0	38.2	0	0	38.2	0	0	38.2
Segment 1									
F11	13.01	-1.138	36.61	0	0	38.2	0	0	38.2
F11,F12	65.24	-1.138	30.24	0	0	38.2	0	0	38.2
F11,F12,F13	78.25	0	28.65	0	0	38.2	0	0	38.2
Segment 2									
F21	1.138	13.01	36.61	0	0	38.2	0	0	38.2
F21,F22	1.138	65.24	30.24	0	0	38.2	0	0	38.2
F21,F22,F23	0	78.25	28.65	0	0	38.2	0	0	38.2
Segment 3									
F31	-13.01	1.138	36.61	0	0	38.2	0	0	38.2
F31,F32	-65.24	1.138	30.24	0	0	38.2	0	0	38.2
F31,F32,F33	-78.25	0	28.65	0	0	38.2	0	0	38.2
Segment 4									
F41	-1.138	-13.01	36.61	0	0	38.2	0	0	38.2
F41,F42	-1.138	-65.24	30.24	0	0	38.2	0	0	38.2
F41,F42,F43	0	-78.25	28.65	0	0	38.2	0	0	38.2

Table 3.3: Comparison of forces between non-fault tolerant, fault tolerant and decoupled & fault Tolerant Control model with different fault configurations

Figures 3.18, 3.19, 3.20 and 3.21 show how forces are produced in the non-fault tolerant, fault tolerant (based on integral equations & lumped parameter model) and decoupled fault tolerant model with the change in rotor angle and the introduction of faults. Figure 3.18 shows forces produced in different models, when segment 1 phase 1 is faulted and the rotor angle turns through a pole pitch, $\mathbf{i}_c = [1\ 0\ 0]$ .Both the fault tolerant and the decoupled fault tolerant models are better than the non-fault tolerant model. They produce the desired force even with faults. Figure 3.19 and Figure 3.20 shows the forces produced when segment 1 phase 1,2 are faulted and segment 1 phase 1,2,3 are faulted respectively. The forces produced are invariant to faults in the case of fault tolerant and decoupled fault tolerant model. Unlike the decoupled fault tolerant model, the fault tolerant model (based on integral equations & lumped parameter model) has some crosscoupling. The degree of crosscoupling is less than the non-fault tolerant model as shown in the figures. Though not clear from the figures, the force produced F_{xx} in the fault tolerant model is close to the desired force, but not precisely equal to that value for most rotor angles. This is due to the fact that the fault tolerant model is not decoupled. From the figures one can infer that the fault tolerant model based on the integral equations is better than the fault tolerant model based on the lumped parameter model even in faulted conditions. With some fault combinations, however the model might fail. One of them is shown in Figure 3.21, where the force produced drastically falls at half-a-pole pitch for the fault configuration [000100010001]. It was inferred from the simulations that decoupled and fault tolerant model is the better than any of the other models. This model was hence implemented in dspace and

the performance of fault tolerance is evaluated. From here on, the decoupled and fault tolerant control model will be referred to as fault tolerant model.

1-Lumped Parameter model
2-Fault tolerant control model based on integral equations
3-Fault tolerant control model based on lumped parameter model
4-Decopled and fault tolerant control

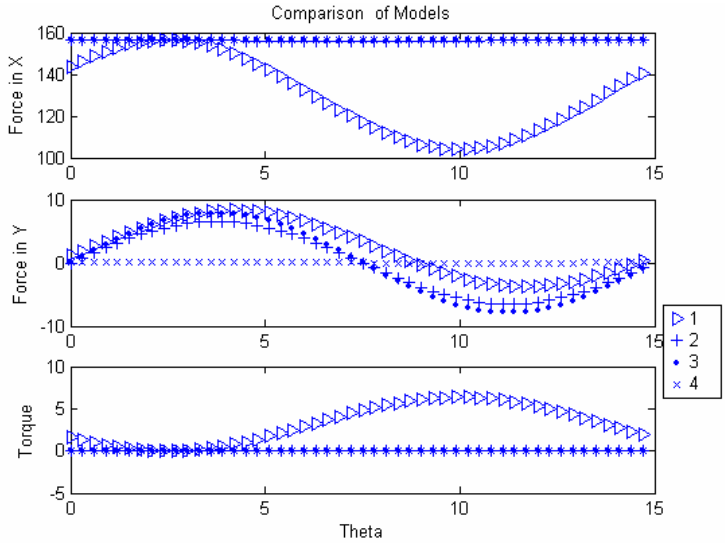


Figure 3.18: Forces produced when segment 1 phase 1 is faulted and rotor angle turns through a pole pitch, $\mathbf{i}_c = [1\ 0\ 0]$

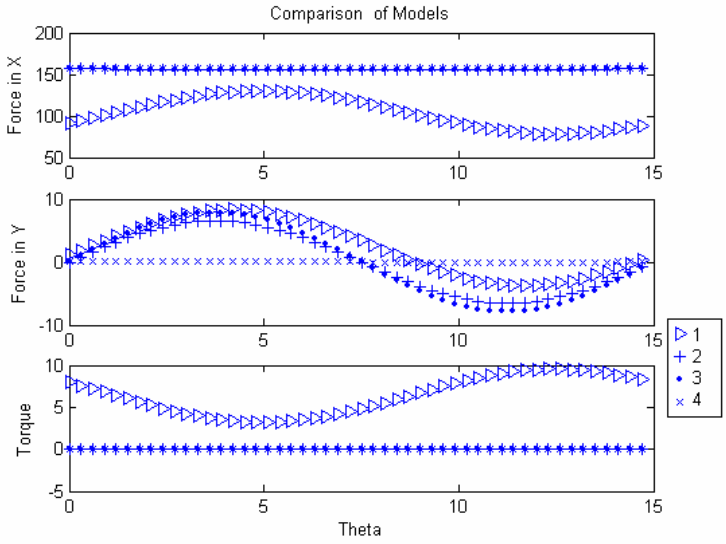


Figure 3.19: Forces produced when segment 1 phase 1,2 is faulted and rotor angle turns through a pole pitch, $\mathbf{i}_c = [1\ 0\ 0]$

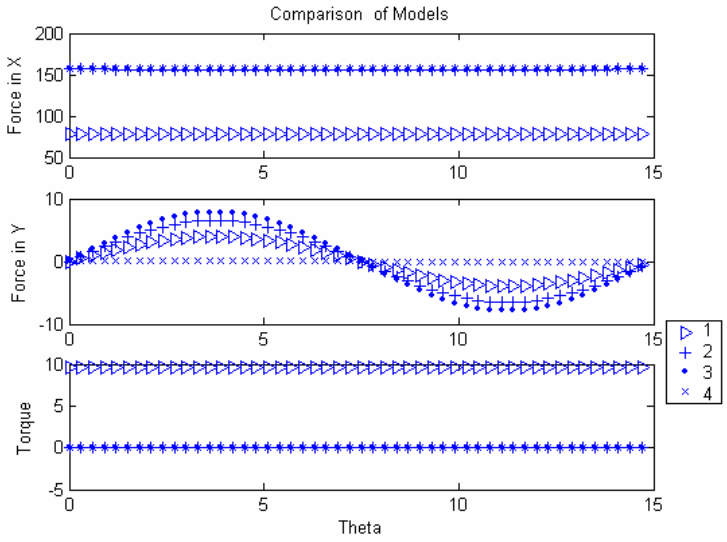


Figure 3.20: Forces produced when segment 1 phase 1,2,3 is faulted and rotor angle turns through a pole pitch, $\mathbf{i}_c = [1\ 0\ 0]$

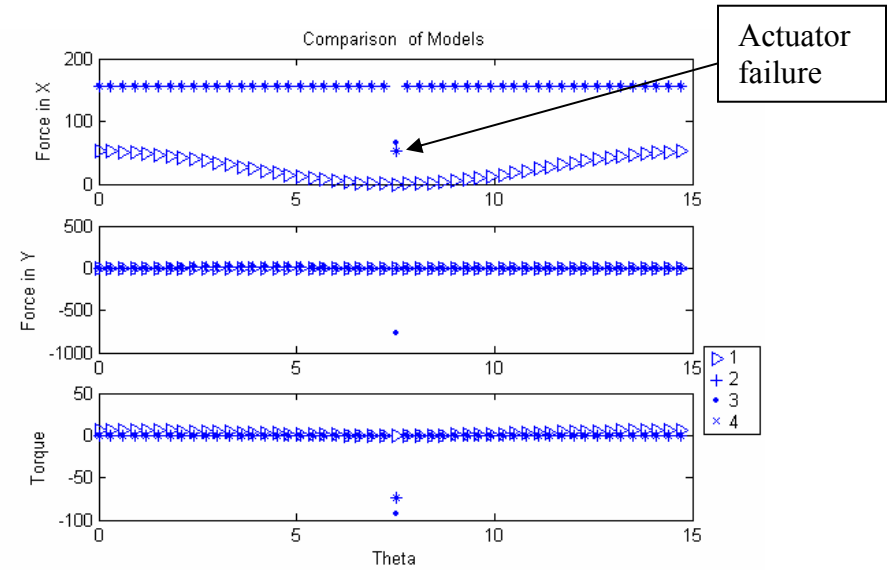


Figure 3.21: Forces produced for the fault configuration [000 100 010 001] and rotor angle turns through a pole pitch, $\mathbf{i}_c = [1\ 0\ 0]$

3.5 Comparison of perturbation voltages

Figure 3.22 shows the perturbation voltages in all twelve phases of the self-bearing motor for the same input control currents i_x=1, i_y=1, i_θ=1 with “no fault” condition. In “no fault” conditions, both the fault tolerant and Non-fault tolerant model send in the same output voltages for the same input control currents. The plot shows how the perturbation currents are nearly the same for both the models. This implies that both decoupled fault tolerant model are as good as the non-fault tolerant model in “no fault” condition. It remains an uncertain from the above simulation if decoupled fault tolerant model is better than non-fault tolerant model. This can be found out only by maintaining the same input control currents for both the fault tolerant and non-fault tolerant model, and then introducing faults in different phases. Now the forces produced in the both these models are compared . Care was taken so that the perturbation voltage values already closer to zero were not zeroed by introduction of faults. Zeroing of perturbation voltage values close to zero would not result in a significant change in the output force and therefore not result in any useful conclusion.

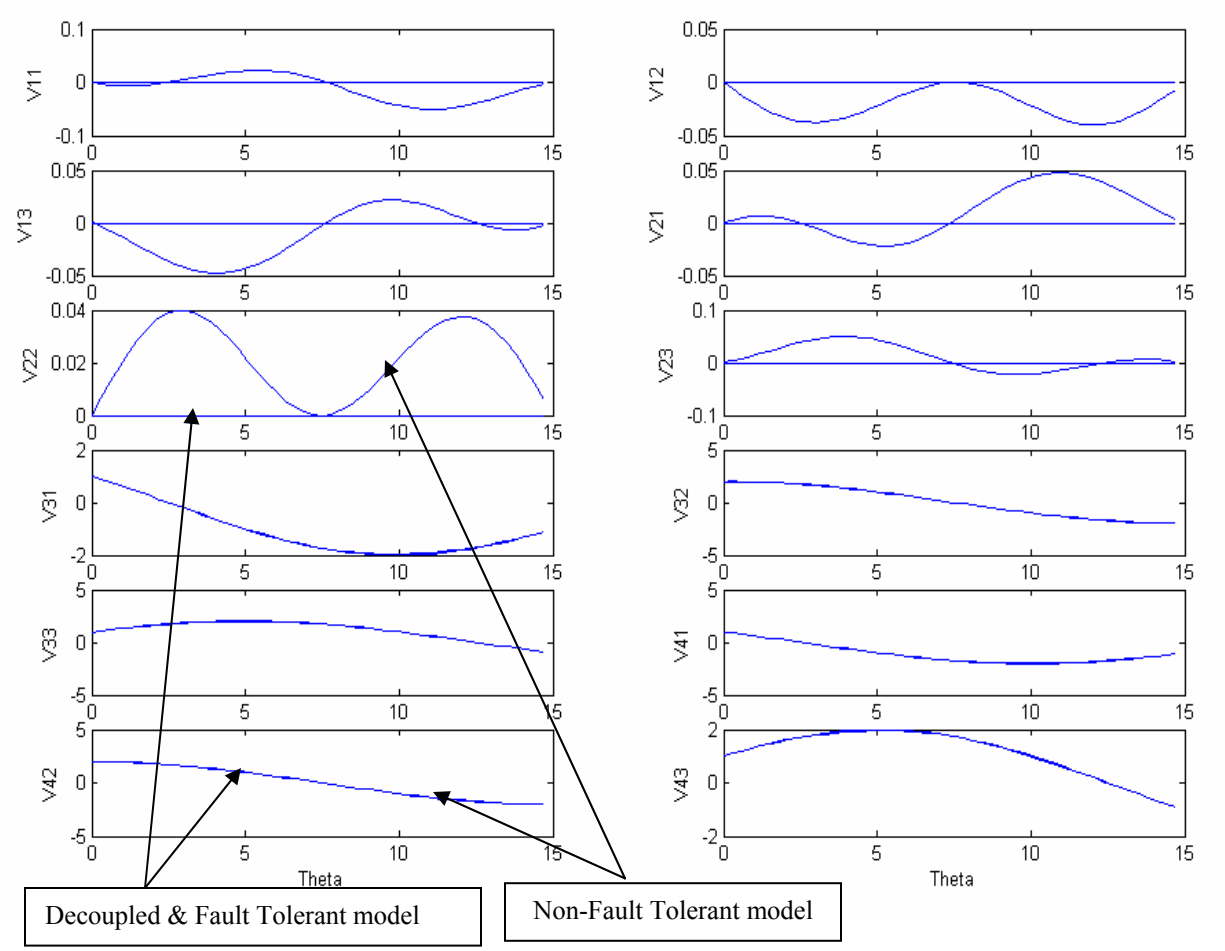


Figure 3.22: Comparison of 12 phase currents between decoupled-fault tolerant and non-fault tolerant model for rotation of the rotor through 1 pole pitch and i_x=1, i_y=1, i_θ=1, "no fault" condition

3.6 Implementation of Fault Tolerant Model

The decoupled and fault tolerant model was found to be the best of the three fault tolerant models. The software equivalents of the model (Φ and λ) shown in the figure are replaced by the actual hardware (power amplifier and the self-bearing motor). Fault tolerance was implemented in the motor based on the decoupling and fault tolerant algorithm using simulink/matlab/rtw/dspace. The model was simulated for any modeling errors and was verified for consistency. Test cases were designed in simulink to verify that the output of the model met with the algorithmic output

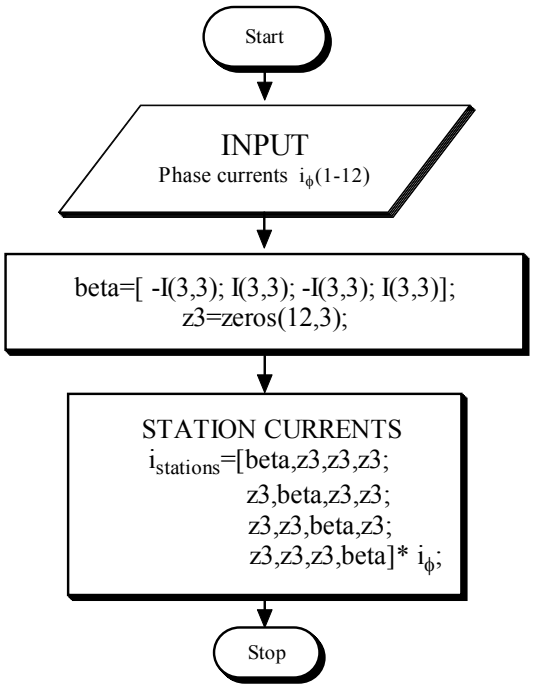
requirements. Now the model was ready for some real-time testing. The non-real-time simulink model based on the decoupled and fault tolerant algorithm was modified by replacing the m-file matlab function with a C-mex file S-function block for pseudo-inverse. The simulink model was modified to interface with the dspace controller hardware. A realistic testing of the algorithm functionality was performed by combining concepts from the rapid control prototyping approach and hardware-in-the-Loop Simulation. Real-Time Workshop was used to generate and execute a stand-alone C code for developing and testing the fault tolerant algorithm modeled in simulink. rtw / simulink / matlab are used to build the simulink model into C codes. The resulting code was used for real-time rapid prototyping and hardware-in-the-loop testing. The generated code can be interactively tuned and monitored using dspace environment.

3.7 Functions used in the simulink models

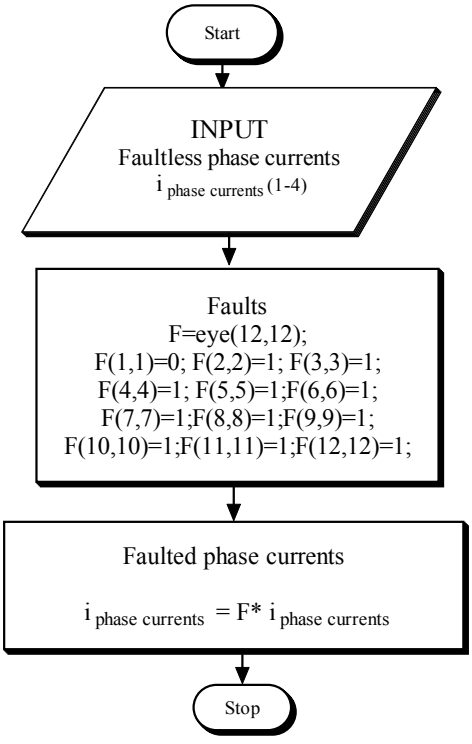
The following flowcharts give the different user defined matlab functions used for simulink blocks in the various models described earlier. There are four medit file functions and two C-mex file S-functions used in the models. The medit files used are used only in simulations and are replaced by using other blocks. The rest of blocks in the decoupled and fault tolerant model, which are impossible to replace with any built-in simulink functions, are made up of C-mex file S-functions so that the simulink model file can be auto-compiled in the dspace environment for real-time control. The two functions written in C are for the pseudo inverse of the model and the flux linkage matrix. Both functions are compiled in the matlab command prompt

by typing “mex file name. c” to convert them into “dll” file. Now the code was dynamically linked with their appropriate block in the model.

