8-1-2012

Electromechanical actuator bearing fault detection using empirically extracted features

Rahulram Sridhar

Electromechanical Actuator Bearing Fault Detection using Empirically Extracted Features

by

Rahulram Sridhar

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of
Master of Science
in Mechanical Engineering

Supervised by

Assistant Professor Dr. Jason R Kolodziej
Department of Mechanical Engineering
Kate Gleason College of Engineering
Rochester Institute of Technology
Rochester, New York
August 2012

Approved by:

Dr. Jason R Kolodziej, Assistant Professor
Thesis Advisor, Department of Mechanical Engineering

Dr. Mark Kempski, Professor
Committee Member, Department of Mechanical Engineering

Dr. Agamemnon Crassidis, Associate Professor
Committee Member, Department of Mechanical Engineering

Dr. Wayne W Walter, Professor
Department Representative, Department of Mechanical Engineering

Thesis Release Permission Form

Rochester Institute of Technology
Kate Gleason College of Engineering

Title:

Electromechanical Actuator Bearing Fault Detection using Empirically Extracted Features

I, Rahulram Sridhar, hereby grant permission to the Wallace Memorial Library to reproduce my thesis in whole or part.

Rahulram Sridhar

Date

Dedication

To the future of graduate-level research at the Department of Mechanical Engineering, Rochester Institute of Technology, Rochester, New York

Acknowledgments

I was introduced to this area of research by Dr. Jason Kolodziej of the Department of Mechanical Engineering (ME) at the Rochester Institute of Technology (RIT) who also provided the research problem and guidance with the solution. During the preliminary stages of the research, experiments were conducted with a DC motor control module. Assistance with obtaining and using the module was provided by Dr. Mark Kempski, also of the ME department at RIT. Funding for the research was provided by Moog Inc., East Aurora, NY. External supervision and feedback was provided by Mr. Larry Hall and Mr. Anthony J. Chirico III, both of Moog Inc. All the work was conducted at the Department of Mechanical Engineering, Rochester Institute of Technology with help being extended by Dr. Edward Hensel, Head of the Mechanical Engineering Department, whenever it was sought. And as always, my parents Mr. C. Sridhar and Mrs. Srividya Sridhar offered their support. I also acknowledge the following individuals.

Dr. Frank Sciremammano Jr.
Dr. Wayne W. Walter
Dr. Agamemnon Crassidis
Ms. Venessa Mitchell
Ms. Diane Selleck

Abstract

Electromechanical Actuator Bearing Fault Detection using Empirically Extracted Features

Rahulram Sridhar

Supervising Professor: Dr. Jason R Kolodziej

Model parameter estimation when coupled with Principal Component Analysis (PCA) and Bayesian classification techniques form a potentially effective fault detection scheme for Electromechanical Actuators (EMAs). This work uses parameter estimation algorithms based on linear system identification methods, derives a novel feature extraction algorithm based on PCA and analyzes its performance through simulations and experiments. A Bayesian classifer is used to create well defined EMA health classes from the extracted features.

Research contributions on fault detection in EMAs are significant because EMA faults and their detection are not yet well understood. Potential future applications - such as in primary flight control actuation in aircraft - require that quality fault detection systems be in place. Therefore, fault detection of EMAs is a vast area of ongoing research where highly capable solutions are gradually becoming available. Prior work in parameter estimation methods for feature extraction in DC motor drives - which includes EMAs - are amongst those available. While PCA is a popular feature extraction solution in a number of frequency-based fault detection approaches, the use of PCA for feature extraction from model parameters for detecting bearing faults in EMAs has not been previously reported.

In this work, a linear difference model is applied to the EMA system data such that fault information is distributed amongst the estimated model parameters. A direct comparison of the parameter estimates from healthy and degraded systems offers little insight into health conditions because of the weak effects of faults on the signal data. However, the application of PCA to uncorrelate the linearly correlated model

parameters while minimizing the loss of variance information from the data effectively brings out fault information. The present algorithm is successfully applied to data collected from a Moog MaxForce EMA. The results are consistent and display effective fault detection characteristics, making the developed approach a suitable starting point for future work.

Contents

List of Tables

List of Figures

Nomenclature

f_c	Bearing fault frequency reflected in the stator current
f_e	Supply frequency or fundamental current frequency
f_v	Characteristic vibration frequency
$y(t)$, $u(t)$	Generic discrete-time output and input data
$y^F(t)$, $u^F(t)$	Filtered generic discrete-time output and input data
$e(t)$	Generic discrete-time white-noise process/Equation-error
q^{-1}	Backward shift operator
θ, $\hat{\theta}$	Unknown and estimated parameter vectors
$G(q,\theta), H(q,\theta)$	Transfer functions from $u(t)$ and $e(t)$ to $y(t)$
$f_e(x,\theta)$	Probability density function of $e(t)$
a, b, c	Output and input term coefficients in a linear difference equation
n_a, n_b, n_c	Number of a, b and c coefficients
$A(q), B(q), C(q)$	Vectors of a, b and c parameters
$w(t)$, f, n_f	Output error model regressors, coefficients and number of terms
$F(q)$	Vector of f parameters
$\phi(t)$	Equation error model regression vector
$\hat{y}(t\|\theta)$	Estimated output given θ
$V(\theta)$, N	Scalar-valued norm and length of data set
X, Z, Y	Regression matrix, Instrumental variables matrix and output data
v_m, i_m, ω_m	Armature voltage, current and angular speed respectively
K_e, K_t	Back-emf and electromagnetic torque constant
B, J	Net viscous friction coefficient and system inertia
R, L, T_{ex}	Armature resistance, inductance and net external load torque
Φ, T	State transistion matrix and sampling interval

f_{OD}, f_{rm}	Outer-race defect frequency and motor angular speed in Hz
BD, PD, n, δ	Ball bearing and pitch diameter, number of balls and contact angle
σ_i, σ_i^2	Standard deviation and variance of a parameter i
$r(s)$, $y(s)$, $y_m(s)$	Residual, measured output and simulated model output
G_c, G_p	Controller and Plant
G_m, G_s	Plant model and System model
$Q(s)$, $P(s)$	Controller numerator and denominator coefficients
$B_p(s)$, $A_p(s)$	Plant numerator and denominator coefficients
$B_m(s)$, $A_m(s)$	Plant model numerator and denominator coefficients
$B_s(s)$, $A_s(s)$	System model numerator and denominator coefficients
G_{cp}, G_{cs}	Position and speed controllers
$e_1(s)$, $e_2(s)$	Position and velocity errors
$\mathbf{z}$, Ω	Feature vector and set of possible health classes
$\hat{w}(\mathbf{z})$, $C(w\|w_k)$	Assigned class and Bayesian classification cost function
$p(\mathbf{z}\|w_k)$, $P(w_k)$	Conditional density and unconditional prior probability of each class
μ_k, $\mathbf{C}_k$, N	Expectation vector, covariance matrix and dimension of $\mathbf{z}$
k, k_e	Individual and combined (effective) spring constants
p	Number of brushless DC motor poles
i_a, i_b, i_c	Individual phase currents of brushless DC motor
I_d, I_q, I_o	Brushless DC motor direct, quadrature and zero axis currents

Abbreviations

EMA	Electromechanical Actuator
EHA	Electrohydrostatic Actuator
FBW	Fly-By-Wire Technology
PBW	Power-By-Wire Technology
PMSM	Permanent Magnet Synchronous Motor
FFT	Fast Fourier Transform
SISO	Single Input Single Output
LSE	Least Squares Estimation
ARX	Auto-Regressive with Exogeneous Input
ARMAX	Auto-Regressive Moving Average with Exogeneous Input
OE	Output Error
IV	Instrumental Variables
PMDC	Permanent Magnet Direct Current
SNR	Signal-to-Noise Ratio
PCA	Principal Component Analysis
PI	Proportional-Integral
LVDT	Linear Variable Differential Transformer

Chapter 1

Introduction

Electromechanical actuation is used extensively in various applications and trying to list all possibilities in this document alone is impractical. The most important applications are in aircraft, defense and industry as these specific solutions are commonly advertised by companies like Moog Inc., East Aurora, NY which specialize in offering them. For instance, the advantages of using EMAs in aircraft and launch vehicles is adequately summarized in a technology review by S Botten et.al. [6] and in a technical report by MA Davis of Moog Inc. [1]. In this chapter, all subsequent information related to fault detection in EMAs pertains to its application in aircraft, although a lot of it can be directly applied to defense and industrial applications as well.

The concept of an all-electric aircraft emerged in the 1970s [6]. Since then, increased research activity attempting to achieve this goal is noticeable based on the breadth and depth of results from research articles published in the last decade. One of the areas of on-going research is in flight actuation technologies. Such technologies are typically classified into two main areas namely fly-by-wire (FBW) and power-by-wire (PBW) [6]. FBW is a technology that replaces manual flight controls by an electronic interface. In such systems, the control or actuation signals are provided electrically but the power required for the actuation is typically hydraulic and requires a central hydraulic system. By the late 90s, technological advancements in FBW were developed to the extent that they were introduced in the primary flight control systems of in-service aircraft such as the Airbus A320 and Boeing 777 [6]. PBW is a more recent technology that differs from FBW in that the power required for actuation is

also electrical in nature, thus eliminating the need for a central hydraulic system. Examples of PBW actuators as shown in [7] are reproduced in Fig. 1.1. Until the late 90s, PBW technology was not as mature as FBW [6]. For instance, preliminary testing of an EMA replacement for hydraulic actuation for primary flight control surfaces in an F-18 Systems Research Aircraft (SRA) was conducted as recently as the year 2000 [7]. The results of the investigation showed some promise but there were a number of serious thermal-related issues that needed attention, let alone the high possibility of inherent defects in the EMA which were not investigated.

The last decade has seen some increase in research related to PBW actuation primarily because of the advantages that electric actuation has to offer. Recent implementation of PBW technology for secondary tasks such as spoiler and trim tabs actuation is observed [1, 6, 7, 8]. However, the use of PBW actuation for primary flight control surfaces is yet to be effectively implemented. For instance, even the relatively new Airbus A380 unveiled in 2005 incorporates electrohydraulic actuation - an FBW technology - for its primary flight control surfaces and uses Electrohydrostatic Actuators (EHAs) - a PBW technology - only as a "back-up". Detailed information about the actuation systems of the A380 is provided in [8]. More information about Electrohydrostatic Actuators (EHAs) is provided in [6, 1, 7]. EMAs are not yet associated with primary actuation surfaces in aircraft mainly because they are not sufficiently well understood - especially fault characteristics and their detection. Consequently, they are currently the subject of a vast amount of research - this work being one of them.

1.1 Electric Actuation in Aircraft: A Brief Comparative Study between EHAs and EMAs

According to S Botten et.al. [6], some of the advantages that electric actuation provide over current flight actuation systems are as follows:

1. Improved aircraft maintainability since fewer hydraulic components are required.

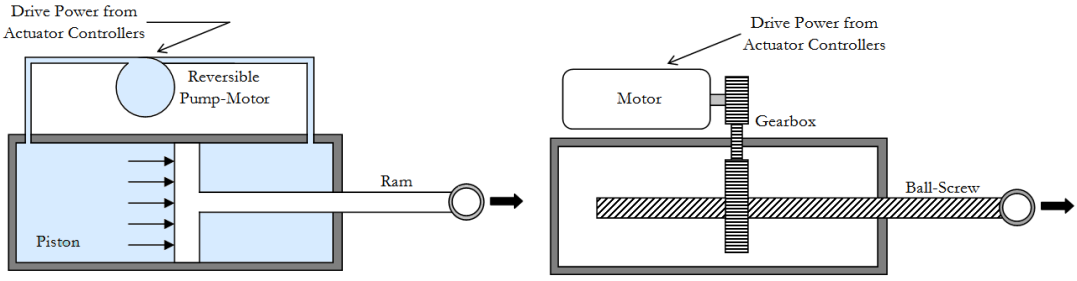

Figure 1.1: Examples of Power-By-Wire actuators - Left: Electrohydrostatic Actuator; Right: Electromechanical Actuator

2. More flexibility with respect to system reconfiguration.

3. Reduced system weight due to the reduction in the number of bulky hydraulic components.

All of the above, in turn, reduce the aircraft operating costs. EMAs offer many benefits that EHAs do not, such as improved reliability, and reduced complexity, weight and maintenance requirements. Furthermore, hydraulic technology related issues - windage losses, pump efficiency issues, hydraulic fluid leakages - are eliminated. Despite all these benefits, the EHA is still the currently preferred PBW actuation technology for aircraft flight controls because of the following reasons [8]:

1. The jamming probability of an EMA used in primary flight control is difficult to predict based on current experience and any knowledge from secondary flight control applications cannot be directly used for primary flight control due to very different operating cycles.

2. Wearing of mechanical components are not yet well understood and the generation of unacceptable limit cycles cannot be prevented.

Both the above points highlight a general lack of knowledge. In line with requirements, this work attempts to contribute to the growth in knowledge of EMAs - specifically of the detection of certain fault types.

1.2 The Electromechanical Actuator

This work utilizes a Moog MaxForce Electromechanical Actuator. The system is shown schematically in Figs. 1.2 and 1.3. The actuation power is provided by a

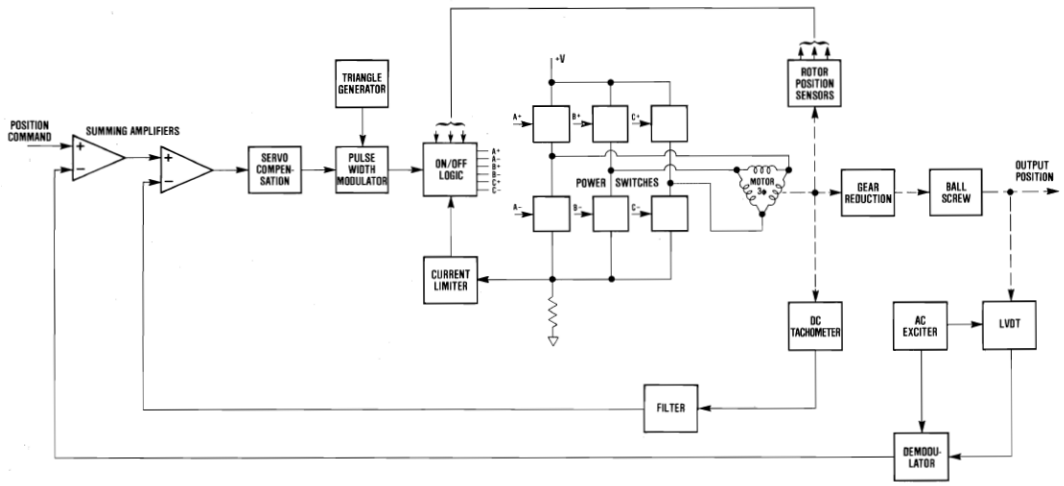

Figure 1.2: Moog MaxForce Electromechanical Actuator Block Diagram [1]

DC three-phase brushless permanent magnet synchronous motor (PMSM). There is an outer position feedback loop and an inner velocity feedback loop. The inner loop is used to provide higher performance servoactuation by providing a convenient way of establishing a stable position loop having a high static gain for precise actuator positioning. It also reduces the effects of motor cogging [1]. Information regarding the operational characteristics of the EMA is provided in Chapter 5.

The cross-sectional view of the actuator is shown in Fig. 1.4. The screw is rigidly coupled to the rotor and the nut is coupled to the screw by steel balls that circulate along the single race of the screw. The rotational motion of the screw is converted to translational motion by constraining the ball-nut's motion along the screw. The translation of the nut produces the linear motion of the piston rod. The angular contact bearing connects the motor's shaft to the ball-screw.

When these bearings develop defects, the actuation performance is compromised and when EMAs are used on aircraft, they are considered safety-critical components

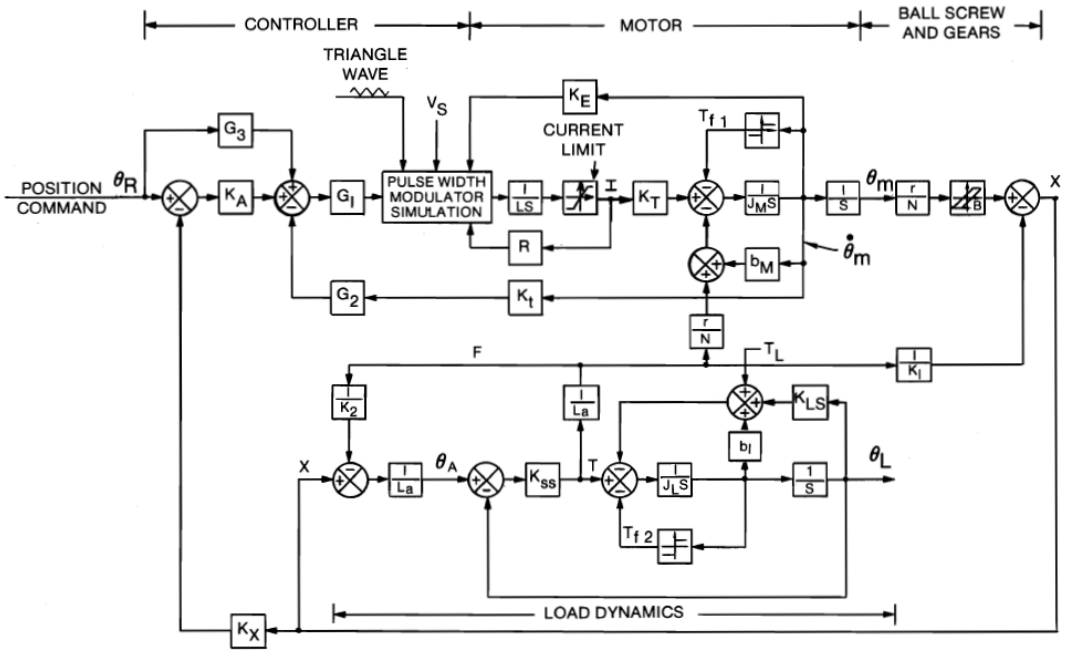

Figure 1.3: Moog MaxForce Electromechanical Actuator simulation model [1]

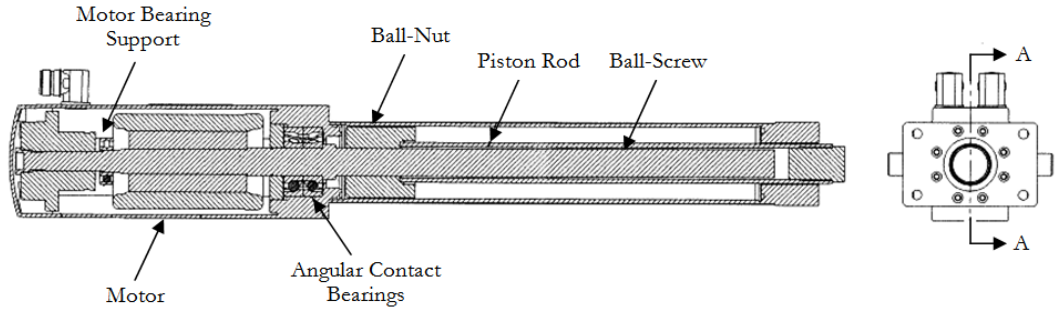

Figure 1.4: Cross-section AA of the Moog MaxForce EMA (Courtesy of Moog Inc., East Aurora, NY)

making it imperative that faults be detected in their early stages of development. The next two sections highlight the need for fault detection in EMAs and the work that has already been done in this regard.

1.3 The Need for Fault Detection in EMAs

As mentioned earlier and according to a 2009 paper by P Bansal et.al. [9], EMAs are relatively new to the aerospace industry and have not yet been used for a long enough time or in large enough quantities to be able to understand their operating characteristics and flaws specific to this application. In most of the current commericial and military applications, EMAs are restricted for use in secondary tasks such as trim tabs actuation and spoiler or speed brake deployment [9]. The most recent research developments allow a marginal increase in the application areas of EMAs. For example, usage in state-of-the-art commericial aircraft such as the Airbus A380 and Boeing 787 increased and EMAs are now used to operate landing gear brakes in addition to spoilers and trim tabs [9]. However, the benefits that EMAs offer are not yet fully exploited, primarily because they are susceptible to a number of faults, the identification and mitigation of which isn't understood well enough for flight certification on primary surfaces. Currently available solutions are still too complex to be incorporated without adding costs and increasing the weight of the system. With respect to EMAs, mechanical and structural faults caused by excessive loads, lubrication issues or manufacturing defects are the main causes for concern [9]. Other fault types include motor faults, electrical and electronic faults and sensor faults. More information on these types of faults and critical failure modes of EMAs can be found in [9, 10, 11]. In this work, the focus is on mechanical faults and lubrication issues only.

The type of mechanical defect considered in this work is a spall in the inner-race of the roller bearing that is in contact with the output shaft of the PMSM. Additionally, the lubrication in the bearing is decreased by 25 percent of the nominal value. The spall in the inner-race is generated using Electric Discharge Machining (EDM).

According to the study of mechanical fault modes in EMAs by P Bansal et.al. [9], a spalled bearing results in severe vibrations and separation of metal flakes. This type of fault has - on a scale from one to ten with ten being the highest - a relative probability of occurence of five and a relative criticality of three with respect to the use of EMAs in secondary aircraft systems. These values increase when considering the EMAs for primary flight surface actuation. Detecting spalled bearings are consequently an important problem to solve. Also, according to the same paper, using a grey-box model - one that is developed both empirically and by utilising physical knowledge about the system - is the recommended approach for feature extraction in this case. Furthermore, a lack of lubrication in the bearing causes seizure and this type of defect has a relative criticality of four although the relative probability of occurence is around two on the same scales as before. In this work, Moog seeded both faults simultaneously and therefore addressing fault detection in EMAs with multiple and simultaneously occuring fault types is vital.

After a thorough review of prior work (most of which are published within the last decade - see Section 1.4), it is found that solutions exist for each of the fault types treated in this work. However, it is pointed out that detecting the presence of both fault types simultaneously using parameter estimation and the feature extraction approaches devised here is not reported.

It is thus concluded that the research problem addressed in this work is relevant in relation to the state-of-the-art in fault detection of EMAs. It follows directly from the last statement that the need for fault detection in EMAs is evident primarily because of the various other fault types that can occur in EMAs (see [9, 10, 11]) that are not addressed in this work.

1.4 Summary of Prior Work

There are a vast number of research publications regarding fault detection of technical processes. Those pertaining to EMAs in particular are however significantly fewer in number and have only appeared within the last five years. A 1997 review

paper by Isermann and Balle [12] present model-based fault detection approaches in technical processes in general. The paper provides some useful definitions regarding standard terminology used in this field. One of particular importance to this work are the definitions of fault detection and isolation. While detection merely refers to acknowledging the presence of a fault in a system, isolation includes determining the kind and location of the fault in the system. As specified in the title, this thesis deals with detection alone. For instance, the Moog MaxForce EMA from which data is collected consists of two kinds of bearing faults present simultaneously. The developed algorithms only detect the presence of the faults and is unable to tell which particular fault amongst the two has been detected. Some of the model-based methods identified include (i) use of state and output observers which include generation of residuals for state variables or output variables with fixed parametric models, (ii) parity equations which include fixed parametric or nonparametric models and (iii) identification and parameter estimation which utilize adaptive nonparametric or parametric models.

Another commonly used approach is a signal-model-based approach based on a spectral analysis of particular signals to extract features that are compared to nominal feature values in order to detect faults. The paper [12] states that, as of 1995 for detecting actuator faults (which includes various types of actuators and not EMAs alone), parameter estimation accounts for about ten percent of the techniques used.

An updated review paper by Isermann on model-based fault detection published in 2005 [13] makes an important statement regarding the requirements of a good fault detection scheme. These requirements include (i) early detection of small faults with abrupt time behavior and (ii) ability to detect faults in closed loops. Both points are important considerations for this thesis because the Moog EMA operates in closed loop and the inner-race bearing fault present in the EMA is a low magnitude fault, thus making its detection a challenging task. Also, a much sought after application of EMAs is in aircraft where early detection of such faults is necessary. The paper [13] presents the same model-based approaches discussed in the 1997 paper [12] but additionally presents an example related to fault diagnosis of a cabin pressure

outflow valve actuator of a passenger aircraft. In that example [13], a combined parameter-estimation and parity equation approach is presented where the estimated parameters are used to develop parity equations which are essentially residual equations used to indicate faulty systems. The approach offers good fault coverage as it attempts to combine the benefits of two different fault detection techniques. However, the drawback lies in the requirement of a number of signals that are not always readily available.

Obtaining a large number of signals from a system requires the use of additional sensors. This is not always practical and is definitely more expensive. Therefore, it is advantageous to make use of signal information that is readily available from the system without having to use additional sensors. An example is the motor current signal of an EMA which can be readily obtained from sensors already present in the system (usually for electrical protection [14]). This is unlike vibration signals which require the additional use of an accelerometer. There are a number of fault detection approaches that make use of current signals to detect faults. In the case of bearing faults in electrical motor drives, current signals are used because it is stated in [14] that bearing fault frequencies in system vibrations can be reflected in stator current. The relationship between the vibration frequencies and current frequencies for bearing faults is described by Eq. (1.1).

$$f_c = \mid f_e \pm m f_v \mid \tag{1.1}$$

where f_c is the bearing fault frequency reflected in the stator current, f_e is the supply frequency or fundamental current frequency, f_v is one of the characteristic vibration frequencies and $m = 1, 2, 3, \ldots$ account for harmonic contributions.

Zhou et.al. [14] also explain that bearing faults account for over 40 percent of all induction machine faults making them an important class of faults to study. Bearing faults can be classifed into single-point defects and generalized roughness type faults. Both these faults are addressed in this thesis with the single-point defect present in the inner-race of the EMA angular contact bearings (see Fig. 1.4) and the generalized

roughness type defect occuring due to a reduction in the amount of bearing lubrication. Zhou et.al. [14] go on to suggest that single-point defects are commonly detected using frequency-based approaches that identify characteristic fault frequencies in the stator current while such an approach is not possible for detecting generalized roughness type faults because such faults cause broad changes in the frequency spectrum of the stator current.

Some of the techniques presented in [14] that deal with single-point defects include a neural-network clustering approach which samples the stator current, computes the spectrum using a Fast Fourier Transform (FFT) and selects frequency components to be used in a neural network for clustering. Once clusters are formed from a healthy system, the formation of additional clusters when monitoring a system of unknown health indicates the occurrence of a fault. The main drawback of this approach is that rules based on knowledge of the spectral distribution of the stator current need to be made and this is not easily accomplished. Other approaches dealing with single-point bearing defects presented in [14] include an Adaptive Statistical Time Frequency Method, Wavelet Packet Decomposition Method and an Extended Park's Vector Approach. All the methods presented are frequency-based and do not involve the use of a model or parameter estimation of any kind as is done in this thesis.

In their 2007 paper [14], Zhou et.al. also mention the dearth in the amount of literature available that deals with generalized roughness faults and also stress the need for further research in this area. One technique called the mean spectrum deviation method proposed in 2004 is discussed. It is also a frequency-based approach that uses a filtering mechanism and an averaging process to detect generalized roughness faults. While successful fault detection is claimed, the authors of [14] point out two main drawbacks with the approach that refer to the need for extensive knowledge of the current spectrum distribution as well as near-perfect signal filtering to avoid loss in fault information, thus making the solution a cumbersome one. Unlike the popular approaches presented in [14], this thesis attempts to utilize a model-based parametric estimation approach coupled with pattern recognition techniques to detect both types

of faults without increasing the complexity of the approach.

Based on the content of the three review papers, the discussed approaches are generally capable of detecting specific fault types accurately and are either unable to address issues regarding alternative types of faults or require significant modifications making them impractical and expensive. Therefore, the problems that still require attention are:

1. Ability of a single approach to detect both single-point defects and generalized roughness defects.

2. Ability of the proposed fault detection technique to utilize inherent signals without using additional sensors.

3. Ability to detect low magitude faults occuring in systems operating in a closed loop.

4. Addressing all the above issues simultaneously without compromising on computational efficiency and cost.

Keeping these points in mind, an extensive review of literature is carried out to identify which approaches have already been attempted along with their positive and negative attributes. This allows the research problem to be effectively constructed.

S.M. Mahdi Alavi et.al. [15] present a fault detection and isolation technique for an SISO closed loop control system exhibiting actuator and sensor faults. In the proposed approach, a frequency-based method is employed that is based on system frequency response shaping. The user is presented with a graphical environment to design a fault detection filter by manually shaping the frequency response of the system. The developed filter is then used to generate a residual signal that is sensitive to actuator and sensor faults. The authors test the efficiency of the approach on a single machine infinite bus power system and a satisfactory level of performance is reported, although no classification scheme is employed. The fault detection results are not intuitive and there is no discussion on fault isolation.

A 1995 paper by C. Aubrun et.al. [16] discusses an important problem related to

fault detection in a control loop. A model-based fault detection approach is applied to a control loop containing a non-linear pneumatic actuator to detect mainly actuator drift and sensor faults such as bias. Model parameters are calculated using a fuzzy C-means algorithm which the authors claim is able to occasionally detect deviations of the order of one percent of the nominal value. One of the limitations identified by the authors is related to determination of initial conditions of the models used. Also, the approach is application specific and the techniques employed cannot be directly applied on an alternative system such as an EMA.

The need for fault detection methods that do not require additional sensors was previously stressed upon. An important contribution in this regard comes from a 2011 paper by G Sreedhar Babu et.al. [17] that deals with condition monitoring of brushless DC motor-based electromechanical linear actuators using motor current signature analysis. The information in this paper [17] is important to this thesis because of the similarity in the area of application as well as in the nature of the EMA signals used. Unlike the work in this thesis however, the work in [17] detects faults such as gear tooth breakage and errors in gear mounting, the effects of which are generally more prominent compared to bearing defects that are dealt with in this thesis. Furthermore, the proposed fault detection approach in [17] is frequency-based that involves computing a fast fourier transform (FFT) of the motor current signal and then determining the energy spectrum density which gives the modulus of the amplitude of the signal at different frequencies. The amplitude differences in vibration and current spectrums are compared amongst each other for systems of different health conditions. No classification scheme is proposed in [17] and the paper appears to focus on the application to different types of gear related faults as well as study the effect of load on fault detection. The paper concludes by stating the successful use of current signals to detect faults based on a simple frequency analysis. This confirms that motor current carries information about faults in the system and the information is generally picked up with relatively the same ease when compared to using vibration signals. This is an important result in the context of this thesis because vibration

signals are not used.

A large number of other publications highlight the benefits and potential of using current signals for fault detection in electric motor drives. Fabio Immovilli et.al. [18] in 2009 discussed the use of current signals to detect generalized-roughness bearing faults in a DC motor drive by a frequency-based method that analyzes the kurtogram to identify the frequency bandwidth where the effect of the fault is strongest. Chirico et.al. [19] recently demonstrated a frequency-based approach that utilizes order analysis of motor current signals to extract features containing information about faults through pattern recognition techniques. The approach is tested on the same EMA used in the present thesis and shows generally favorable results when current signals are used and significantly improved results with vibration signals, a trend that is observed amongst the large number of other frequency-based approaches available [20, 21, 22, 23, 24, 25, 26].