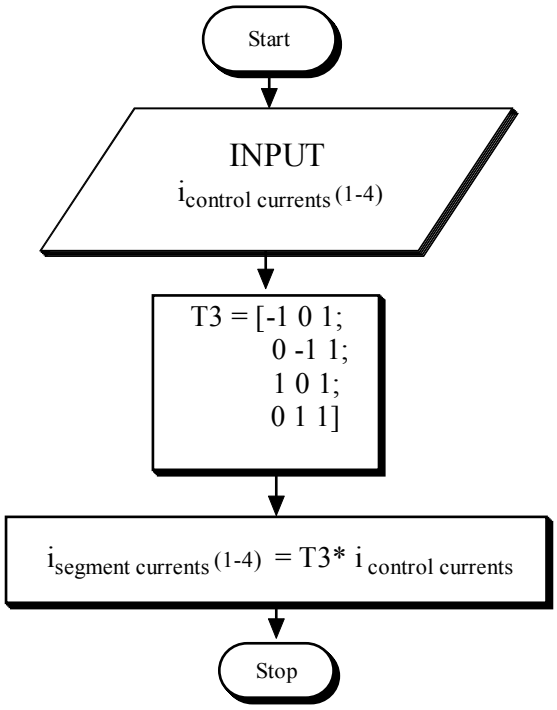
3.7.1 Medit-file function for phase distribution (λ)



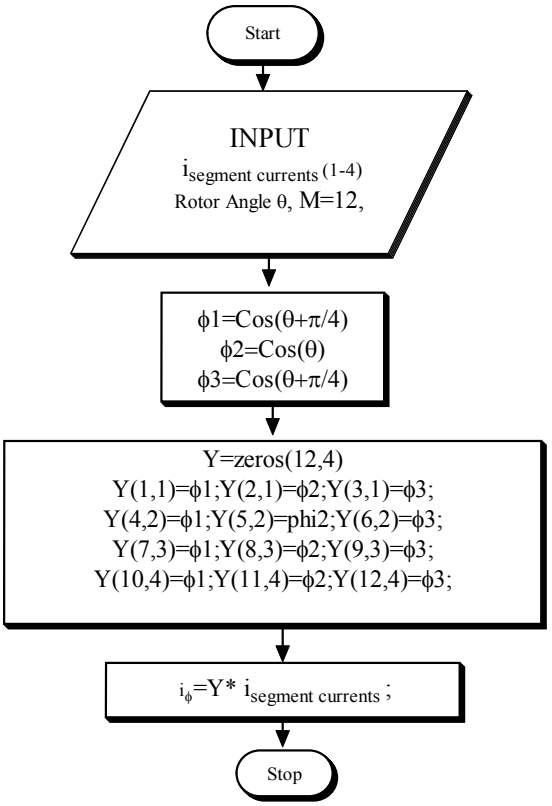
3.7.2 Medit-file function for phase currents after faults introduced (F)



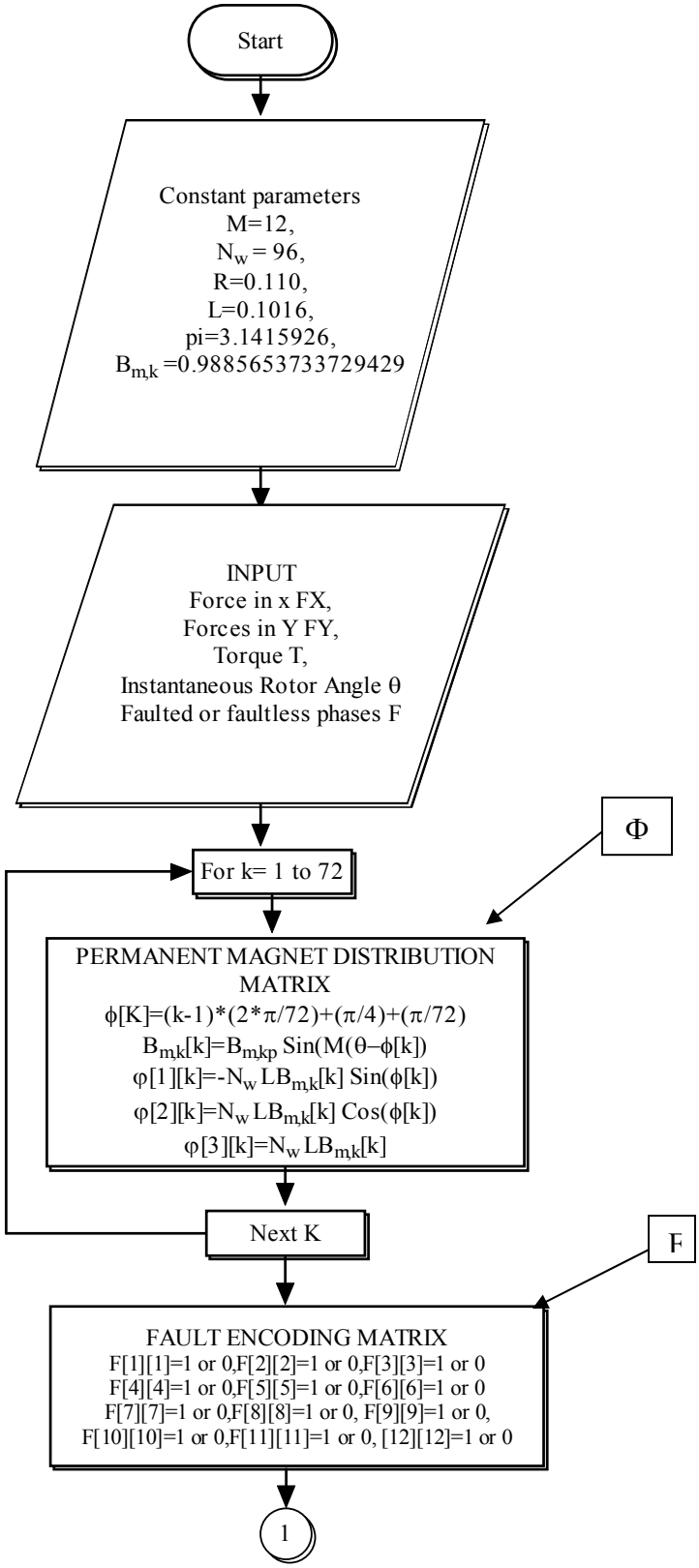
3.7.3 Medit-file Function for segment-Control current mapping (T3)

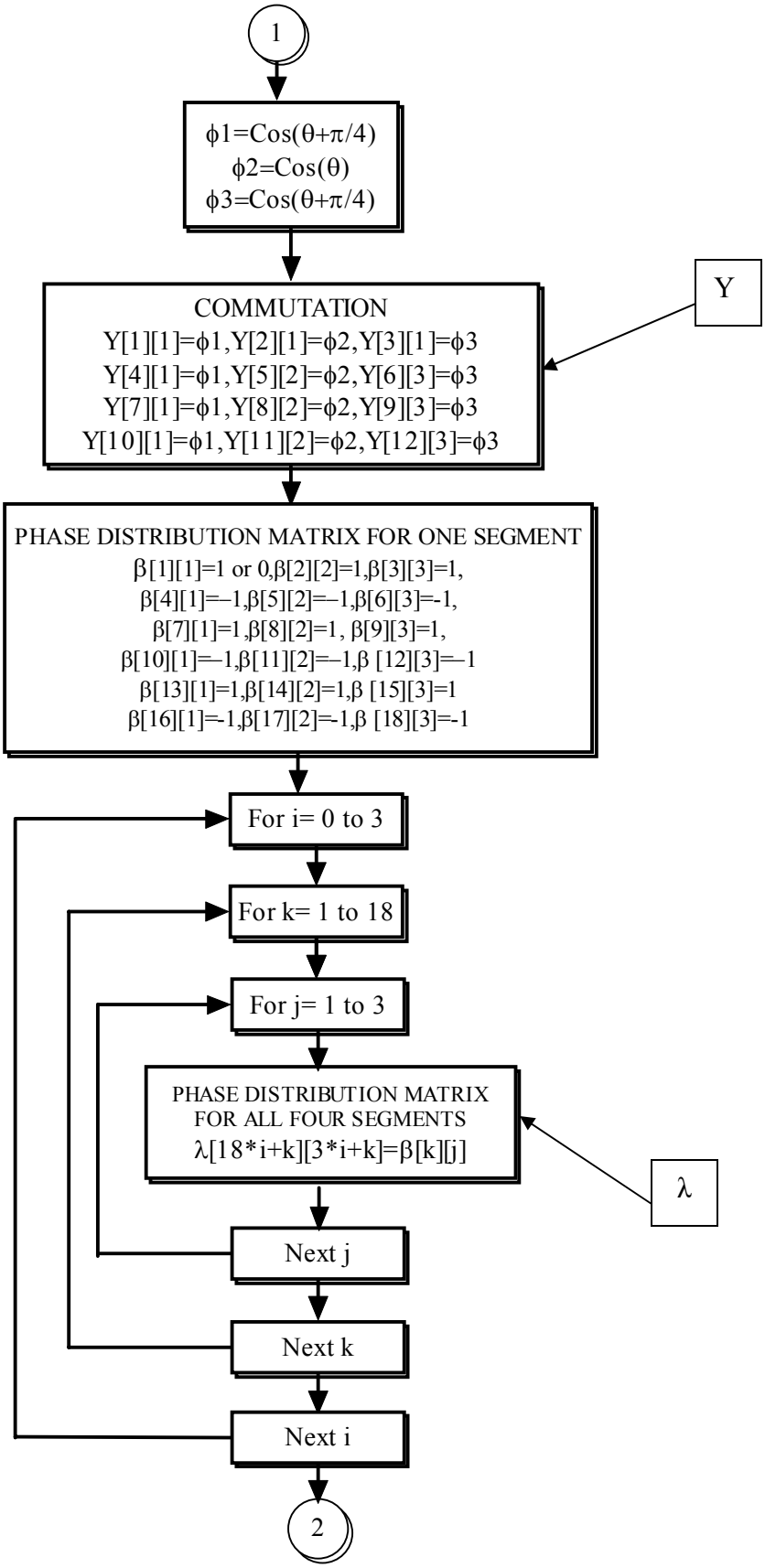


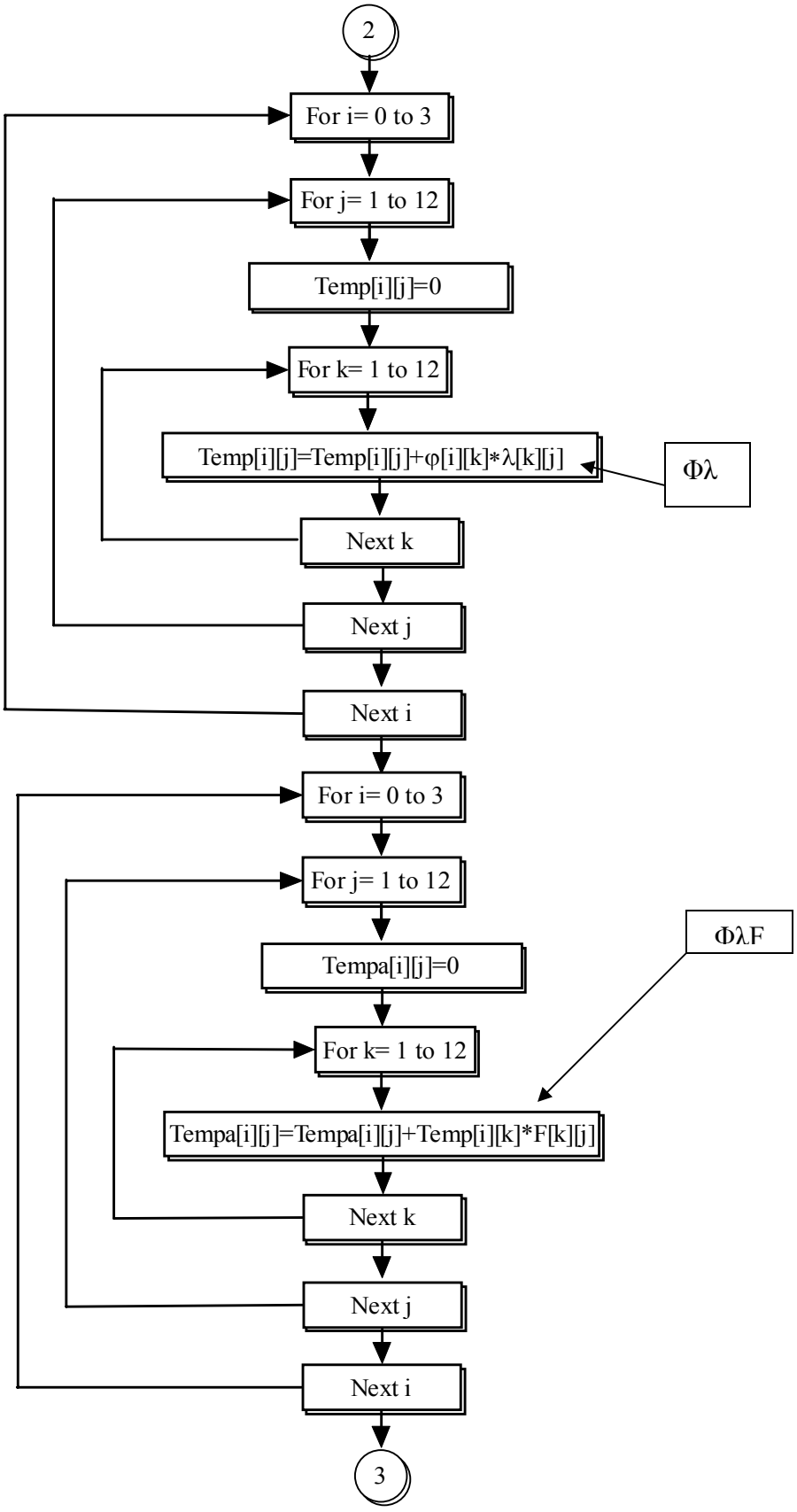
3.6.4 Medit-file function for commutation (Y)