There are papers published within the last five years that deal with the specific problem of fault detection for EMAs in aerospace systems [9, 10, 21]. One such paper [9] published in 2009 analyzes some critical failure modes documented for EMAs and also describes fault detection methods applied to a subset of them. Some of the important points taken from this study are that spalled bearings - one of the defect types investigated later in this thesis - have a moderate to high probability of occuring and are generally more critical when compared to an extensive list of other fault types that broadly include electrical and electronic faults, sensor faults, motor faults and other mechanical faults. Also of relevance to the present thesis is the fact that experiments in [9] are conducted using a Moog MaxForce actuator almost identical to that used in this thesis. The fault detection approach employed however is based on Artifical Neural Networks which is also employed in other fault detection applications as indicated in [27, 28, 29], allowing for the investigation of model-based parameter estimation methods.

Engineers at NASA and Impact Technologies published in 2009 an important paper [2] dealing with data collection and modeling for nominal and fault conditions

on EMAs. The paper primarily investigates techniques for modeling different fault types into a Simulink model of an EMA as well as developing models to accurately describe mechanical and thermal behavior of actuators. However, what is of most importance to the research in this thesis is a section on the development of a flyable data collection test stand. The flyable test stand, which was under development at the NASA Ames Research Center at the time this paper [2] was written (a working prototype and successful testing is reported in a later paper by the same authors [30]), allows aircraft to fly a scaled-down EMA test stand (shown in Fig. 1.5 reproduced from [2]) holding two actuators - one nominal and one injected with a fault. This is

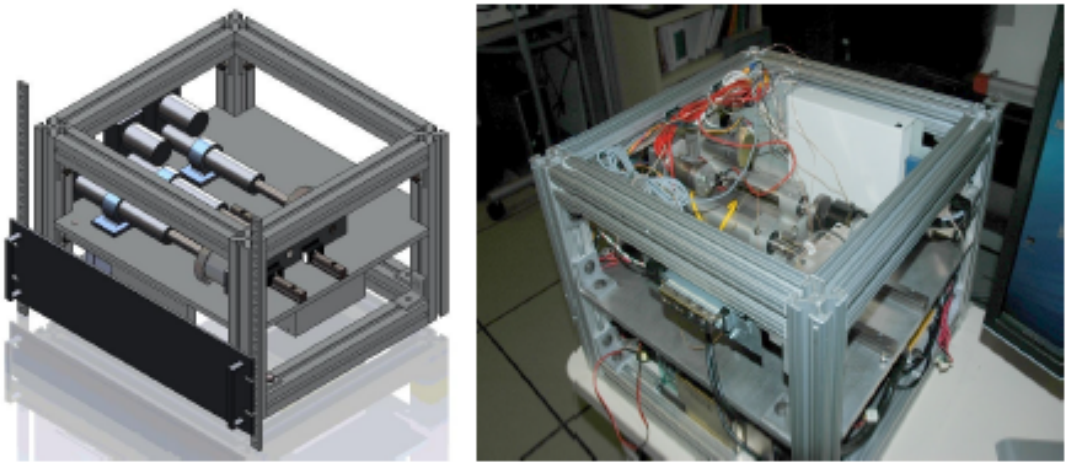

Figure 1.5: Portable EMA test stand [2]

useful to obtain realistic load information, one of the signals required as an input in the model-based approach proposed in this thesis. Also, the work described in this thesis conducts an offline analysis of an EMA's health condition. So, once an EMA is mounted on an aircraft, offloading it for fault testing would be undesirable. The flyable test stand offers at least two solutions: (i) either the test stand could be used to run tests on the EMA while it is on an aircraft that is stationary on the ground, or (ii) the proposed algorithm could be modified slightly to function in real-time in which case the test stand would be used in a manner as described in [2]. This development allows easy implementation of the fault detection approach proposed in this thesis.

Prior to 2009, the lack of a flyable test stand would have been a major drawback in implementing the fault detection approach proposed in this thesis. Other recent publications related to research on flyable test stands include [31, 32], the latter of the two being the most recent and providing results from preliminary investigations into a second-generation test stand.

Since the research in this thesis deals with model-based approaches to fault detection in EMAs, it is pointed out that a number of model-based approaches exist that are applied to electric drives in general and with a small subset applied to EMAs in particular. Amongst the most notable is a 2002 paper by Dixon and Pike [33] that presents a parameter estimation-based approach to EMA condition monitoring. In this paper [33], a linear model of the system is assumed following which parameters are estimated using a simplified refined Instrumental Variables (IV) method as it is found to be more robust when compared to a least-squares or a standard IV approach in the presence of noise of an unknown nature. Therefore, the approach adopted by this method about ten years ago is, so far, almost identical to the approach used in this thesis. However, the differences lie in the following: (i) the approach in [33] uses a discrete-time transfer function model of the system that utilizes a single input. The research in this thesis utilizes a discrete-time state space model to accommodate the use of two inputs as the non-zero external load acting on the EMA cannot be neglected to achieve accurate estimates of the model parameters; (ii) once the parameters are estimated, the authors of this paper [33] transform the discrete model to continuous time and map the model parameters to the physical parameters of the system. To do this, two open-loop transfer function approximations of the single closed loop present in the EMA is used. Additionally, errors are bound to exist due to approximations while mapping the model parameters to physical parameters. In this thesis, the EMA system itself is significantly more complicated as it consists of two loops (an outer position feedback loop and an inner, nested velocity feedback loop). Therefore, the selection of model structure and estimation technique needs to take into account the

presence of both these loops making back calculating physical parameters impractical. Furthermore, due to the presence of two controllers, any fault signatures in motor current are likely heavily compensated. Consequently, an appropriate feature extraction technique is required to extract maximum fault information and a direct analysis of model parameters is not possible, as is done in [33], and lastly (iii) the authors present a fuzzy-logic based fault classification scheme which is different from the Bayesian classification approach pursued in this thesis. Since this [33] is one of the first publications employing parameter estimation for fault detection (coulomb friction faults in particular) in EMAs, it indicates that this area of research is still in its infancy and further research is warranted before effective solutions are developed and validated. The following additional concluding points offered by the authors is worth mentioning: (i) that the results obtained in this paper [33] indicate significant potential in using parameter estimation approaches for fault detection in EMAs and (ii) that the use of discrete-time models over continuous-time ones (as is done in this thesis) is more robust in terms of numerical stability and convergence.

Although advancements in model-based (and amongst them, parameter estimation approaches) since 2002 are reported (refer to [10, 11, 12, 13, 28, 34, 35]), not all of them apply to complex systems such as the EMA dealt with in this thesis. In the few instances that do [10, 35], the approach is too simple with unaddressed issues [10] or accurate but not cost effective enough to be put into operation [35]. As indicated earlier, confidence in the use of EMAs for safety critical applications such as primary flight control on aircraft is yet to be achieved. The present work develops a novel fault detection approach that makes use of techniques similar to certain established ones such as the instrumental variables based parameter estimation scheme first published in [33], principal component analysis or an equivalent pattern recognition approach (similar to those found in [19, 36]) for feature extraction (sometimes used with frequency-based fault detection approaches for dimension reduction [19]) and a Bayesian classifier to generate definite health classes based on the features extracted. The sequence of approaches used and their method of integration is not previously

reported in any of the literature reviewed in this summary. Therefore, based on the review of previous works, the scope for improvement in fault detection in EMAs and the goals sought by Moog Inc. - the research sponsors and providers of the EMA signal data - a formal research problem is constructed.

1.5 The Research Problem: A Formal Definition

Devise a fault detection algorithm for detecting a spalled inner-race bearing defect and a reduction in lubrication in a Moog MaxForce Electromechanical Actuator by utilizing an empirical model-based parameter estimation approach along with an effective feature extraction scheme to extract features that contain sufficient information about the faults despite the closed-loop operation of the EMA, and which are subsequently used to effectively classify the EMA as defective or healthy by implementing a classification algorithm.

Chapter 2

Theory

The general fault detection architecture is shown in Fig. 2.1. After obtaining mea-

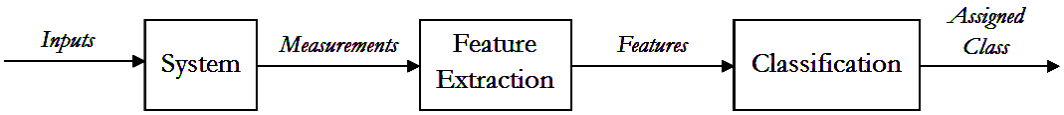

Figure 2.1: General fault detection architecture

surements from the system, feature extraction is performed. As indicated in Chapter 1, there are a number of feature extraction techniques available. Two commonly utilized approaches are frequency-based and model-based. In this work, a model-based approach is used.

Amongst model-based approaches, there are a number of solutions available as presented in Chapter 1. This work utilises two different solution approaches. The first is based directly on system identification techniques and is previously reported in literature for detecting faults in DC motor systems although reports of its use in EMAs in particular is not reported. In this approach, system identification is used to derive a model of the system from available measurements. It is expected that healthy and degraded systems produce different models. Model characteristics are directly utilized as features which are then compared to make a judgement about the health of the system. The approach is one of the simplest model-based approaches available but is later found to be quite inaccurate when applied to complex systems such as EMAs. This is possibly why there are no published results available. Consequently, an alternative approach that couples a similar system identification-like approach

with pattern recognition techniques is investigated. The latter approach is presented later on in the chapter and such an approach is not previously reported to detect inner-race bearing faults and reduced lubrication faults in DC motor electric drives - let alone EMAs in particular. However, since the former approach is also partially used in the latter one, its theory and application to simpler systems is presented. As a start, the Section 2.1 reviews the basic concepts of system identification.

2.1 System Identification

System identification is an approach to modeling that utilizes input and output data recorded from a system to infer a model [3]. For example, consider developing a mathematical model of a DC motor without using any knowledge about the physics of the system. To do this, first assume that a monitored voltage is supplied to the armature and the motor's shaft speed is recorded. The output speed versus time plot shown in Fig. 2.2 shows that the motor displays characteristics of a first-order system.

Selecting a suitable model structure is the first step in the system identification technique [37]. In this example, a first-order discrete-time model structure is chosen which is mathematically represented as shown in Eq. (2.1) with t being a positive integer.

$$y(t) + ay(t-1) = bu(t-1) \tag{2.1}$$

Equation (2.1) is simply a first order difference equation. The equation is represented in discrete-time because data sets are usually obtained by sampling a continuous-time signal and characteristics such as sampling interval are important choices to make in order to achieve the most effective and economical solution [3]. For instance, very fast sampling can lead to numerical problems while slow sampling times may not pick up potentially important high frequency information and even worse, result in signal aliasing if an appropriate anti-aliasing filter is not utilized. In Eq. (2.1), the sampling interval is assumed to be one second for convenience in notation. Advice on how to choose the right sampling time is available in [3]. In general, for the purpose of fault detection, the sampling time should ideally be small enough to capture information

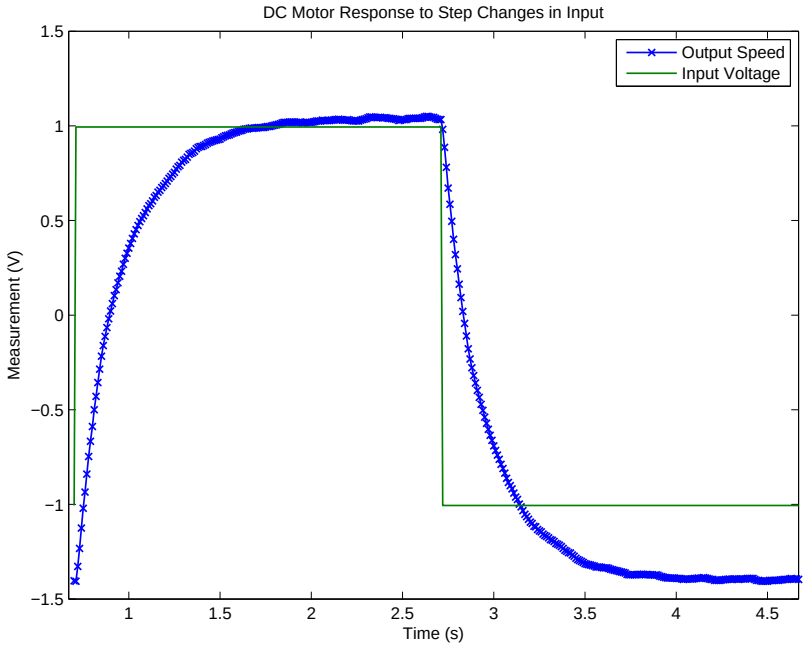

Figure 2.2: Sample response of a DC motor to step changes in input

about the defect.

Having identified the model structure, the next step is estimating the parameters (in this case a and b). There are a number of parameter estimation techniques available. Here, a simple linear least squares estimation (LSE) approach is used. This method is discussed in detail in the following sections. However, to put simply, LSE uses the available data set to determine the values of a and b such that the expression $bu(t-1) - ay(t-1)$ is as close to $y(t)$ as possible. In other words, some norm of the difference between $y(t)$ and $(bu(t-1) - ay(t-1))$ is minimized.

Model validation involves assessing how the model relates to the observed data, any prior knowledge of the system and how it performs the tasks required. Deficient models are rejected and a new model structure is selected. More on model validation is discussed in subsequent sections. A convenient way of describing the steps involved in system identification is by means of a block diagram. Ljung [3] provides a useful representation which is reproduced in Fig. 2.3.

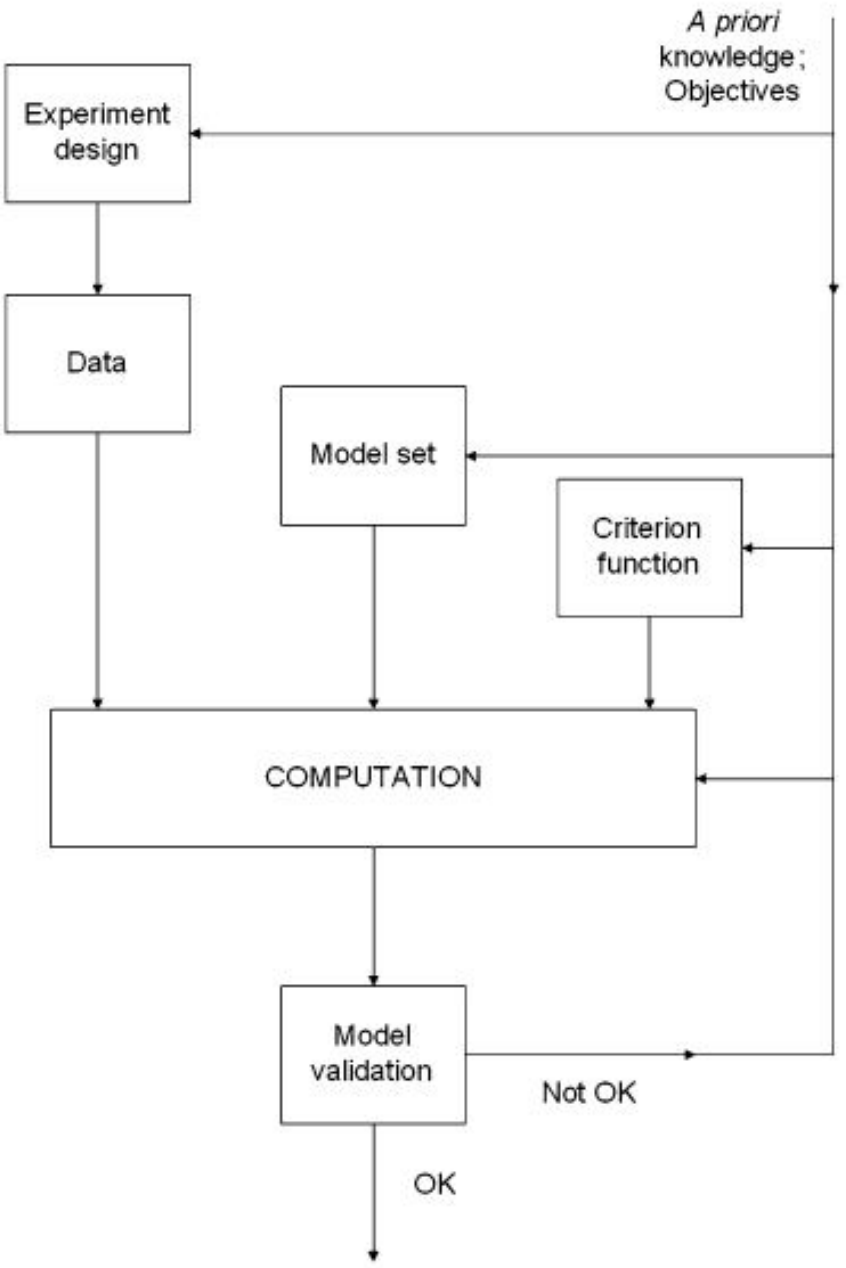

Figure 2.3: System identification block diagram [3]

2.1.1 Linear Time-Invariant Models

Figure 2.3 shows that after collecting data from a system, the first step in the system identification procedure is to determine a class of models (or model set) from which the most suitable model for the intended purpose is selected. This section discusses a class meant for linear time-invariant systems.

Linear time-invariant systems form the most important class of dynamical systems both in theory and practice despite the fact that they often represent idealizations of real-life processes [3]. A system is time-invariant if its response to a particular input signal does not vary with time. It is linear if the output to some linear combination of inputs is the same as a linear combination of its outputs to each individual input. Such a system is completely characterized by its impulse response when its output at a particular instant of time depends only on the inputs up to that instant. They are mathematically represented as shown in Eq. (2.2) or by parameterizing the coefficients into rational functions as subsequently discussed.

$$y(t) = G(q, \theta)u(t) + H(q, \theta)e(t) \tag{2.2}$$

$e(t)$ is white noise having a probability density function (pdf) $f_e(x, \theta)$ that depends on θ - the parameter vector (see Eq. (2.4)). It is common to assume the pdf as Gaussian [3].

Equation Error Model

One form of black-box model - a parameterized model derived empirically without using physical insight - has an equation error form and is a simple input-output relationship obtained by describing a linear difference equation. This model form, also known as an Auto-Regressive with Exogeneous Input (ARX) model (although it is noted here that other models such as the Auto-Regressive Moving-Average with Exogeneous Input (ARMAX) model presented later also fall under the class of equation

error models), was shown earlier in Eq. (2.1) and is shown here more generally.

$$\begin{aligned} y(t) &+ a_1 y(t-1) + \cdots + a_{n_a} y(t-n_a) \\ &= b_1 u(t-1) + \cdots + b_{n_b} u(t-n_b) + e(t) \end{aligned} \tag{2.3}$$

The parameters in this case are

$$\theta = [\, a_1 \quad a_2 \ \ldots \ a_{n_a} \quad b_1 \quad b_2 \ \ldots \ b_{n_b} \,]^T \tag{2.4}$$

where $(n_a + 1)$ and n_b are the number of output and input terms respectively in Eq. (2.3) and are often referred to as output and input delays respectively [3]. If

$$A(q) = 1 + a_1 q^{-1} + \cdots + a_{n_a} q^{-n_a}$$

and

$$B(q) = b_1 q^{-1} + \cdots + b_{n_b} q^{-n_b}$$

where q^{-1} denotes the backward shift operator and is used for notational convenience where $q^{-1}u(t) = u(t-1)$ and so on, then Eq. (2.3) is written in the form of Eq. (2.2) with

$$G(q,\theta) = \frac{B(q)}{A(q)}, \quad H(q,\theta) = \frac{1}{A(q)} \tag{2.5}$$

The ARX model is conveniently represented in block diagram form as shown in Fig. 2.4. $A(q)y(t)$ forms the autoregressive part and $B(q)u(t)$ represents the exogeneous inputs.

The main advantages of using an ARX model are that it's simple and the predictor form defines a linear regression allowing for prediction of model parameters using simple estimation techniques such as the least squares estimation (LSE) method. The drawback with this model is that it generally does not represent the physics of the system well. This is because the noise is fed through the denominator dynamics of the system before being added to the process output (see Fig. 2.4) while in most real systems, noise affects the system at the output (see Fig. 2.7) [3, 37, 4]. This also means that if noise enters the system primarily at the input, then the ARX model

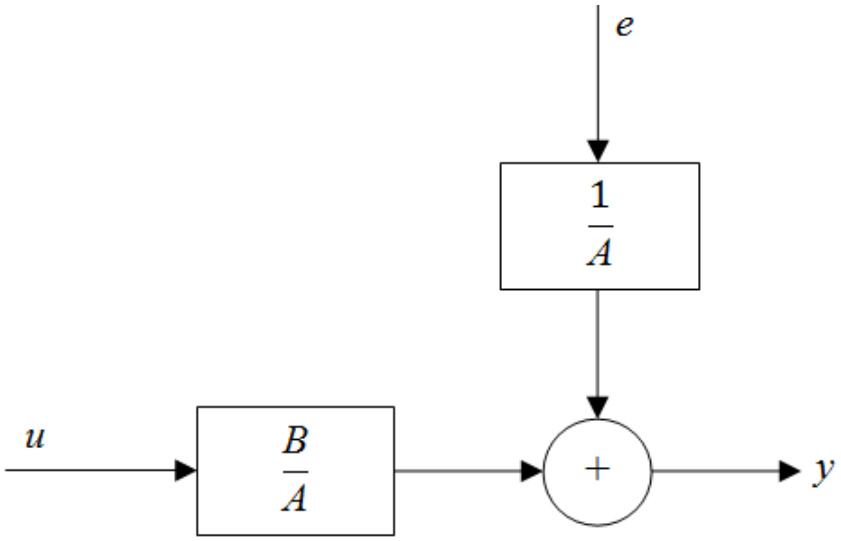

Figure 2.4: ARX model block diagram

structure will perform well.

ARMAX Model

Another disadvantage with using the ARX model is that there isn't any freedom in describing the disturbance term. The ARMAX model attempts to rectify this by allowing some flexibility by describing the equation error as a moving average of white noise [3]. This indirectly allows better prediction of the physics of the system. The model is described mathematically as shown in Eq. (2.6).

$$\begin{aligned} y(t) \quad &+ \quad a_1 y(t-1) + \cdots + a_{n_a} y(t-n_a) = b_1 u(t-1) + \ldots \\ &+ \quad b_{n_b} u(t-n_b) + e(t) + c_1 e(t-1) + \cdots + c_{n_c} e(t-n_c) \end{aligned} \tag{2.6}$$

In comparison to the ARX model (Eq. (2.3)), there are added terms to the right of the disturbance $e(t)$. In the absence of these terms, the minimization of $e(t)$ in the ARX model means that only $A(q)$ and $B(q)$ are permitted to change. This means that the characteristics of the white noise disturbance is minimized and these two parameters are absorbing these characteristics. That is why the physics of the system is often not well approximated due to corruption by the white noise characteristics.

In the case of ARMAX however, the disturbance at each previous time interval is also a part of the terms being minimized. Each such term $e(t - n_c)$ has a coefficient c_{n_c} that can change. Consequently, additional characteristics of the error signal are absorbed into these terms. However, this still doesn't solve the issue that the noise is fed through the denominator dynamics. The only improvement is that more information about the noise is used to develop the noise model and this may or may not offer improved results. Again, in the unlikely scenario that the noise enters the actual system at the input, the ARMAX model structure is more likely to develop a more accurate noise model $H(q)$ compared to the ARX model, indirectly causing an improvement in the system model $G(q)$. The moving average part of the model is presented by the term $C(q)e(t)$ with $C(q)$ being

$$C(q) = 1 + c_1 q^{-1} + \cdots + c_{n_c} q^{-n_c}$$

Eq. (2.6) is then rewritten as

$$A(q)y(t) = B(q)u(t) + C(q)e(t) \tag{2.7}$$

This corresponds to Eq. (2.2) with

$$G(q,\theta) = \frac{B(q)}{A(q)}, \quad H(q,\theta) = \frac{C(q)}{A(q)} \tag{2.8}$$

and where

$$\theta = [\, a_1 \quad a_2 \ldots a_{n_a} \quad b_1 \quad b_2 \ldots b_{n_b} \quad c_1 \quad c_2 \ldots c_{n_c} \,]^T \tag{2.9}$$

The ARMAX model is represented in block diagram form as shown in Fig. 2.5. It is perhaps the second most popular model after the ARX [4]. Like the ARX, it belongs to the class of equation error models because the noise filter contains the denominator dynamics of the model but provides added flexibility over the ARX in dealing with the disturbance terms. Although this makes the model nonlinear in its parameters, sufficiently efficient multi-stage least squares algorithms exist that help in estimating the parameters. Other nonlinear optimization and recursive algorithms

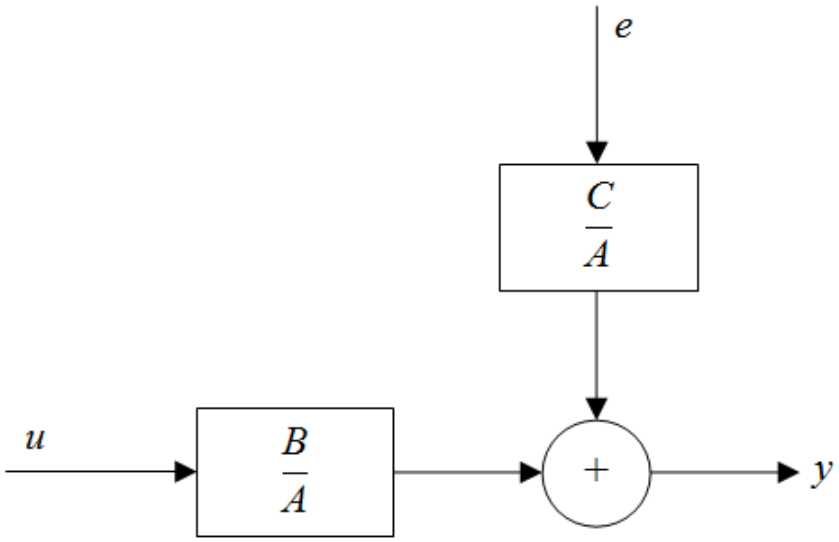

Figure 2.5: ARMAX model block diagram

also exist [3, 37, 4].

Output Error Model

The primary concern with the equation error model form is that both the transfer functions $G(q,\theta)$ and $H(q,\theta)$ have $A(q)$ as the common denominator. From a physical aspect, this is not desirable as it tends to model the disturbance as part of the physical system with ARMAX only offering some flexibility in forming $H(q,\theta)$. The output error (OE) model attempts to rectify this by parameterizing the transfer functions independently. The output error is best described mathematically as shown in Eq. (2.10) [3].

$$\begin{aligned} w(t) &+ f_1 w(t-1) + \cdots + f_{n_f} w(t-n_f) \\ &= b_1 u(t-1) + \cdots + b_{n_b} u(t-n_b) \\ y(t) &= w(t) + e(t) \end{aligned} \tag{2.10}$$

with

$$F(q) = 1 + f_1 q^{-1} + \cdots + f_{n_f} q^{-n_f}$$

The model is then written as

$$y(t) = \frac{B(q)}{F(q)}u(t) + e(t) \tag{2.11}$$

and is represented in block diagram form as shown in Fig. 2.6. The parameter vector

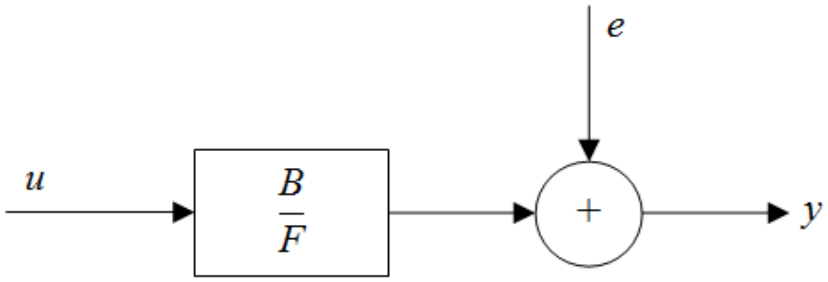

Figure 2.6: OE model block diagram

is determined as

$$\theta = [\, b_1 \quad b_2 \ \ldots \ b_{n_b} \quad f_1 \quad f_2 \ \ldots \ f_{n_f} \,]^T \tag{2.12}$$

$w(t)$ is derived from u using Eq. (2.10).

2.1.2 Parameter Estimation Methods

A parameter estimation method is a way of estimating values for model parameters (such as a, b, c and f from the previous section) using available experimental data. It is desired that the best possible method be chosen so that the model of the system is "good".

A "good" dynamic model is one that satisfactorily predicts the dynamics of the actual system. A "good" prediction occurs when the difference between the model's predicted output for a given input is as close as possible, if not exactly the same, as the observed output. That is the same as desiring a "small" prediction error. "Small" is perceived in many ways and various ways of acheiving a "small" prediction error are described. One way is to define a scalar-valued norm that measures the size of the prediction error. For example, if a least-squares approach is utilized for prediction, the norm is quadratic (i.e. the square of the prediction error). An attempt to

minimize this norm to zero is made, allowing the parameters to vary and therefore be estimated in the process. Another approach is to require that the prediction error be uncorrelated with the given data set - a method introduced later under the name of Instrumental Variables (IV). This method requires that some projection of the prediction error (the "instrumental variable" is what causes the projection) be minimized.

There are a number of other techniques for parameter estimation that are presented in [3, 37, 4]. The next section discusses the estimation of parameters for the ARX model structure.

Estimating Parameters for the ARX Model

It was shown earlier that the parameter vector θ for the ARX model is given by Eq. (2.4). Introduce the regression vector $\phi(t)$ as shown in Eq. (2.13).

$$\phi(t) = [\, -y(t-1) \ \ldots \ -y(t-n_a) \quad u(t-1) \ \ldots \ u(t-n_b) \,]^T \tag{2.13}$$

Thus Eq. (2.3) is rewritten as

$$\hat{y}(t|\theta) = \phi^T(t)\theta \tag{2.14}$$

where $\hat{y}(t|\theta) = y(t) - e(t)$. Since $\hat{y}(t|\theta)$ is a function of θ, this is better written as $\hat{y}(t|\theta) = y(t) - e(t, \theta)$.

The Least Squares Criterion

Define a scalar-valued norm or criterion function as shown in Eq. (2.15).

$$V(\theta) = \frac{1}{N}\sum_{t=1}^{N} \frac{1}{2}\big[\, y(t) - \phi^T(t)\theta \,\big]^2 \tag{2.15}$$

where N is the length of the data set available. This norm is minimized to zero and is the square of the prediction error term. Consequently, this criterion or norm is known as the *Least Squares Criterion*. Since it is quadratic in θ, the minimization is carried out analytically by setting the first derivative of $V(\theta)$ with respect to θ equal to zero and solving for the unknown parameter vector. Mathematically, this is represented

as follows [3]:

$$\begin{aligned} \hat{\theta} &= \arg\min_{\theta} V(\theta) \\ \Rightarrow 0 &= \tfrac{d}{d\theta}V(\theta) = \tfrac{2}{N}\sum_{t=1}^{N}\phi(t)(y(t)-\phi^T(t)\theta) \end{aligned}$$

which gives

$$\sum_{t=1}^{N}\phi(t)y(t) = \sum_{t=1}^{N}\phi(t)\phi^T(t)\theta \tag{2.16}$$

or

$$\hat{\theta} = \left[\sum_{t=1}^{N}\phi(t)\phi^T(t)\right]^{-1}\sum_{t=1}^{N}\phi(t)y(t) \tag{2.17}$$

The parameters $\hat{\theta}$ are now easily calculated using software packages such as MATLAB. The covariance matrix of $\hat{\theta}$ is calculated as

$$C_\theta = \left[\sum_{t=1}^{N}\phi(t)\phi^T(t)\right]^{-1}\sigma_n^2$$

where σ_n^2 is the variance of the noise and is estimated as

$$\hat{\sigma_n}^2 = \frac{1}{N-n_a-n_b}\sum_{i=1}^{N}\hat{e}_i^2 \tag{2.18}$$

where $\hat{e}_i$ are the residuals obtained by subtracting the simulated/model output from the actual/measured output. The variance of each of the parameter estimates is the corresponding value of the covariance term along the principal diagonal of

$$\hat{C}_\theta = \left[\sum_{t=1}^{N}\phi(t)\phi^T(t)\right]^{-1}\hat{\sigma}_n^2$$

A confidence interval for each $\hat{\theta}$ is then obtained by choosing a confidence level c used to represent the intervals as follows

$$\hat{\theta_j} \pm c\sqrt{\hat{\text{var}}(\hat{\theta_j})} \;\forall\; j = 1, 2, \ldots, (n_a+n_b) \tag{2.19}$$

The solution presented in Eq. (2.17) is conveniently represented in matrix form as shown in the following equations, provided the number of input and output data terms used in the regression vector are the same (i.e. $n_a = n_b = p$). The matrix form is more readily adopted into MATLAB.