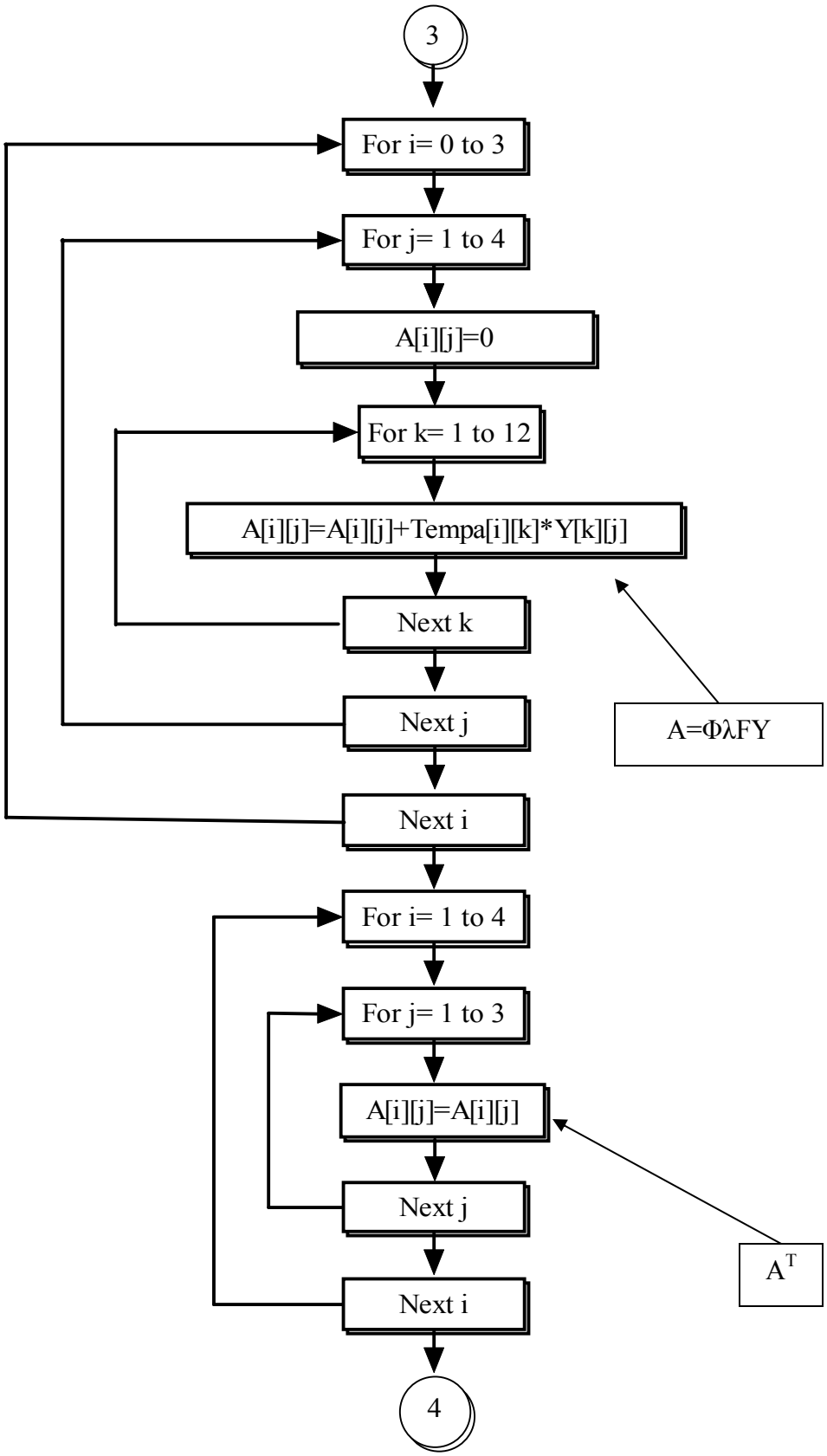


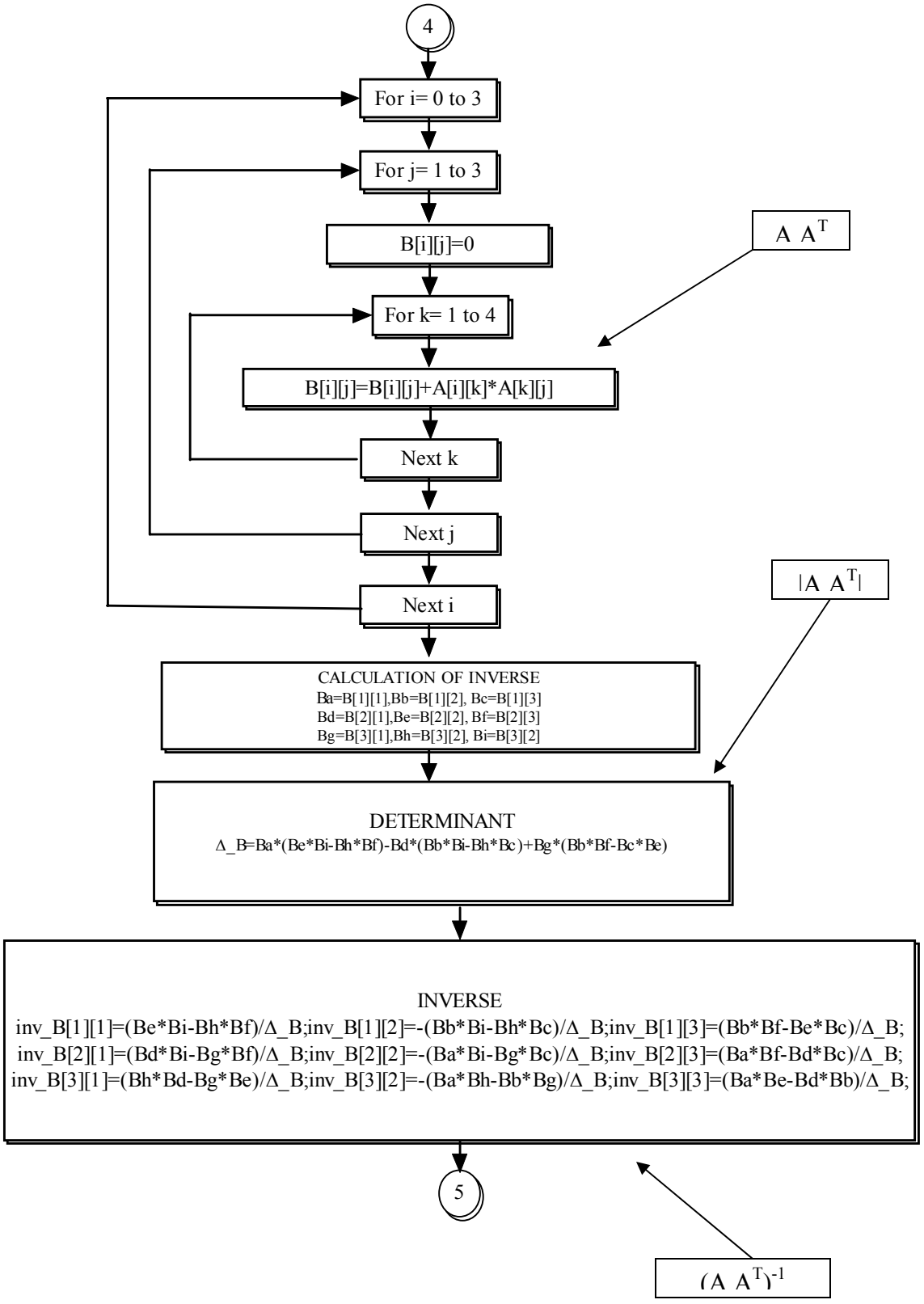
3.6.5 Flowchart to compute the pseudo-inverse in the simulink model (A^{-1})

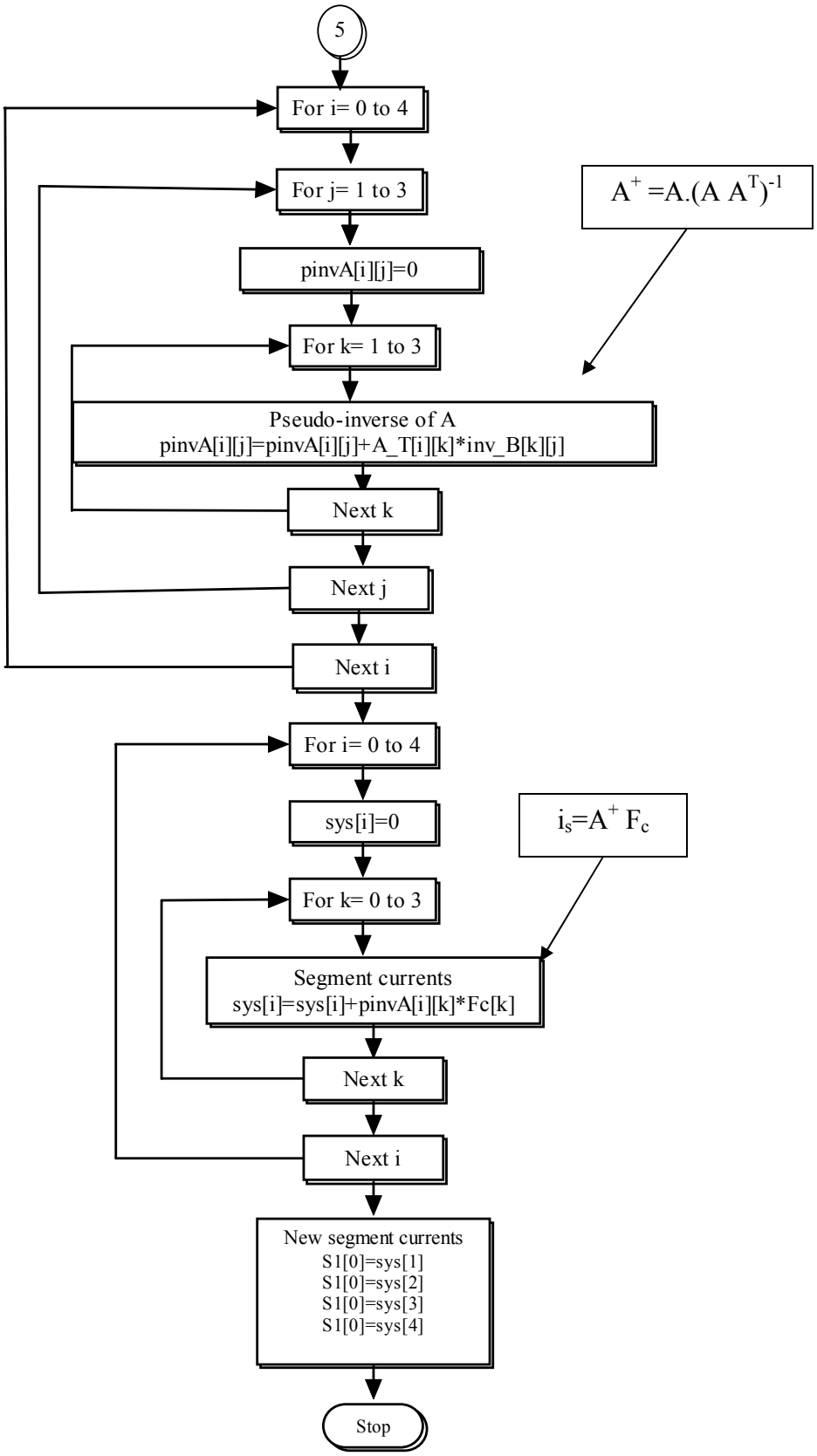



1
ϕ1=Cos(θ+π/4)
ϕ2=Cos(θ)
ϕ3=Cos(θ+π/4)
COMMUTATION
Y[1][1]=ϕ1,Y[2][1]=ϕ2,Y[3][1]=ϕ3
Y[4][1]=ϕ1,Y[5][2]=ϕ2,Y[6][3]=ϕ3
Y[7][1]=ϕ1,Y[8][2]=ϕ2,Y[9][3]=ϕ3
Y[10][1]=ϕ1,Y[11][2]=ϕ2,Y[12][3]=ϕ3
Y
PHASE DISTRIBUTION MATRIX FOR ONE SEGMENT
β[1][1]=1 or 0,β[2][2]=1,β[3][3]=1,
β[4][1]=–1,β[5][2]=–1,β[6][3]=-1,
β[7][1]=1,β[8][2]=1, β[9][3]=1,
β[10][1]=–1,β[11][2]=–1,β [12][3]=–1
β[13][1]=1,β[14][2]=1,β [15][3]=1
β[16][1]=-1,β[17][2]=-1,β [18][3]=-1
For i= 0 to 3
For k= 1 to 18
For j= 1 to 3
PHASE DISTRIBUTION MATRIX
FOR ALL FOUR SEGMENTS
λ[18*i+k][3*i+k]=β[k][j]
λ
Next j
Next k
Next i
2


2
For i= 0 to 3
For j= 1 to 12
Temp[i][j]=0
For k= 1 to 12
Temp[i][j]=Temp[i][j]+φ[i][k]*λ[k][j]
Φλ
Next k
Next j
Next i
For i= 0 to 3
For j= 1 to 12
Tempa[i][j]=0
For k= 1 to 12
ΦλF
Tempa[i][j]=Tempa[i][j]+Temp[i][k]*F[k][j]
Next k
Next j
Next i
3

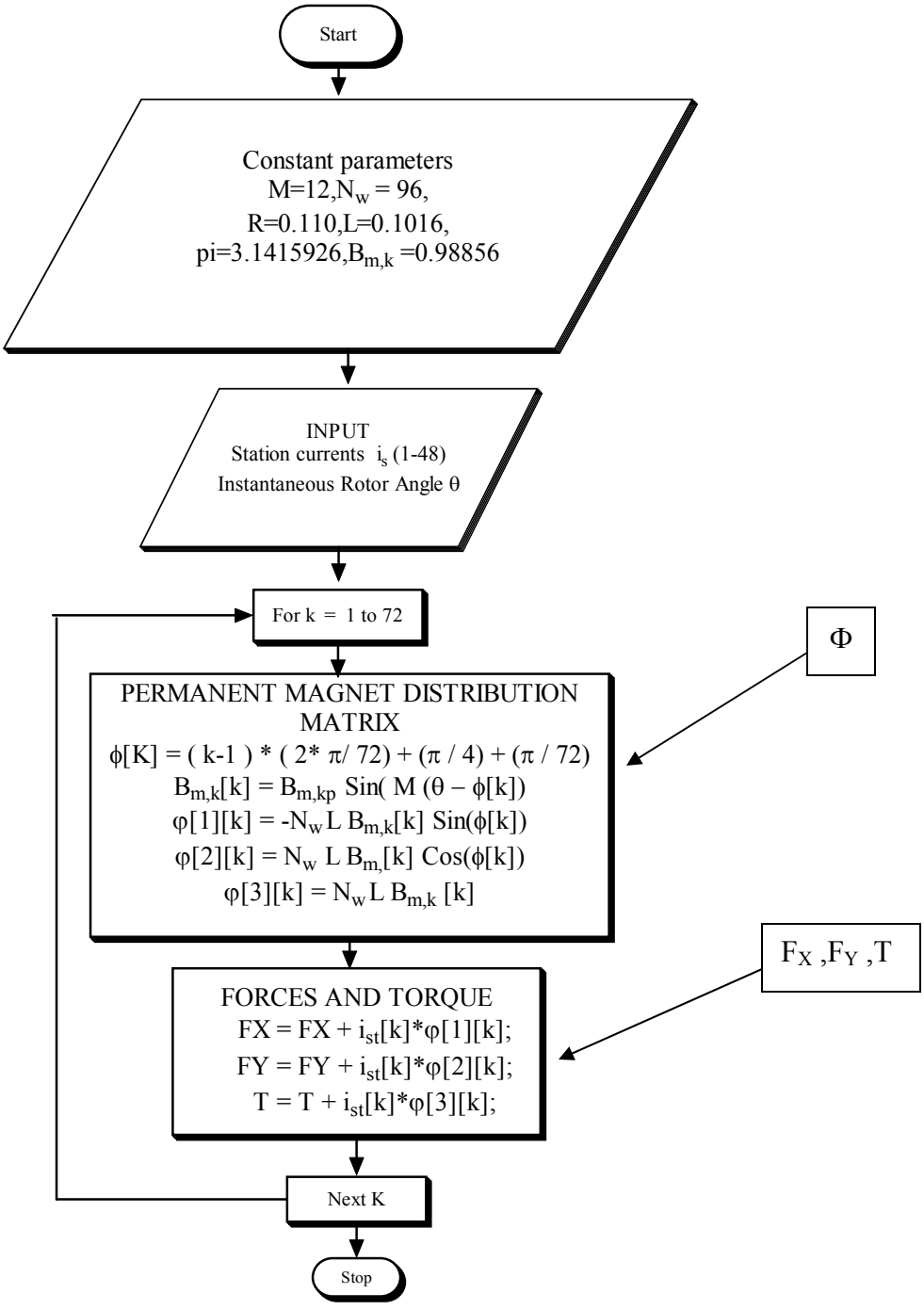

3
For i= 0 to 3
For j= 1 to 4
A[i][j]=0
For k= 1 to 12
A[i][j]=A[i][j]+Tempa[i][k]*Y[k][j]
Next k
Next j
Next i
A=ΦλFY
For i= 1 to 4
For j= 1 to 3
A[i][j]=A[i][j]
Next j
Next i
A^T
4


4
For i= 0 to 3
For j= 1 to 3
B[i][j]=0
For k= 1 to 4
B[i][j]=B[i][j]+A[i][k]*A[k][j]
Next k
Next j
Next i
A A^T
|A A^T|
CALCULATION OF INVERSE
Ba=B[1][1],Bb=B[1][2], Bc=B[1][3]
Bd=B[2][1],Be=B[2][2], Bf=B[2][3]
Bg=B[3][1],Bh=B[3][2], Bi=B[3][2]
DETERMINANT
Δ_B=Ba*(Be*Bi-Bh*Bf)-Bd*(Bb*Bi-Bh*Bc)+Bg*(Bb*Bf-Bc*Be)
INVERSE
inv_B[1][1]=(Be*Bi-Bh*Bf)/Δ_B;inv_B[1][2]=-(Bb*Bi-Bh*Bc)/Δ_B;inv_B[1][3]=(Bb*Bf-Be*Bc)/Δ_B;
inv_B[2][1]=(Bd*Bi-Bg*Bf)/Δ_B;inv_B[2][2]=-(Ba*Bi-Bg*Bc)/Δ_B;inv_B[2][3]=(Ba*Bf-Bd*Bc)/Δ_B;
inv_B[3][1]=(Bh*Bd-Bg*Be)/Δ_B;inv_B[3][2]=-(Ba*Bh-Bb*Bg)/Δ_B;inv_B[3][3]=(Ba*Be-Bd*Bb)/Δ_B;
5
(A A^T)^-1


5
For i= 0 to 4
For j= 1 to 3
pinvA[i][j]=0
For k= 1 to 3
Pseudo-inverse of A
pinvA[i][j]=pinvA[i][j]+A_T[i][k]*inv_B[k][j]
Next k
Next j
Next i
For i= 0 to 4
sys[i]=0
For k= 0 to 3
Segment currents
sys[i]=sys[i]+pinvA[i][k]*Fc[k]
Next k
Next i
New segment currents
S1[0]=sys[1]
S1[0]=sys[2]
S1[0]=sys[3]
S1[0]=sys[4]
Stop
$A^+ = A.(A\ A^T)^{-1}$
$i_s = A^+ F_c$

3.6.6 Flowchart to compute the forces and torque in the simulink model (Φ)

.



Chapter 4

Experimental performance of decoupled and fault tolerant control

This chapter will discuss the experimental results obtained from the self-bearing motor test-rig after the implementation of the fault tolerant control. The fault tolerant control was added to the non-fault tolerant control in the same simulink model and then auto-compiled into the dspace environment. Both the models are subjected to same set of experiments and the performance was evaluated. A dspace layout was created for testing the fault tolerant control. The goal of this chapter is to prove experimentally that the decoupled and fault tolerant control is better than the non-fault tolerant control. The experimental performance of the decoupling and fault tolerant algorithm was evaluated and compared with the simulations as well as the non-fault tolerant control model. This chapter compares the non-fault and fault tolerant control in terms of stability, closed loop stiffness, torsional stiffness and power consumption.

The layout in dspace is shown in figure 4.1. As shown in the figure, the layout has indicators and controls to manipulate the output voltages using both fault tolerant and non-fault tolerant control. Faults can be deliberately introduced in the phases by changing the value of the fault control in the dspace layout from “1” to “0”

in that particular phase. “1” represents the presence of the phase and “0” represents the absence of the phase. There are indicators showing the 12 phase voltages to compare the fault tolerant control with the non-fault tolerant control. The total voltage is the sum of the 12 phase voltages and was compared in both the models. A button was used to switch from non-fault tolerant control to fault tolerant control in real-time.