Define the matrix X as shown in Eq. (2.20).

$$X = \begin{bmatrix} -y(p) & -y(p+1) & \dots & -y(N-1) \\ \vdots & \vdots & \vdots & \vdots \\ -y(1) & -y(2) & \dots & -y(N-p) \\ u(p) & u(p+1) & \dots & u(N-1) \\ \vdots & \vdots & \vdots & \vdots \\ u(1) & u(2) & \dots & u(N-p) \end{bmatrix}^T \tag{2.20}$$

The parameter vector $\hat{\theta}$ is estimated as follows:

$$\hat{\theta} = \left[X^T X\right]^{-1} X^T Y \tag{2.21}$$

where Y is defined as

$$Y = [\, y(p+1) \quad y(p+2) \ \dots \ y(N) \,]^T$$

It is noted here that in order to successfully estimate $\hat{\theta}$, the matrix $X^T X$ has to be invertible. This situation is more likely if the input data $u(t)$ is sufficiently and persistently exciting [4]. The MATLAB script file demonstrating this simple technique of estimating the parameter vector is shown in Appendix B. The script file that utilizes a direct application of Eq. (2.17) is also provided for completeness. A more convenient way of extracting a solution is by utilizing the built-in System Identification Toolbox [38] that offers the benefits of generality and ease-of-use at the same time.

The main drawbacks of utilizing this prediction approach for the ARX model are bias and consistency. A bias occurs when the parameters systematically deviate

from their optimal values (i.e. they are either over- or under-estimated) [4]. Non-consistency occurs when the bias fails to approach zero even when the number of data samples N approaches infinity [4]. These two characteristics occur in this case primarily because of the nature of the ARX model structure as well as the least squares estimate. As mentioned earlier, the ARX model does a poor job of dealing with noise because it uses an unrealistic noise model of $1/A(q)$. Instead, a model with additive output noise is more realistic. This difference between a real process and an ARX model is best described in the form of the block diagram shown in Fig. 2.7 [4]. This is the cause of bias.

The lack of consistency is due to the characteristics of the least squares esti-

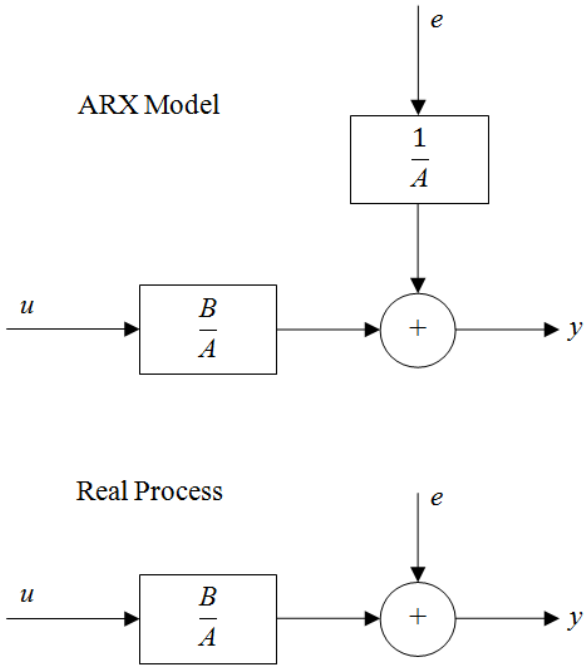

Figure 2.7: ARX model versus a real process [4]

mation approach. To get around these problems, an output error model structure is chosen or a different approach to parameter estimation is selected. The next section deals with the consistency issue by introducing a new estimation approach known an the Instrumental Variables (IV) method.

Instrumental Variables Method

The Instrumental Variables method offers a simple remedy to the consistency problem discussed earlier. Any given system (even one that is fully understood) is such that the output data at any given time contains some disturbance information. Typically, it is additive as shown in Fig. 2.7. In order for the LSE to be consistent (i.e. for $\hat{\theta}$ to converge to some finite value (ideally θ_0 which is considered a "true value" of the parameter vector as N tends to infinity [3])), it is required that the average of the expected values of $\phi(t)\phi^T(t)$ over the entire data set be non-singular (i.e. the bracketed term in Eq. (2.17) must be invertible). It is also required that the time average of the expected values (since $y(t)$ isn't deterministic) of the term outside the brackets in Eq. (2.17) over the entire data set be equal to zero. This is possible only if the disturbance is a strong white noise process having zero mean and some variance because then the disturbance is uncorrelated, making the required expected value zero; or if $\phi(t)$ is not auto-regressive because previous output data contains previous disturbance information which, if not strongly white, will be correlated to the disturbance signal at the present time instant thus making the required expected value possibly non-zero. More on consistency is available in [3, 4].

The easiest way to implement the IV method is to replace the regression matrix X in Eq. (2.21) by a new matrix Z that has the same dimensions as X [4]. If $Z^T X$ is non-singular (which again is controlled by choosing appropriate input data and making appropriate changes in Z), the estimate $\hat{\theta}$ using the IV method is given by

$$\hat{\theta} = \left[Z^T X\right]^{-1} Z^T Y \tag{2.22}$$

Since Z is chosen to be uncorrelated with the noise, the estimate is consistent. The choice of Z is also such that the parameter variance is as small as possible so that $\hat{\theta}$ converges as close to θ_0 as possible. The best choice to ensure zero variance is to choose the regression matrix X itself to occupy Z. However, this reverts to the LSE method and brings back the problem of consistency. Since the presence of the output

terms in the regressor matrix is the cause of the consistency problem, the easiest solution to choosing Z is to equate it to an undisturbed version of the regression matrix X. The following algorithm is proposed in view of this [4] and is used in the rest of this thesis.

1. Estimate an ARX model from the data using Eq. (2.21).
2. Simulate the undisturbed model with the estimated parameters to obtain the undisturbed simulated outputs. That is, obtain $y_u(t)$ such that

$$y_u(t) = \frac{B(q)}{A(q)}u(t)$$

   where $B(q)$ and $A(q)$ are estimated.
3. Construct the instrumental variable matrix Z to resemble X in Eq. (2.20) except replace all the output terms $y(t)$ with each of the corresponding terms $y_u(t)$.
4. Estimate the new parameters $A(q)$ and $B(q)$ using Eq. (2.22) where Y consists of the original output data and not the simulated data.

By adopting the above algorithm, it is assumed that the simulated outputs are relatively close to the actual process outputs and the effect of the bias due to the initial ARX estimate is small enough to be neglected. However, this may not always be true. In such cases, the IV method is improved by repeating steps 2 to 4 by using the previous IV estimate for every new iteration. Convergence is usually very fast and there is not much additional gain by running the algorithm more than thrice [4]. A more advanced algorithm is presented in [3], however its benefits over the above algorithm is not always assured. A sample MATLAB script file implementing this method is available in Appendix B.

Estimating Parameters for the ARMAX Model

One of the differences cited earlier between the ARMAX model when compared with the ARX model is that the former is nonlinear in its parameters. Consequently,

the estimation of these parameters either needs to utilize nonlinear estimation techniques or efficient multi-stage linear estimation ones. The latter of the two techniques is presented because it is more conveniently implemented and offers comparable performance compared to the nonlinear approach. In fact, according to [4], the nonlinear approach demands far greater computational power and still takes longer to converge on the optimal solution.Therefore, the following algorithm is presented for estimating the parameters of the ARMAX model [4].

1. Estimate an ARX model from the data using Eq. (2.21).
2. Calculate the prediction errors of this ARX model using the following equation.

$$e(t) = A(q)y(t) - B(q)u(t)$$

   where $A(q)$ and $B(q)$ are estimated in step 1.

3. Estimate the ARMAX model parameters a_i, b_i and c_i using the LSE method on the following difference equation

$$\begin{aligned} e(t) &= a_1 y(t-1) + \cdots + a_m y(t-m) \\ &- b_1 u(t-1) - \cdots - b_m u(t-m) \\ &- c_1 e(t-1) - \cdots - c_m e(t-m) \end{aligned}$$

   where the terms $e(t-i)$ are initially obtained from the ARX estimate.

Steps 2 to 3 are then repeated using the most recent estimates of the ARMAX parameters with each subsequent calculation of the parameter error. While programming this algorithm in a software such as MATLAB the initial values of the parameter error terms are assumed zero. This is a valid assumption according to [3]. A sample MATLAB script file implementing this method is available in Appendix B.

Estimating Parameters for the OE Model

The Output Error (OE) model structure is such that the noise model does not include the process denominator dynamics making the model more realistic at the

expense of it becoming nonlinear and consequently making the parameters harder to estimate. Like the ARMAX model, the OE model parameters are estimated using either a nonlinear optimization approach or by repeated application of a linear estimation technique. The following steps present the linear approach [4].

1. Estimate an ARX model from the data using Eq. (2.21).
2. Filter the input data $u(t)$ and the output data $y(t)$ through a filter $F(q)$ as follows:
$$u^F(t) = \frac{1}{F(q)}u(t) \quad \text{and}\, y^F(t) = \frac{1}{F(q)}y(t) \tag{2.23}$$
where $F(q) = A(q)$ for the first iteration only.
3. Estimate the OE model parameters f_i and b_i by an ARX model estimate using the filtered versions of the input and output data obtained in step 2.

Steps 2 to 3 are repeated until convergence is reached while for all future iterations, the filter consists of the estimated f_i parameters of step 3. A sample MATLAB script file implementing this method is available in Appendix B.

The next section presents an example showing the application of the above techniques to detect faults seeded in a DC motor system through both simulations and experiments. An ARX model structure is selected and the method of Instrumental Variables is used for parameter estimation. These choices are made because the systems considered in the example are well explained using a first order model and implementing an ARX model structure is more convenient compared to the other model structures presented earlier. The IV method is used to offset the consistency issues that occur when using a least squares estimation approach.

2.2 Example: Fault Detection in DC Motor Drives

This example shows the application of a fault detection approach based solely on the system identification concepts presented in Section 2.1. Once the measurement data is obtained, the features used for classification (refer to Fig. 2.1) are the estimated

parameters of the selected model structure. As indicated in the preceding section, a first order ARX model structure is selected and the method of Instrumental Variables is used for parameter estimation. The final "classification" assumes that the estimated parameters are normally distributed and a feature plot is generated and analyzed. The example utilizes a simulation of a PM DC motor developed from a physics-based model as well as experiments conducted with an LJ MS15 DC motor control module manufactured by LJ Technical Systems Inc., Holtsville, NY and provided for use by Dr. Mark Kempski of the Department of Mechanical Engineering at the Rochester Institute of Technology, Rochester, NY. Basic information about the control module is presented while detailed information can be found in the module's user manual [5]. The simulated PM DC motor is used due to the relative simplicity in seeding the system with single-point defects having specific characteristics. Furthermore, the purpose of using both a simulation and a real system is to compare the proposed algorithm's performance using real experimental data as opposed to simulated data which may not accurately represent real life situations.

2.2.1 DC Motor Control Module

The module consists of two DC motors - one of which behaves as a tachogenerator, a continuous rotation potentiometer (this component is not used in the experiments conducted), a gray-coded disc and a slotted disc both of which are used to generate information about the shaft speed, a digital tachometer which gives a continuous 3-digit display of the output shaft speed and an eddy current brake. The digital tachometer is used to calibrate the motor module. These components are marked in the schematic of the motor module shown in Fig. 2.8.

The DC motor control module used is shown in Fig. 2.9. The setup is such that the drive motor - a coreless DC motor - receives an input voltage signal from a signal generator. In this work, the two types of input signals generated are a square wave of amplitude 1 V (2 V peak-to-peak) and frequency 250 mHz, and a sine wave having the same characteristics. The drive motor rotates a shaft that supports the

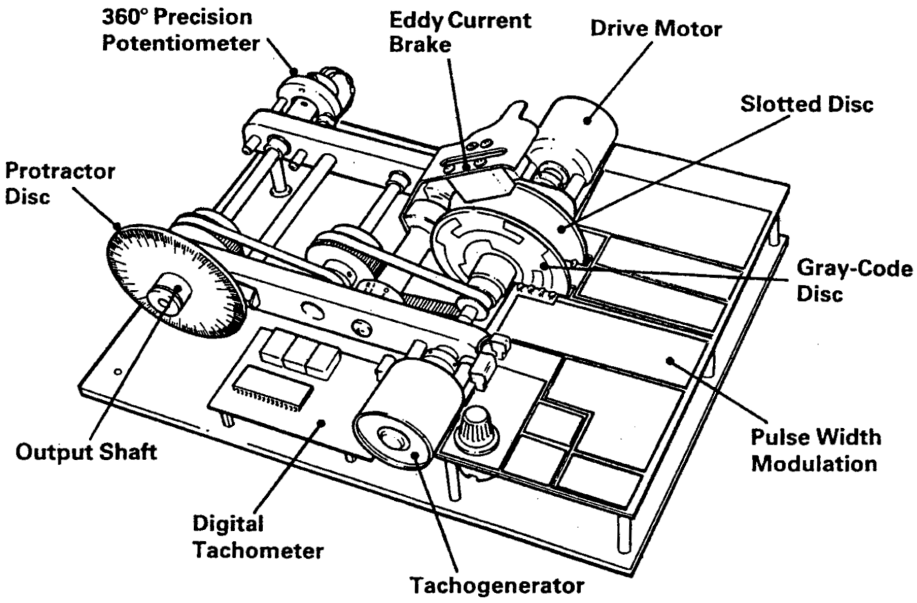

Figure 2.8: Schematic of the DC Motor Control Module [5]

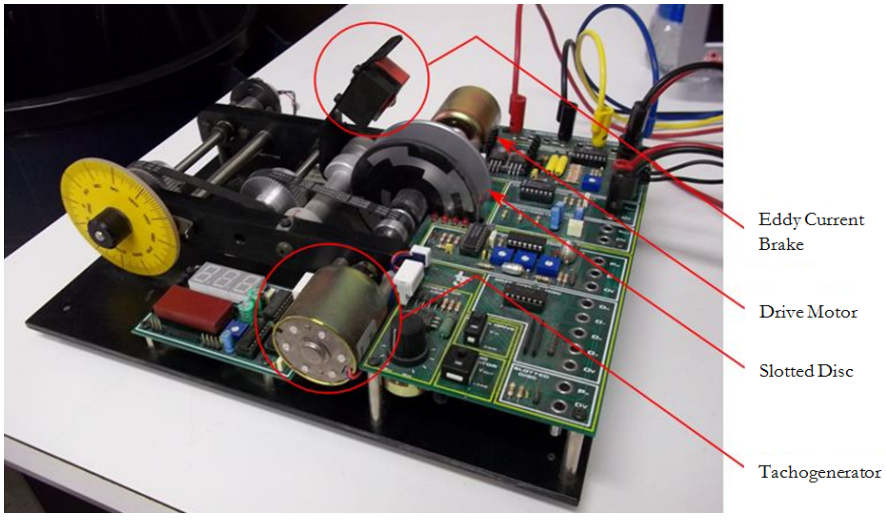

Figure 2.9: DC Motor Control Module

slotted disc and tachogenerator amongst other devices. The tachogenerator generates a voltage signal that corresponds with the speed of rotation of the shaft. This signal is captured using a data acquisition system and serves as the output data signal for the fault detection process. An eddy current brake is used to dampen the system response by adjusting the amount by which it covers the slotted disc. The brake functions as a viscous damper and its application results in a damped system response [39]. This is verified by plotting the step response at the different brake positions as shown in Fig. 2.10.

The application of the eddy current brake is taken to represent a generalized

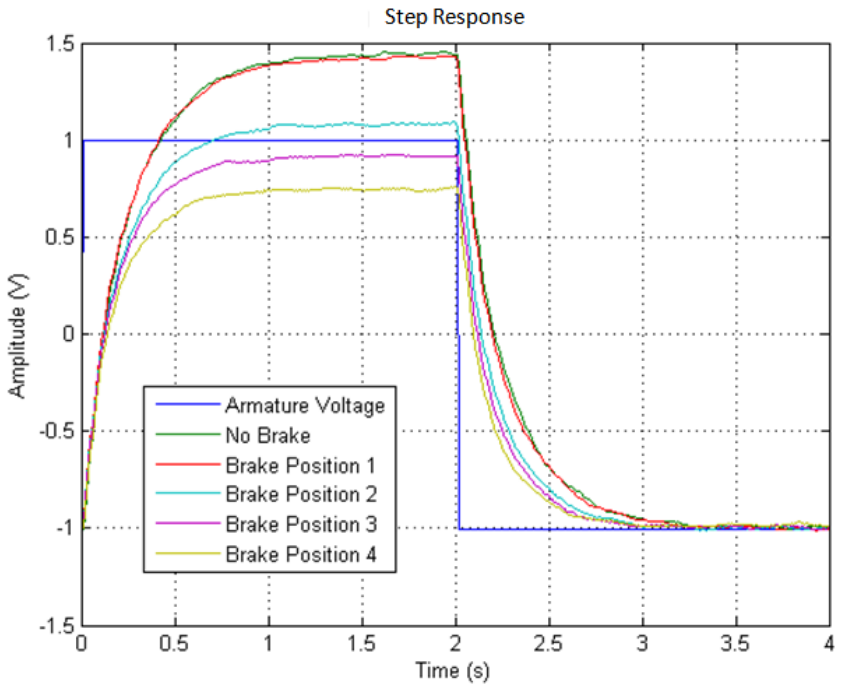

Figure 2.10: Motor Module step response characteristics

roughness type defect (the same effect as the reduced lubrication defect in the EMA). Seeding the control module with a single-point defect is also attempted. The seeding is done by attaching a protrusion to the slotted disc. This protrusion periodically comes in contact with a stopper that momentarily slows down the speed of rotation of the disc. The fault is therefore periodic with respect to the angular position of the

disc. The response of the defected system to a sine wave input is plotted in Fig. 2.11. It is important to mention that despite the presence of a nonlinearity in the motor response on account of the dead-zone about zero velocity (notice in Fig. 2.11 that the peak amplitudes are not equal and that the 0 V amplitude does not occur at $t = 2$ seconds), the chosen model structure for system identification is assumed linear as it is expected that this assumption does not affect comparisons between models derived from healthy and unhealthy data.

2.2.2 Permanent Magnet DC Motor Model

The primary limitation of using the DC motor control module is the lack of flexibility in seeding the system with single-point defects. Therefore, in order to show the performance of the developed methods in detecting such faults, a data set is collected by running a simulation of a linear model of a DC motor derived from physical laws and seeded with a single-point defect. Details of the derivation of the DC motor model and the seeding of the defect follows.

The DC motor system is modeled as shown in Fig. 2.12. The operation of a DC motor is described mathematically as follows:

$$v_m = K_e\omega_m + i_m R + L\frac{di_m}{dt} \tag{2.24}$$

$$K_t i_m - T_{ex} = B\omega_m + J\frac{d\omega_m}{dt} \tag{2.25}$$

where v_m is armature voltage, i_m is armature current, ω_m is armature angular speed, T_{ex} is external torque load applied to the shaft, L is armature inductance, $J = J_m + J_L$ is the net system inertia (with contributions from the motor and the load), $B = B_m + B_L$ is the net viscous friction coefficient (again with contributions from the motor and the load), K_e is the back-emf constant and K_t is the electromagnetic torque constant. It is assumed that the shaft is massless, rigid and undamped so that the load viscous damping coefficient and inertia can be algebraically added to the corresponding motor values [40]. Assuming zero initial conditions for ω_m and i_m and

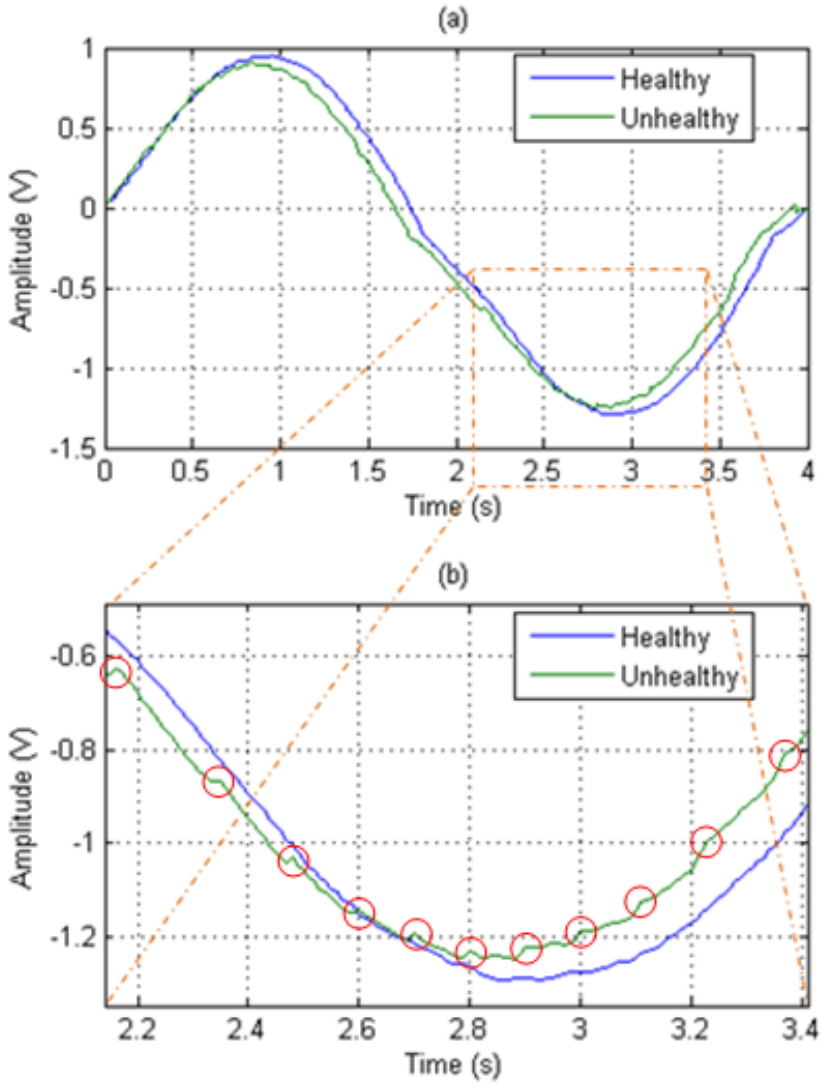

Figure 2.11: Response to a sine wave input with seeded single-point defect - (a) Response over one time period; (b) Zoomed-in version showing single-point defects encircled

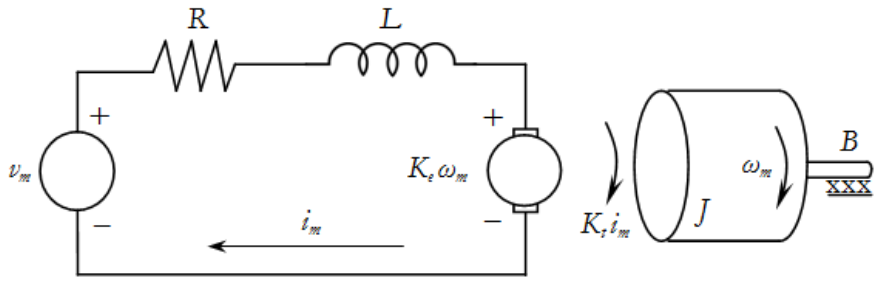

Figure 2.12: Electromechanical system with DC motor

taking the Laplace transform of Eqs. (2.24) and (2.25) gives

$$\begin{aligned} V_m &= K_e W_m + I_m R + L I_m s \\ K_t I_m - T_{ex} &= B W_m + J W_m s \end{aligned} \tag{2.26}$$

Continuous-Time State-Space Representation

For the two-input (v_m and T_{ex}), two-output (ω_m and i_m) system described, the following choices are made: state variables $i_m = x_1$ and $\omega_m = x_2$; input variables $v_m = u_1$ and $T_{ex} = u_2$; and output variables $i_m = x_1 = y_1$ and $\omega_m = x_2 = y_2$. Then,

$$\begin{aligned} L\dot{x}_1 + Rx_1 + K_e x_2 &= u_1 \\ J\dot{x}_2 - K_t x_1 + Bx_2 &= -u_2 \end{aligned} \tag{2.27}$$

Therefore in matrix notation, the state-space equations in continuous time are

$$\begin{aligned} \dot{\mathbf{x}} &= \begin{bmatrix} -\frac{R}{L} & -\frac{K_e}{L} \\ \frac{K_t}{J} & -\frac{B}{J} \end{bmatrix} \mathbf{x} + \begin{bmatrix} \frac{1}{L} & 0 \\ 0 & -\frac{1}{J} \end{bmatrix} \mathbf{u} \\ \mathbf{y} &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \mathbf{u} \end{aligned} \tag{2.28}$$

with the bold face letters denoting vectors. Eq. (2.28) is then written as follows:

$$\begin{aligned} \dot{\mathbf{x}} &= A\mathbf{x} + E\mathbf{u} \\ \mathbf{y} &= C\mathbf{x} + D\mathbf{u} \end{aligned} \tag{2.29}$$

The next section involves deriving the equivalent discrete-time version of the state-space representation. The matrices are referred to by the labels provided in Eq. (2.29). It is necessary to obtain a discrete-time form of the model to allow convenient control of sampling interval because the latter plays an important role in the parameter estimation process [3, 4]. The importance of the sampling interval also becomes evident during the next derivation.

Discrete-Time State-Space Representation

Taking the Laplace transform of Eq. (2.29) gives

$$X(s) = (sI - A)^{-1}(EU(s) + x(t_0)) \tag{2.30}$$

The inverse Laplace transform of Eq. (2.30) gives back the system in the time-domain where $\mathcal{L}^{-1}((sI - A)^{-1})$ is the state transition matrix Φ. The state transition matrix, when multiplied with the initial condition of the state variable (in this case $x(t_0)$) gives the value of the state variable at a later time t. For a time-invariant system $\Phi(t, t_0) = e^{A(t-t_0)}$. Although this work deals only with such systems, the rest of the presentation remains general.

$$x(t) = \mathcal{L}^{-1}(X(s)) = \Phi(t - t_0)x(t_0) + \int_{t_0}^{t} \Phi(t - \tau)Eu(\tau)\, d\tau \tag{2.31}$$

Applying a zero-order hold on the input and setting $t_0 = kT$ gives,

$$u(t) = u(kT), \quad kT \leq t \leq (k + 1)T$$

where a sample interval of T is assumed. So,

$$x(t) = \Phi(t, kT)x(kT) + \left(\int_{kT}^{t} \Phi(t, \tau)E\, d\tau\right)u(kT) \tag{2.32}$$

Define a new function Γ as follows:

$$\Gamma(t, kT) = \int_{kT}^{t} \Phi(t, \tau)E\, d\tau \tag{2.33}$$

Inserting Eq. (2.33) into Eq. (2.32) and setting $t = (k+1)T$ gives

$$x((k+1)T) = \Phi((k+1)T, kT)x(kT) + \Gamma((k+1)T, kT)u(kT) \tag{2.34}$$

Therefore, the required discrete-time state-space representation is

$$\begin{aligned} x((k+1)T) &= \Phi((k+1)T, kT)x(kT) + \Gamma((k+1)T, kT)u(kT) \\ y(kT) &= Cx(kT) + Du(kT) \end{aligned} \tag{2.35}$$

where Φ is the state transition matrix and Γ is as defined in Eq. (2.33). For a time-invariant system, Eq. (2.35) is significantly simplified as shown below.

Assume the final discrete-time state space equations for the time-invariant system is

$$\begin{aligned} x((k+1)T) &= G(T)x(kT) + H(T)u(kT) \\ y(kT) &= Cx(kT) + Du(kT) \end{aligned} \tag{2.36}$$

the following transformations apply:

$$G(T) = e^{AT} \tag{2.37}$$

$$\begin{aligned} H(T) &= \left(\int_{0}^{T} e^{A\lambda} d\lambda\right)E \\ &\approx (e^{AT} - I)A^{-1}E \end{aligned} \tag{2.38}$$

provided A is non-singular. Additionally,

$$\begin{aligned} e^{AT} &= I + AT + \frac{A^2T^2}{2!} + \dots \\ &\approx I + AT \quad \forall \quad T \ll 1 \end{aligned} \tag{2.39}$$

leading to the first constraint on the sampling interval. Furthermore

$$H(T) \approx (e^{AT} - I)A^{-1}E \quad \textit{iff.} \quad \det(A) \neq 0$$

where

$$\det(A) = \frac{RB + K_eK_t}{LJ}$$

$\therefore H(T)$ exists when $RB \neq -K_eK_t$ and this is true in the present case.
The discrete-time state space representation is then simplified to the following:

$$\begin{aligned} x((k+1)T) &= \begin{bmatrix} 1 - \frac{RT}{L} & -\frac{K_eT}{L} \\ \frac{K_tT}{J} & 1 - \frac{BT}{J} \end{bmatrix} x(kT) + \begin{bmatrix} \frac{T}{L} & 0 \\ 0 & -\frac{T}{J} \end{bmatrix} u(kT) \\ y(kT) &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} x(kT) + \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} u(kT) \end{aligned} \tag{2.40}$$

Representing Eq. (2.40) completely in terms of the physical parameters and simplifying algebraically results in

$$i((k+1)T) = a_0 i(kT) + a_1 i((k-1)T) + b_1 v(kT) + c_1 \omega((k-1)T) + d_1 T_{ex}((k-1)T) \tag{2.41}$$

where

$$a_0 = \frac{RT}{L} - 1, \quad a_1 = \frac{K_eK_tT^2}{J^2}, \quad b_1 = \frac{T}{L}, \quad c_1 = -\frac{K_eT}{J} + \frac{K_eBT^2}{J^2} \quad \text{and} \quad d_1 = \frac{K_eT^2}{J^2}$$

Upon simplifying further, Eq. (2.41) is notationally reduced to

$$\hat{y}((k+1)T \,|\, \theta(T)) = \phi^T(kT)\theta(T) \tag{2.42}$$

where $i(kT)$ is replaced by $y(kT)$ and

$$\phi^T(kT) = [y(kT) \quad y((k-1)T) \quad v(kT) \quad \omega((k-1)T) \quad T_{ex}((k-1)T)]$$

and

$$\theta^T(T) = [a_0(T) \quad a_1(T) \quad b_1(T) \quad c_1(T) \quad d_1(T)]$$

As evidenced by the parameter vector $\theta^T(T)$ recently presented, the choice of sampling interval clearly affects the estimation of the parameters and is an important consideration when performing system identification. The above formulation also provides an example of a grey-box model where the model structure in Eq. (2.41) is derived from a physical insight into the DC motor system. As a result, the estimated parameters have physical meaning and back-calculation of the physical parameter values from the vector of values in $\theta^T(T)$ is possible.