4.1 Risk-free testing of the fault tolerant model

A risk free testing of the fault tolerant control model can be done by looking at the 12 phase currents for both fault tolerant and non-fault tolerant model (Figure 4.2). This experiment was performed even without switching the power amplifier on. However the dspace controller and sensor signals are still kept on. Hence the sensor signals are taken in the dspace and serve a common input control current and angular position to both the models. The angular position was varied by rotating the shaft by hand and hence changing it. The input control currents are varied by changing the reference signal. For “no fault” condition, the phase voltages in both models are nearly identical proving the fact that the test rig can be levitated using the decoupled and fault tolerant control at “no fault” condition.

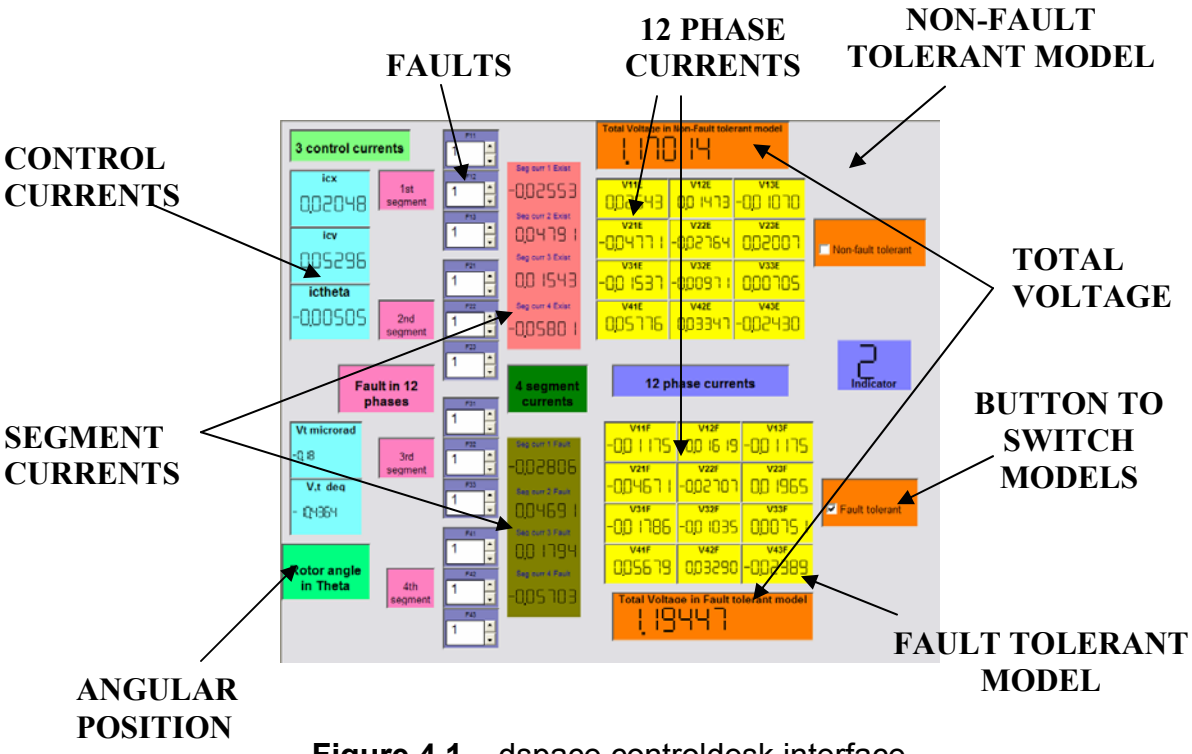


Figure 4.1 – dspace controldesk interface

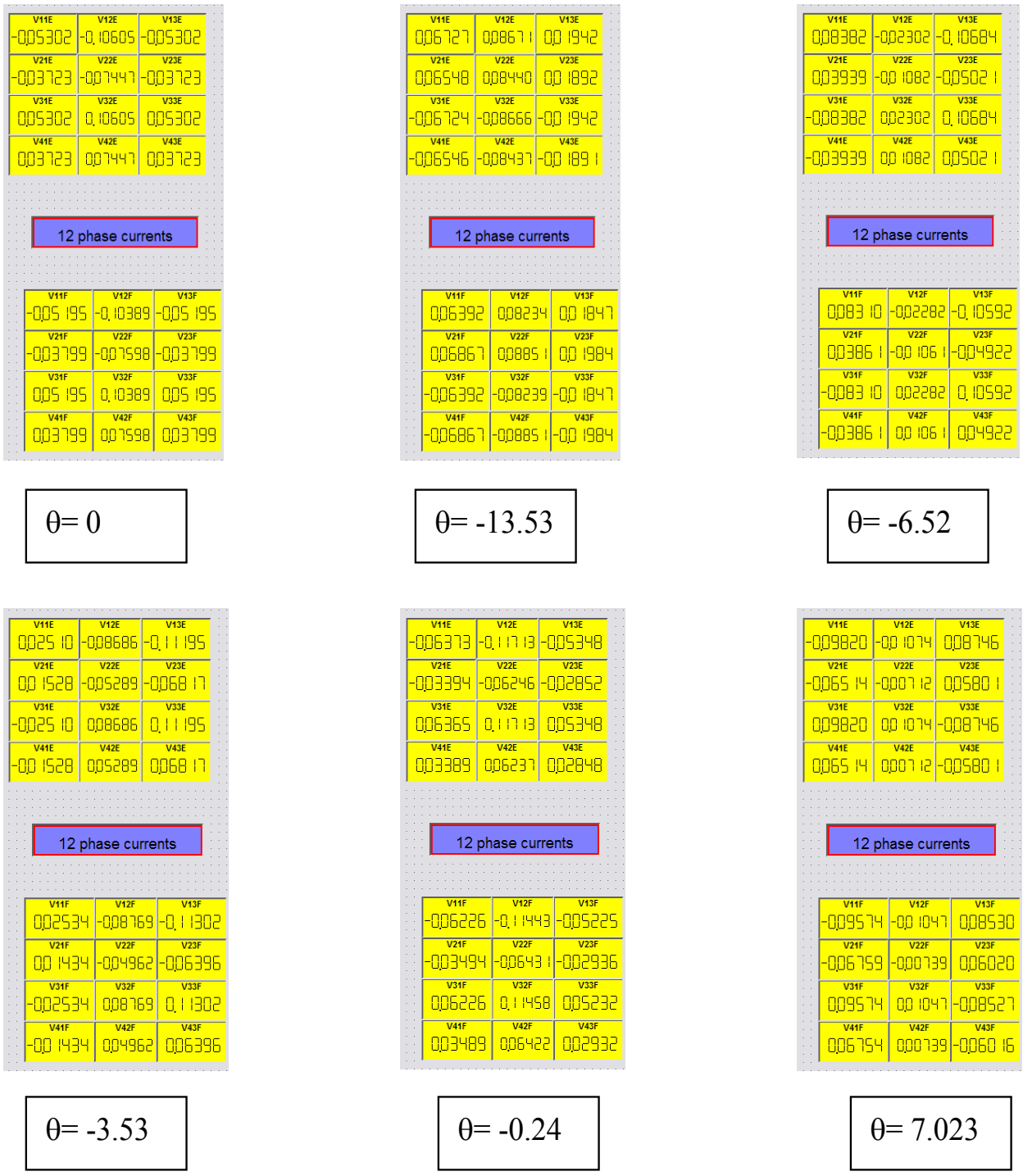


Figure 4.2: Phase voltages with change in rotor angles
(Power amplifier switched off)

4.2 Test of stiffness in fault tolerant and non-fault tolerant control

The motor was levitated and faults are introduced in the non-fault tolerant model & fault tolerant control model. The x and y sensor signals are looked at in dspace control desk as shown in the figure. The sensor signals gave the position of shaft in x and y directions and indirectly represent the stiffness of the motor. In the non-fault tolerant control, the shaft moves away from the center (to the left in this case) indicating a decrease in the closed loop stiffness of the motor. The decrease in stiffness would cause the system to be less stable. But in the fault tolerant model, the shaft remains at the center and hence retains the stiffness. The purpose of the control is to keep the shaft at the center even with faults. Hence the fault tolerant control was better than the non-fault tolerant control in a faulty environment.

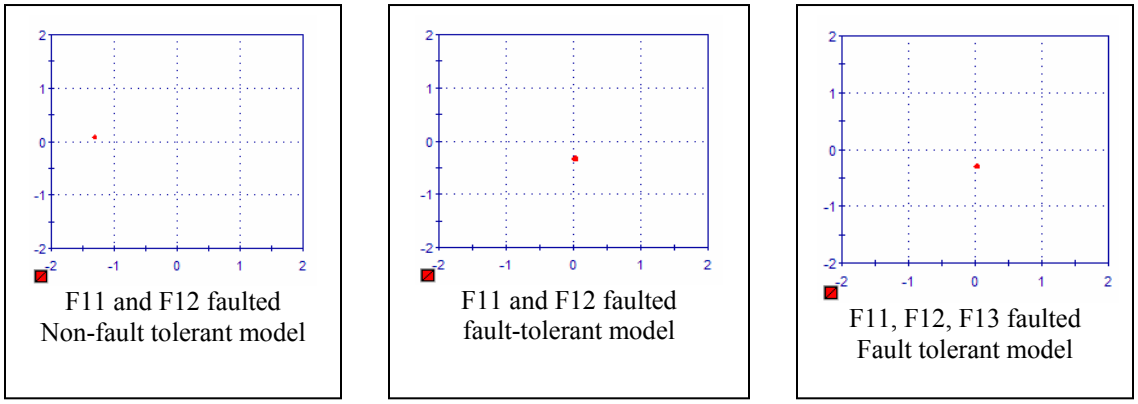


Figure 4.3 – X and Y position of the shaft indicating the stiffness of the motor

4.3 Sine sweep test

In this test, fault tolerance was evaluated when the motor shaft was made to do a sine sweep. Both the non-fault tolerant control and fault tolerant control are subjected to this test. Faults are introduced while the shaft was executing a sine sweep. When F_{11} and F_{12} are faulted, the non-fault tolerant control (NF) was stable

in its θ direction but was unstable in its radial direction (Figure 4.4). If F_{11},F_{12} and F_{13} are faulted, then the motor goes unstable in both radial as well as θ direction. For the same fault configuration, the motor would be stable in radial direction and θ direction with the fault tolerant control (F).

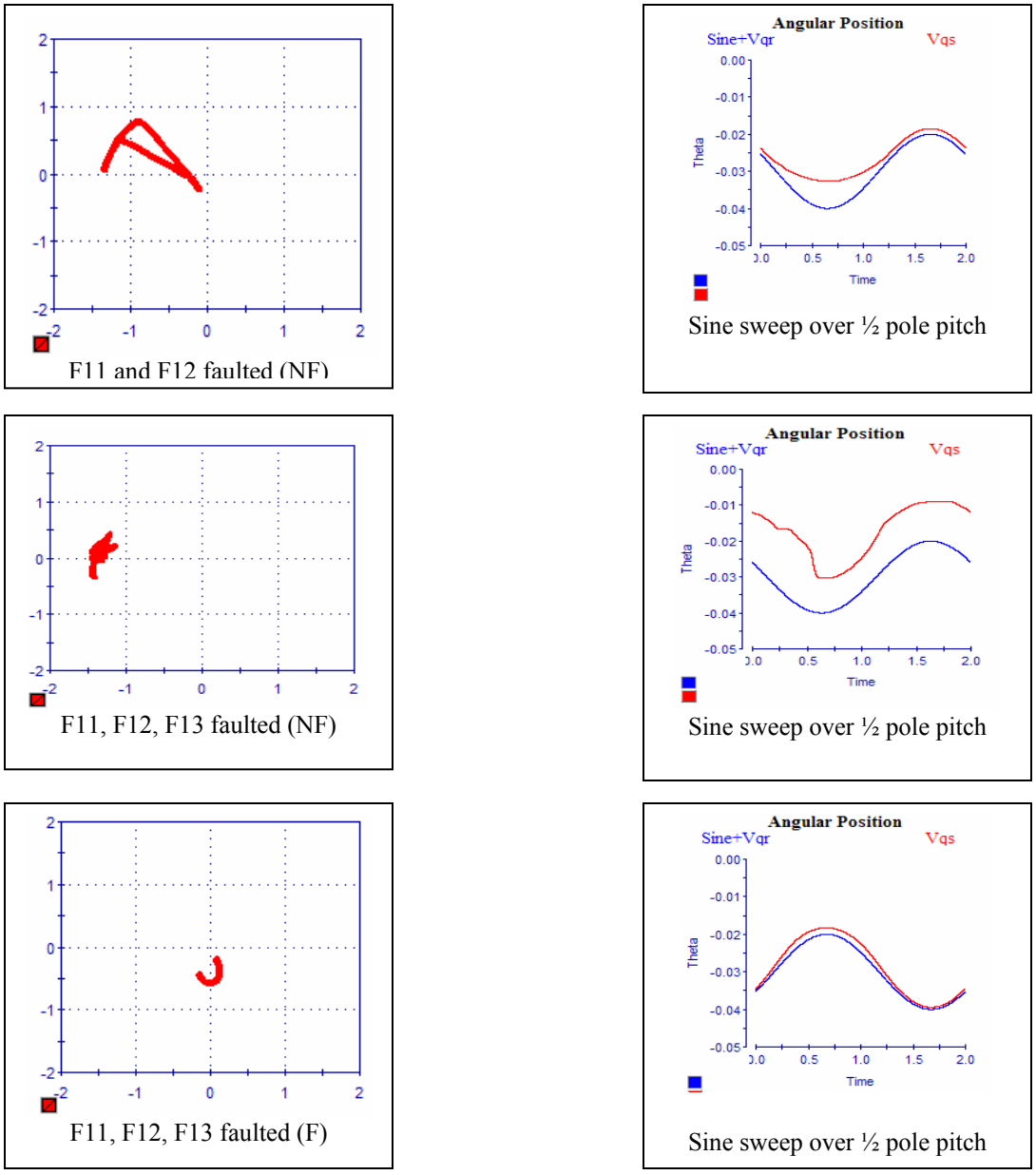


Figure 4.4: Sine sweep test

4.4 Power consumed in Non-fault tolerant model and Fault tolerant model

The motor was levitated and the total voltage was seen from the dspace layout for both the non-fault tolerant and fault tolerant control. This total voltage is the sum of all the 12 phase voltages. A transconductance amplifier was used to amplify the phase voltages to currents. A 1V supplied to such an amplifier produces 4 A of current. Hence the total current is four times the magnitude of the total voltage. The power consumed is the square of the total current consumed times the resistance of the winding of the motor. When there are no faults in the system, the total power required was about the same in both fault tolerant and non-fault tolerant system. When faults are introduced, the power consumed was more for the fault tolerant control than non-fault tolerant control. But in a fault-tolerant control, the total power consumed in the motor with faulty phases was found to be more than that required in a fault less system. As the faults increase, the power consumption increases as well. Fault tolerance was achieved in the motor at the cost of increased power consumption.

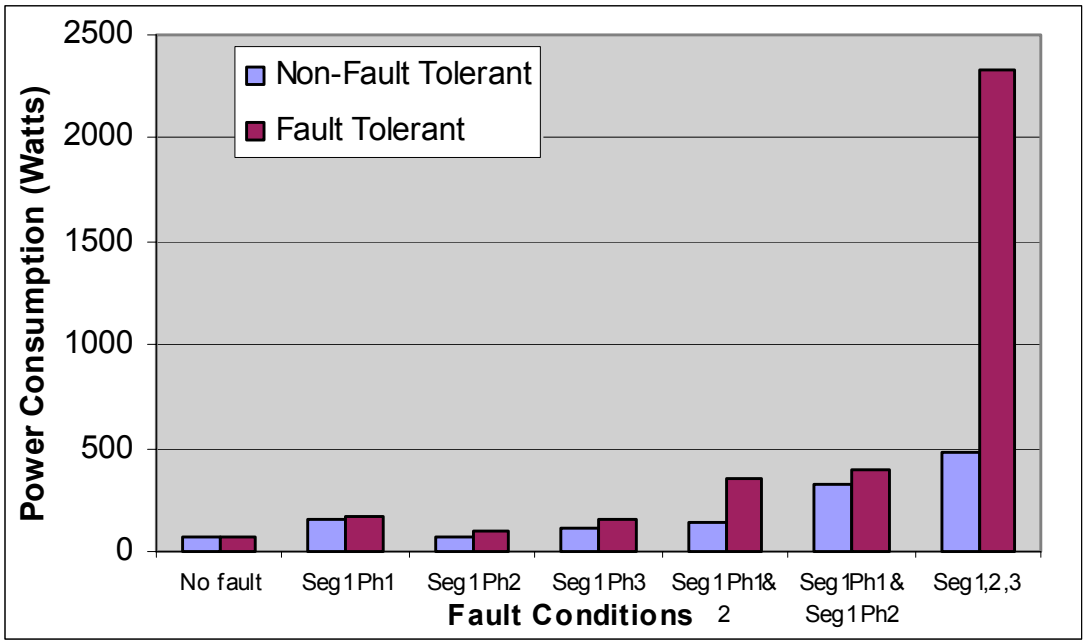


Figure 4.5 – Instantaneous Power consumption in non-fault and fault tolerant control

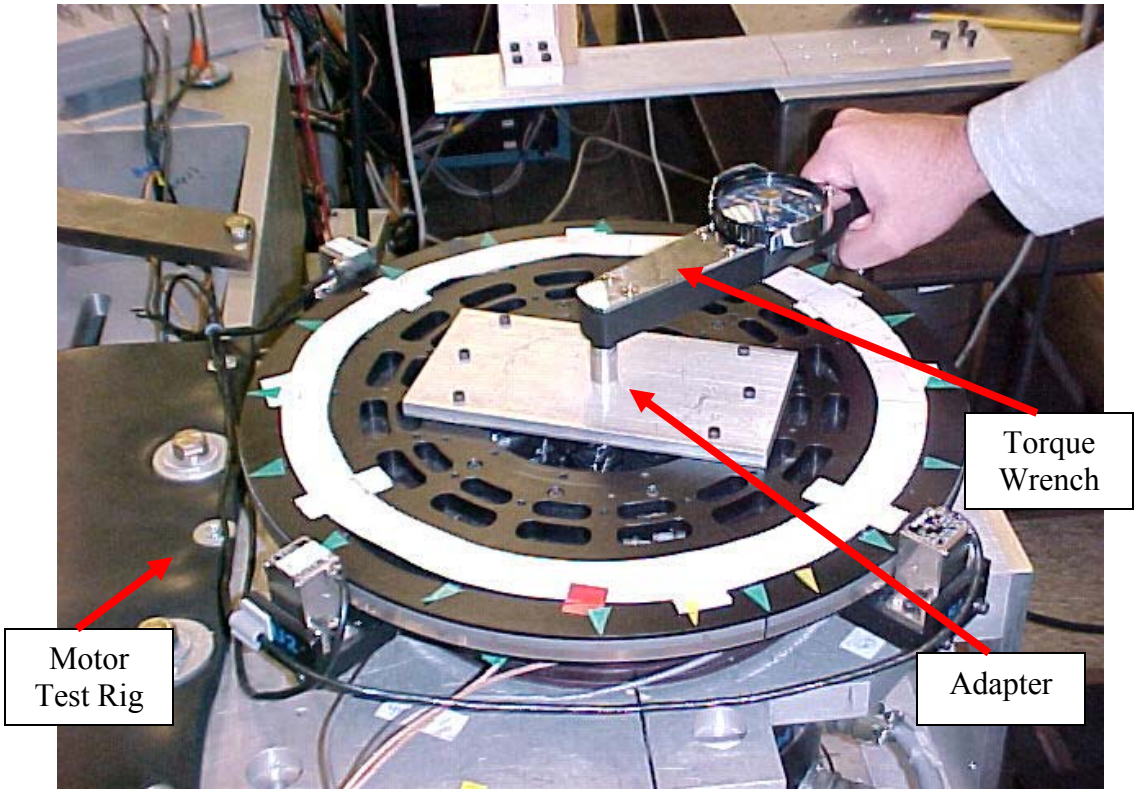


Figure 4.6: Closed Loop Torsional stiffness of non-fault and fault tolerant control

4.5 Torsional stiffness in Non-fault tolerant model and Fault tolerant model

The motor was levitated and a torque wrench was used for imparting a specific amount of torque to the motor shaft. The shaft rotates through a small angle and the change in angle of rotation was measured from the dspace layout for both the non-fault tolerant as well as the fault tolerant control. The torsional stiffness of both the models was computed for every fault configuration. The stiffness of the fault tolerant control remains nearly the same even after the phase faults are introduced. The stiffness of the non-fault tolerant control drops as the number of faults increases as shown in table 4.1 and figure 4.7.

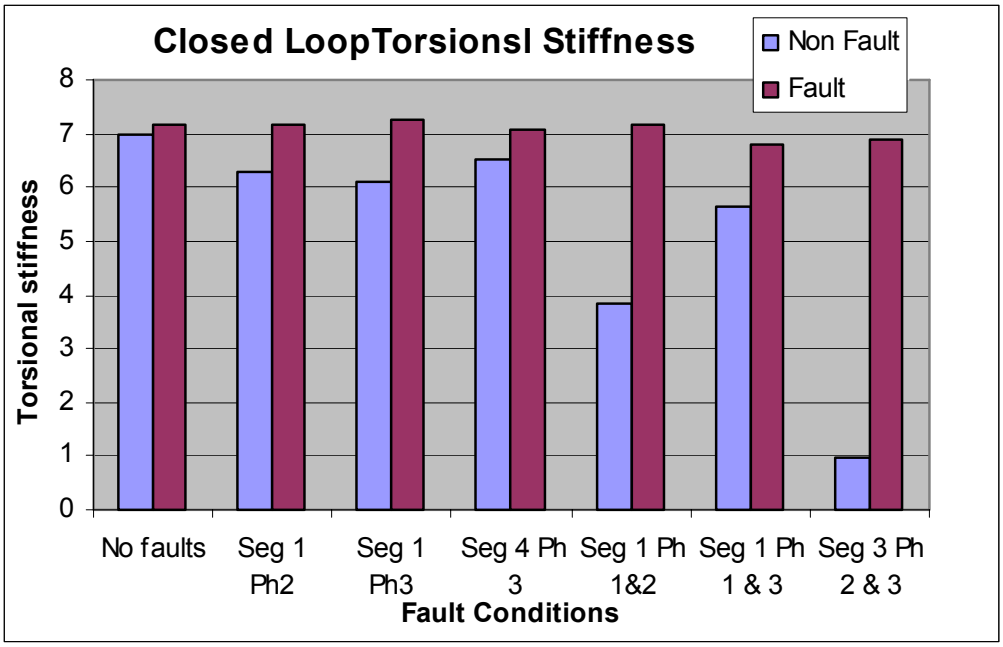


Figure 4.7: Closed loop torsional stiffness of non-fault and fault tolerant control

Faulted phases	Torque applied N-m	Models	Initial position of the shaft Θ_1	Final position of the shaft Θ_2	Rotor angle changed in μ radians	Rotor angle changed in deg	Closed loop torsion stiffness N-m/rad 10^2
No fault	7	NF	-0.1722	-0.1822	-0.0100	-0.5752	6.9700
	7	F	-0.1736	-0.1833	-0.0098	-0.5587	7.1758
F12	7	NF	-0.1779	-0.1890	-0.0112	-0.6387	6.2763
	7	F	-0.1768	-0.1866	-0.0097	-0.5579	7.1861
F13	7	NF	-0.1847	-0.1962	-0.0114	-0.6556	6.1151
	7	F	-0.1903	-0.2000	-0.0097	-0.5531	7.2486
F43	7	NF	-0.1859	-0.1966	-0.0107	-0.6126	6.5439
	7	F	-0.1797	-0.1895	-0.0099	-0.5655	7.0893
F11, F12	7	NF	-0.1672	-0.1805	-0.0133	-0.7622	5.2600
	7	F	-0.1841	-0.1922	-0.0081	-0.4615	8.6859
F11, F13	7	NF	-0.1627	-0.1808	-0.0181	-1.0373	3.8648
	7	F	-0.1877	-0.1975	-0.0097	-0.5580	7.1839
F32, F33	7	NF	-0.1902	-0.2027	-0.0125	-0.7135	5.6189
	7	F	-0.1865	-0.1969	-0.0103	-0.5909	6.7843
F11, F12,F13	7	NF	-0.1030	-0.1746	-0.0716	-4.0983	0.9782
	7	F	-0.1883	-0.1984	-0.0102	-0.5822	6.8864

Table 4.1: Closed loop torsional stiffness of non-fault and fault tolerant control

4.6 Closed loop stiffness for Non-fault tolerant and fault-tolerant control

The closed loop stiffness of the system can be measured in the non-fault tolerant and the fault-tolerant control by the following method. The self-bearing motor was levitated and a specified external force was applied in the x-direction on the shaft in Newton using weight-pulley system. The displacements of the shaft in x-direction are measured for different values of external force. It was ensured that the angular displacements were less, so that the permanent magnet distribution relative to the shaft stator segments remains nearly the same. Closed loop stiffness of the motor was determined by finding force per unit displacement for each weight. The experiment was repeated for different weights so that reliable data can be obtained. But collecting the displacement data remains a challenge due to the shaft moving back to initial position after being displaced by the weights. This happens due to the integral gain moving the shaft back to the initial position. It was therefore necessary that the displacement data be taken in immediately after loading. To avoid hysteresis, the weights are fully removed after obtaining every displacement data and then the new weights are added to obtaining new data. Closed loop stiffness of the fault tolerant and the non-fault tolerant models was compared. From the table it was found that the stiffness of the fault tolerant control was higher than that of the non-fault tolerant control model even with coil faults. This shows that the motor is more stable in case fault tolerant control.

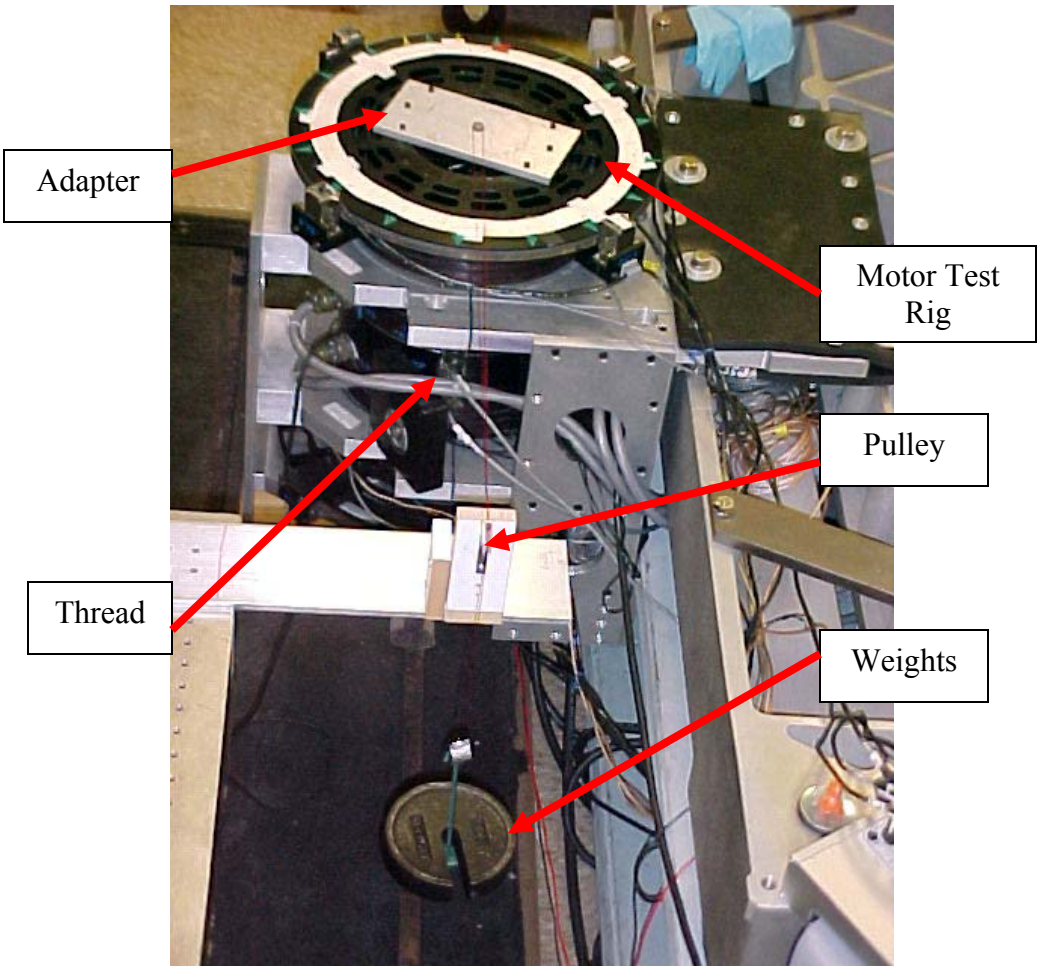


Figure 4.8: Experimental set-up for measuring closed loop stiffness

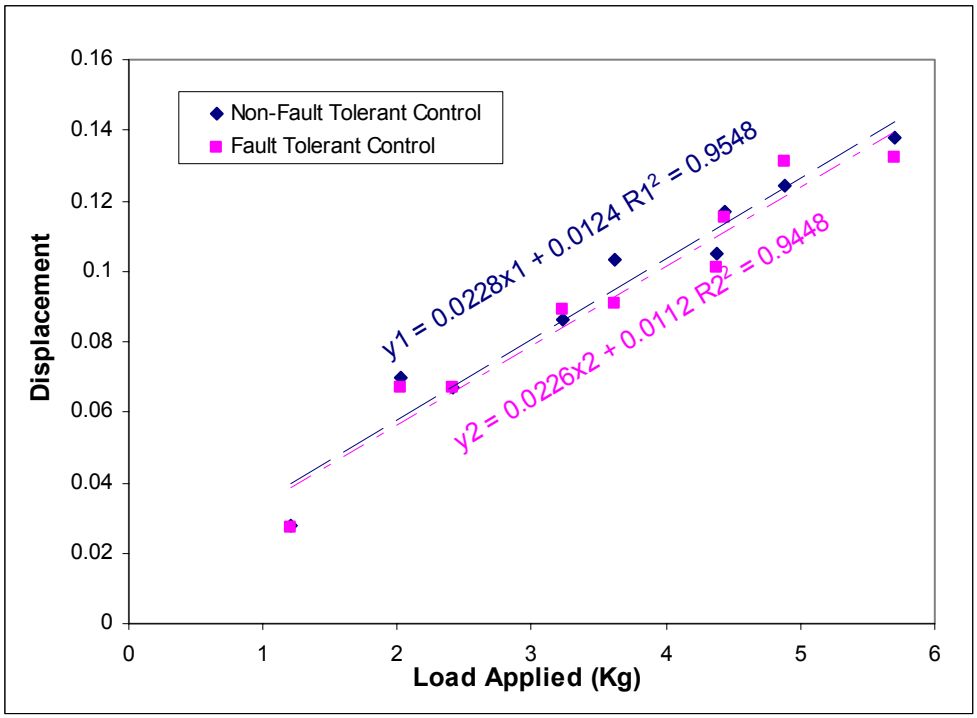


Figure 4.9: Closed loop torsional flexibility stiffness of non-fault and fault tolerant control in "No fault" configuration

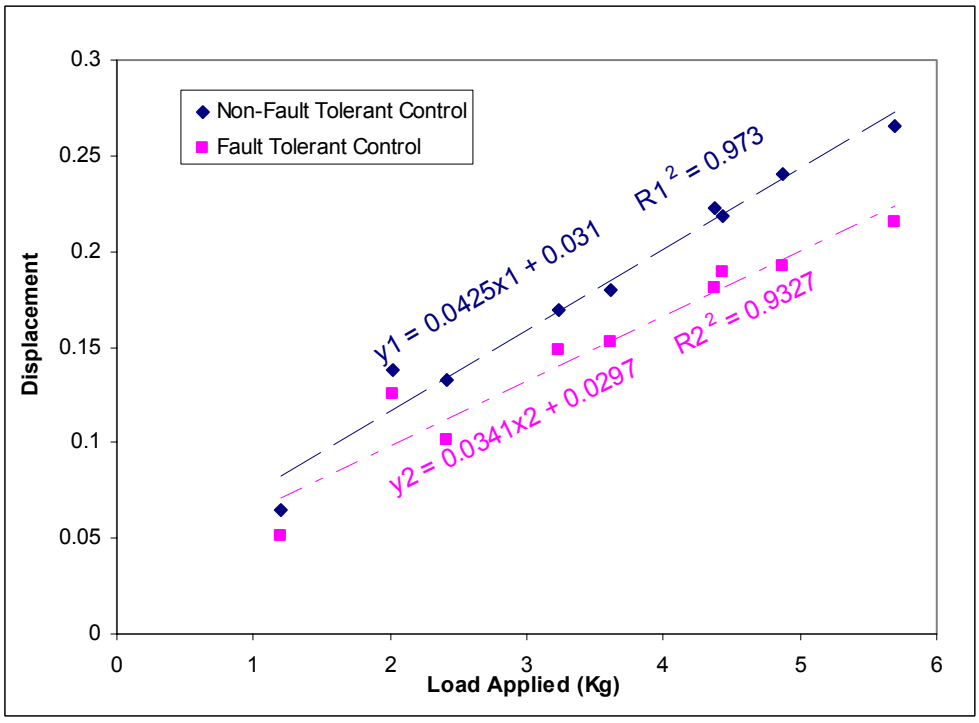


Figure 4.10: Closed loop torsional flexibility/stiffness of non-fault and fault tolerant control in segment 1 phase1 was faulted

4.7 Stability of the fault tolerant model in different fault configurations

The fault tolerant control was found to be stable under different fault configurations at different rotor angles that would cause the non-fault tolerant control to fail. Though better than the non-fault tolerant control, the fault tolerant control was not found to be as good as the simulation results suggested.

Fault Conf	Rotor Angles									
	5.4	6.3	7.2	8.1	9.0	9.9	10.8	11.7	12.6	13.5
111 111 111 111	Stable	Stable	Stable	Stable	Stable	Stable	Stable	Stable	Stable	Stable
011 111 111 111	Stable	Stable	Stable	Stable	Stable	Stable	Stable	Stable	Stable	Stable
001 111 111 111	Stable	Stable	Stable	Stable	Stable	Stable	Stable	Stable	Stable	Stable
000 111 111 111	Stable	Stable	Stable	Stable	Stable	Stable	Stable	Stable	Stable	Stable
000 011 111 111	Stable	Stable	Stable	Stable	Stable	Stable	Stable	Stable	Stable	Stable
000 101 111 111	Unstable	Stable	Stable	Stable	Stable	Stable	Stable	Stable	Stable	Stable
000 110 111 111	Unstable	Unstable	Unstable	Unstable	Unstable	Stable	Stable	Stable	Stable	Stable
000 010 111 111	Unstable	Unstable	Unstable	Unstable	Unstable	Unstable	Unstable	Unstable	Unstable	Unstable
000 001 111 111	Unstable	Unstable	Unstable	Unstable	Unstable	Unstable	Unstable	Unstable	Unstable	Unstable
000 100 111 111	Unstable	Unstable	Unstable	Unstable	Unstable	Unstable	Unstable	Unstable	Unstable	Unstable

Table 4.2: Stability of the motor under fault tolerant control

Chapter 5

Conclusions and future work

5.1 Conclusions

The self-bearing motor used in this research work is a 12 phase 4 segment motor that operates on the principles of radial bearing force and motoring torque. A pair of opposite segments produces the force in x and y directions and adjacent segments cause the motoring torque.