Continuous-Time Transfer Function Representation

Revert to the governing differential equations shown in Eq. (2.26). Apply an initial condition of zero to the state variable x in Eq. (2.30). This gives

$$X(s) = (sI - A)^{-1}(EU(s)) = \Phi(s)EU(s) \tag{2.43}$$

Also in this case

$$\begin{aligned} Y(s) &= CX(s) = C\Phi(s)EU(s) \\ \Rightarrow H(s) &= \frac{Y(s)}{U(s)} = C\Phi(s)E \end{aligned} \tag{2.44}$$

$H(s)$ represents the required continuous-time transfer function. The individual transfer functions for each combination of input and output signals are now derived as follows. $\Phi(s)$ can be represented in matrix form as

$$\Phi(s) = \frac{LJ}{(sJ+B)(sL+R)+K_eK_t}\begin{bmatrix} s+\frac{B}{J} & -\frac{K_e}{L} \\ \frac{K_t}{J} & s+\frac{R}{L} \end{bmatrix} \tag{2.45}$$

Therefore

$$\begin{aligned} H(s) = C\Phi(s)E &= \frac{LJ}{(sJ+)(sL+R)+K_eK_t}\begin{bmatrix} \frac{1}{L}\left(s+\frac{B}{J}\right) & \frac{K_e}{LJ} \\ \frac{K_t}{LJ} & -\frac{1}{J}\left(s+\frac{R}{L}\right) \end{bmatrix} \\ \Rightarrow H(s) &= \frac{1}{(sJ+B)(sL+R)+K_eK_t}\begin{bmatrix} sJ+B & K_e \\ K_t & -(sL+R) \end{bmatrix} \end{aligned} \tag{2.46}$$

The two transfer functions of interest in this example are those between armature current i_m and armature voltage v_m, and armature angular speed ω_m and armature voltage v_m. These two transfer functions are individually represented as follows after multiplying $H(s)$ with $U(s)$ and abandoning the matrix representation.

$$\begin{aligned} H_1(s) = \frac{I_m(s)}{V_m(s)} &= \frac{sJ+B}{(sJ+B)(sL+R)+K_eK_t} \\ &+ \frac{K_e}{(sJ+B)(sL+R)+K_eK_t}\left(\frac{T_{ex}(s)}{V_m(s)}\right) \end{aligned} \tag{2.47}$$

$$\begin{aligned} H_2(s) = \frac{W_m(s)}{V_m(s)} &= \frac{K_t}{(sJ+B)(sL+R)+K_eK_t} \\ &- \frac{sL+R}{(sJ+B)(sL+R)+K_eK_t}\left(\frac{T_{ex}(s)}{V_m(s)}\right) \end{aligned} \tag{2.48}$$

$H_1(s)$ represents the continuous-time transfer function for armature current versus armature voltage while $H_2(s)$ represents the continuous-time transfer function for armature angular speed versus armature voltage.

Discrete-Time Transfer Function Representation

In order to obtain the discrete-time equivalents of the transfer functions represented in Eqs. (2.47) and (2.48), Tustin's approximation (or the Bilinear Transform) is applied to them. After the required substitution (viz. $s = \frac{2}{T}(\frac{z-1}{z+1})$) and the ensuing

algebraic manipulations, the following discrete-time transfer functions are derived.

$$H_1(z) = \frac{I_m(z)}{V_m(z)} = \frac{z(2J + BT) + (T - 2J)}{\Psi(z)} + \frac{K_e T(z+1)}{\Psi(z)}\left(\frac{T_{ex}(z)}{V_m(z)}\right) \tag{2.49}$$

$$H_2(z) = \frac{W_m(z)}{V_m(z)} = \frac{K_t T(z+1)}{\Psi(z)} - \frac{z(2L + RT) + (RT - 2L)}{\Psi(z)}\left(\frac{T_{ex}(z)}{V_m(z)}\right) \tag{2.50}$$

where

$$\Psi(z) = \frac{z^2(4JL + 2TM + T^2N) + z(2T^2N - 8JL) + (T^2N - 2TM + 4JL)}{T(z+1)}$$

$$M = BL + JR,\ N = BR + K_e K_t$$

$H_1(z)$ represents the discrete-time transfer function for armature current versus armature voltage while $H_2(z)$ represents the discrete-time transfer function for armature angular speed versus armature voltage.

For the simulations in this example, $T_{ex}(z) = 0$ in all cases and therefore the transfer function is reduced to

$$H(z) = \frac{W_m(z)}{V_m(z)} = \frac{K_t T(z+1)}{\Psi(z)} \tag{2.51}$$

2.2.3 DC Motor Model Validation

Both the continuous-time and discrete-time transfer functions are observed independently to see how they simulate the DC Motor's response to a pulse input. The parameters for the DC motor are borrowed from the DC motor component of a validated EMA model provided by Lockheed Martin Control Systems, Johnson City, New York, (now BAE Systems Inc.), and available in the Master's thesis of Mr. Konstantin P. Louganski [41]. These parameters are also provided in Table 2.1. Note that different parameter values are used for testing the fault detection algorithm in Section 2.2.5. Both forms of the DC motor model are developed in Simulink. A pulse input (square wave) of 0.1 V amplitude is provided. The simulation is run for a duration of 0.1 seconds. For the discrete-time transfer function model, a sample interval T

Table 2.1: DC motor parameters

Parameter	Value
K_t	1.141 in-lb/A
L	4.5×10^{-4} H
R	$0.5\,\Omega$
K_e	0.129 V/(rad/s)
J	2.43×10^{-4} in-lb-s^2
B_m	3.125×10^{-4} in-lb-s

of 0.0001 seconds is chosen. The output response from the continuous-time model is shown in Fig. 2.13 while that from the discrete-time model is shown in Fig. 2.14.

Since both forms of the model give similar outputs for the same input signal and

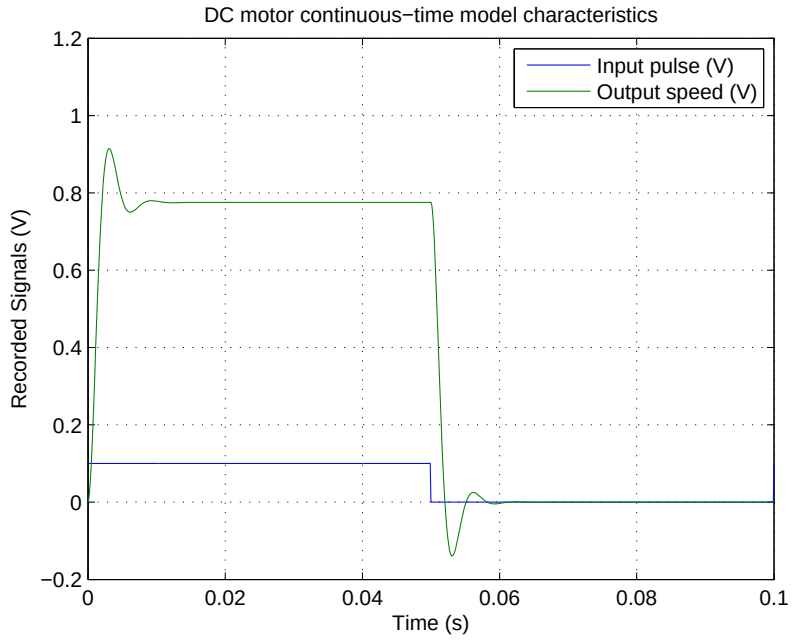

Figure 2.13: DC motor continuous-time simulated response to a pulse input

the fact that the nature of the response resembles that of a DC motor leads to the conclusion that the models derived in the previous section are valid.

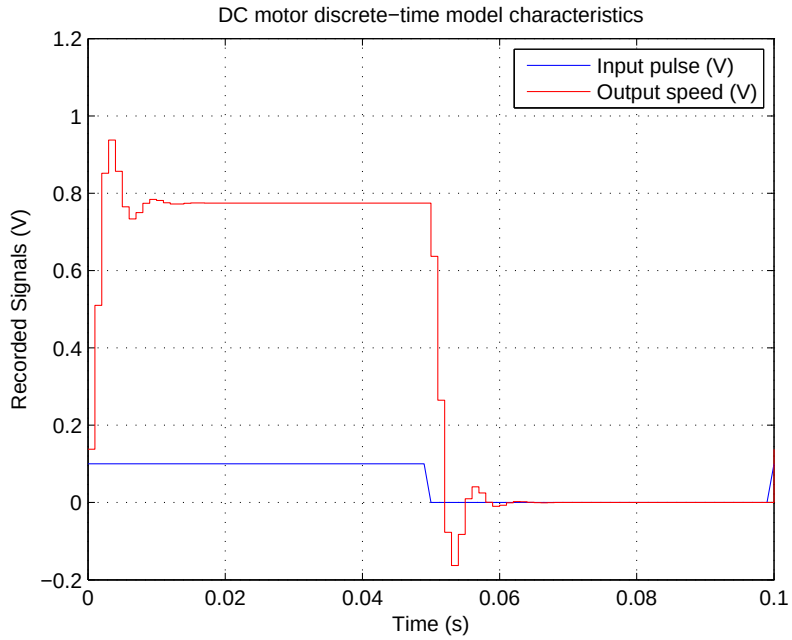

Figure 2.14: DC motor discrete-time simulated response to a pulse input

2.2.4 Modeling the Single-Point Defect

In order to simulate the single-point defect, it is required that the viscous friction coefficient of the load B_L be varied in a periodic manner. Additionally, the period chosen is not random but based on the effect a spalled bearing has on an electric motor drive system [10, 20]. According to [10], bearing seizure is approximated by an increase in the friction damping coefficient. Therefore, to simulate a single-point defect, bearing seizure is assumed to occur periodically with motor position. The defect frequency is calculated based on the presence of a defect in the outer-race of a bearing [20].

$$f_{OD} = \frac{n}{2} f_{rm} \left(1 - \left(\frac{BD}{PD} \right) \cos \delta \right) \tag{2.52}$$

where PD represents the pitch diameter and is the outer diameter of the bearing's outer race, BD represents the ball bearing diameter, n is the number of balls, δ is the contact angle and f_{rm} is the motor angular speed in Hz. According to [20], Eq. (2.52) is satisfactorily approximated by $f_{OD} = 0.41 n f_{rm}$ for most bearings with between

6 to 12 balls. This approximation is used. Both healthy and defected systems are simulated with step inputs in voltage. The Simulink models are presented in Appendix A while the MATLAB script file used for fault detection is published in Appendix B.

2.2.5 Fault Detection Results

To recap, for the LJ MS15 Control Module, data sets for five brake positions are captured to form five cases of which one (no brake condition) depicts a healthy system while the others depict various levels of degradation (refer Fig. 2.10). Furthermore, the degradation is assumed to represent a generalized roughness defect. In order to show the performance of the proposed fault detection procedure, the output-data signal in each case is corrupted with Gaussian white noise to a point when the signal-to-noise ratio (SNR) is 25 because the signals aquired from the module are mostly noise-free - a situation seldom encountered in more complex systems such as EMAs. The SNR value is chosen based on the inability to comment on the system's health merely by observing its step response. Results from the motor module with brake at position 1 are shown in Fig. 2.15.

It is concluded that the approach barely distinguishes between the healthy and

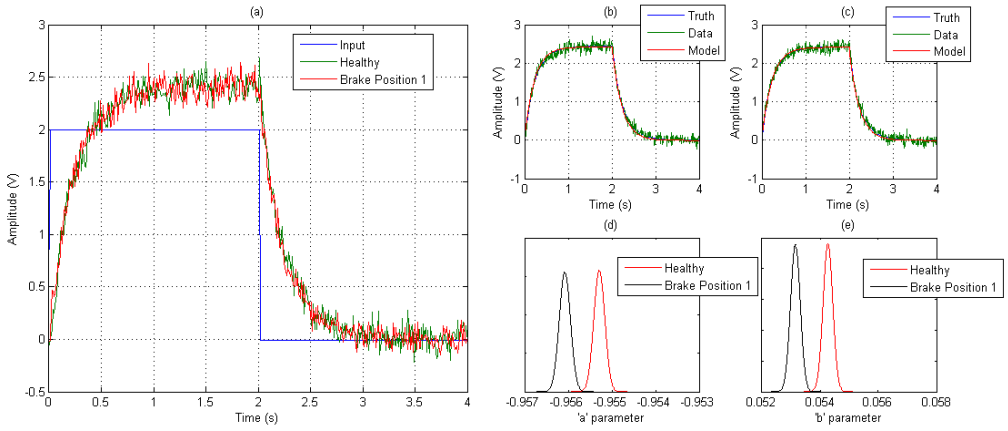

Figure 2.15: Results of motor module with brake at position 1 - (a) Step response with added noise; (b) Healthy step response; (c) Step response with brake; (d) Distribution of predicted 'a' parameter; (e) Distribution of predicted 'b' parameter

the degraded motor module and it is likely to have a fairly high misclassification rate even though the algorithm appears to predict the system model accurately in each case as indicated by the model fit percentage given in Table 2.2. When the normal

Table 2.2: DC Motor Module Generalized Fault Detection Results (Values correspond to Fig. 2.16)

Characteristic	No Brake	Brake 1	Brake 2	Brake 3	Brake 4
a	-0.9565	-0.9554	-0.9485	-0.9429	-0.94044
σ_a^2	0.0002	0.0001	0.0004	0.0003	0.0003
b	0.05275	0.05382	0.05353	0.05467	0.05194
σ_b^2	0.0003	0.0002	0.0003	0.0003	0.0002
Fit %	96.84	96.68	96.68	95.91	96.54

distribution of the predicted parameters from the healthy and the degraded cases are plotted, the separation of the means is quite small and there exists some overlap of the distributions, indicating a higher likelihood of misclassifcation. It is noted that the distribution of the a parameter (Fig 2.15 (d)) is likely of greater significance than the b parameter because it depicts the denominator dynamics of the system in each case.

The parameter distribution plots for all the other brake positions are consolidated into Fig. 2.16 while Table. 2.2 provides useful information related to these distributions. Notice the distributions of the healthy and brake position 1 systems in Fig. 2.16 (a) and compare them to those in Fig. 2.15 (d). This is a clear example of blatant misclassification due to a marginal change in the noise added to the system.

In the case of the seeded single-point defect in the motor module, the proposed approach does a poor job of detecting the presence of the defect. The approach is unable to capture the periodic fluctuations in the output. Instead, the approach "assumes" that the effect of these disturbances are distributed over the entire output. Since the fluctations are small, the parameter estimates for the two health cases are similar. Figure 2.17 shows the distributions of the parameter estimates in this case.

Therefore, the proposed fault detection scheme based on system identification

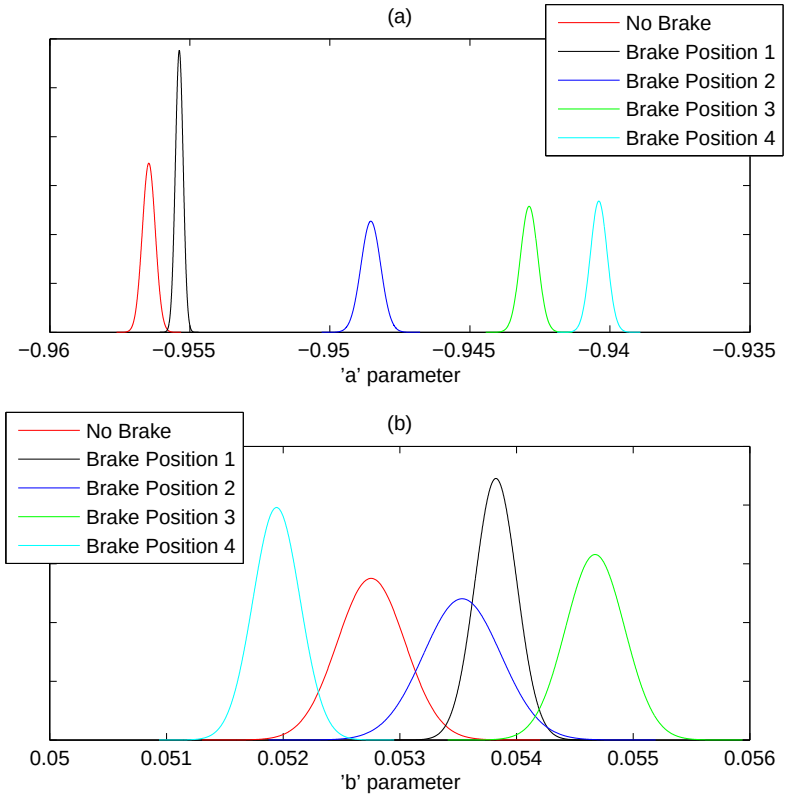

Figure 2.16: Distributions of parameters for each of the brake positions - (a) Distributions of 'a' parameters; (b) Distributions of ' b' parameters

techniques where the estimated parameters that represent the system are used as the features for classification, is inadequate to effectively detect faults in the DC Motor control module, especially in the case of single-point defects. Although some successful detection is observed, these isolated cases of success are attributed to significant changes in the system parameters. For instance, except for brake position 1, the other cases show a significant change in the system output (see Fig. 2.10) and this approach is expected to succeed in those cases.

As mentioned earlier, although the approach detects the application of the brake at position 1, it is likely that misclassification will occur with minor changes in the system that may not represent a fault. Although the model prediction is generally very accurate and the use of an IV estimation method reduces the influence of noise in the estimation of the parameters to some extent, it is likely that changes in noise will contribute to misclassification.

Since the algorithm is not effectively tested in the case of a single-point defect as the seeding of such a defect on the DC motor module is difficult, a simulated defect is introduced in a Simulink model of a DC motor system. The Simulink block diagram with the single-point defect modeled based on Eq. (2.52) is provided in Appendix A. The simulated outputs generated in each case are corrupted with Gaussian white noise with a signal-to-noise ratio of 20 instead of 25 to test how the algorithm performs in the presence of more powerful noise. The motor model parameter values and the defect characteristics are consolidated in Table 2.3.

Figure 2.18 shows the result for a simulated single-point defect. It is seen that the proposed approach is still unable to offer effective fault detection for such fault types even though the system is generally identified accurately. The problem with this approach is the choice of the parameter estimates as the features for classification. The deviations in the estimates between healthy and degraded systems are too small to allow effective classification (assuming 'effective' classification is achieved with at least a 5σ separation between peaks).

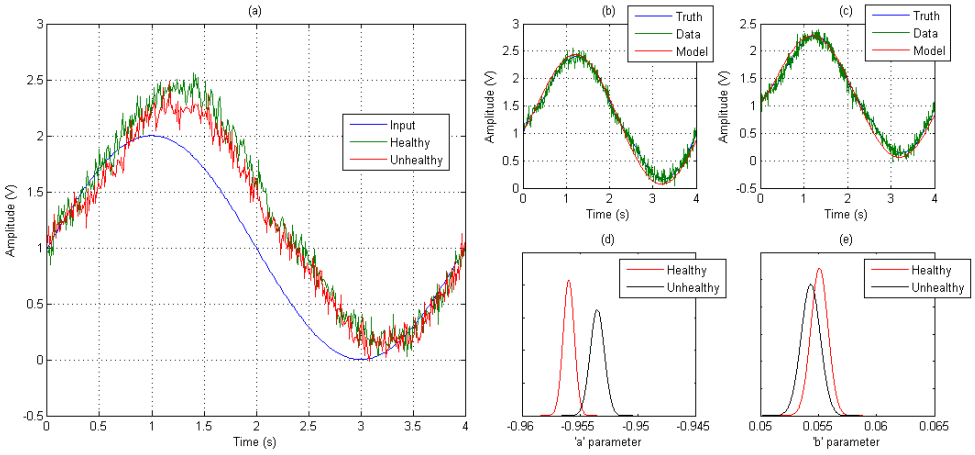

Figure 2.17: Results of Motor Module with single-point defect - (a) Response with added noise; (b) Healthy response; (c) Response with brake; (d) Distribution of predicted 'a' parameter; (e) Distribution of predicted 'b' parameter

Table 2.3: Motor model and defect parameter values

Parameter	Value
K_t	0.05 Nm/A
n	9
K_e	0.05 Vs/rad
PD	$1.7\text{x}10^{-2}$ m
L	$4.5\text{x}10^{-7}$ H
BD	$0.3\text{x}10^{-2}$ m
R	0.5 Ω
δ	0.1745 rad
J_m	0.00025 Nms^2/rad
B_m	0.0001 Nms/rad

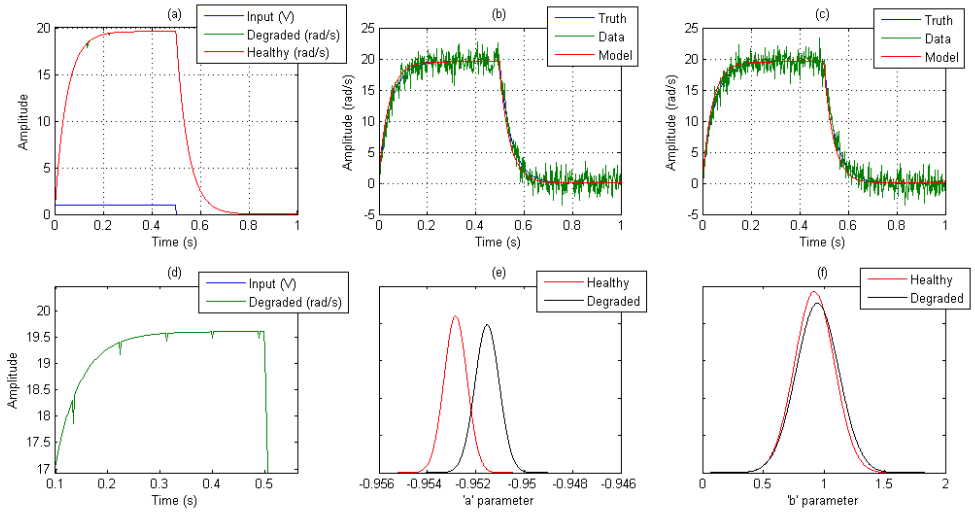

Figure 2.18: Results of a DC motor model with a simulated outer-race bearing defect - (a) Healthy vs. Degraded response; (b) Healthy response with added noise; (c) Degraded response with added noise; (d) Single-point defects in degraded response; (e) Distribution of predicted 'a' parameter; (f) Distribution of predicted 'b' parameter

2.2.6 Limitations of the Fault Detection Approach

Based on the results shown in the previous sections, the limitations of the fault detection approach are noted as follows:

1. The parameter estimates, when used as features, result in ineffective fault detection in the case of both single-point defects as well as generalized roughness defects. Any fault information that may be present in the b parameter is neglected and all the models selected are first-order models that are sufficient to identify the system accurately but may not be sufficient to obtain all the information about the faults present.

2. Although the systems are accurately identified by the model structures, the algorithm is not tested on more complex systems such as EMAs. The DC motor model is a first-order system operating in open-loop and the control module is also quite accurately estimated by a first-order open loop system. The Moog MaxForce EMA however consists of two nested control loops making it more

complex.

3. The parameter estimates are not sufficiently insensitive to noise.

4. Model validation is not performed nor is a definite model order selection methodology in place. All the cases are fitted with a standard first-order linear difference equation and although the fit percentage might be good, the model may not be robust and might depend on the nature of the input and noise. Both these effects are undesirable.

5. The selected model structure is for a single-input, single-output system. In the case of the EMA, an external torque load affects the output significantly and hence must be used as a second input to accurately fit a model to the data. Therefore, flexibility in choice of model structure is essential but is not available in the present approach.

2.3 Proposed Revisions to the Fault Detection Approach

Having identified the limitations of the previous approach, the following modifications are proposed:

1. Test various model structures and make a choice based on model fit and a residual analysis to ensure that the model estimation is accurate and that the parameter estimates are not significantly affected by variations in noise or external disturbances. Allow flexibility in selecting higher order models if necessary.

2. Allow the usage of multiple inputs to the system to better estimate the model parameters.

3. Investigate the effects a closed loop system has on fault detection and test various combinations of input and output data to be used for parameter estimation.

4. In case higher order models are chosen, the number of parameters to be estimated increases and each of the parameters can contain fault information to

different extents. Therefore, devise a pattern recognition or other feature extraction technique that identifies those parameters that contain the most information or perform a mathematical transformation on the parameters to bring out the fault information better.

5. Develop a classifer that effectively groups features from systems having the same health condition by minimizing the rate of misclassification.

Some of the revisions proposed above are incorporated in a revised fault detection approach, details of which are presented in Chapter 3. The major changes include (i) added flexibility in model structure selection and choice of inputs and outputs based on a mathematical study of the effect of closed loop system controllers on faults, (ii) a well defined model validation procedure, (iii) feature extraction from estimated parameters using Principal Component Analysis (PCA) and (iv) application of a Bayesian classifier to effectively group features belonging to the same health class with each other. The theoretical concepts behind the above modifications are presented in the next few sections.

2.4 Fault Detection in Closed-Loop Systems

Systems that require precise tracking of reference variables operate in closed loop under the action of a controller. The EMA used in this work is one such system. Small faults in the actuator and process are usually compensated by the feedback controller and they are not detectable by considering the desired and actual outputs alone, as is done for systems operating in open loop [42]. Furthermore according to [42], faults generating (or equivalent to generating) a change in the amount of friction in the system operating in closed loop - such as the spalled bearing and generalized roughness defects dealt with in this research - can be detected using parametric identification techniques similar to those already discussed for systems operating in open loop. For fault detection of the closed-loop system in Fig. 2.19, the

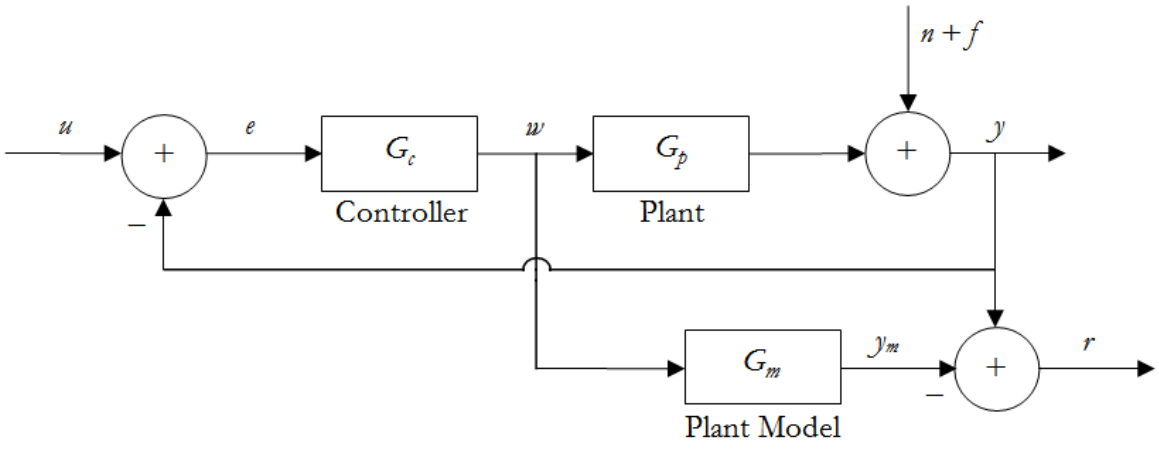

Figure 2.19: Model-based fault detection of closed-loop system with known controller output

output error is given by

$$r(s) = y(s) - y_m(s) = y(s) - G_m(s)w(s) = (G_p - G_m)w + n + f \tag{2.53}$$

where n and f denote noise and fault information. This simplifies to

$$\begin{aligned} r(s) &= (G_p - G_m)G_c\left[\frac{u - (n+f)}{1 + G_cG_p}\right] + n + f \\ \Rightarrow r(s) &= \frac{G_c(G_p - G_m)}{1 + G_cG_p}u + \frac{1 + G_cG_m}{1 + G_cG_p}(n+f) \end{aligned} \tag{2.54}$$

To identify the model G_m using the model parameters as arguments minimizing the equation error $r(s)$ where

$$\begin{aligned} r(s) &= A_m(s)y(s) - B_m(s)w(s) \\ \Rightarrow r(s) &= \frac{Q(A_mB_p - A_pB_m)}{PA_p + QB_p}u + \frac{A_p(PA_m + QB_m)}{PA_p + QB_p}(n+f) \end{aligned} \tag{2.55}$$

and where

$$G_p(s) = \frac{B_p(s)}{A_p(s)} \quad \text{and} \quad G_c(s) = \frac{Q(s)}{P(s)}$$

it is desirable that the contributions from u vanish. That is, if $r(s)$ is minimized such that it only consists of contributions from n and f, then $A_m = A_p$ and $B_m = B_p$ in Eq. (2.55) and the plant is exactly identified. As the magnitude of n or f or both increase, the arguments minimizing $r(s)$ are affected by a filtered amount (the filter being the coefficient of $(n + f)$ in Eq. (2.55)) and the chances of identifying the plant accurately are reduced. In the present work however, fault identification is the primary objective and the parameter estimates (A_m and B_m or their equivalents) are used to extract features for classification. In this case, assuming the data used for the parameter estimation are $y(s)$ and $w(s)$, it is desirable that A_m and B_m are affected, as far as possible, by f. In most practical situations, the magnitude of f is the smallest amongst n and u and thus it has the smallest effect during the estimation of the features. This is true even in the open-loop case but the filters for each of the terms are mathematically less complex, offering better chances for weaker signals such as f to influence the estimation of the features.

In this work however, the problems are more complicated. Due to the nature of the experiments conducted by Moog Inc., the closed loop systems used are equivalent to the ones shown in Figs. 2.20 and 2.21. Figure 2.20 is representative of the EMA system setup at the Rochester Institute of Technology while Fig. 2.21 is similar to the setup at Moog, Inc. For the system in Fig. 2.20,

$$r(s) = y(s) - y_s(s)$$

and from the equation error consideration

$$r(s) = A_s(s)y(s) - B_s(s)u(s)$$

where $A_s(s)$ and $B_s(s)$ are parameter vectors. Proceeding in a similar manner,

$$w(s) = G_c(s)(u(s) - y(s)) = \frac{QA_p(u - (n + f))}{PA_p + QB_p} \tag{2.56}$$

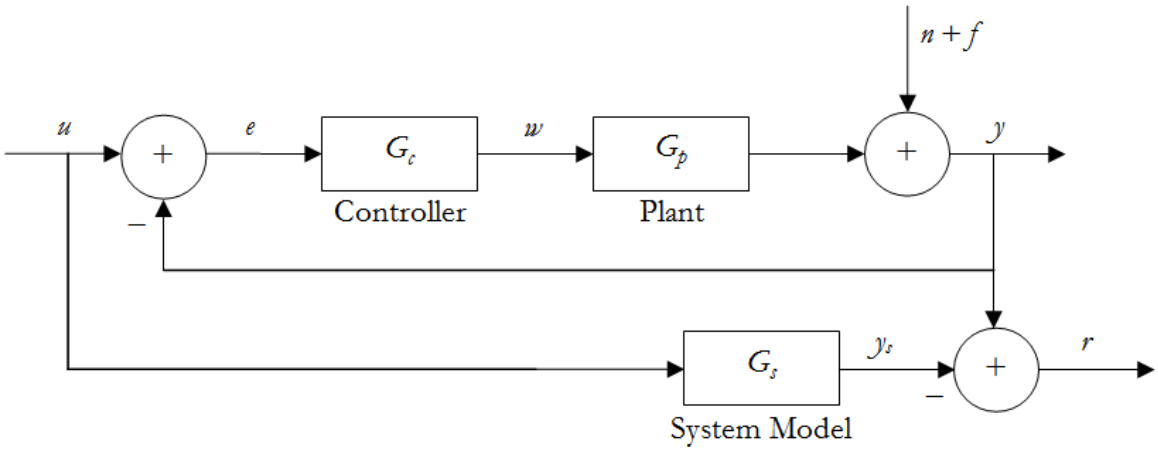

Figure 2.20: Model-based fault detection of closed-loop system with unknown controller output

Therefore,

$$r(s) = A_s\left[\left(\frac{B_pQ(u-(n+f))}{PA_p+QB_p}\right)+n+f\right]-B_su$$

$$\Rightarrow r(s) = \left[\frac{A_sB_pQ-B_sPA_p-B_sQB_p}{PA_p+QB_p}\right]u+\left[\frac{A_sPA_p}{PA_p+QB_p}\right](n+f) \tag{2.57}$$

In this case, to estimate A_s and B_s to exactly identify the plant alone would require setting these parameters as arguments that reduce the function $r(s)$ to

$$r(s) = \left[\frac{ABQ}{PA+QB}-B\right]u+\left[A-\frac{ABQ}{PA+QB}\right](n+f) \tag{2.58}$$

where $A = A_p = A_s$ and $B = B_p = B_s$ for an exact agreement of the system model and plant shown in Fig. 2.20. If u vanishes from Eq. (2.57), then the system being identified is given by

$$A_sB_pQ-B_sPA_p-B_sQB_p = 0$$

$$\Rightarrow \frac{B_s}{A_s} = \frac{QB_p}{QB_p+PA_p}$$

$$\Rightarrow G_s(s) = \frac{G_cG_p}{1+G_cG_p} \tag{2.59}$$

For fault detection, the situation is similar to the previous one. In this case however, the filters for u and $(n + f)$ are different. With u being the dominating signal, more accurate minimization routines are required that are robust enough to minimize $r(s)$ sufficiently while at the same time, allowing the parameters to absorb as much information from f as possible. Reducing the magnitude of u by choosing an appropriate input signal is a solution that is tested as part of the revised fault detection approach presented in the next chapter.