The self-bearing motor produces independent bearing force and motoring torque using the common coil windings and return flux path. It has the advantage of lesser heating, higher efficiency, lesser iron losses and smooth angular slewing. It has a high level of precision pointing and tracking accuracies [Ren 05].

This thesis used a model based adaptive control approach, which is the preferred method for software fault tolerance. The reference model inversion was computed instantaneously to obtain the desired control currents for a given rotor displacement and set of external forces and torque for achieving fault tolerance. The decoupling was accomplished by identifying that the force-current relationship, K_i, is invariant under a fault.

This work involved simulating, implementing and validating the decoupled and fault tolerant model in a 4 segment 12 phase self-bearing motor. The summary and conclusions of this thesis is as follows

1. In this work, many different models were simulated in simulink inclusive of the decoupled and fault tolerant algorithm and the conclusions were derived from the simulation. Force and torque produced are analyzed in every model for many common input control currents and rotor angles.
2. It was found from the simulations that the decoupled and fault tolerant control was better than the fault tolerant model based on the integral equations and the fault tolerant model based on the lumped parameter. The decoupled and fault tolerant model was the only model which could remove the crosscoupling in addition to providing fault tolerance.
3. The simulink model of the fault tolerant control was modified so that the model could be used for real-time control with dspace. The m-file s-function used for finding the pseudo inverse of the model in the simulation was replaced by c-mex file s-function so that the model was downloadable to the dsp boards of the dspace.
4. The pseudo-inverse involves intensive mathematical computations. The C-mex file S-function block of the pseudo-inverse makes the simulink model bigger in terms of computation time. But this does not change the operating bandwidth for the system, since the power amplifier puts a smaller limit on the bandwidth.

5. After downloading to the DSP boards of the dspace system, a graphical user interface was designed for the fault tolerant control to test the controller.
6. Experiments are designed so that fault tolerance can be validated.
7. A risk-free testing of the fault tolerant control was performed by looking at the phase currents of both the non-fault tolerant and fault tolerant control in the dspace layout without switching on the power amplifier. The phase currents of both the models for different control currents and rotor angle.
8. The shaft was levitated and the x - y positions of the shaft are looked at, for both fault tolerant and non-fault tolerant models, with phase faults. The shaft was found to move away from the center with the addition of phase faults to the non-fault tolerant model, indicating a decrease in stiffness. The shaft would not move from the center in the fault tolerant control, indicating that the stiffness remains the same. The lower the stiffness, the lower the stability of the actuator. Thus the stability of the motor under non-fault tolerant control decreases with the introduction of faults in the phases.
9. Both the non-fault tolerant control and fault tolerant control are subjected to a sine sweep test. Faults are introduced while the shaft was executing a sine sweep. It was found that the performance of the non-fault tolerant control decreased with faults. A stable bearing force and motoring torque was accomplished even under coil faults.

10. The penalty paid for the fault tolerance was an increased power usage and operating temperature. The power consumed was found to increase with the number of faults in the fault tolerant control.
11. The torsional stiffness of the non-fault tolerant control drops as the number of faults increases, but it remains the same for the fault tolerant control. Thus the stability of the non-fault tolerant control drops drastically with faults that of the fault tolerant control. However, it was noted that the stiffness of the fault tolerant control also dropped but dropped slowly with faults indicating a steady degradation in performance of the motor with an increasing number of faults.
12. A load test with a weight-pulley system was performed with the motor under the fault tolerant and the non-fault tolerant control. It was found that the closed loop stiffness of the motor was found to be higher for the motor with the fault tolerant control. The test was performed with the addition of faults, and it was found that the stiffness of the motor was higher for the fault tolerant control. It was noted that the stiffness of the motor falls even with fault tolerant control, thus indicating some degradation in the performance. Higher stiffness of the motor in the fault tolerant control indicated greater stability in the motor.
13. The fault tolerant control was found to be stable under different fault configurations at different rotor angles at which the non-fault tolerant control would fail. Though better than the non-fault tolerant control, the

fault tolerant control was not found to be as good as the simulation results suggested.

14. Since the motor was higher in stability and less crosscoupled with the fault tolerant control, it can be used at higher speeds.
15. The saturation block in the simulink file, puts a limit on the phase currents output to the D/A. When more faults are introduced, the magnitude of current in faultless phases was more than the saturation limit in the saturation block. Hence the output currents in those phases will be equal to the saturation limit. The forces produced will also be different from required, hence resulting in deterioration in performance of the fault tolerant control.

5.2 Future work

1. Exploring the possibility of using a look-up table in the simulink model instead of the C-mex file S-function, so that intensive computations involved in the fault tolerant model be reduced.
2. Detect faults using fault detection circuitry and feeding into dspace so that it can be used in space applications.
3. Incorporate phase fault tolerance for non-centered rotors.
4. Identify other types of faults like cracked rotor, temperature excess, amplifier faults, etc. Investigate the different methods of correcting the different faults, identify the best method and implement.
5. Investigate and implement fault tolerance for short circuit faults.

Bibliography

[Beat 00] Beat Aeschlimann,, Matthias Kummerle, Jurjen Zoethout and Hannes Bleuler, "Model based decoupling control of a disc rotor on active magnetic bearing," *Proc. 7th Int. Symp. Magn. Bearings,* ETH, Zurich, Switzerland, pp. 245-249, August, 2000.

[Bischel 91] Bischel, J., "The bearingless electrical machine," in *Proc. Int. Symp Magn. Suspension Technol.'91,* NASA Publication 3152, Langley Research Center, Hampton, VA, pp. 561-573, August, 1991.

[Bleuler 91] Bleuler, H., "A survey of magnetic levitation and magnetic bearing types" JSME international journal, Series III, Vol. 35,No.3,1992.

[Chiba 91] Chiba, A., Rahman, M.A., and Fukao, T., "Radial forces in bearingless reluctance motor," *IEEE Trans. Magnetics,* vol. 27, p. 786, Mar. 1991.

[Chin 03:1] Hooi-mei Chin and Stephens, L.S., "Closed loop performance of a slotless Lorentz self-bearing motor" Proc. of IGTI, ASME Turbo Expo Expo,June, 2003.

[Chin 03:2] Hooi-mei Chin, "Controller design issues and performance of the lorentz self-bearing motor", Masters thesis, University of Kentucky,December,2003

[Dominick 99] Dominick Montie, "Self-sensing in fault tolerant magnetic bearing" in Trans. Of the ASME,99-GT-178,Jun. 1999.

[Hertel 00] Lars Hertel, Wilfried Hofmann, "Basic approach for the design of bearingless motors," *Proc. 7th Int. Symp. Magn. Bearings,* ETH, Zurich, Switzerland, pp. 341-346, August, 2000.

[Kim 00] Kim, D.G., and Stephens, L.S., "Fault Tolerance of a Lorentz-Type Slotless, Self Bearing Motor According to the Coiling Schemes", *Proc. 7th Int. Symp. Magn. Bearings,* ETH, Zurich, Switzerland, pp. 219-224, August, 2000.

[Maslen 95] Maslen,E.H. and Meeker, D.C."Fault tolerance of magnetic bearing by generalized bias current linearization," IEEE transactions of magnetics, Vol. 31, No.3, May 1995.

[Meeker 96] Meeker, D.C."Optimal solutions to inverse problems in quadratic magnetic actuators," Doctoral Dissertation, University of Virginia, May 1996.

[Na 99] Na, U.J. and Palazzola, A.B., "Optimized realization of fault tolerant hetroppolar magnetic bearing for active vibration control" Proceedings of the ASME Design Engineering Technical Conferences, September 1999.

[Na 99] Na, U.J. and Palazzola, A.B., "Fault tolerant, Decoupling control of magnetic bearings including material path reluctances" *Proc. 7th Int. Symp. Magn. Bearings,* ETH, Zurich, Switzerland, pp. 213-218, August, 2000.

[Okada 96] Okada, Y., Miyamoto, S., and Ohishi, T., "Levitation and torque control of internal permanent magnet type bearingless motor," *IEEE Trans. Control Syst. Techn.,* vol. 4, no. 5, pp. 565-571, 1996.

[Okada 00] Okada, Y., Konishi, H., Kannebako, H., and Lee, C.W., "Lorentz Force Type Self-Bearing Motor," *Proc. 7th Int. Symp. Magn. Bearings,* ETH, Zurich, Switzerland, pp. 353-358, August, 2000.

[Salazar 00] Salazar, A.O., Chiba, A., and Fukao, T., "A Review of Developments in Bearingless Motors," *Proc. 7th Int. Symp. Magn. Bearings,* ETH, Zurich, Switzerland, pp. 335-339, August, 2000.

[Ren 05] Zhaohui Ren, "Model, Control and Performance of a six degree-of-freedom precision pointing and tracking systems" , PhD Thesis, University of Kentucky, May, 2005

[Schoeb 94] Schoeb, R. and Bischel, J., "Vector control of bearingless motor," in *Proc. 4th Int. Symp. Magn. Bearings,* ETH Zurich, Switzerland, pp. 327-332, 1994.

[Steele 00] Steele, B.A. and Stephens, L.S., "A Test Rig for Measuring Force and Torque Production in A Lorentz,

Slotless Self Bearing Motor," *Proc. 7th Int. Symp. Magn. Bearings,* ETH, Zurich, Switzerland, pp. 407-412, August, 2000.

[Stephens 00] Stephens, L.S. and Kim, D.G., "Analysis and Simulation of a Lorentz-type, Slotless Self-Bearing Motor", Proceedings of the 1st IFAC Conference on Mechatronics, Darmstadt, Germany, September, 2000.

[Stephens 02:1] Stephens, L.S. and Kim, D.G., "Force and torque characteristics for a slotless Lorentz Self-Bearing Servomotor," IEEE transactions of magnetics, Vol. 38, No.4, July 2002.

[Stephens 02:2] Stephens, L.S. and Hooi-Mei Chin., "Robust stability of the lorentz-type self-bearing servomotor," JSME international journal, Series C, Vol. 46,No.2,2003.

[Stephens 04] Stephens, L.S. and Anand Ranganathan "Fault Tolerance of a Lorentz Self Bearing Motor Considering open coil faults," *Proc. 9th Int. Symp. Magn. Bearings, Lexington*, USA, August, 2004.

Appendix A

C-mex-file function for calculation of pseudo inverse

```
/* File:inv_A.c is C-mex S-function used in conjunction with fault_algo.mdl
 *  Abstract:
 *      c code for pseudo-inverse of A */
#define S_FUNCTION_NAME inv_A
#define S_FUNCTION_LEVEL 2
#include "simstruc.h"
/*====================*
 * S-function methods *
 *====================*/
/* Function: mdlInitializeSizes ===============================================
 * Abstract:
 *    The sizes information is used by Simulink to determine the S-function
 *    block's characteristics (number of inputs, outputs, states, etc.).
 */
static void mdlInitializeSizes(SimStruct *S)
{
    ssSetNumSFcnParams(S, 0);  /* Number of expected parameters */
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
        return; /* Parameter mismatch will be reported by Simulink */
    }

    ssSetNumContStates(S, 0);
    ssSetNumDiscStates(S, 0);

    if (!ssSetNumInputPorts(S, 19)) return;/*7 in two out*/
    ssSetInputPortWidth(S, 0, 3); /*format: s,port,width */
    ssSetInputPortWidth(S, 1, 1); /*format: s,port,width */
    ssSetInputPortWidth(S, 2, 3); /*format: s,port,width */
    ssSetInputPortWidth(S, 3, 12); /*format: s,port,width */
    ssSetInputPortDirectFeedThrough(S, 0, 3);
    ssSetInputPortDirectFeedThrough(S, 1, 1);
    ssSetInputPortDirectFeedThrough(S, 2, 3);
    ssSetInputPortDirectFeedThrough(S, 3, 12);
    if (!ssSetNumOutputPorts(S, 4)) return;
    ssSetOutputPortWidth(S, 0, 1); /*port and width*/
    ssSetOutputPortWidth(S, 1, 1);
    ssSetOutputPortWidth(S, 2, 1);
    ssSetOutputPortWidth(S, 3, 1);
    ssSetNumSampleTimes(S, 1);
    ssSetNumRWork(S, 0);
    ssSetNumIWork(S, 0);
    ssSetNumPWork(S, 0);
    ssSetNumModes(S, 0);
    ssSetNumNonsampledZCs(S, 0);
    /* Take care when specifying exception free code - see sfuntmpl_doc.c */
    ssSetOptions(S, SS_OPTION_EXCEPTION_FREE_CODE);
}
/* Function: mdlInitializeSampleTimes =========================================
 * Abstract:
 *    Specifiy that we inherit our sample time from the driving block.
 */
```

```
static void mdlInitializeSampleTimes(SimStruct *S)
{
   ssSetSampleTime(S, 0, INHERITED_SAMPLE_TIME);
   ssSetOffsetTime(S, 0, 0.0);
}
#define MDL_INITIALIZE_CONDITIONS
/* Function: mdlInitializeConditions ========================================
 * Abstract:
 *    Initialize both discrete states to one.
 */
static void mdlInitializeConditions(SimStruct *S)
{
}
/* Function: mdlOutputs =======================================================
 * Abstract:
 *      y = Cx
 */
static void mdlOutputs(SimStruct *S, int_T tid)
{
   real_T            *S1    = ssGetOutputPortRealSignal(S,0);
   real_T            *S2  = ssGetOutputPortRealSignal(S,1);
   real_T            *S3  = ssGetOutputPortRealSignal(S,2);
   InputRealPtrsType uPtrFx    = ssGetInputPortRealSignalPtrs(S,0,1);
   InputRealPtrsType uPtrFy    = ssGetInputPortRealSignalPtrs(S,0,2);
   InputRealPtrsType uPtrT    = ssGetInputPortRealSignalPtrs(S,0,3);
   InputRealPtrsType uPtrtheta    = ssGetInputPortRealSignalPtrs(S,1,1);
   InputRealPtrsType uPtrY1    = ssGetInputPortRealSignalPtrs(S,2,1);
   InputRealPtrsType uPtrY2    = ssGetInputPortRealSignalPtrs(S,2,2);
   InputRealPtrsType uPtrY3    = ssGetInputPortRealSignalPtrs(S,2,3);
   InputRealPtrsType uPtrF1    = ssGetInputPortRealSignalPtrs(S,3,1);
   InputRealPtrsType uPtrF2    = ssGetInputPortRealSignalPtrs(S,3,2);
   InputRealPtrsType uPtrF3    = ssGetInputPortRealSignalPtrs(S,3,3);
   InputRealPtrsType uPtrF4    = ssGetInputPortRealSignalPtrs(S,3,4);
   InputRealPtrsType uPtrF5    = ssGetInputPortRealSignalPtrs(S,3,5);
   InputRealPtrsType uPtrF6    = ssGetInputPortRealSignalPtrs(S,3,6);
   InputRealPtrsType uPtrF7    = ssGetInputPortRealSignalPtrs(S,3,7);
   InputRealPtrsType uPtrF8    = ssGetInputPortRealSignalPtrs(S,3,8);
   InputRealPtrsType uPtrF9    = ssGetInputPortRealSignalPtrs(S,3,9);
   InputRealPtrsType uPtrF10    = ssGetInputPortRealSignalPtrs(S,3,10);
   InputRealPtrsType uPtrF11    = ssGetInputPortRealSignalPtrs(S,3,11);
   InputRealPtrsType uPtrF12    = ssGetInputPortRealSignalPtrs(S,3,12);
   /*uPtrs[element])  * Pointer to Input Port0 */
  real_T U1,U2,U3,U4,U5,U6;
  UNUSED_ARG(tid); /* not used in single tasking mode */
     Fx=*uPtrFx[0];
     Fy=*uPtrFy[0];
     T=*uPtrT[0];
     theta=*uPtrtheta[0];
M=8;Nw=85;R=50.8e-3;L=25.4e-3;
R=50.8e-3;
 BMKP=0.78;
for k=1:48
phi(k)=(k-1)*(2*pi/48)+(pi/4)+(pi/48); %ORIENTATION ALONG ANY STATOR STATION RELATIVE TO X COORDINATE
AXIS
 BMK(k)=BMKP*sin(M*(theta-phi(k)));
ph(1,k)=-Nw*L*BMK(k)*sin(phi(k));
```

```
    ph(2,k)=Nw*L*BMK(k)*cos(phi(k));
     ph(3,k)=Nw*L*R*BMK(k);
    end
    phi1=*uPtrY1[0];
    phi2=*uPtrY2[0];
    phi3=*uPtrY3[0];
    Fa[1,1]=*uPtrF1[0];
    Fa[2,2]=*uPtrF2[0];
    Fa[3,3]=*uPtrF3[0];
    Fa[4,4]=*uPtrF4[0];
    Fa[5,5]=*uPtrF5[0];
    Fa[6,6]=*uPtrF6[0];
    Fa[7,7]=*uPtrF7[0];
    Fa[8,8]=*uPtrF8[0];
    Fa[9,9]=*uPtrF9[0];
    Fa[10,10]=*uPtrF10[0];
    Fa[11,11]=*uPtrF11[0];
    Fa[12,12]=*uPtrF12[0];
    Y=zeros(12,4);
    Y[1,1]=phi1;Y[2,1]=phi2;Y[3,1]=phi3;
    Y[4,2]=phi1;Y[5,2]=phi2;Y[6,2]=phi3;
    Y[7,3]=phi1;Y[8,3]=phi2;Y[9,3]=phi3;
    Y[10,4]=phi1;Y[11,4]=phi2;Y[12,4]=phi3;
    beta=[-eye(3,3);eye(3,3);-eye(3,3);eye(3,3)];
    z3=zeros(12,3);
    lambda=[beta,z3,z3,z3;
                z3,beta,z3,z3;
                z3,z3,beta,z3;
                z3,z3,z3,beta];
    A=ph*lambda*Fa*Y;                       %A with no fault
    sys=pinv(A)*Fc';      %Matrix pseudo-inverse of A_Fault

    S1[0]=sys[0];S2[0]=sys[1];S3[0]=sys[2];
    }
    #define MDL_UPDATE
    /* Function: mdlUpdate ======================================================
     * Abstract:
     *      xdot = Ax + Bu
     */
    static void mdlUpdate(SimStruct *S, int_T tid)
    {
      UNUSED_ARG(S); /* unused input argument */
    }
    /* Function: mdlTerminate =====================================================
     * Abstract:
     *    No termination needed, but we are required to have this routine.
     */
    static void mdlTerminate(SimStruct *S)
    {
       UNUSED_ARG(S); /* unused input argument */
    }
    #ifdef  matlab_MEX_FILE    /* Is this file being compiled as a MEX-file? */
    #include "simulink.c"      /* MEX-file interface mechanism */
    #else
    #include "cg_sfun.h"       /* Code generation registration function */
#endif
```