For the system in Fig. 2.21, the scenario is even more complex. Once again

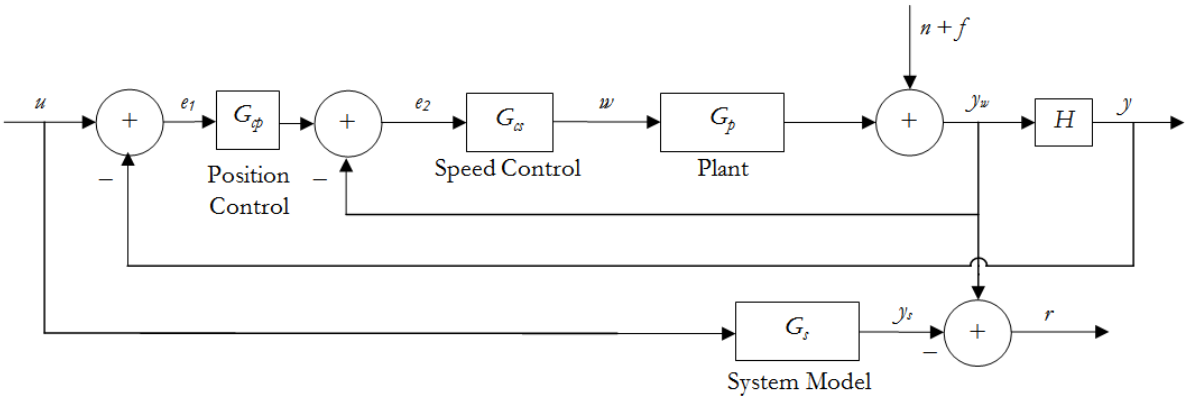

Figure 2.21: Model-based fault detection of nested closed-loop system with unknown controller outputs

considering an equation error form for parameter estimation,

$$r(s) = A_s(s)y_w(s) - B_s(s)u(s)$$

$$w(s) = e_2(s)G_{cs}(s) = (G_{cp}(s)e_1(s) - y_w(s))G_{cs}(s)$$

where $y_w = wG_p + n + f$. Also, $y = Hy_w$ and $e_1 = u - y = u - Hy_w$. Therefore, $w(s)$ simplifies to,

$$w(s) = \frac{G_{cs}G_{cp}u - G_{cs}(HG_{cp} + 1)(n + f)}{1 + HG_{cs}G_{cp}G_p + G_{cs}G_p} \tag{2.60}$$

and

$$y_w(s) = \frac{G_pG_{cs}G_{cp}u - G_pG_{cs}(HG_{cp} + 1)(n + f)}{1 + G_{cs}G_p(HG_{cp} + 1)} + n + f \tag{2.61}$$

giving

$$\begin{aligned} r(s) &= \left[\frac{A_s G_p G_{cs} G_{cp} - B_s\left[1 + G_{cs} G_p (H G_{cp} + 1)\right]}{1 + G_{cs} G_p (H G_{cp} + 1)}\right] u \\ &+ \left[\frac{A_s}{1 + G_{cs} G_p (H G_{cp} + 1)}\right] (n + f) \end{aligned} \tag{2.62}$$

The vanishing of the filter for u gives the system being identified. That is,

$$\begin{aligned} & A_s G_p G_{cs} G_{cp} - B_s\left[1 + G_{cs} G_p (H G_{cp} + 1)\right] = 0 \\ & \Rightarrow \frac{B_s}{A_s} = G_s(s) = \frac{G_p G_{cs} G_{cp}}{1 + G_{cs} G_p (H G_{cp} + 1)} \end{aligned} \tag{2.63}$$

The comments about fault detection remain similar to the previous cases. Based on the preceding analysis, selecting e in the case of Fig. 2.20 and e_1 in the case of Fig. 2.21 as one of the inputs in the parameter estimation process is proposed. It is expected that doing so will highlight the fault information in the output signal while the IV method proposed for parameter estimation is expected to reduce the influence of noise variations in the parameter estimates.

2.5 Principal Component Analysis (PCA)

Another modification to the previous fault detection approach is to use PCA to extract features from estimated model parameters instead of using the parameters as features themselves. In particular, this work focuses on utilizing PCA to uncorrelate linearly correlated model parameters without discarding any significant variance information from the data. This is unlike most other applications of PCA (such as in [19]) where dimension reduction is more prominent. The idea behind this approach is explained as follows.

Suppose a second order model is required to generate a "good" fit to the data selected (more on model structure selection is provided in the next chapter). In this case, a linear difference equation of the form $y(t) + a_1 y(t-1) + a_2 y(t-2) = b_1 u_1(t-1) + c_1 u_2(t-1)$ is used (a two input system is depicted because in the case

of the Moog EMAs, the applied external torque significantly affects the output and hence cannot be neglected). Assuming, as is done before, that a_1 and a_2 carry most of the information about the fault, it is expected that the two parameters are correlated and that their individual variances are useful parameters to be analyzed for fault information. If analyzed independently as is done in the previous approach, the differences between estimated parameters from a healthy and an unhealthy system are minimal. When PCA is applied on these parameter distributions, a transformation occurs that results in two "features" that are now uncorrelated with one another and with the distribution of one of the features having greater variance than the other. Thus when this feature of larger variance (or both features taken together) is selected and compared across different health classes, it is expected that nearly all the information about the fault that is picked up by the parameters is available to make a comparison. In most cases in this work, a correlated two-dimensional or three-dimensional data set is transformed into an uncorrelated two-dimensional one (i.e. none or a minimal amount of the information from the data is discarded). The following information on PCA is consequently related to its application in this work. An example consisting of a randomly generated data set having a normal distribution is used. The data set is split equally in two with each set signifying a given "health condition". A scatter plot of the entire dataset is shown in Fig. 2.22.

To perform the PCA, the first step is to center the data set about the origin by subtracting the means from each corresponding dimension. This makes the expected value of the dataset zero. The centered data is shown in Fig. 2.23.

The covariance matrix is then computed. Typically, the non-diagonal elements will be non-zero indicating that the two dimensions are correlated. If they are zero, it is likely that using the data in the current form as features would yield the best possible results. The eigenvalues and eigenvectors of the covariance matrix are computed and the two eigenvectors corresponding to the two largest eigenvalues are selected as the feature vectors. The two feature vectors form the feature matrix W which is then used to transform the centered dataset $\mathbf{y}$ into a "feature set" $\mathbf{z}$ according to the

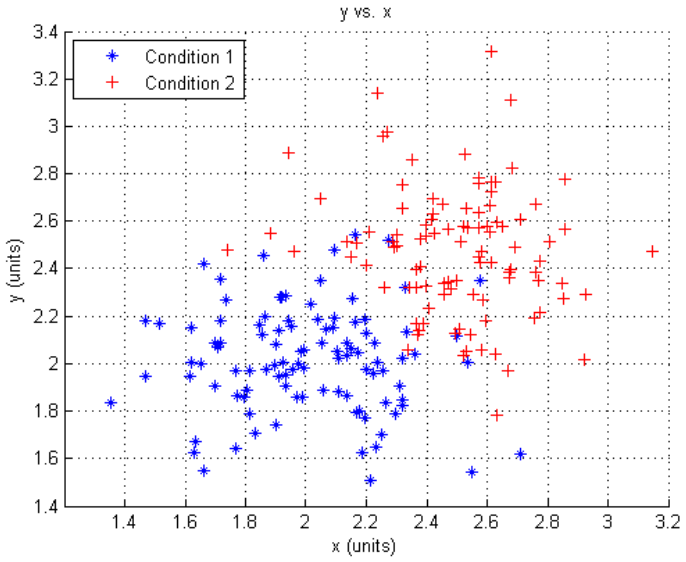

Figure 2.22: PCA sample data set

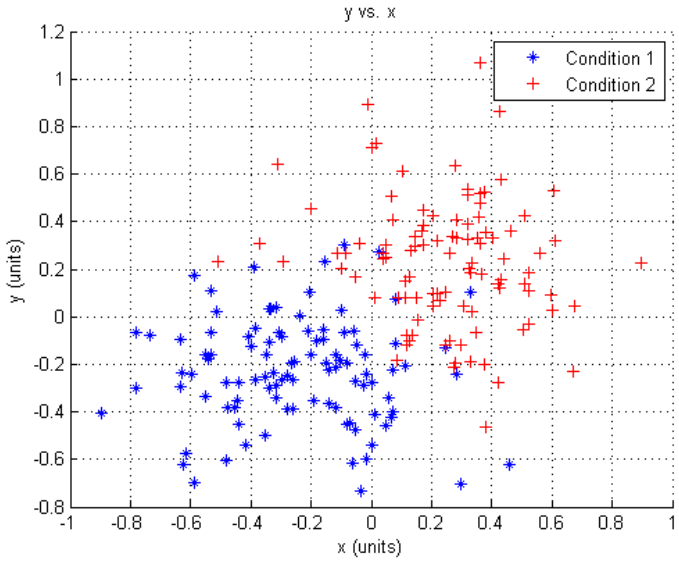

Figure 2.23: PCA sample data centered about the origin

relation

$$\mathbf{z} = W\mathbf{y}$$

The "features" (i.e. transformed data) are now distributed along two reoriented axes (shown in Fig. 2.24) in such a manner that they are no longer correlated and the variance along one of the transformed axes is maximum. This axis becomes the first

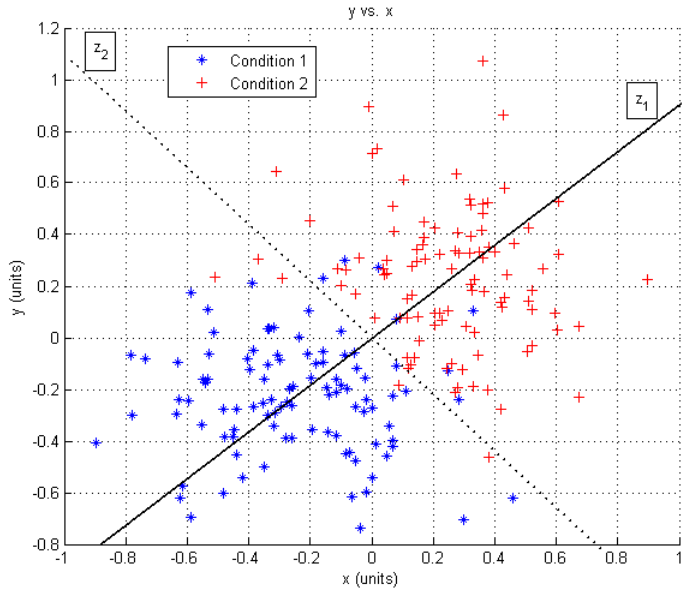

Figure 2.24: Scatter plot of centered data showing the two principal components z_1 and z_2. The data now has maximum variance along z_1 and minimum variance along z_2

principal component while its complement is the second. It is possible that the second component be discarded without losing any significant information, however that is generally not performed in this work. The "feature set" $\mathbf{z}$ comprising of z_1 and z_2 is shown in Fig. 2.25 as the final outcome of PCA on the original dataset.

2.6 Bayesian Classification

Bayesian classification is an approach that uses Bayes' theorem to select the class of an object from a set of possible classes $\Omega = \{w_1, w_2, \ldots, w_K\}$ that are assumed

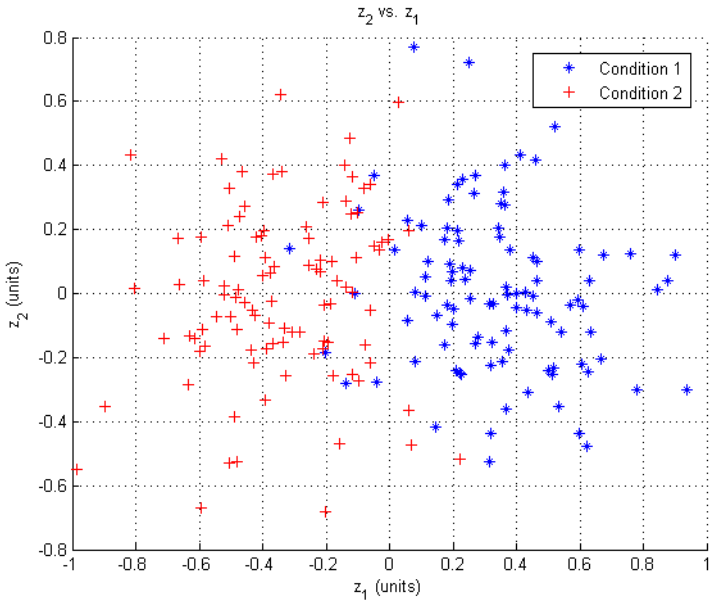

Figure 2.25: Feature set derived by performing PCA on the data shown in Fig. 2.22

to be mutually exclusive. For example, in the case of the Moog MaxForce EMA, two classes exist such that $\Omega = (\text{'Healthy'},\text{'Degraded'})$. The class with the minimum amount of risk is the class selected by the Bayesian Classifier. According to Bayes' theorem,

$$p(w_k|\mathbf{z}) = \frac{p(\mathbf{z}|w_k)P(w_k)}{p(\mathbf{z})} \tag{2.64}$$

The classifier decision function is shown in Eq. (2.65).

$$\hat{w}(\mathbf{z}) = \underset{w\,\epsilon\,\Omega}{\text{argmin}}\left\{ \sum_{k=1}^{K} C(w|w_k)p(\mathbf{z}|w_k)P(w_k) \right\} \tag{2.65}$$

where the cost function $C(w|w_k)$ is the penalty of assigning the class w to an object actually belonging to class w_k. In this work, the cost function is assumed to be uniform which implies that

$$C(\hat{w}_i|w_k) = 1 - \delta(i,k) \quad \text{with} \quad \delta(i,k) = \begin{cases} 1 & \text{if} \quad i = k \\ 0 & \text{elsewhere} \end{cases}$$

the unconditional prior probability of each class $P(w_k)$ is equal, and

$$\sum_{k=1}^{K} P(w_k) = 1 \tag{2.66}$$

Lastly, it is assumed that the conditional density, $p(\mathbf{z}|w_k)$, is normally distributed. That is

$$p(\mathbf{z}|w_k) = \frac{1}{\sqrt{(2\pi)^N |\mathbf{C}_k|}} \exp\left(\frac{-(\mathbf{z}-\mu_k)^T \mathbf{C}_k^{-1}(\mathbf{z}-\mu_k)}{2}\right) \tag{2.67}$$

where μ_k and $\mathbf{C}_k$ represent the expectation vector and covariance matrix of the feature vector $\mathbf{z}$ of true class w_k and N is the dimension of the vector $\mathbf{z}$. For the sample data set from the previous section, the Bayes' classifier generates the classification bound shown in Fig. 2.26.

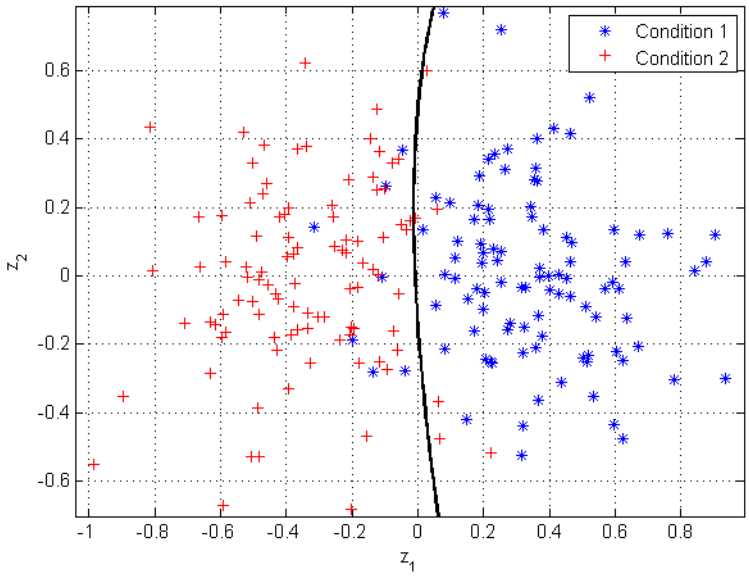

Figure 2.26: Bayesian classification bound for sample dataset shown in Fig. 2.22

Chapter 3

Methodology

This chapter provides a step-by-step account of the proposed fault detection approach. All subsequent sections discuss each step of the approach in detail. An example involving a speed controlled PM DC motor system seeded with an outer-race bearing defect is used, showing results along each step of the process. The Simulink block diagram of the system (see Fig. 3.1) is evaluated to generate the required data sets. The internal architecture and the model parameters are available in Appendix A.

The approach is then used on more cases of the same DC motor system addition-

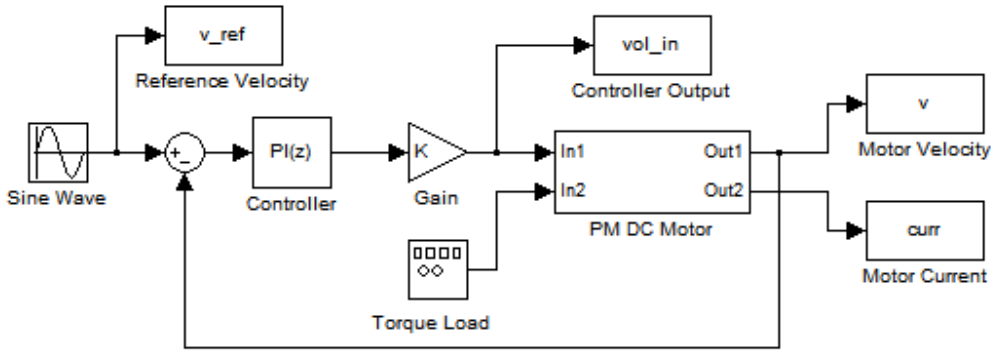

Figure 3.1: Simulink model of a permanent magnet DC motor with inbuilt fault generation system

ally seeded with a reduced lubrication/generalized roughness type defect as shown in Chapter 4. This system is selected because of the following reasons:

1. The EMA is driven by a PMSM and the data used for fault detection is based

on a dq-model of it (i.e. quadrature current is used during the feature extraction process).

2. Approximations to periodic and generalized roughness type faults are readily simulated.

3. A single feedback loop with a PI controller is sufficient for accurate speed control and this is easily implemented in Simulink.

After testing the algorithm through simulations and analyzing its performance, it is applied on experimental data collected from a Moog MaxForce Electromechanical Actuator. Two sets of experiments are conducted. The first set consists of experiments conducted on an actively loaded test rig at Moog Inc., East Aurora, New York while the second set is conducted at the Rochester Institute of Technology, Rochester, New York on a passively loaded test rig.

The following sections present the development of the approach in a step-wise manner, starting with data set generation. Figure 3.2 shows the fault detection approach employed in block diagram form.

3.1 Data Set Generation

As indicated in Fig. 3.2, the first step in the fault detection process is obtaining measurement data from the system. The important factors to consider are: (i) determining which signals are to be captured and (ii) choosing an appropriate sampling interval that allows information about the fault to be captured while making sure that an excessive number of samples are not recorded to prevent computational issues.

Example: DC Motor Seeded with a Single-Point Defect

Figure 3.3 shows healthy and unhealthy motor current signals and the effect of the simulated bearing defect and lack of bearing lubrication on the current signal for the closed loop system of Fig. 3.1. Modeling of the outer-race defect is based on

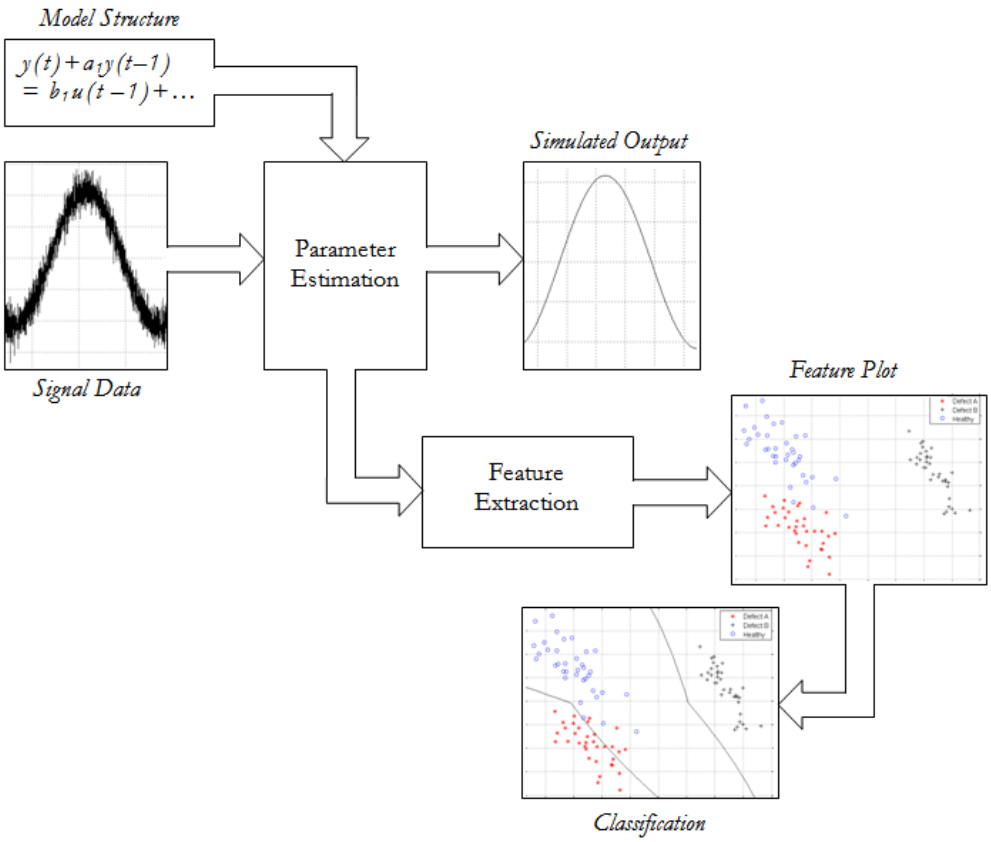

Figure 3.2: Block diagram showing fault detection approach employed in this thesis

Eqs. (1.1) and (2.52) where f_e from Eq. (1.1) is given by,

$$f_e = \frac{p}{2} f_{rm} \tag{3.1}$$

and where f_v from Eq. (1.1) equals f_{OD} from Eq. (2.52), and an increase in the friction-damping coefficient is used to simulate a reduction in the amount of lubrication. The latter approach is based off a similar simulation technique employed in [10] and is therefore validated.

The effect of the bearing defect is easily noticeable while that of the lack of lubrication causes a marginal shift in the phase and increase in the peak amplitude of the current. During the simulations presented in Chapter 4, both defects are considered although they are seeded one defect at a time. In this example, only the single-point defect case is considered.

Such signals are either directly measured using current sensors or are calculated

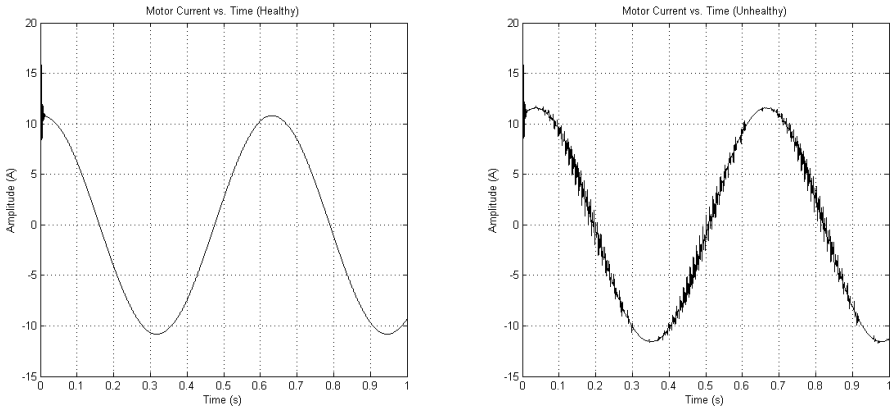

Figure 3.3: Effect of inner-race bearing defect and reduced bearing lubrication on PM DC motor current - Left: Healthy system output; Right: Degraded system output

based on signals captured using sensors. Therefore, such signals are typically corrupted by noise. To simulate this effect, the realization of a Gaussian white noise process having zero mean and a specific variance (see corresponding MATLAB program in Appendix B) is added to the current signal at the output. To ensure that the noise profiles vary with each case, the 'seed' value in the Simulink random number

generator is set to a non-zero random number. This is shown in the internal architecture of Fig. 3.1. According to the literature in [3, 4, 37], simulating noise encountered in real life by a realization of a white noise process is generally a valid assumption. To independently validate the assumption of a Gaussian nature for white noise, current and position sensor noise is acquired from the Moog MaxForce EMA and Gaussian distributions are fitted to it. The results of the fits are shown in Fig. 3.4.

The plots indicate that the EMA current and position sensor noise (these sen-

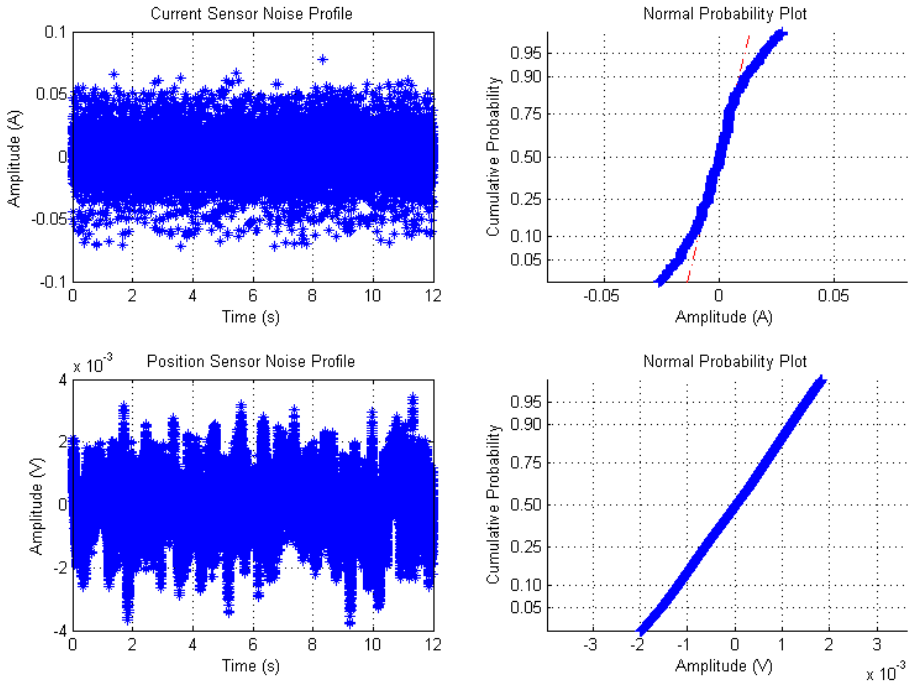

Figure 3.4: Analysis of actual Moog EMA test stand current (top) and position (bottom) sensor noise profiles

sors provide the output signals for use in Chapter 5) assume normal distributions. Therefore, adding the realization of a Gaussian white noise process is appropriate for simulation. The noise corrupted PM DC motor current signals for the healthy and unhealthy cases are shown in Fig. 3.5.

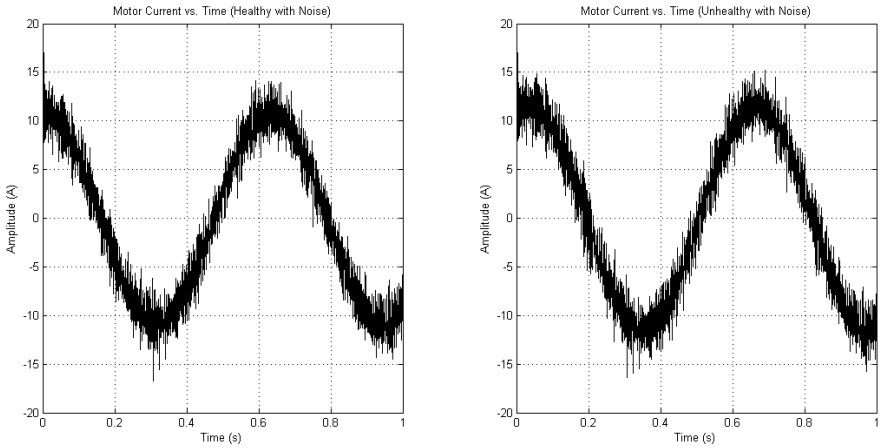

Figure 3.5: Noise corrupted PM DC motor current signals for healthy (left) and unhealthy (right) cases

3.2 Choosing Input and Output Data

Once the measurement data from the system is aquired, the next step is choosing the appropriate input and output data for estimating the parameters of the chosen model structure. The choice of inputs in the case of the MaxForce EMA is limited because the controller outputs are internal to the system and are not collected. Therefore, the chosen input combinations are either the position reference signal or the position error signal along with the motor torque load input although it is expected that the position error signal provides better results. Additionally, the velocity command signal is available as an input in the case of the EMA data collected at RIT. The reasoning behind these choices for inputs is related to the explanation in the section on closed-loop fault detection in Chapter 2. The choice of outputs in the case of the EMA is provided in Chapter 5.

For the simulations in Chapter 4, since the system is a PM DC motor in a closed-loop velocity control configuration, the input combinations are either the velocity reference signal or the velocity error signal along with the sinusoidal torque load input to the motor while the output is the motor current (see Fig. 3.1). Again, it is expected that use of the velocity error signal as input along with the torque load

produces better results.

Example: DC Motor Seeded with a Single-Point Defect

In this example, the velocity error signal and torque load are used as inputs to the system while the motor current is the output. The time plots of these signals are shown in Fig. 3.6. The first two rows show the inputs to the system while the last one shows the system output.

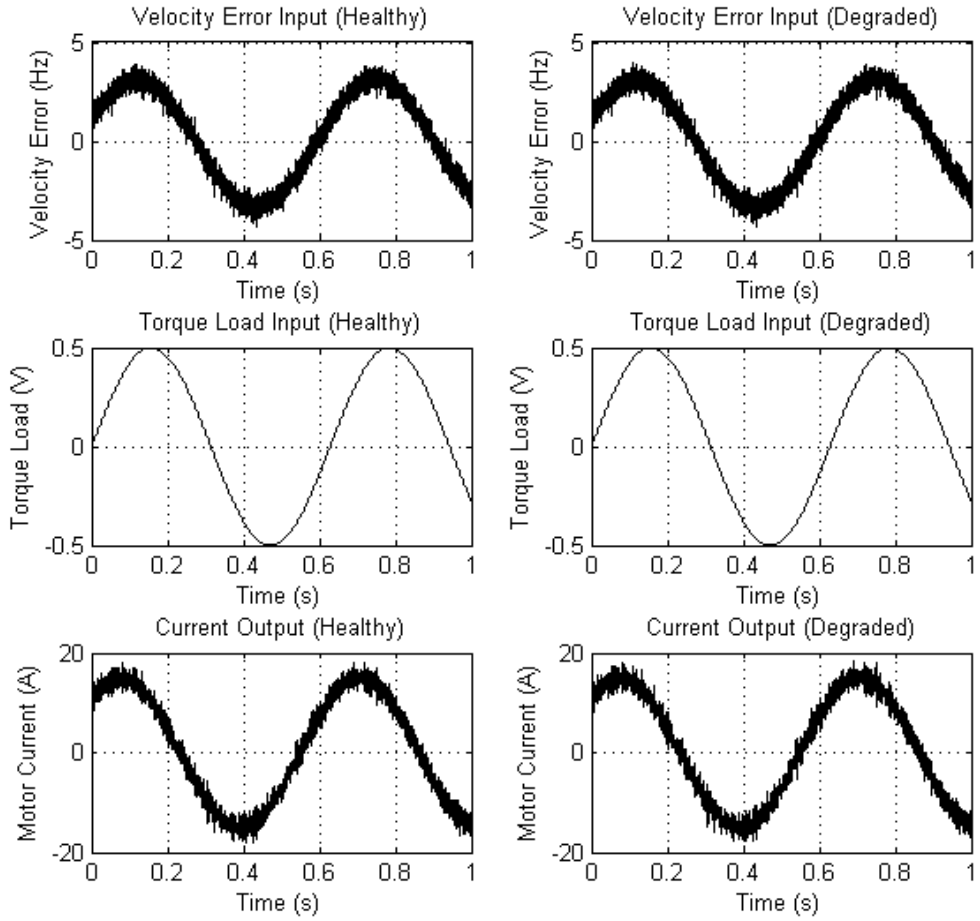

Figure 3.6: Time-plots of the signals generated from the PM DC motor simulation with the first two rows showing the inputs and the third row showing the output for the two health cases considered

3.3 Model Structure Selection

Once the input/output data are chosen, model order selection is the next step. An initial estimate of the model order is obtained by estimating all possible model structures using a particular 'training' dataset and computing the value of the normalized quadratic fit between the measured output and the modeled output as shown in Eq. (3.2).

$$\text{Fit}\,\% = 100 \times \left(1 - \left|\frac{y_m - y}{y - \frac{1}{N}\sum_{k=1}^{N} y_k}\right|\right) \tag{3.2}$$

This value is then used to pick the initial model structure that results in a best fit for the dataset and simultaneously for a different 'validation' dataset.

In addition to obtaining a high fit percentage value, residual analysis and pole-zero placement are both important factors that are considered. Therefore, the approach initially places a constraint on the minimum fit percentage needed to proceed with the next steps of feature extraction. Unless the fit exceeds this set threshold, an alternative model structure is analyzed. Additionally, during the choice of the model structure, a residual analysis plot as well as a pole-zero map are generated. The user utilizes these plots and decides whether or not to modify the fit. Parameter estimation is achieved using the method of Instrumental Variables on a linear difference model. For this approach, only the independence test (i.e. the cross-correlation between the residuals and the inputs must be as close to zero as possible within a set threshold) needs to be satisfied [38]. The residual analysis plot displays a 99% confidence region around zero in the form of a yellow box. This represents the range of residual values that have a 99% probability of being statistically insignificant [38]. The pole-zero map shows the location of the model poles and zeros along with confidence intervals corresponding to three standard deviations (a confidence interval in excess of 99 percent). The number of poles and zeros is equal to the number of sampling intervals between the most delayed and least delayed outputs and inputs respectively. For example, in a linear difference model structure such as

$y(t) + a_1 y(t-1) + a_2 y(t-2) = b_1 u_1(t-1) + b_2 u_1(t-2) + c_1 u_2(t-1)$, there are two poles and one zero from u_1 to y and just the two poles from u_2 to y assuming the sampling interval is one second. Overlapping or close proximity of the confidence intervals of a pole-zero pair indicates model order reduction is possible.

Example: DC Motor Seeded with a Single-Point Defect

In order to demonstrate the advantages of using this fault detection methodology, two cases are considered - one that utilizes PCA for feature extraction and another that doesn't. The primary benefit of PCA is that it allows the user to choose a high order model to obtain a good representation of the system and collect information about any faults that are present while maintaining a small number of features to make classification more convenient. Since PCA suppresses a loss of information when reducing the dimension of a dataset, choosing a high order model, generating more than two features and then applying PCA on them to obtain two new features in the form of principal components is expected to yield an improvement during classification. To demonstrate this, two model structures are derived for the example in this chapter. One of them consists of two 'a' parameters while the other consists of three. Since PCA is employed when there are more than two parameters in order to reduce the number of features for classification to two, a comparison across the two cases reveals the benefits that PCA offers. Before proceeding, it is important to understand the use of a particular notation. The order vector is always a $[1 \times 5]$ matrix (i.e. a row vector). The first element is the order of the output terms, the next two denote the orders of each of the inputs while the last two denote the delays of each of the inputs. For example, for a model structure of the form $y(t) + a_1 y(t-1) + a_2 y(t-2) = b_1 u_1(t-1) + b_2 u_1(t-2) + c_1 u_2(t-3)$, the order vector is published as [2 2 1 1 3]. This format is used in the rest of this thesis.

Based on the plots in Fig. 3.7, it is observed that choosing a lower order model generates a less accurate fit although the residual analysis is very good. This is the best second order model structure that fits the data. Additionally, when increasing

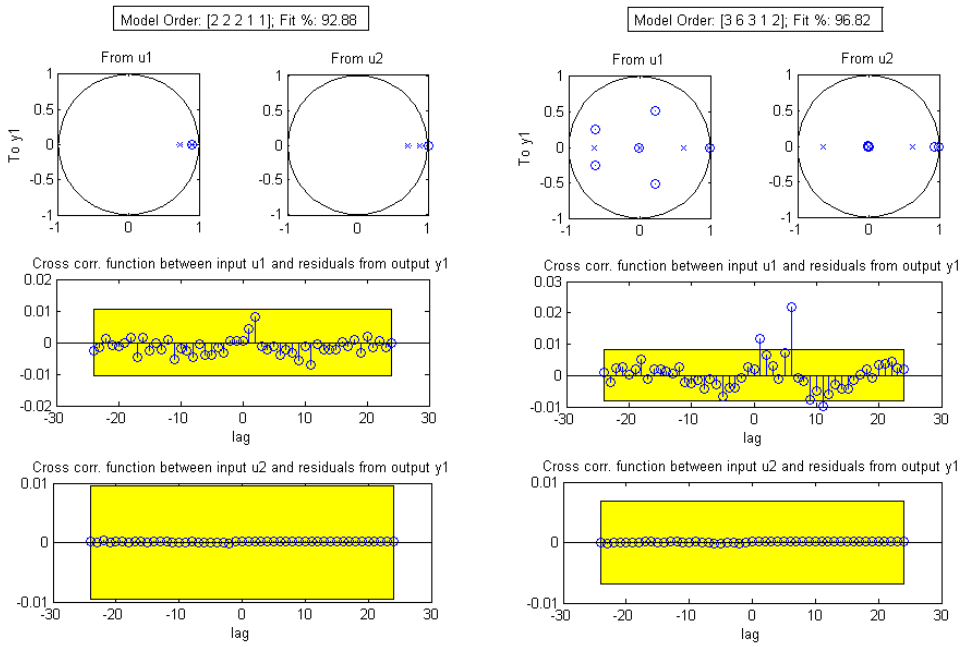

Figure 3.7: Order selection analysis plots - Top: Pole-zero maps; Middle: Cross-correlation plots between u_1 and residuals; Bottom: Cross-correlation plots between u_2 and residuals

the order of the output terms to three, a marked improvement in the fit percentage is noticed. The residual analysis showing cross correlation between u_1 and the residuals however is not perfect. Nevertheless, the two points outside the confidence interval merely indicate that the output $y(t)$ that originates from the inputs $u_1(t-1)$ and $u_1(t-6)$ is not perfectly described by the model (once again assuming a sampling interval of one second). However, since the deviations outside the 99% confidence interval are marginal and is still offset by the 4% increase in the model fit, the selected model structure is acceptable. Lastly, an analysis of the pole-zero map for both cases indicates that all the poles and zeros are separated from one another by more than 3 standard deviations of each value, indicating that the step-up to three 'a' terms is not redundant.

Although a similar approach is followed in Chapters 4 and 5, it is noted that when comparing certain cases where the order is larger than two, a two-dimensional feature plot is generated even when PCA is not used. This is done by assuming that the third

'a' parameter is less significant compared to the other two when it comes to comparing them for differences, as shown in a similar case in the text by Keesman [37]. This is done to provide uniformity across the results sections of those chapters.

3.4 Parameter Estimation

Once the required model order is selected, the approach computes the parameter estimates using the Instrumental Variables method according to the theory in Chapter 2. Two features are then selected for classification. This is done because it is observed that in most cases, the model-order lies in the neighborhood of two output terms (i.e. there are usually only one to three output terms used to satisfy the model structure selection criteria). Furthermore, if the order selection routine chooses one output term and (n_b, n_c) input terms for each of the inputs, then the approach selects one output term and the n_b terms as these terms are expected to carry most information about the fault. The n_c terms are not selected because this coefficient is associated with the external torque load which, in general, is not part of the feedback system and is therefore unlikely to carry information about the fault. If the order selection routine selects two or more output parameter terms, then the algorithm selects both these terms for feature extraction because they are expected to contain most information about the fault.

Example: DC Motor Seeded with a Single-Point Defect

The estimated parameters and the measured and modeled outputs for each of the two cases are presented in Figs. 3.8 and 3.9.

3.5 Feature Extraction

Once the required parameters are selected, feature extraction is performed. The program allows the user to choose whether or not to implement PCA to extract the required features. It is once again pointed out that the PCA techniques are used,

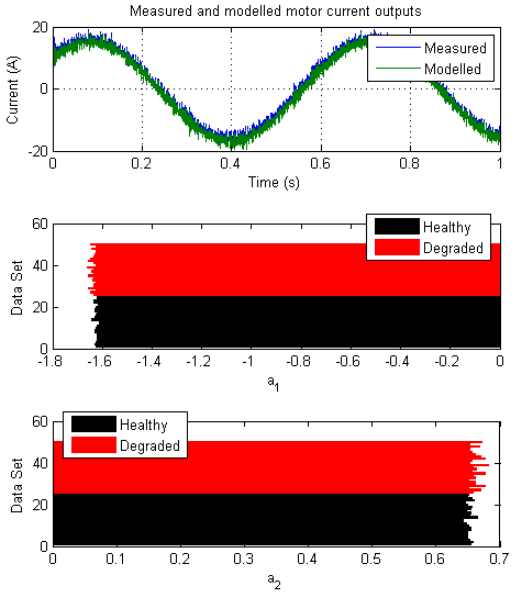

Figure 3.8: Modeled and measured motor current outputs of a healthy PM DC motor model and the model parameter estimates for 25 data sets each of the healthy and degraded conditions of the same system using a model order of [2 2 2 1 1]

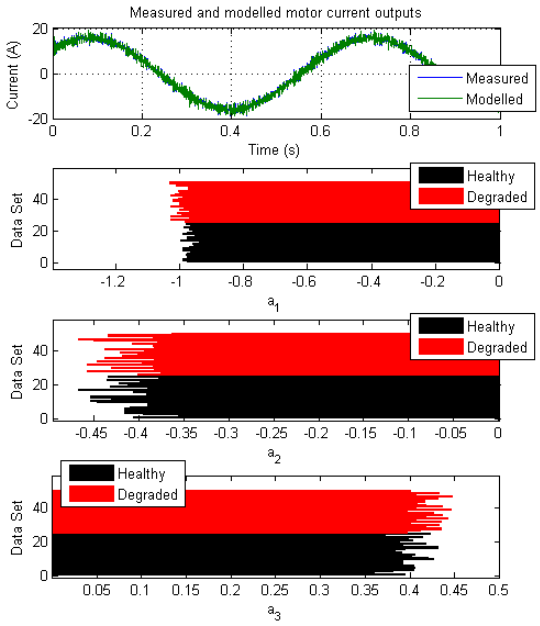

Figure 3.9: Modeled and measured motor current outputs of a healthy PM DC motor model and the model parameter estimates for 25 data sets each of the healthy and degraded conditions of the same system using a model order of [3 6 3 1 2]

both for dimension reduction and, when there is no reduction, to transform a pair of linearly correlated parameter estimates to two linearly uncorrelated features such that the variance of the parameter estimates is maximized along one of the feature axes. Once the features are computed, a feature plot is generated in the form of a scatter diagram showing points from the feature vectors that belong to classes of different health conditions.

Example: DC Motor Seeded with a Single-Point Defect

In the case where PCA is applied, the following table shows the eigenvalues and the contribution of the corresponding eigenvector to the total variance of the data set. Additionally, Fig. 3.10 compares the original parameter estimates alongside the

Table 3.1: Computed Eigenvalues and Eigenvectors during PCA

Eigenvector	Eigenvalue	Contribution %
e_1	0.0015	80.57
e_2	3.6167×10^{-4}	19.43
e_3	1.0872×10^{-7}	0.00584

calculated features. The features from both cases (the parameter estimates for the case with model order [2 2 2 1 1] are the features themselves) are then plotted on the 2D scatter diagram shown in Fig. 3.11.

3.6 Classification and Validation

As indicated in Chapter 2, Bayesian classification is employed to segregate the features into different classes. The features obtained from the training data are grouped in the same order as the data and the corresponding health condition labels are paired to them to create the dataset required for drawing the decision boundaries described in Chapter 2. Once the features from the training data set are used to create the classification bounds, a 'validation' dataset is used to estimate model parameters, extract

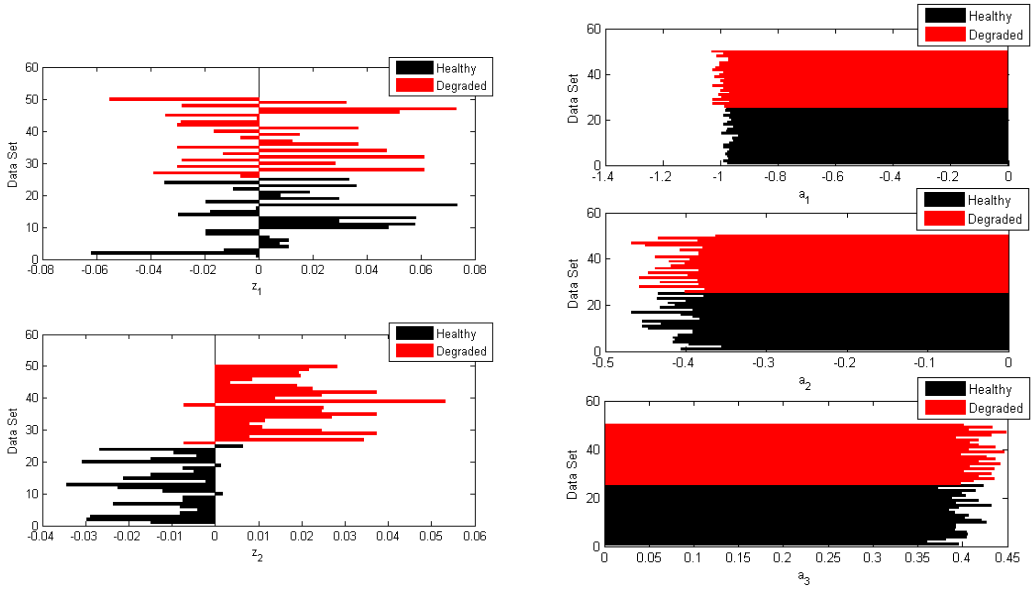

Figure 3.10: Comparison of features calculated using PCA (Left) to original parameter estimates (Right) for case with model order [3 6 3 1 2]

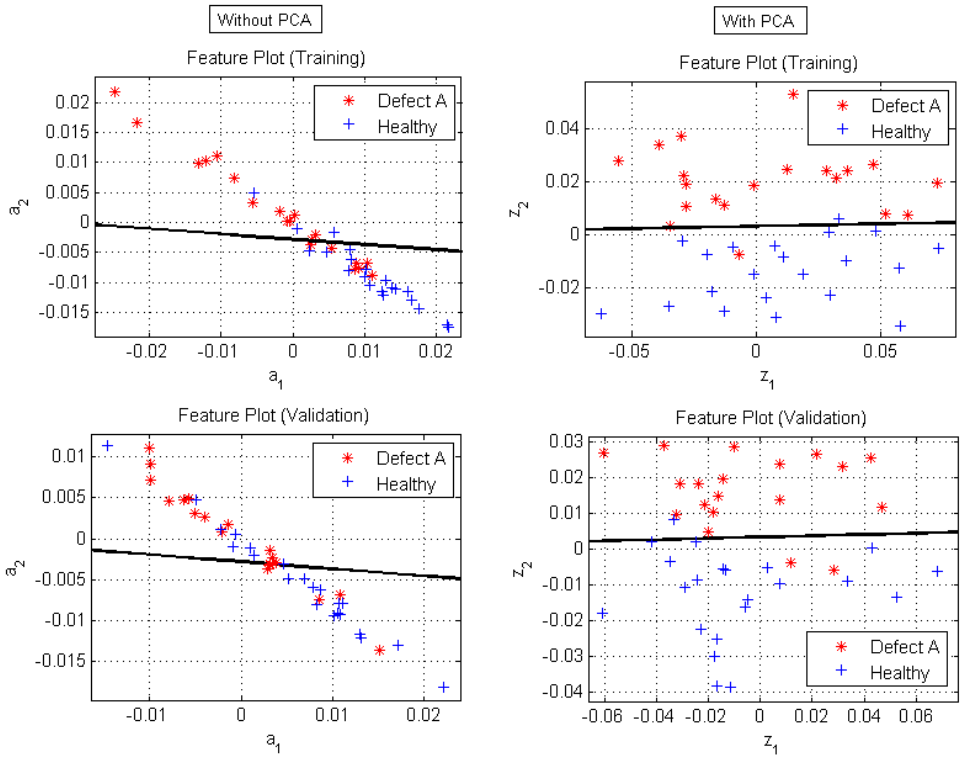

Figure 3.11: Classification plots - Left: Model order - [2 2 2 1 1] without PCA for feature extraction; Right: Model order - [3 6 3 1 2] with PCA for feature extraction

features and correctly place them in the appropriate class. The transformation matrix used to transform the parameter estimates from the training data into features using PCA is stored and applied on parameter estimates from the validation data set. This eliminates the need to recompute a transformation matrix using more than one set of validation data. Once the features are extracted from the validation data set, they are classified in accordance with the theory behind Bayesian classification provided in Chapter 2. For the proposed approach, a uniform cost function is assumed, the conditional probability density function that a feature vector belongs to a particular health class is normally distributed and the prior probabilities of being assigned to a particular health class is equal for each class.

Example: DC Motor Seeded with a Single-Point Defect

For this example, the feature plots with classification and validation are shown in Fig. 3.11. Table 3.2 summarizes the results observed. The approach is programmed in MATLAB and SIMULINK, the algorithms and block diagrams of which are available in the appendices.

It is therefore concluded that using a higher order model with PCA for feature

Table 3.2: Classification and Validation Results

Model Order	Features	Misclassification %
[2 2 2 1 1]	a_1 and a_2	30
[3 6 3 1 2]	z_1 and z_2	6

extraction produces superior results as opposed to the previous approach of using the parameter estimates themselves as features. The simulation and experimental results shown in the following chapters reflect this conclusion.

Chapter 4

Simulations

The results of the simulations used to build and validate the fault detection algorithms based on the theory of Chapter 2 are presented here. The algorithms are developed, adjusted and validated based on its performance on data from a simulated permanent magnet DC motor with a seeded single-point defect and reduction in lubrication as described in the previous chapter.

For the simulations, a total of 120 data sets are used for both supervised training of the classification algorithm and validation. Amongst each of the 120 data sets, forty sets each from a healthy model, one with a simulated outer-race defect and another with a simulated reduction in lubrication are generated. Twenty sets from each class are then used for supervised training of the classifier while the balance from each health condition are used for validation. The motor and single-point defect parameter values are partially published in Table 2.3 and completely in the corresponding MATLAB program in Appendix B. Consistent with prior works, the output signal used for feature extraction is the motor current signal as the defects are expected to affect it more than they would affect the motor velocity signal [14, 17, 19, 20, 23]. The two input signals considered are the velocity command and the controller input based on the discussion on fault detection in closed-loop systems in Chapter 2. The results using each of these inputs are analyzed and a choice is made based on which offers better performance. The controller input is expected to perform marginally better in general.

Section 4.1 presents the results for the various cases tested. As mentioned earlier,

the major modifications to the fault detection approach compared to the earlier approach used in the example of Chapter 2 are: (i) a thorough analysis of fault detection in closed-loop systems (see Chapter 2) to hypothesize the best possible combination of inputs and outputs for parameter estimation, (ii) use of an intuitive order selection routine to select a model order that yields a fit above a certain minimum acceptable limit while ensuring that results from a residual analysis are not compromised - an effect that might lead to incorrect classification based on model differences other than that due to the inherent faults, and (iii) application of principal component analysis to either take the two available linearly correlated model parameter estimates (in the case of a first-order or second-order linear difference equation as elucidated in the previous chapter) and uncorrelate them by performing a transformation that results in two new features, one of which possesses the largest variance in the data, or achieving the same result after choosing two features from more than two parameters (in the case of higher order models). Therefore, the results in the following section are arrived at by using the approach that incorporates the above modifications. Section 4.2 discusses the inferences drawn from the results. A similar approach is used for the experiments of Chapter 5.

4.1 Results

Note: Amongst the statistics for each case is listed the misclassification percentage as a measure of the classification performance. In all cases, the output signal 'y_1' is motor current and the external load torque is always used as an input 'u_2' during parameter estimation. This provides better parameter estimation performance. Two types of 'u_1' inputs are used. They are presented separately under Conditions 1 and 2 in Sections 4.1.1 and 4.1.2.

4.1.1 Condition 1

In this section, input 'u_1' is the Reference Velocity signal. The time plots of the input and output signals are shown in Fig. 4.1. Two model structures are tested and

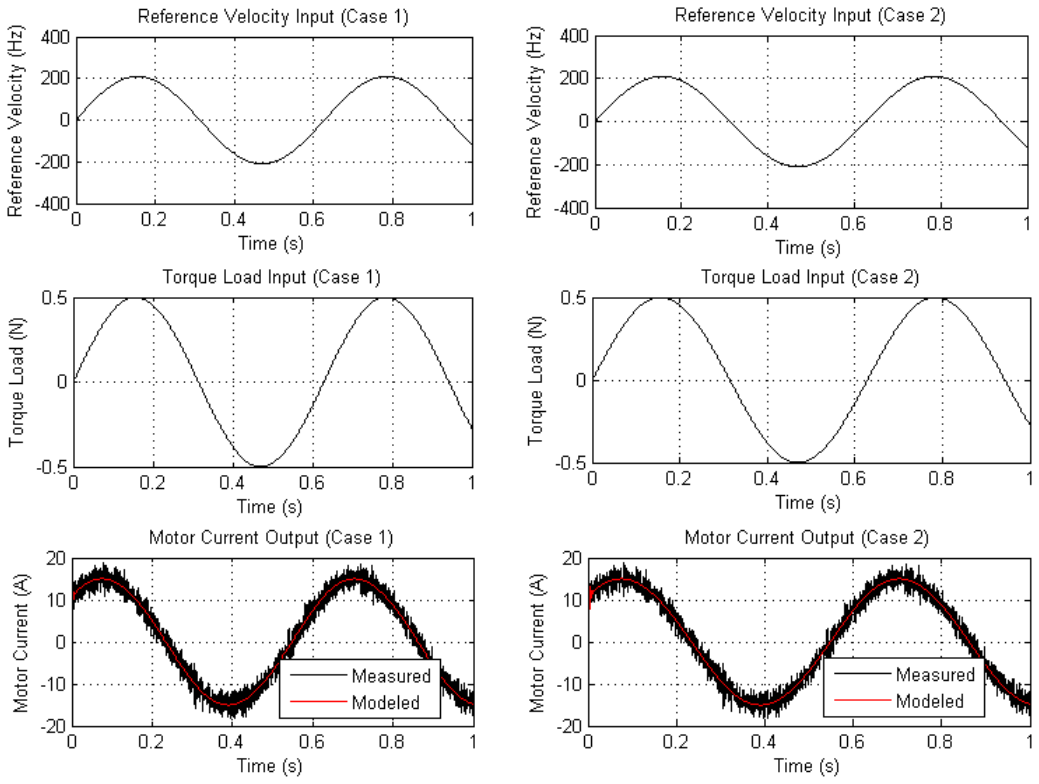

Figure 4.1: Time plots for Condition 1. Left - Case 1: Order Vector [2 1 2 1 1]; Right - Case 2: Order Vector [3 1 2 1 1]

the results, after applying the developed algorithm in a manner indentical to that shown in Chapter 3, are presented as Cases 1 and 2.

Table 4.1: Case 1: Order Vector: [2 1 2 1 1]; Fit: 86.58 %

Associated Plots: Figs. 4.2 to 4.4

Case	Feature Extraction	Misclassification %
Training	Without PCA	13.33
Validation	Without PCA	33.33
Training	With PCA	13.33
Validation	With PCA	28.33

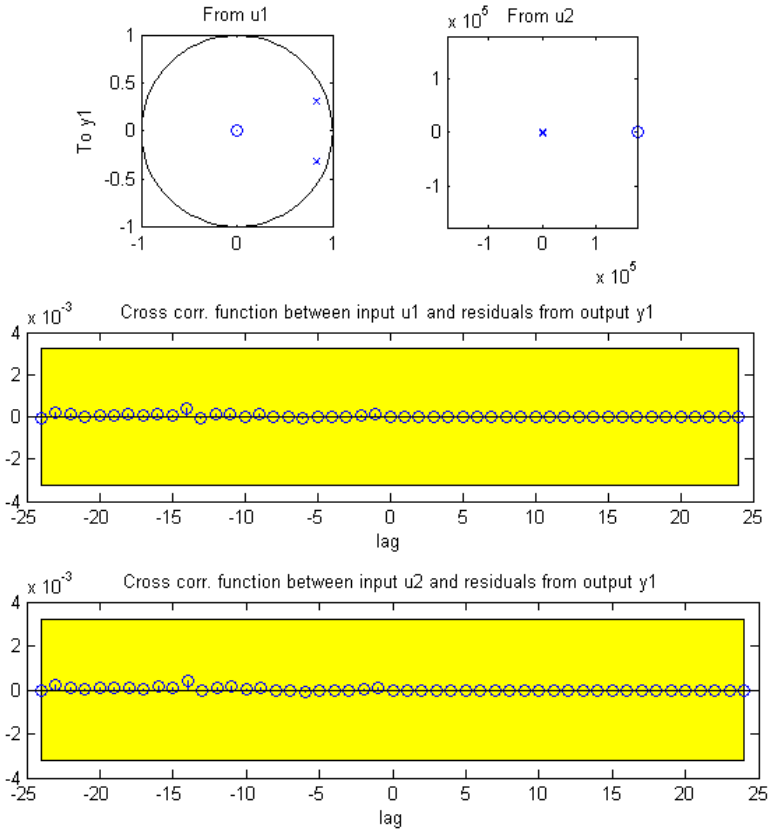

Figure 4.2: Order selection analysis plots - Top: Pole-zero map; Middle: Cross-correlation between u_1 and residuals; Bottom: Cross-correlation between u_2 and residuals

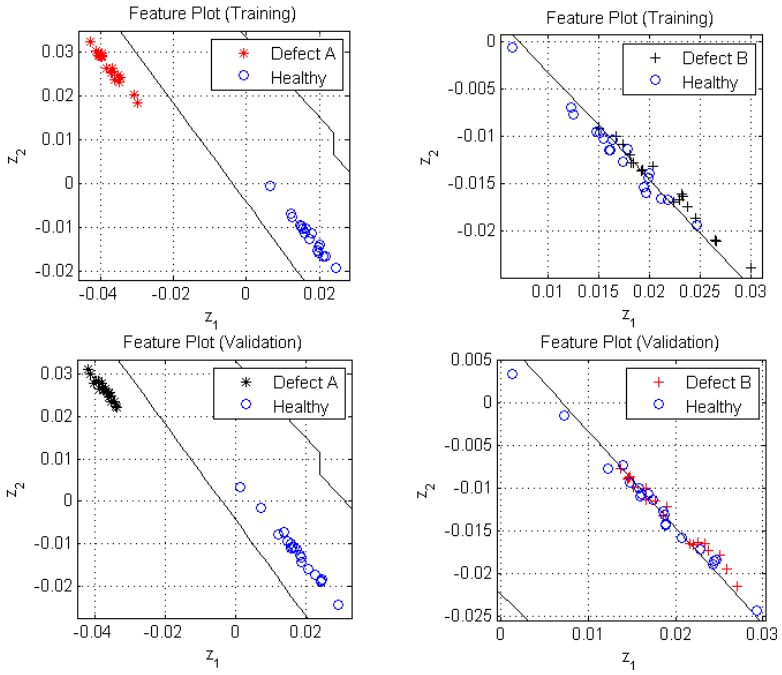

Figure 4.3: Feature plots - Top: Training Cases; Bottom: Validation Cases

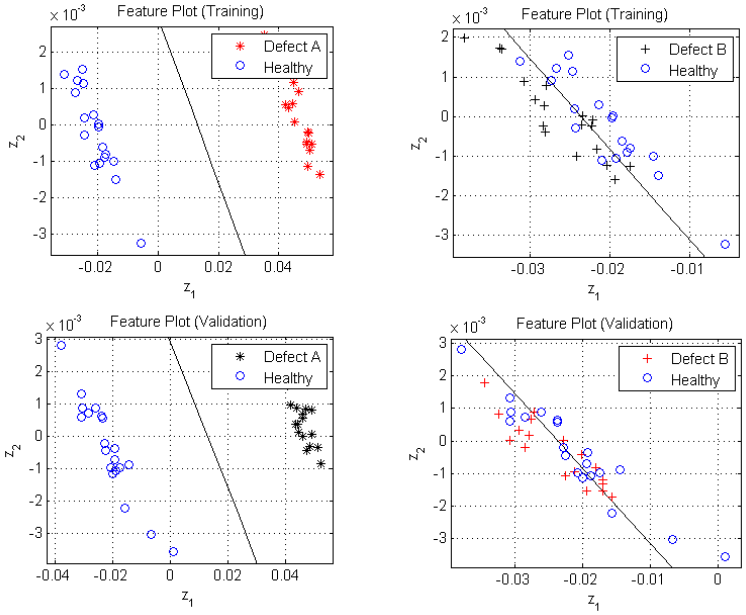

Figure 4.4: Feature plots - Top: Training Cases; Bottom: Validation Cases

Table 4.2: Case 2: Order Vector: [3 1 2 1 1]; Fit: 86.48 %

Associated Plots: Figs. 4.5 to 4.7

Case	Feature Extraction	Misclassification %
Training	Without PCA	13.33
Validation	Without PCA	21.67
Training	With PCA	13.33
Validation	With PCA	15