Appendix B

Medit-file function for calculation of force and torque using integral equations

```
%input of x,y,zeta,ix,iy,izeta values
clear

clc

 x=input(' Rotor motion in x = ')

 y=input(' Rotor motion in Y = ')

 zeta=input(' Rotor motion in zeta = ')

 gamma=input(' Phase Angle of the current wrt permenent magnet flux = ')

 ix=input(' control current in x direction = ')

 iy=input(' control current in y direction = ')

 izeta=input(' control current in zeta direction = ')

 M=8;                          %NUMBER OF POLE PAIRS

 Nseg=4;                       %NUMBER OF SEGMENTS

 Ns=12;                        %NUMBER OF WINDING STATIONS PER SEGMENT

 Nw=85;                        %NUMBER OF WIRES PER WINDING STATION

 tm=7.75e-3;                   %RADIAL THICKNESS OF PERMANENT MAGNETS IN m

 tc=3.87e-3;                   %RADIAL THICKNESS OF COIL WINDINGS IN m

 go=0.762e-3;                  %NOMINAL RADIAL AIR GAP

 R=50.8e-3;                    %ROTOR OUTER RADIUS

 L=25.4e-3;                    %MOTOR LENGTH

 %BMKP=0.77;                   %PM FLUX DENSITY IN Tesla

 mur=1.1;                      %RECOIL PERMEABILITY

 muo=4*3.143*(10e-7);          %PERMEABILITY IN FREE SPACE

 Br=1.08;                      %REMNANCE FLUX DENSITY IN Tesla

 Kml=1.82;                     %MAGNETIC LEAKAGE FACTOR

 Cphi=0.8;                     %FLUX CONCENTRATION FACTOR

 %iAmp=12.0;                   %PEAK INSTANTANEOUS CURRENT PER PHASE IN
Amps
```

```
irms=3.0;                          %MAXIMUM CONTINOUS CURRENT PER PHASE IN Amps
for k=1:480
   theta(k)=(k-1)*2*pi/480;%GLOBAL SEGMENT ANGLE
end
%ORIENTATION ALONG ANY STATOR STATION RELATIVE TO X COORDINATE AXIS
for k=1:480
  phi(k)=theta(k)+pi/4;
end

%CURRENT IN EACH SEGMENT
 i(1)=izeta-ix;
 i(2)=izeta-iy;
 i(3)=izeta+ix;
 i(4)=izeta+iy;
%AMPLITUDE OF CURRENT
for k=1:4
   iAmp(k)=(2*sqrt(2)/3)*Nw*i(k);
   sprintf('AMPLITUDE OF CURRENT IS %5.5f',iAmp(k))
end
for k=1:480
     if  (k>=1) & (k<=120)
         q=1;
  elseif (k>=121) & (k<=240)
         q=2;
  elseif (k>=241) & (k<=360)
         q=3;
  elseif (k>=361) & (k<=480)
         q=4;
end
```

```
%PM FLUX AMPLITUDE
BWKP(k)=(muo*iAmp(q))/2*(tm+tc+go-x*cos(phi(k))-y*sin(phi(k)));
%WINDING FLUX AMPLITUDE
BMKP(k)=0.77;        %(sqrt(2)*Br*tm)/(tm+mur*Cphi*Kml*(go+tc-x*cos(phi(k))-y*sin(phi(k))));
%PM FLUX
BMK(k)=BMKP(k)*sin(M*(zeta-(theta(k)-(q-1)*pi/2)));
%WINDING CURRENT DISTRIBUTION
IK(k)=iAmp(q)*sin(M*(zeta-(theta(k)-(q-1)*pi/2)-gamma));
sprintf('WINDING CURRENT IN %d IS %5.5f',k,IK(k));
%WINDING FLUX
BWK(k)=BWKP(k)*sin(M*(zeta-(theta(k)-(q-1)*pi/2)-gamma+pi/(2*M)));
end
%THE NET FORCE AND TORQUE ON THE SELF BEARING ROTOR
 FXL=0; FYL=0; Tzeta=0;
 FXM=0; FYM=0;
 FXW=0; FYW=0;
for k=1:480
%LORENTZ TYPE FORCES FOR BEARING FORCE AND TORQUE CONTROL
FXL=FXL+BMK(k)*IK(k)*cos(phi(k));
FYL=FYL+BMK(k)*IK(k)*sin(phi(k));
Tzeta=Tzeta+BMK(k)*IK(k);
%MAXWELL TYPE FORCES ON THE ROTOR DUE TO THE PM FLUX
FXM=FXM+BMK(k)*BMK(k)*cos(phi(k));
FYM=FYM+BMK(k)*BMK(k)*sin(phi(k));
%MAXWELL TYPE FORCES ON THE ROTOR DUE TO THE WINDING FLUX
FXW=FXW+BWK(k)*BWK(k)*cos(phi(k));
FYW=FYW+BWK(k)*BWK(k)*sin(phi(k));
end
```

```
%LORENTZ TYPE FORCES FOR BEARING FORCE AND TORQUE CONTROL
FXL=M*L*FXL;
FYL=M*L*FYL;
Tzeta=M*R*L*Tzeta;
%MAXWELL TYPE FORCES ON THE ROTOR DUE TO THE PM FLUX
FXM=(R*L/2*muo)*FXM;
FYM=(R*L/2*muo)*FYM;
% MAXWELL TYPE FORCES ON THE ROTOR DUE TO THE WINDING FLUX
FXW=(R*L/2*muo)*FXW;
FYW=(R*L/2*muo)*FYW;
%NET FORCE AND TORQUE ON THE SELF BEARING ROTOR
FX=FXL+FXM+FXW
FY=FYL+FYM+FYW
Tzeta=Tzeta
```

Appendix C

Medit-file function for phase distribution matrix

```
function [lambda_output] = lambda(lambda_input)
beta=[-eye(3,3);eye(3,3);-eye(3,3);eye(3,3)];
z3=zeros(12,3);
lambda_output=[beta,z3,z3,z3;
               z3,beta,z3,z3;
                z3,z3,beta,z3;
                z3,z3,z3,beta]*lambda_input;
```

Medit-file function for commutation matrix

```
function [Y_output] = Y(Y_input)
M=8;zeta=Y_input(5);gamma=0;
phi1=cos(M*(zeta-gamma)+pi/3);
phi2=cos(M*(zeta-gamma));
phi3=cos(M*(zeta-gamma)-pi/3);
Y=zeros(12,4)
Y(1,1)=phi1;Y(2,1)=phi2;Y(3,1)=phi3;
Y(4,2)=phi1;Y(5,2)=phi2;Y(6,2)=phi3;
Y(7,3)=phi1;Y(8,3)=phi2;Y(9,3)=phi3;
Y(10,4)=phi1;Y(11,4)=phi2;Y(12,4)=phi3;
Y_output=Y*Y_input(1:4);
```

Medit-file function for segment current-control current mapping

```
function [T3_output] = T3(T3_input)
T3_output=[-1 0 1;0 -1 1;1 0 1;0 1 1]*T3_input;
```

Medit-file function for faulted phase currents

```
function [F_output] = F(F_input)
F=eye(12,12);
F(1,1)=0;
F(2,2)=0;
F(3,3)=0;
F(4,4)=0;
F(5,5)=0;
F(6,6)=0;
F(7,7)=0;
%  F(8,8)=0;
F(9,9)=0;
F(10,10)=0;
F(11,11)=0;
%  F(12,12)=0;
F_output=F*F_input;
```

Appendix D

C-mex file function for permanent magnet flux distribution matrix

```
/*  File    : ph_sfunc.c
 *  Abstract:
 *      c code for ph_sfunc     */

#define S_FUNCTION_NAME ph_sfunc
#define S_FUNCTION_LEVEL 2
#include "simstruc.h"
#include "math.h"

/*====================*
 * S-function methods *
 *====================*/

/* Function: mdlInitializeSizes ===============================================
 * Abstract:
 *    The sizes information is used by Simulink to determine the S-function
 *    block's characteristics (number of inputs, outputs, states, etc.).
 */
static void mdlInitializeSizes(SimStruct *S)
{
    ssSetNumSFcnParams(S, 0);  /* Number of expected parameters */
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
        return; /* Parameter mismatch will be reported by Simulink */
    }
    ssSetNumContStates(S, 0);
    ssSetNumDiscStates(S, 0);

   if (!ssSetNumInputPorts(S, 49)) return;/*7 in two out*/
    ssSetInputPortWidth(S, 0, 1); /*format: s,port,width */
    ssSetInputPortWidth(S, 0, 2); /*format: s,port,width */
    ssSetInputPortWidth(S, 0, 3); /*format: s,port,width */
    ssSetInputPortWidth(S, 0, 4); /*format: s,port,width */
    ssSetInputPortWidth(S, 0, 5); /*format: s,port,width */
    ssSetInputPortWidth(S, 0, 6); /*format: s,port,width */
    ssSetInputPortWidth(S, 0, 7); /*format: s,port,width */
    ssSetInputPortWidth(S, 0, 8); /*format: s,port,width */
    ssSetInputPortWidth(S, 0, 9); /*format: s,port,width */
    ssSetInputPortWidth(S, 0, 10); /*format: s,port,width */
    ssSetInputPortWidth(S, 0, 11); /*format: s,port,width */
    ssSetInputPortWidth(S, 0, 12); /*format: s,port,width */
    ssSetInputPortWidth(S, 0, 13); /*format: s,port,width */
    ssSetInputPortWidth(S, 0, 14); /*format: s,port,width */
    ssSetInputPortWidth(S, 0, 15); /*format: s,port,width */
    ssSetInputPortWidth(S, 0, 16); /*format: s,port,width */
    ssSetInputPortWidth(S, 0, 17); /*format: s,port,width */
    ssSetInputPortWidth(S, 0, 18); /*format: s,port,width */
    ssSetInputPortWidth(S, 0, 19); /*format: s,port,width */
    ssSetInputPortWidth(S, 0, 20); /*format: s,port,width */
    ssSetInputPortWidth(S, 0, 21); /*format: s,port,width */
    ssSetInputPortWidth(S, 0, 22); /*format: s,port,width */
    ssSetInputPortWidth(S, 0, 23); /*format: s,port,width */
```

```
    ssSetInputPortWidth(S, 0, 24); /*format: s,port,width */
    ssSetInputPortWidth(S, 0, 25); /*format: s,port,width */
    ssSetInputPortWidth(S, 0, 26); /*format: s,port,width */
    ssSetInputPortWidth(S, 0, 27); /*format: s,port,width */
    ssSetInputPortWidth(S, 0, 28); /*format: s,port,width */
    ssSetInputPortWidth(S, 0, 29); /*format: s,port,width */
    ssSetInputPortWidth(S, 0, 30); /*format: s,port,width */
    ssSetInputPortWidth(S, 0, 31); /*format: s,port,width */
    ssSetInputPortWidth(S, 0, 32); /*format: s,port,width */
    ssSetInputPortWidth(S, 0, 33); /*format: s,port,width */
    ssSetInputPortWidth(S, 0, 34); /*format: s,port,width */
    ssSetInputPortWidth(S, 0, 35); /*format: s,port,width */
    ssSetInputPortWidth(S, 0, 36); /*format: s,port,width */
    ssSetInputPortWidth(S, 0, 37); /*format: s,port,width */
    ssSetInputPortWidth(S, 0, 38); /*format: s,port,width */
    ssSetInputPortWidth(S, 0, 39); /*format: s,port,width */
    ssSetInputPortWidth(S, 0, 40); /*format: s,port,width */
    ssSetInputPortWidth(S, 0, 41); /*format: s,port,width */
    ssSetInputPortWidth(S, 0, 42); /*format: s,port,width */
    ssSetInputPortWidth(S, 0, 43); /*format: s,port,width */
    ssSetInputPortWidth(S, 0, 44); /*format: s,port,width */
    ssSetInputPortWidth(S, 0, 45); /*format: s,port,width */
    ssSetInputPortWidth(S, 0, 46); /*format: s,port,width */
    ssSetInputPortWidth(S, 0, 47); /*format: s,port,width */
    ssSetInputPortWidth(S, 0, 48); /*format: s,port,width */
    ssSetInputPortWidth(S, 0, 49); /*format: s,port,width */

    ssSetInputPortDirectFeedThrough(S, 0, 49);

    if (!ssSetNumOutputPorts(S, 3)) return;
    ssSetOutputPortWidth(S, 0, 1); /*port and width*/
    ssSetOutputPortWidth(S, 0, 2);
    ssSetOutputPortWidth(S, 0, 3);

    ssSetNumSampleTimes(S, 1);  /*unsure if this is right*/
    ssSetNumRWork(S, 0);
    ssSetNumIWork(S, 0);
    ssSetNumPWork(S, 0);
    ssSetNumModes(S, 0);
    ssSetNumNonsampledZCs(S, 0);

    /* Take care when specifying exception free code - see sfuntmpl_doc.c */
    ssSetOptions(S, SS_OPTION_EXCEPTION_FREE_CODE);
}

/* Function: mdlInitializeSampleTimes =========================================
 * Abstract:
 *    Specifiy that we inherit our sample time from the driving block.
 */
static void mdlInitializeSampleTimes(SimStruct *S)
{
    ssSetSampleTime(S, 0, INHERITED_SAMPLE_TIME);
    ssSetOffsetTime(S, 0, 0.0);
}

#define MDL_INITIALIZE_CONDITIONS
```

```
/* Function: mdlInitializeConditions ========================================
 * Abstract:
 *    Initialize both discrete states to one.
 */
static void mdlInitializeConditions(SimStruct *S)
{}

/* Function: mdlOutputs =====================================================
 * Abstract:
 *      y = Cx
 */
static void mdlOutputs(SimStruct *S, int_T tid)
{
   real_T            *S1    = ssGetOutputPortRealSignal(S,0);
   real_T            *S2  = ssGetOutputPortRealSignal(S,0,1);
   real_T            *S3  = ssGetOutputPortRealSignal(S,0,3);

   InputRealPtrsType uph1    = ssGetInputPortRealSignalPtrs(S,0,1);
   InputRealPtrsType uph2    = ssGetInputPortRealSignalPtrs(S,0,2);
   InputRealPtrsType uph3  = ssGetInputPortRealSignalPtrs(S,0,3);
   InputRealPtrsType uph4  = ssSetInputPortWidth(S, 0, 4);
   InputRealPtrsType uph5  = ssSetInputPortWidth(S, 0, 5);
   InputRealPtrsType uph6  = ssSetInputPortWidth(S, 0, 6);
   InputRealPtrsType uph7  = ssSetInputPortWidth(S, 0, 7);
   InputRealPtrsType uph8  = ssSetInputPortWidth(S, 0, 8);
   InputRealPtrsType uph9  = ssSetInputPortWidth(S, 0, 9);
   InputRealPtrsType uph10  = ssSetInputPortWidth(S, 0, 10);
   InputRealPtrsType uph11  = ssSetInputPortWidth(S, 0, 11);
   InputRealPtrsType uph12  = ssSetInputPortWidth(S, 0, 12);
   InputRealPtrsType uph13  = ssSetInputPortWidth(S, 0, 13);
   InputRealPtrsType uph14  = ssSetInputPortWidth(S, 0, 14);
   InputRealPtrsType uph15  = ssSetInputPortWidth(S, 0, 15);
   InputRealPtrsType uph16 = ssSetInputPortWidth(S, 0, 16);
   InputRealPtrsType uph17  = ssSetInputPortWidth(S, 0, 17);
   InputRealPtrsType uph18  = ssSetInputPortWidth(S, 0, 18);
   InputRealPtrsType uph19  = ssSetInputPortWidth(S, 0, 19);
   InputRealPtrsType uph20  = ssSetInputPortWidth(S, 0, 20);
   InputRealPtrsType uph21  = ssSetInputPortWidth(S, 0, 21);
   InputRealPtrsType uph22  = ssSetInputPortWidth(S, 0, 22);
   InputRealPtrsType uph23  = ssSetInputPortWidth(S, 0, 23);
   InputRealPtrsType uph24 = ssSetInputPortWidth(S, 0, 24);
   InputRealPtrsType uph25  = ssSetInputPortWidth(S, 0, 25);
   InputRealPtrsType uph26  = ssSetInputPortWidth(S, 0, 26);
   InputRealPtrsType uph27  = ssSetInputPortWidth(S, 0, 27);
   InputRealPtrsType uph28  = ssSetInputPortWidth(S, 0, 28);
   InputRealPtrsType uph29  = ssSetInputPortWidth(S, 0, 29);
   InputRealPtrsType uph30  = ssSetInputPortWidth(S, 0, 30);
   InputRealPtrsType uph31  = ssSetInputPortWidth(S, 0, 31);
   InputRealPtrsType uph32  = ssSetInputPortWidth(S, 0, 32);
   InputRealPtrsType uph33  = ssSetInputPortWidth(S, 0, 33);
   InputRealPtrsType uph34  = ssSetInputPortWidth(S, 0, 34);
   InputRealPtrsType uph35  = ssSetInputPortWidth(S, 0, 35);
   InputRealPtrsType uph36  = ssSetInputPortWidth(S, 0, 36);
   InputRealPtrsType uph37  = ssSetInputPortWidth(S, 0, 37);
   InputRealPtrsType uph38  = ssSetInputPortWidth(S, 0, 38);
   InputRealPtrsType uph39  = ssSetInputPortWidth(S, 0, 39);
```

```
 InputRealPtrsType uph40  = ssSetInputPortWidth(S, 0, 40);
 InputRealPtrsType uph41  = ssSetInputPortWidth(S, 0, 41);
 InputRealPtrsType uph42  = ssSetInputPortWidth(S, 0, 42);
 InputRealPtrsType uph43  = ssSetInputPortWidth(S, 0, 43);
 InputRealPtrsType uph44  = ssSetInputPortWidth(S, 0, 44);
 InputRealPtrsType uph45  = ssSetInputPortWidth(S, 0, 45);
 InputRealPtrsType uph46  = ssSetInputPortWidth(S, 0, 46);
 InputRealPtrsType uph47  = ssSetInputPortWidth(S, 0, 47);
 InputRealPtrsType uph48  = ssSetInputPortWidth(S, 0, 48);
 InputRealPtrsType utheta  = ssSetInputPortWidth(S, 0, 49);

 /*uPtrs[element])  * Pointer to Input Port0 */

real_T  BMK[49],ph[4][49];
real_T  temp[4][13],sys[3];
real_T M=8,Nw=85,R=50.8e-3,L=25.4e-3,pi=3.1415926,theta;
real_T BMKP=0.78;
int_T ii,jj,k;

 ph1  = *uph1;
 ph2  = *uph2;
 ph3  = *uph3;
 ph4  = *uph4;
 ph5  = *uph5;
 ph6  = *uph6;
 ph7  = *uph7;
 ph8  = *uph8;
 ph9  = *uph9;
 ph10  = *uph10;
 ph11  = *uph11;
 ph12  = *uph12;
 ph13  = *uph13;
 ph14  = *uph14;
 ph15  = *uph15;
 ph16 = *uph16;
 ph17  = *uph17;
 ph18  =*uph18;
 ph19  = *uph19;
 ph20  = *uph20;
 ph21  = *uph21;
 ph22  = *uph22;
 ph23  =*uph23 ;
 ph24 = *uph24;
 ph25  =*uph25;
 ph26  =*uph26 ;
 ph27  =*uph27;
 ph28  =*uph28;
 ph29  =*uph29 ;
 ph30  = *uph30;
 ph31  = *uph30;
 ph32  = *uph30;
 ph33  = *uph30;
 ph34  = *uph30;
 ph35  = *uph30;
 ph36  = *uph30;
 ph37  = *uph30;
```

```
    ph38  = *uph30;
    ph39  = *uph30;
    ph40  = *uph30;
    ph41  = *uph30;
    ph42  = *uph30;
    ph43  = *uph43;
    ph44  = *uph44;
    ph45  = *uph45;
    ph46  = *uph46;
    ph47  = *uph47;
    ph48  = *uph48;
    theta  = *utheta;

/* START FROM HERE F0R TOMMOROW*/

   for(k=1;k<=48;k++)
   {
   phi[k]=(k-1)*(2*pi/48)+(pi/4)+(pi/48); /*ORIENTATION ALONG ANY STATOR STATION
                                            RELATIVE TO X COORDINATE AXIS */
   BMK[k]=BMKP*sin(M*(theta-phi[k]));
   ph[1][k]=-Nw*L*BMK[k]*sin(phi[k]);
   ph[2][k]=Nw*L*BMK[k]*cos(phi[k]);
   ph[3][k]=Nw*L*R*BMK[k];
   }

 sys=ph*ph_input  /*Replace with a mult loop */

  S1[0]=sys[1];
  S2[0]=sys[2];
  S3[0]=sys[3];

}

#define MDL_UPDATE
/* Function: mdlUpdate ======================================================
 * Abstract:
 *      xdot = Ax + Bu */
static void mdlUpdate(SimStruct *S, int_T tid)
{
   UNUSED_ARG(S); /* unused input argument */
}

/* Function: mdlTerminate =====================================================
 * Abstract:
 *    No termination needed, but we are required to have this routine.
 */
static void mdlTerminate(SimStruct *S)
{
    UNUSED_ARG(S); /* unused input argument */
}

#ifdef  matlab_MEX_FILE    /* Is this file being compiled as a MEX-file? */
#include "simulink.c"      /* MEX-file interface mechanism */
#else
#include "cg_sfun.h"       /* Code generation registration function */
#endif
```