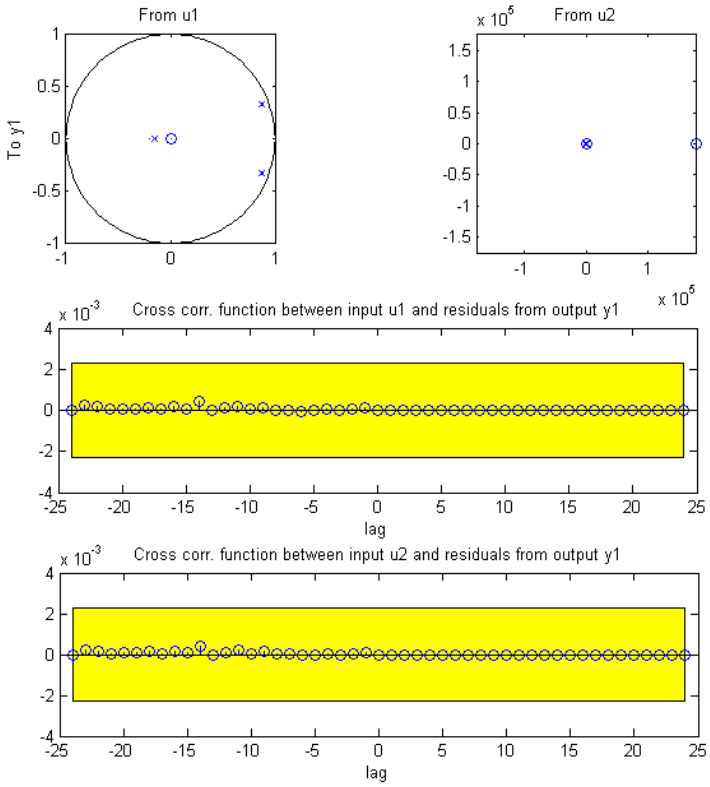

Figure 4.5: Order selection analysis plots - Top: Pole-zero map; Middle: Cross-correlation between u_1 and residuals; Bottom: Cross-correlation between u_2 and residuals

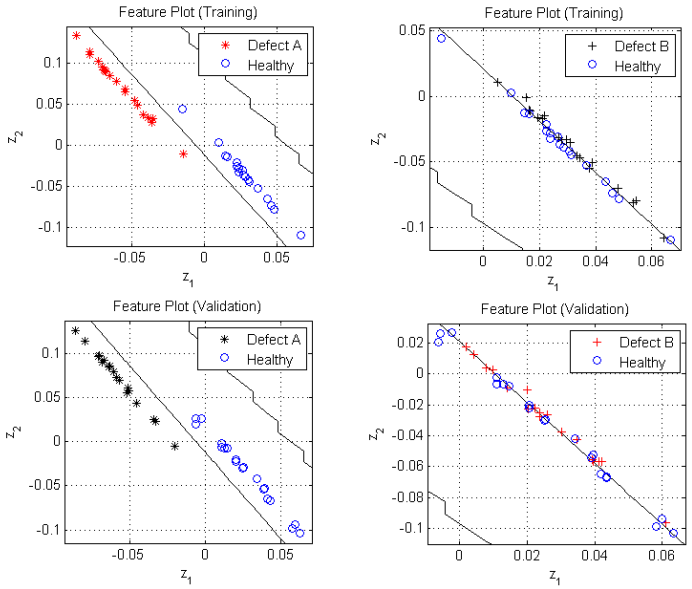

Figure 4.6: Feature plots - Top: Training Cases; Bottom: Validation Cases

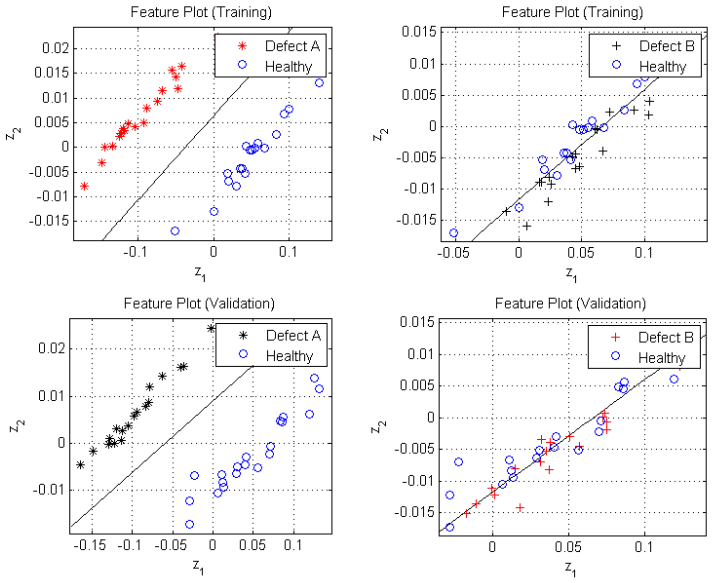

Figure 4.7: Feature plots - Top: Training Cases; Bottom: Validation Cases

4.1.2 Condition 2

In this section, input 'u_1' is the Velocity Error signal. The time plots of the input and output signals are shown in Fig. 4.8. Two model structures are tested and the

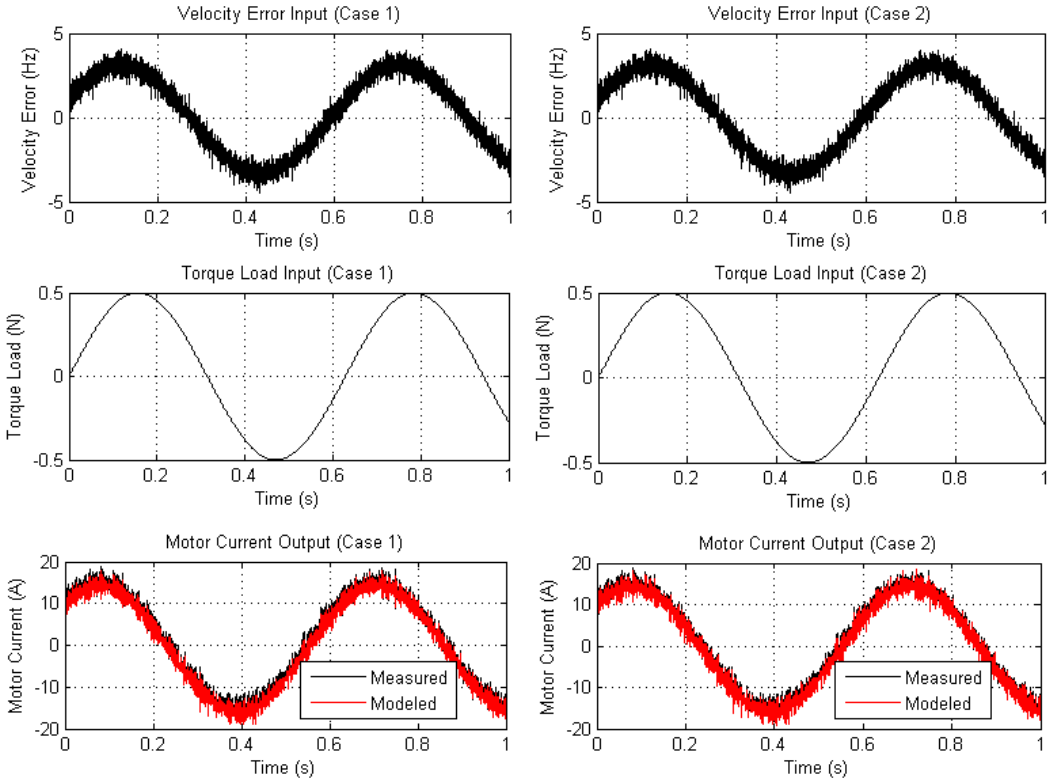

Figure 4.8: Time plots for Condition 2. Left - Case 1: Order Vector [2 2 2 1 1]; Right - Case 2: Order Vector [3 3 3 1 1]

results, after applying the developed algorithm in a manner indentical to that shown in Chapter 3, are once again presented as Cases 1 and 2.

Table 4.3: Case 1: Order Vector: [2 2 2 1 1]; Fit: 92.05 %

Associated Plots: Figs. 4.9 to 4.11

Case	Feature Extraction	Misclassification %
Training	Without PCA	13.33
Validation	Without PCA	21.67
Training	With PCA	15
Validation	With PCA	18.33

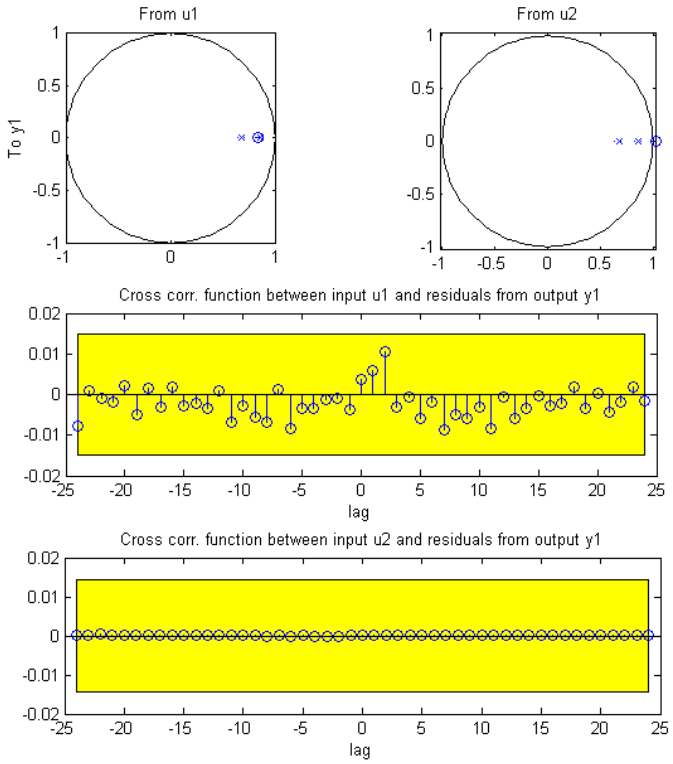

Figure 4.9: Order selection analysis plots - Top: Pole-zero map; Middle: Cross-correlation between u_1 and residuals; Bottom: Cross-correlation between u_2 and residuals

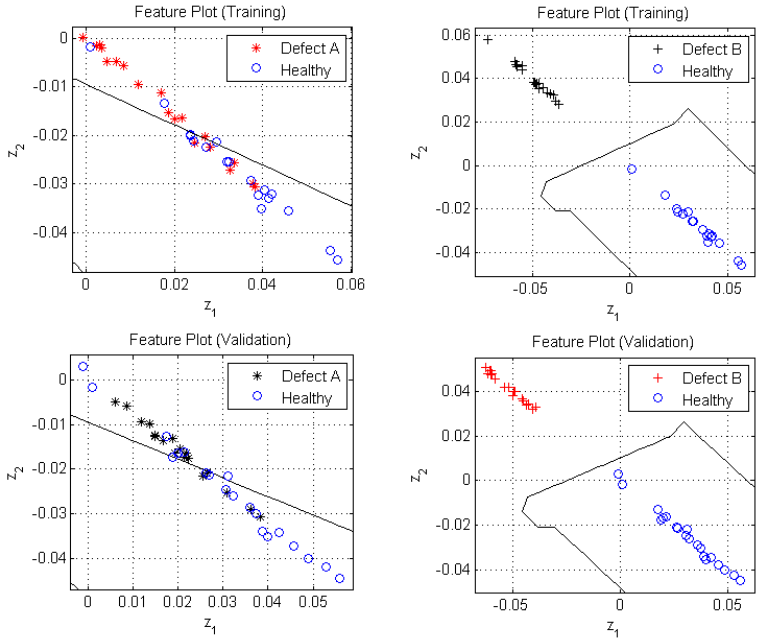

Figure 4.10: Feature plots - Top: Training Cases; Bottom: Validation Cases

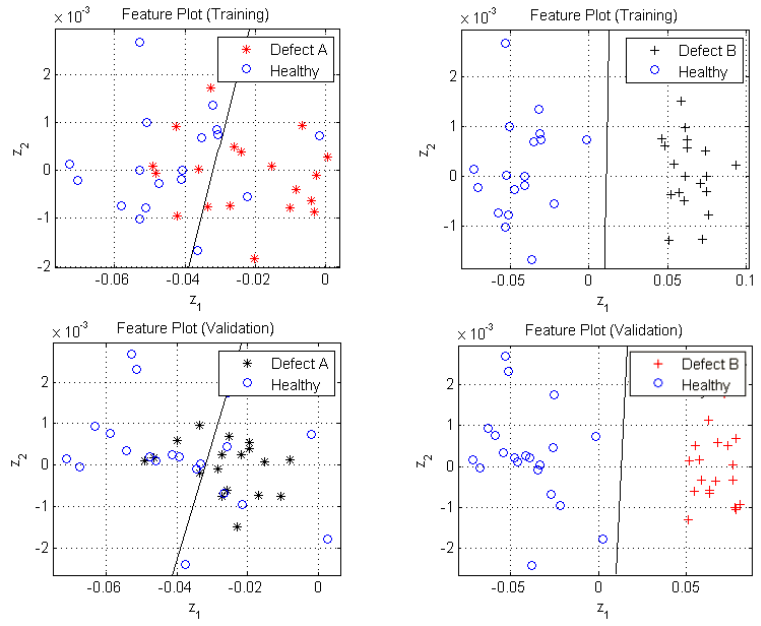

Figure 4.11: Feature plots - Top: Training Cases; Bottom: Validation Cases

Table 4.4: Case 2: Order Vector: [3 3 3 1 1]; Fit: 93.45 %

Associated Plots: Figs. 4.12 to 4.14

Case	Feature Extraction	Misclassification %
Training	Without PCA	13.33
Validation	Without PCA	18.33
Training	With PCA	16.67
Validation	With PCA	15

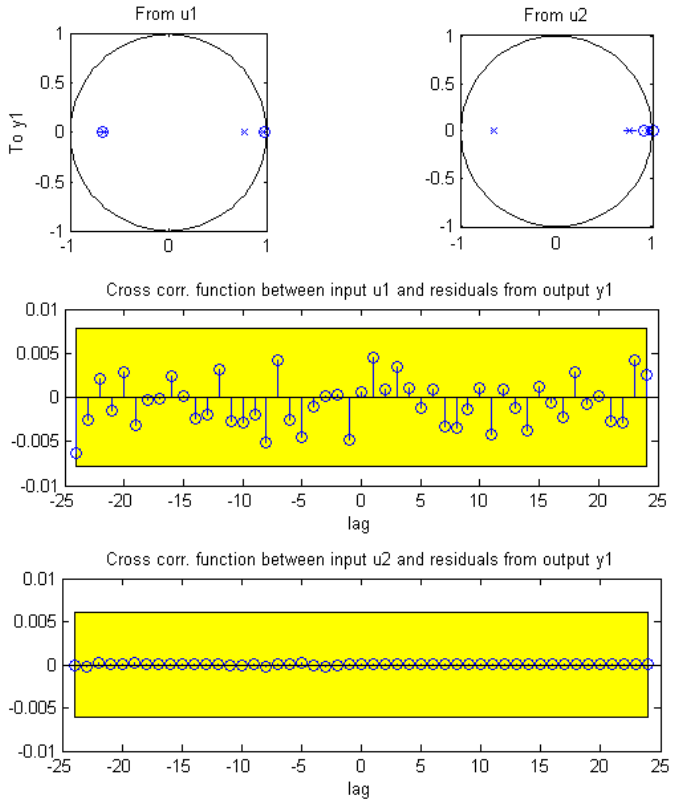

Figure 4.12: Order selection analysis plots - Top: Pole-zero map; Middle: Cross-correlation between u_1 and residuals; Bottom: Cross-correlation between u_2 and residuals

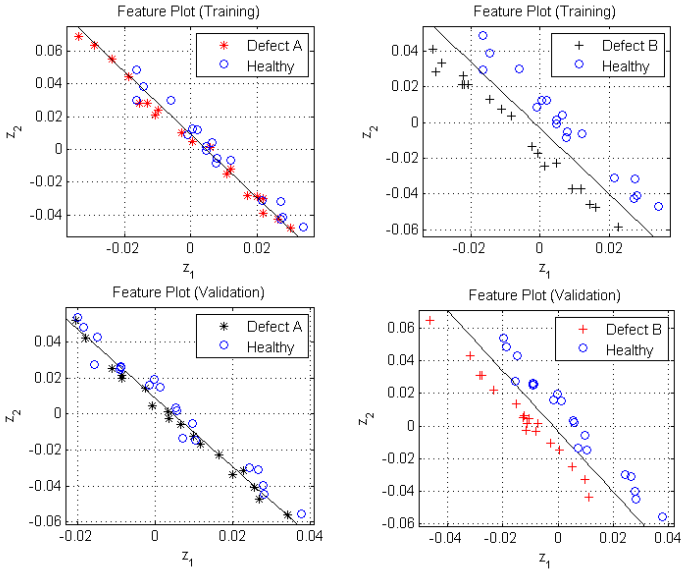

Figure 4.13: Feature plots - Top: Training Cases; Bottom: Validation Cases

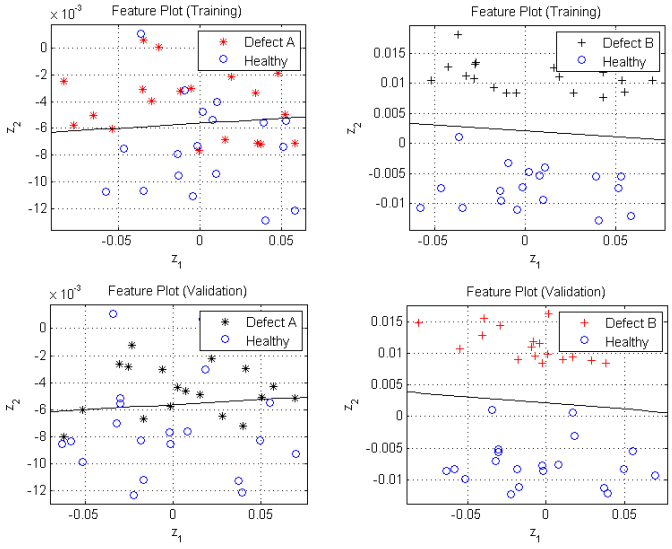

Figure 4.14: Feature plots - Top: Training Cases; Bottom: Validation Cases

4.2 Inferences

When using reference velocity as the input, Defect A (the single-point defect) is effectively classified while Defect B is not and when using velocity error as the input, Defect B (the generalized roughness defect) is effectively classified while Defect A is not. This is partially explained by observing the fit percentages. When reference velocity is the input, the fit percentage is smaller than when the velocity error is used as the input. This particular difference is likely attributed to the nature of the model structure and estimation approach utilized. Since an ARX model structure is used, the noise model is more accurately estimated when the majority of the noise occurs at the input. When using the reference velocity as the input, all the noise in the dataset appears at the output resulting in a lower fit as compared to using the velocity error as the input. Since the velocity error is the difference between the noise-free velocity reference signal and the noisy motor velocity feedback, the input now carries a significant amount of the noise that actually originates at the output, thereby resulting in a better model fit. Further evidence of this effect is noticed by observing the residual analysis plots in each case - especially the cross-correlation plot between the primary input and the residuals. The second input - the external load - is not part of the closed loop system and is always noise-free.

As shown earlier in Fig. 3.3, the single-point defect and the generalized roughness defect affect the output differently. While the single point defect generates minor periodic fluctuations in the output, the overall signal amplitude, frequency and phase are largely unaffected. However, in the case of the generalized roughness defect, a difference in the signal amplitude is observed. This difference is likely captured when calculating the velocity error and this is a plausible explanation as to why the generalized roughness type defect is better approximated when using velocity error as the input. Furthermore, it is proposed that the classification accuracy when using the velocity error as input decreases in the case of Defect A because of the increased effect of the noise in the parameter estimation process. Observing the residual plots in the two cases, it is seen that when Defect A is successfully classified, the cross-correlation

function between both inputs and the residuals are nearly always zero, indicating that all information from the input to the output is properly captured by the parameters. It is expected that this is a key requirement when detecting single-point defects that have only a marginal and periodic effect on the output signal.

The application of PCA for feature extraction produces improved results. The imperfections in classification suggest that an improvement in parameter estimation approaches is required for more efficient functioning of the algorithm. It is also noted that the nature of the defects created in the simulations are not completely realistic and no general conclusions are made regarding the ability of the PCA based method to detect specific types of faults. The results here show that PCA as a feature extraction technique generates improved features to allow more effective classification, thereby providing scope for future investigation. Furthermore, implementation of a more accurate classification routine (perhaps by incorporating a more realistic cost function) is likely to produce enhanced results in the case of features extracted using PCA due to the generally larger spread within and across features originating from systems of different health conditions. This is noticeable in the various feature plots presented earlier.

The next section tests the algorithm's ability to perform on data collected from a real EMA system.

Chapter 5

Experiments

This chapter presents the fault detection results from the experiments conducted on the Moog MaxForce EMA. Since experiments were conducted both at Moog Inc. and RIT on two different test rigs, the chapter is divided into two main sections. Section 5.1 presents the results from the EMA tested on Moog's rig while Section 5.2 presents the results from the EMA tested on RIT's rig.

5.1 EMA at Moog

The Moog EMA test rig shown in Fig. 5.1 consists of the EMA, a hydraulic load actuator, a test fixture, a Moog T200 motor controller, power and signal conditioning equipment and a data acquisition system. The EMA is mounted vertically in the test fixture and a controlled load is applied through the hydraulic actuators. Position, velocity, load and vibration sensors are used to capture relevant information from the system.

The EMA laboratory signal diagram is shown in Fig. 5.2. The dSpace console,

Table 5.1: Moog Maxforce EMA Technical Specifications

Parameter	Value
Stroke	6 in.
Force Capability	3700 lbf. at 15.6 A_{rms}
Peak Motor Velocity	4572 rpm at 220 V AC
Number of Motor Poles	12

along with MATLAB and SIMULINK, is used to implement the EMA position and load actuator controller and is also used for data aquisition. The LVDT provides

position feedback and the differential pressure transducer on the hydraulic load actuator provides load feedback. The accelerometers measure vibration. The position controller (part of the dSpace console) outputs a velocity command signal to the motor controller. The controller uses the resolver and motor current feedback signals to produce compensated voltage commands to the motor coil windings.

In the case of the Moog MaxForce EMA, two sets of data collected by Moog Inc.

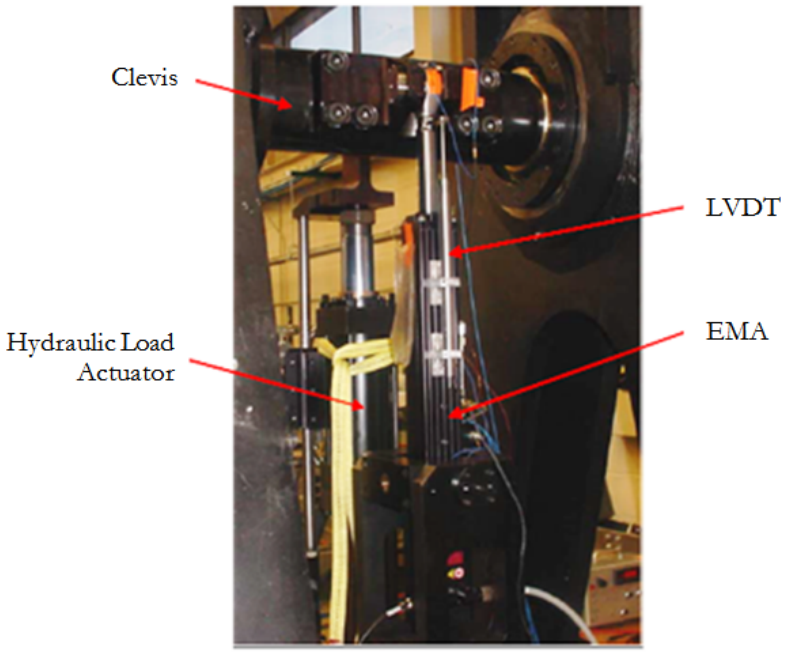

Figure 5.1: Moog Maxforce EMA Test Rig (Courtesy Moog Inc., East Aurora, NY)

is provided for use, one each for training and validation. Amongst the signals collected, those of importance to the present work include position command, position feedback, motor quadrature current and external load. Since the dataset is provided directly by Moog, specific details regarding the aquisition process is not available. The data sets have a sampling interval of $8.3\dot{3} \times 10^{-4}$ seconds. Since Moog did not generate these data sets specifically for the parameter estimation based application (its original application was for an alternative frequency-based approach), a closer investigation into the nature of the signal reveals that the sampling interval is nearly

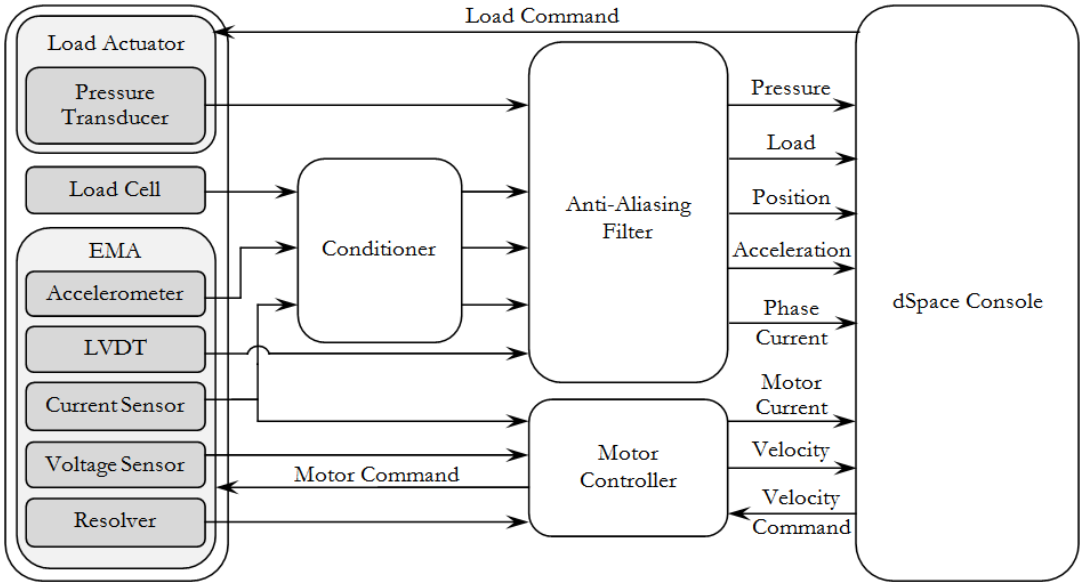

Figure 5.2: EMA Laboratory Signal Diagram

three times higher than what is required to capture the highest frequency fluctations in the data. This is highly undesirable for parameter estimation based approaches as the lack of change in the signal over three samples results in computational issues. Figure 5.3 shows a zoomed in version of one of the signals captured by Moog. The unnecessarily high sampling rate is evident. It is also seen from Fig. 5.3 that the original sampling rate causes sections where there is no change in the signal across three samples and then a sudden change across one sample. This may lead to poor estimation results that subsequently affects fault detection. After resampling however, both theses issues are rectified. It is important to notice that the resampling process does not just involve discarding two sample points periodically starting with the second. Doing so could lead to signal aliasing should there be any fluctations of higher frequencies. Therefore to prevent this, MATLAB 's built-in function 'resample' is used which applies an anti-aliasing filter (in this case, a particular form of low pass filter [38]) prior to resampling.

The number of input and output data options are limited for data collected from the EMA. Amongst the two possible output signals, namely motor velocity and motor

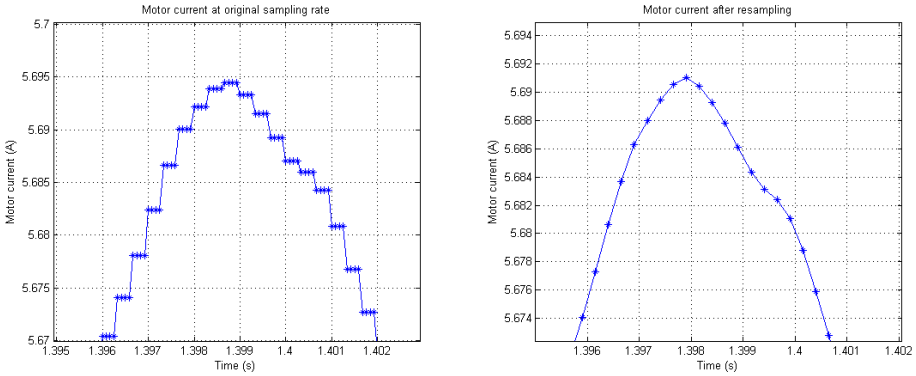

Figure 5.3: Left: Zoomed in version of the motor quadrature current signal captured from the MaxForce EMA by Moog shows a sampling rate three times faster than the optimum required for effecive parameter estimation; Right: The same section of the signal after resampling

quadrature current, the current is used because it is based on validated and verified results from previous works that motor current in the case of brushless motors such as the PMSM in the MaxForce EMA carry information about bearing defects - the type of defect addressed in this work. A short explanation as to why the fault signature of single-point defects is reflected in the motor phase currents (and therefore in the quadrature current as well) is that such a defect causes vibrations in the motor shaft which result in minor deflections in the shaft's axis. Since the shaft is coupled to the rotor, these shifts cause minor variations in the rotors position relative to each of the windings, thereby causing minor variations in the magentic flux in the rotor-stator system resulting in certain characteristic fluctuations in the current drawn by each of the stator windings within the motor. This is why there is a direct relationship between the vibration frequency due to the fault and the frequency reflected in the current signal as indicated by Eqs. (1.1), (2.52) and (3.1).

Sections 5.1.1 to 5.1.5 present for five different conditions. Each condition corresponds to a different dataset. In all cases, the output signal 'y_1' is motor quadrature current and the external load torque is always used as input 'u_2' during parameter estimation. This provides better parameter estimation performance. Additionally, each condition presents two cases of results using reference position and position error as

the two possible input signals for 'u_1'.

5.1.1 Condition 1

The time plots for this dataset are shown in Fig. 5.4.