Appendix E

Fault tolerant model computations

1 Calculation of desired forces using decoupled K_i :

$$\overbrace{\begin{bmatrix} F_x \\ F_y \\ T \end{bmatrix}}^{3x1} = \underbrace{\overbrace{\begin{bmatrix} K_{ixx}(\theta) & 0 & 0 \\ 0 & K_{ixx}(\theta) & 0 \\ 0 & 0 & K_{i\theta} \end{bmatrix}}^{3x3}}_{K_i} \underbrace{\overbrace{\begin{bmatrix} i_{cx} \\ i_{cy} \\ i_{c\theta} \end{bmatrix}}^{3x1}}_{\mathbf{i}_C} = \begin{bmatrix} 155 & 0 & 0 \\ 0 & 155 & 0 \\ 0 & 0 & 37 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 155 \\ 155 \\ 74 \end{bmatrix} \quad (1)$$

2 Calculation of i_s using pseudo inverse A^+:

2.1 Calculation of $A = \Phi(\theta)\Lambda FY$:

$$A = \overbrace{\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{bmatrix}}^{3 \text{ x } 4} \quad (2)$$

2.2 Calculation of A^{-1} :

$$A^{-1} = A^T \ . \ (A \ . \ A^T)^{-1} \quad (3)$$

$$A \ . \ A^T = \overbrace{\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{bmatrix}}^{3 \text{ x } 4} \overbrace{\begin{bmatrix} a_{11} & a_{21} & a_{31} \\ a_{12} & a_{22} & a_{32} \\ a_{13} & a_{23} & a_{33} \\ a_{14} & a_{24} & a_{34} \end{bmatrix}}^{4 \text{ x } 3} \quad (3)$$

$$= \overbrace{\begin{bmatrix} a_{11}.a_{11}+a_{12}.a_{12}+a_{13}.a_{13}+a_{14}.a_{14} & a_{11}.a_{21}+a_{12}.a_{22}+a_{13}.a_{23}+a_{14}.a_{24} & a_{11}.a_{31}+a_{12}.a_{32}+a_{13}.a_{33}+a_{14}.a_{34} \\ a_{21}.a_{11}+a_{22}.a_{12}+a_{23}.a_{13}+a_{24}.a_{14} & a_{21}.a_{21}+a_{22}.a_{22}+a_{23}.a_{23}+a_{24}.a_{24} & a_{21}.a_{31}+a_{22}.a_{32}+a_{23}.a_{33}+a_{24}.a_{34} \\ a_{31}.a_{11}+a_{32}.a_{12}+a_{33}.a_{13}+a_{34}.a_{14} & a_{31}.a_{21}+a_{32}.a_{22}+a_{33}.a_{23}+a_{34}.a_{24} & a_{31}.a_{31}+a_{32}.a_{32}+a_{33}.a_{33}+a_{34}.a_{34} \end{bmatrix}}^{3 \text{x} 3}$$

(4)

$$= \overbrace{\begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix}}^{3 \text{ x } 3} \tag{5}$$

$$(A \,.\, A^T)^{-1} = \frac{Adj(A \,.\, A^T)}{\left|A \,.\, A^T\right|} \tag{6}$$

$$(A \,.\, A^T)^{-1} = \frac{\begin{bmatrix} \left(b_{22} \,.\, b_{33} - b_{32} \,.\, b_{23}\right) & -\left(b_{21} \,.\, b_{33} - b_{31} \,.\, b_{23}\right) & \left(b_{21} \,.\, b_{32} - b_{31} \,.\, b_{22}\right) \\ -\left(b_{12} \,.\, b_{33} - b_{32} \,.\, b_{13}\right) & \left(b_{11} \,.\, b_{33} - b_{31} \,.\, b_{13}\right) & -\left(b_{11} \,.\, b_{32} - b_{31} \,.\, b_{12}\right) \\ \left(b_{12} \,.\, b_{23} - b_{22} \,.\, b_{13}\right) & -\left(b_{11} \,.\, b_{23} - b_{21} \,.\, b_{13}\right) & \left(b_{11} \,.\, b_{22} - b_{21} \,.\, b_{12}\right) \end{bmatrix}}{b_{11}.\left(b_{22} \,.\, b_{33} - b_{32} \,.\, b_{23}\right) - b_{12}\left(b_{21} \,.\, b_{33} - b_{31} \,.\, b_{23}\right) + b_{13}\left(b_{21} \,.\, b_{32} - b_{31} \,.\, b_{22}\right)} \tag{7}$$

$$= \overbrace{\begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix}}^{3 \text{ x } 3} \tag{8}$$

$$A^{-1} = A^T \,.\, (A \,.\, A^T)^{-1} \tag{9}$$

$$A^{-1} = \overbrace{\begin{bmatrix} a_{11} & a_{21} & a_{31} \\ a_{12} & a_{22} & a_{32} \\ a_{13} & a_{23} & a_{33} \\ a_{14} & a_{24} & a_{34} \end{bmatrix}}^{4 \text{ x } 3} \overbrace{\begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix}}^{3 \text{ x } 3} = \overbrace{\begin{bmatrix} a_{11}.c_{11} + a_{21}.c_{22} + a_{31}.c_{33} \\ a_{12}.c_{11} + a_{22}.c_{22} + a_{32}.c_{33} \\ a_{13}.c_{11} + a_{23}.c_{22} + a_{33}.c_{33} \\ a_{14}.c_{11} + a_{24}.c_{22} + a_{34}.c_{33} \end{bmatrix}}^{4 \text{ x } 3} \tag{10}$$

2.3 Calculation of $\mathbf{i}_s$:

$$\overbrace{\mathbf{i}_s}^{4x1} = \overbrace{A^{-1}}^{4x3} \overbrace{\mathbf{F}_c}^{3x1} \tag{11}$$

$$\overbrace{\begin{bmatrix} i_{s1} \\ i_{s2} \\ i_{s3} \\ i_{s4} \end{bmatrix}}^{4x1}_{i_S} = \overbrace{\begin{bmatrix} a_{11}.c_{11} + a_{21}.c_{22} + a_{31}.c_{33} \\ a_{12}.c_{11} + a_{22}.c_{22} + a_{32}.c_{33} \\ a_{13}.c_{11} + a_{23}.c_{22} + a_{33}.c_{33} \\ a_{14}.c_{11} + a_{24}.c_{22} + a_{34}.c_{33} \end{bmatrix}}^{4 \text{ x } 3} \overbrace{\begin{bmatrix} F_x \\ F_y \\ T \end{bmatrix}}^{3 \text{ x } 1} \tag{12}$$

3 Calculation of i_ϕ :

$$\overbrace{i_\phi}^{12x1} = \overbrace{Y}^{12x4} \overbrace{i_S}^{4x1} \tag{13}$$

$$i_\phi = \overbrace{\begin{bmatrix} \Phi_1 & 0 & 0 & 0 \\ \Phi_2 & 0 & 0 & 0 \\ \Phi_3 & 0 & 0 & 0 \\ 0 & \Phi_1 & 0 & 0 \\ 0 & \Phi_2 & 0 & 0 \\ 0 & \Phi_3 & 0 & 0 \\ 0 & 0 & \Phi_1 & 0 \\ 0 & 0 & \Phi_2 & 0 \\ 0 & 0 & \Phi_3 & 0 \\ 0 & 0 & 0 & \Phi_1 \\ 0 & 0 & 0 & \Phi_2 \\ 0 & 0 & 0 & \Phi_3 \end{bmatrix}}^{Y \text{ (4 x 12)}}_{1} \begin{bmatrix} i_1 \\ i_2 \\ i_3 \\ i_4 \end{bmatrix} = \begin{bmatrix} i_1\Phi_1 \\ i_1\Phi_2 \\ i_1\Phi_3 \\ i_2\Phi_1 \\ i_2\Phi_2 \\ i_2\Phi_3 \\ i_3\Phi_1 \\ i_3\Phi_2 \\ i_3\Phi_3 \\ i_4\Phi_1 \\ i_4\Phi_2 \\ i_4\Phi_3 \end{bmatrix}$$

$$\Phi_1 = \cos[m(\theta-\gamma)+\pi/3]$$
$$\Phi_2 = \cos[m(\theta-\gamma)]$$
$$\Phi_3 = \cos[m(\theta-\gamma)-\pi/3]$$

(14)

4 Calculation of i_{st} :

$$i_{st} = \overbrace{\begin{bmatrix} \beta_{12x3} & 0 & 0 & 0 \\ 0 & \beta_{12x3} & 0 & 0 \\ 0 & 0 & \beta_{12x3} & 0 \\ 0 & 0 & 0 & \beta_{12x3} \end{bmatrix}}^{\lambda\ (48\ x\ 12)} \overbrace{\begin{bmatrix} i_1\Phi_1 \\ i_1\Phi_2 \\ i_1\Phi_3 \\ i_2\Phi_1 \\ i_2\Phi_2 \\ i_2\Phi_3 \\ i_3\Phi_1 \\ i_3\Phi_2 \\ i_3\Phi_3 \\ i_4\Phi_1 \\ i_4\Phi_2 \\ i_4\Phi_3 \end{bmatrix}}^{12\ x\ 1} = \overbrace{\begin{bmatrix} i_1\Phi_1 \\ i_1\Phi_2 \\ i_1\Phi_3 \\ i_2\Phi_1 \\ i_2\Phi_2 \\ i_2\Phi_3 \\ i_3\Phi_1 \\ i_3\Phi_2 \\ i_3\Phi_3 \\ i_4\Phi_1 \\ i_4\Phi_2 \\ i_4\Phi_3 \\ i_1\Phi_1 \\ i_1\Phi_2 \\ i_1\Phi_3 \\ i_2\Phi_1 \\ i_2\Phi_2 \\ i_2\Phi_3 \\ i_3\Phi_1 \\ i_3\Phi_2 \\ i_3\Phi_3 \\ | \\ | \\ | \end{bmatrix}}^{48\ x\ 1} \quad (15)$$

5 Calculation of F_x, F_y, T:

$$\begin{bmatrix} F_x \\ F_y \\ T_\theta \end{bmatrix} = \overbrace{\begin{bmatrix} M * L * B_{MK}(1)\ Sin(1) & - & - & - & M * L * B_{MK}(48)\ Sin(48) \\ M * L * B_{MK}(1)\ Cos(1) & - & - & - & M * L * B_{MK}(48)\ Sin(48) \\ M * L * R * B_{MK}(1)\ I(1) & - & - & - & M * L * R * B_{MK}(48) I(48) \end{bmatrix}}^{\Phi(\theta)} \overbrace{\begin{bmatrix} i_1\Phi_1 \\ i_1\Phi_2 \\ i_1\Phi_3 \\ i_2\Phi_1 \\ i_2\Phi_2 \\ i_2\Phi_3 \\ i_3\Phi_1 \\ i_3\Phi_2 \\ i_3\Phi_3 \\ i_4\Phi_1 \\ i_4\Phi_2 \\ i_4\Phi_3 \\ i_1\Phi_1 \\ i_1\Phi_2 \\ i_1\Phi_3 \\ i_2\Phi_1 \\ i_2\Phi_2 \\ i_2\Phi_3 \\ i_3\Phi_1 \\ i_3\Phi_2 \\ i_3\Phi_3 \\ | \\ | \\ | \end{bmatrix}}^{48\ x\ 1} \quad (16)$$

Vita

Anand Ranganathan

Born May 13th 1979 in Chennai, Tamilnadu, India

Education:

Bachelor of Engineering in Mechanical Engineering, 2000

University of Madras, Chennai

Advanced level automotive course work, 2001

Madras Institute of Technology, Anna University, Chennai

Membership: ASME, SAE and I EEE

Scholastic honors:

Federal Government of India GATE scholarship for graduate studies (2000-2001)

Kentucky Graduate Scholarship (2001-2004)

Professional Publication:

Lyndon S. Stephens, Anand Ranganathan, *"Decoupled and fault tolerant algorithm of Lorentz self-bearing motor considering open coil faults,"* 9th International symposium on magnetic bearing, 2004, University of Kentucky