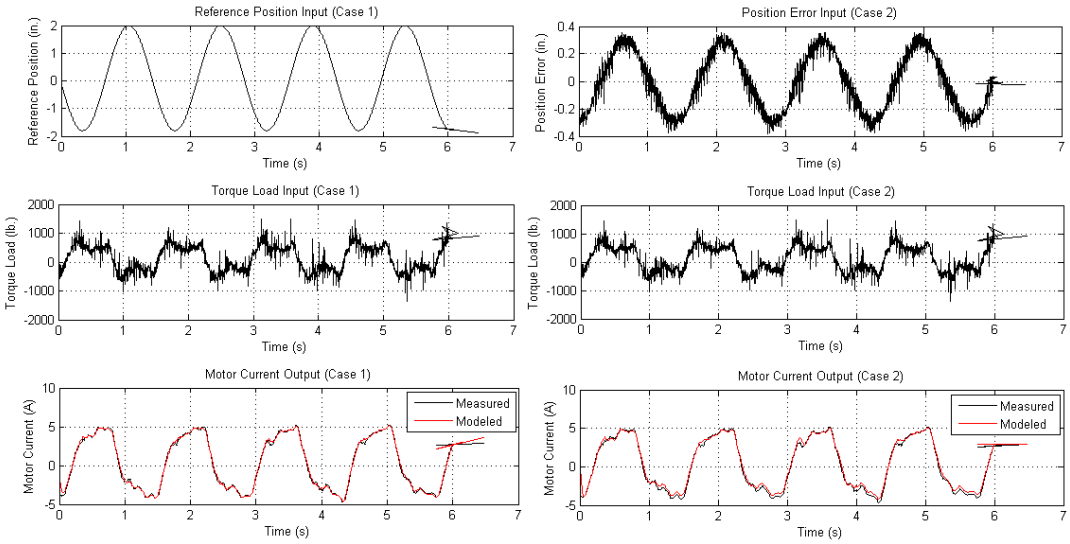

Figure 5.4: Time plots for Condition 1. Left - Case 1: Reference position input; Right - Case 2: Position error input

Table 5.2: Case 1: Input u1 - Reference Position; Order Vector: [1 2 1 1 1]; Fit: 93.24 %

Associated Plots: Figs. 5.5 to 5.7

Case	Feature Extraction	Misclassification %
Training	Without PCA	50
Validation	Without PCA	40
Training	With PCA	40
Validation	With PCA	35

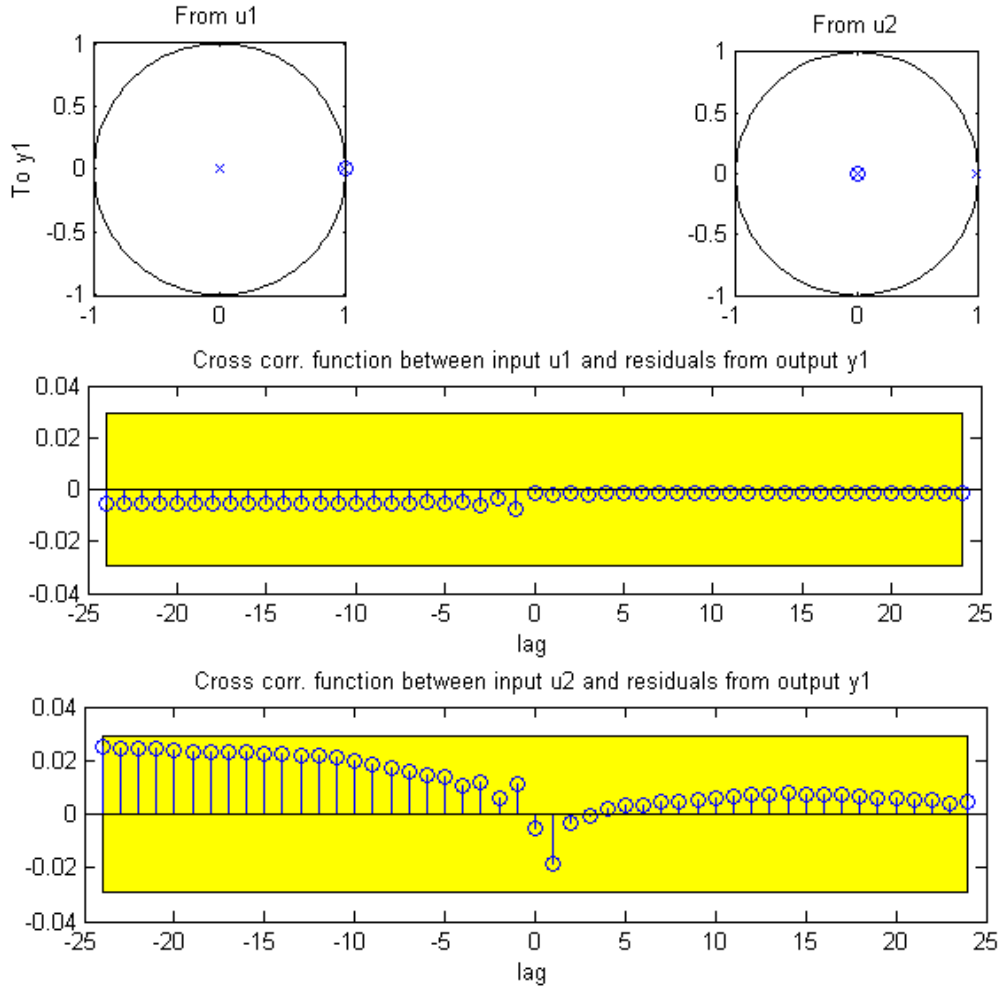

Figure 5.5: Order selection analysis plots for Case 1 - Top: Pole-zero map; Middle: Cross-correlation between u1 and residuals; Bottom: Cross-correlation between u2 and residuals

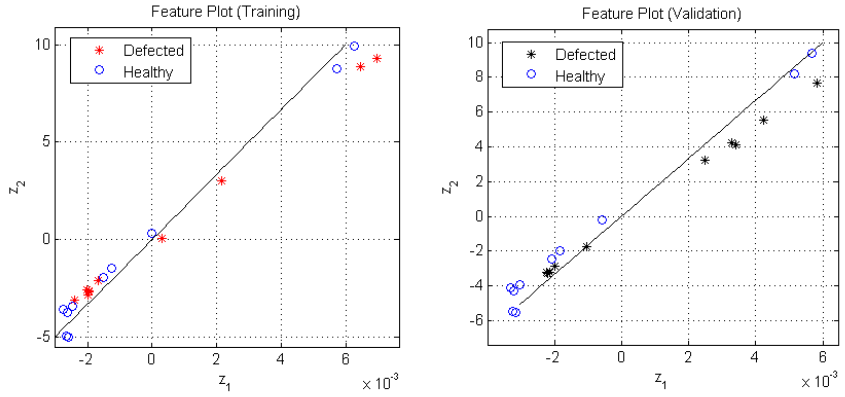

Figure 5.6: Feature plots for Case 1 without PCA - Left: Training Cases; Right: Validation Cases

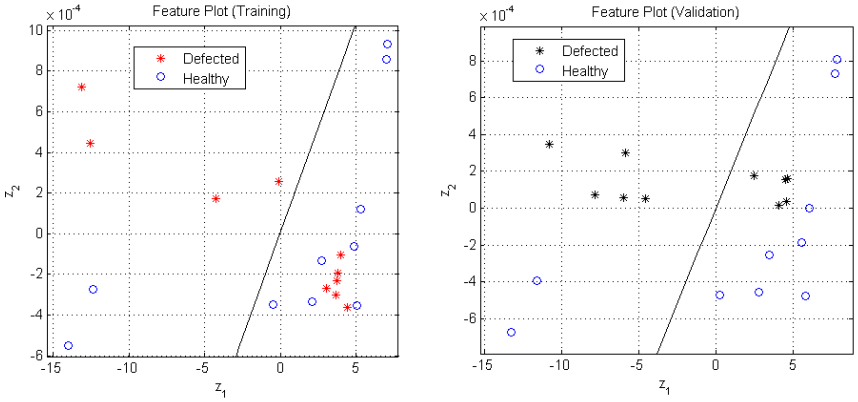

Figure 5.7: Feature plots for Case 1 with PCA - Left: Training Cases; Right: Validation Cases

Table 5.3: Case 2: Input u1 - Position Error; Order Vector: [3 4 4 3 3]; Fit: 89.21 %

Associated Plots: Figs. 5.8 to 5.10

Case	Feature Extraction	Misclassification %
Training	Without PCA	50
Validation	Without PCA	60
Training	With PCA	35
Validation	With PCA	35

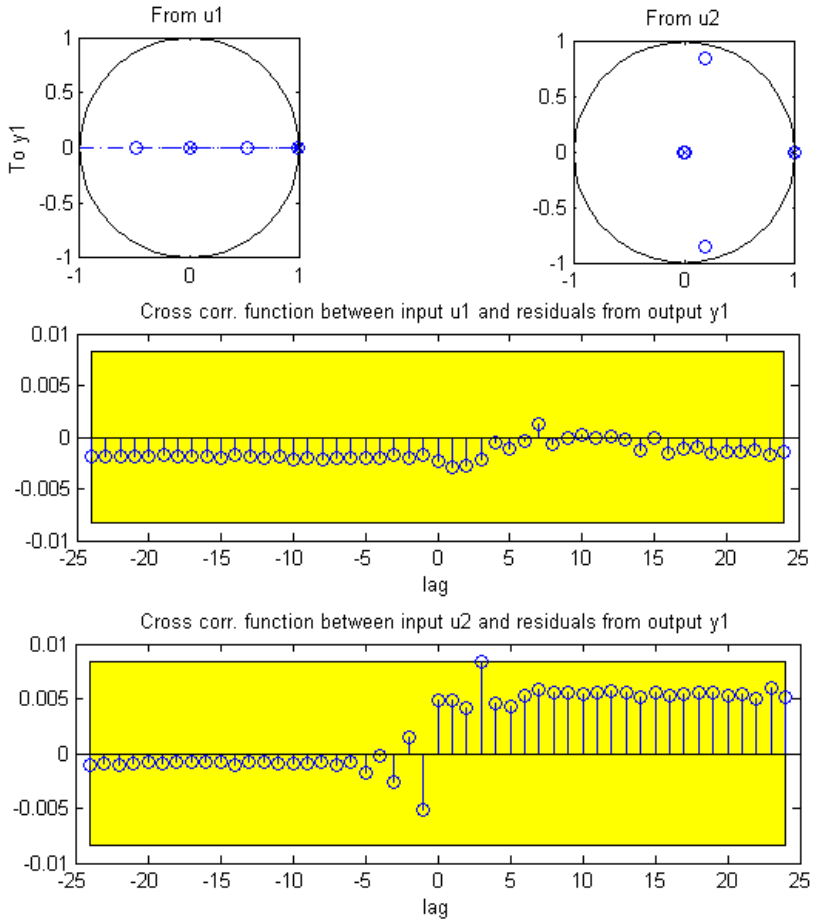

Figure 5.8: Order selection analysis plots for Case 2 - Top: Pole-zero map; Middle: Cross-correlation between u1 and residuals; Bottom: Cross-correlation between u2 and residuals

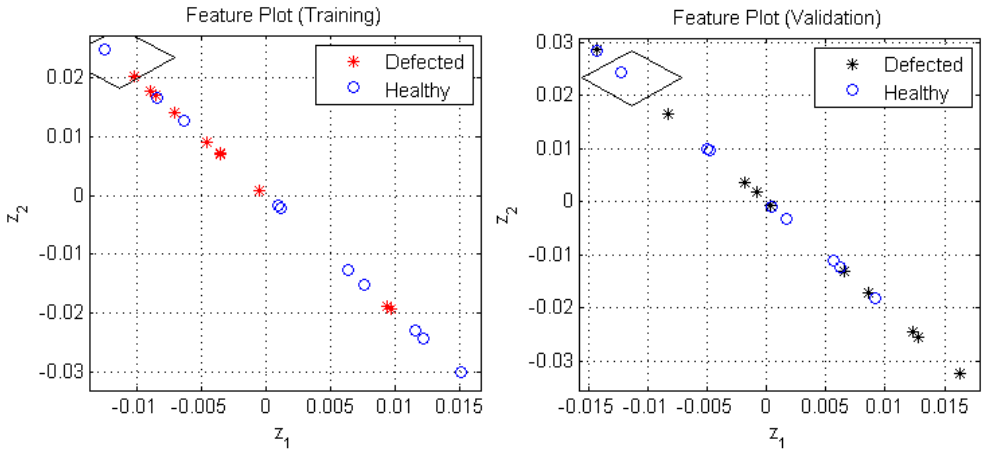

Figure 5.9: Feature plots for Case 2 without PCA - Left: Training Cases; Right: Validation Cases

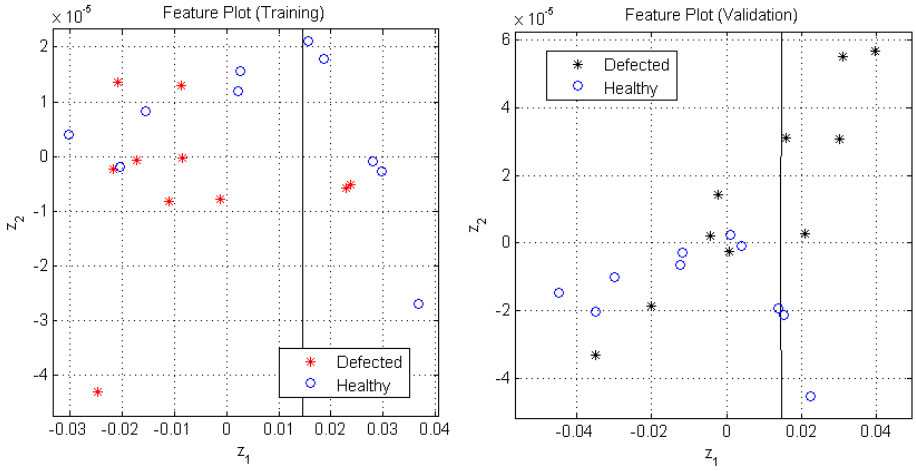

Figure 5.10: Feature plots for Case 2 with PCA - Left: Training Cases; Right: Validation Cases

5.1.2 Condition 2

The time plots for this dataset are shown in Fig. 5.11.

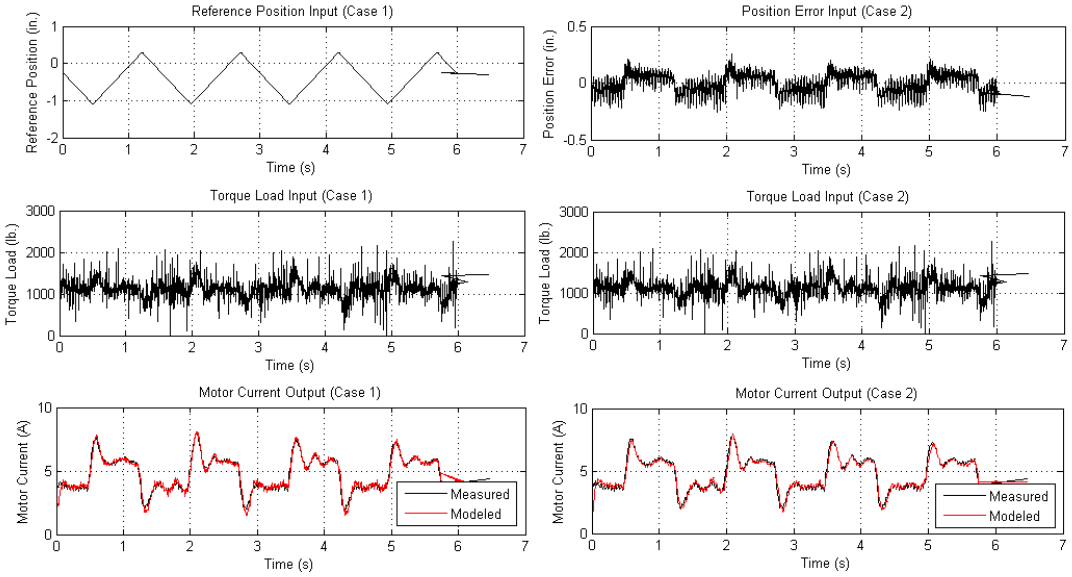

Figure 5.11: Time plots for Condition 2. Left - Case 1: Reference position input; Right - Case 2: Position error input

Table 5.4: Case 1: Input u1 - Reference Position; Order Vector: [2 4 6 2 1]; Fit: 80.10 %

Associated Plots: Figs. 5.12 to 5.14

Case	Feature Extraction	Misclassification %
Training	Without PCA	60
Validation	Without PCA	50
Training	With PCA	35
Validation	With PCA	40

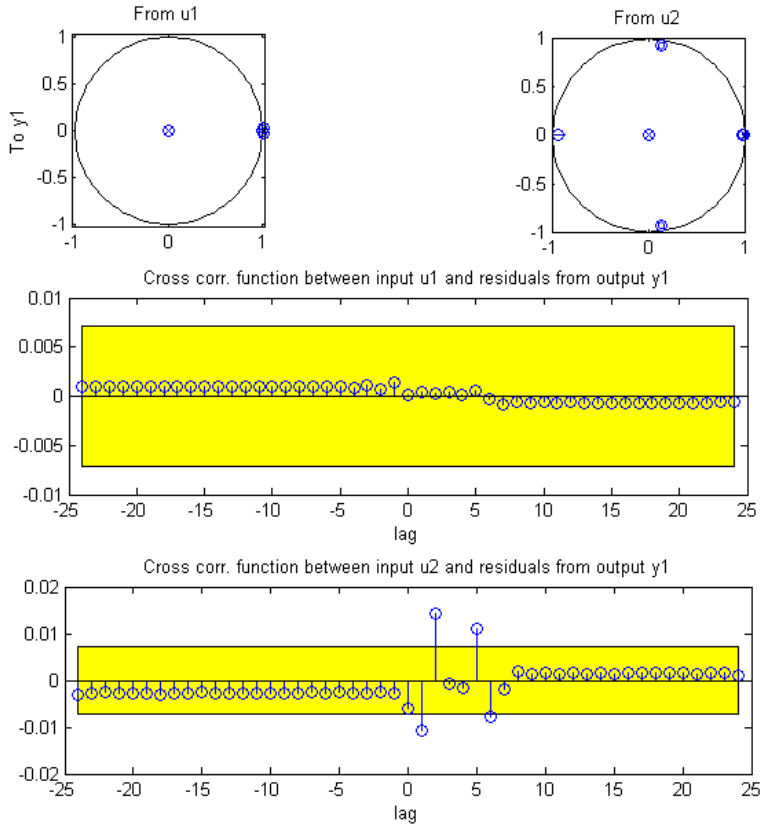

Figure 5.12: Order selection analysis plots for Case 1 - Top: Pole-zero map; Middle: Cross-correlation between u1 and residuals; Bottom: Cross-correlation between u2 and residuals

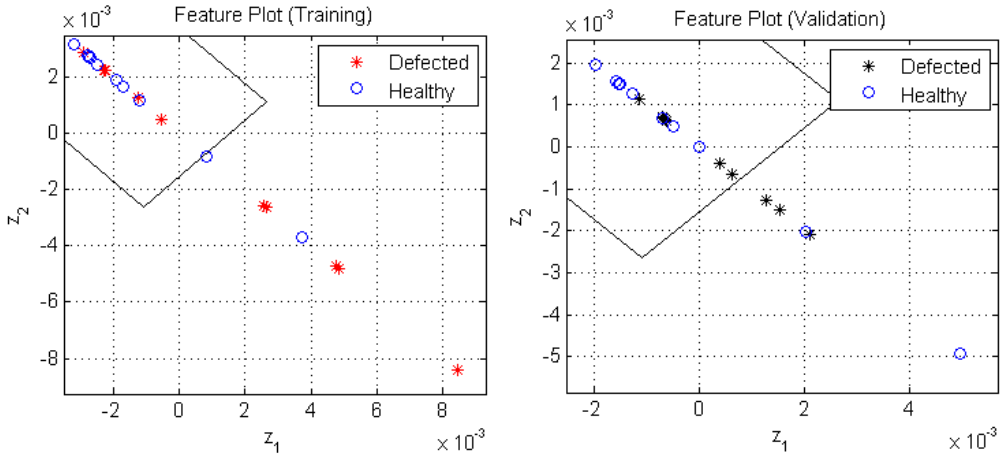

Figure 5.13: Feature plots for Case 1 without PCA - Left: Training Cases; Right: Validation Cases

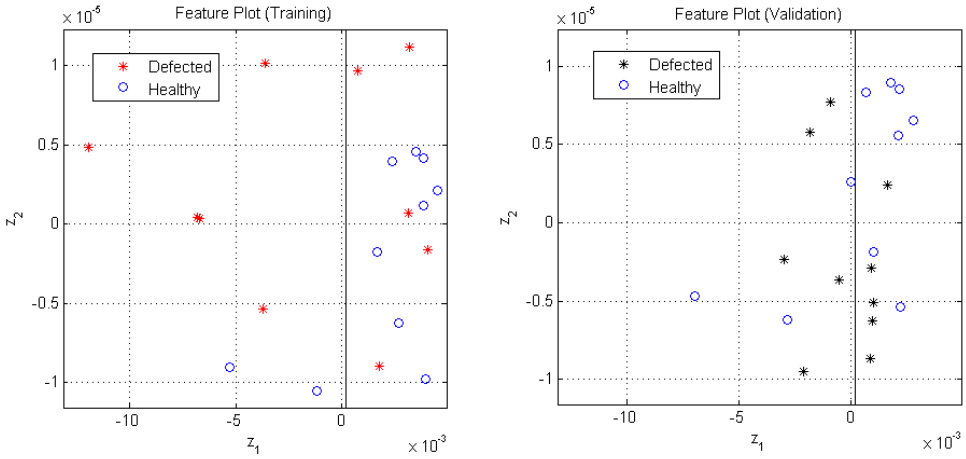

Figure 5.14: Feature plots for Case 1 with PCA - Left: Training Cases; Right: Validation Cases

Table 5.5: Case 2: Input u1 - Position Error; Order Vector: [3 3 3 1 2]; Fit: 80.12 %

Associated Plots: Figs. 5.15 to 5.17

Case	Feature Extraction	Misclassification %
Training	Without PCA	45
Validation	Without PCA	35
Training	With PCA	10
Validation	With PCA	10

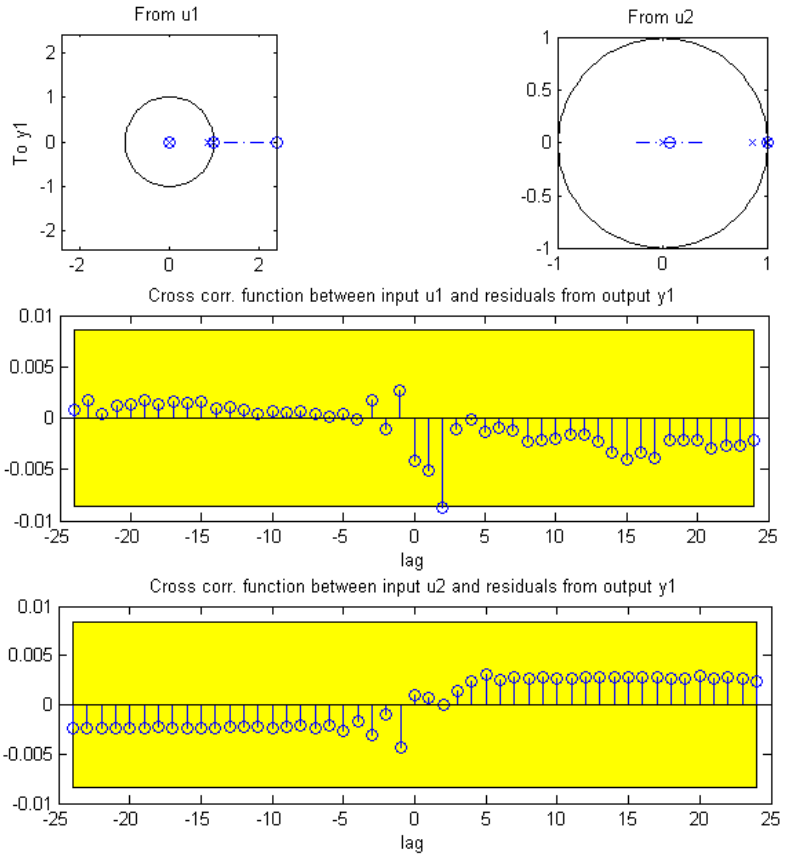

Figure 5.15: Order selection analysis plots for Case 2 - Top: Pole-zero map; Middle: Cross-correlation between u1 and residuals; Bottom: Cross-correlation between u2 and residuals

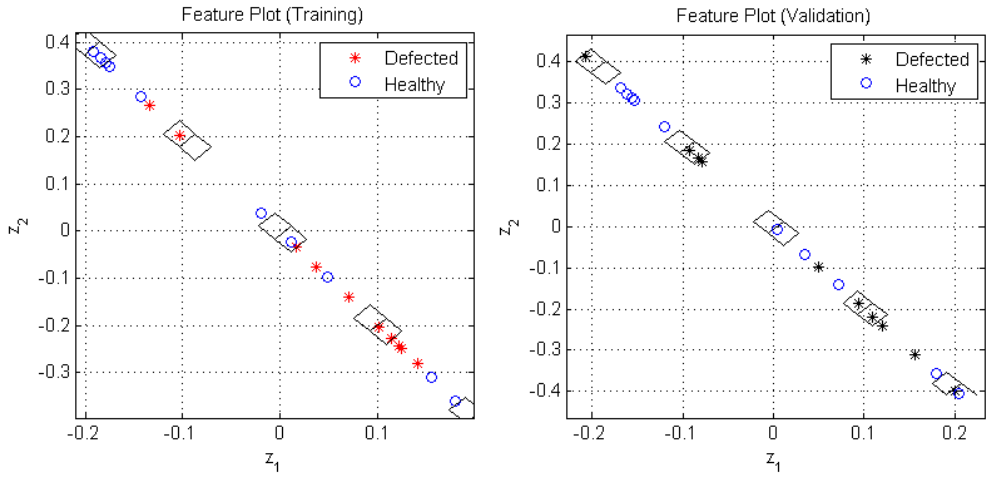

Figure 5.16: Feature plots for Case 2 without PCA - Left: Training Cases; Right: Validation Cases

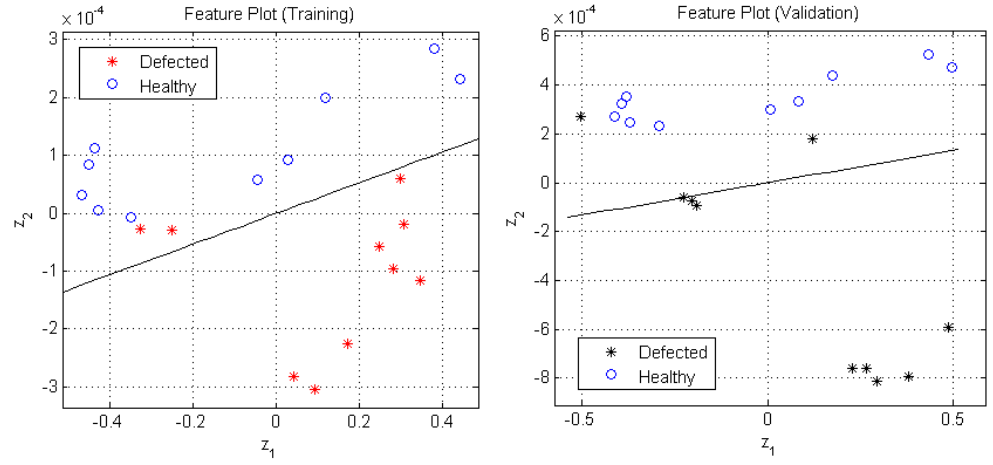

Figure 5.17: Feature plots for Case 2 with PCA - Left: Training Cases; Right: Validation Cases

5.1.3 Condition 3

The time plots for this dataset are shown in Fig. 5.18. It is noted that the dataset does not yield an acceptable model fit when reference position is used as an input (i.e. there are no Case 1 results).

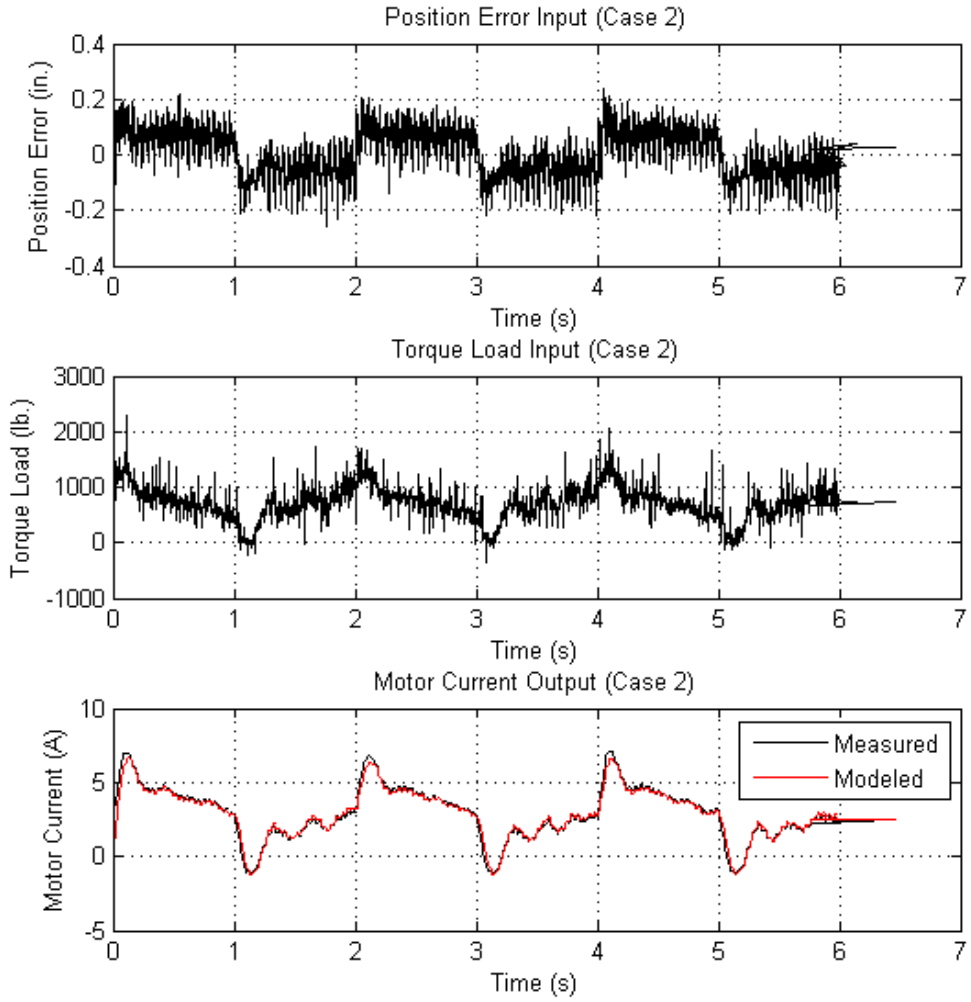

Figure 5.18: Time plots for Condition 3, Case 2: Position error input

Table 5.6: Case 1: Input u1 - Reference Position; Order Vector: [2 2 2 4 4]; Fit: 48.50 %

Associated Plots: None

Case	Feature Extraction	Misclassification %
Training	Without PCA	N/A
Validation	Without PCA	N/A
Training	With PCA	N/A
Validation	With PCA	N/A

Table 5.7: Case 2: Input u1 - Position Error; Order Vector: [3 2 2 1 2]; Fit: 83.34 %

Associated Plots: Figs. 5.19 to 5.21

Case	Feature Extraction	Misclassification %
Training	Without PCA	40
Validation	Without PCA	45
Training	With PCA	40
Validation	With PCA	15

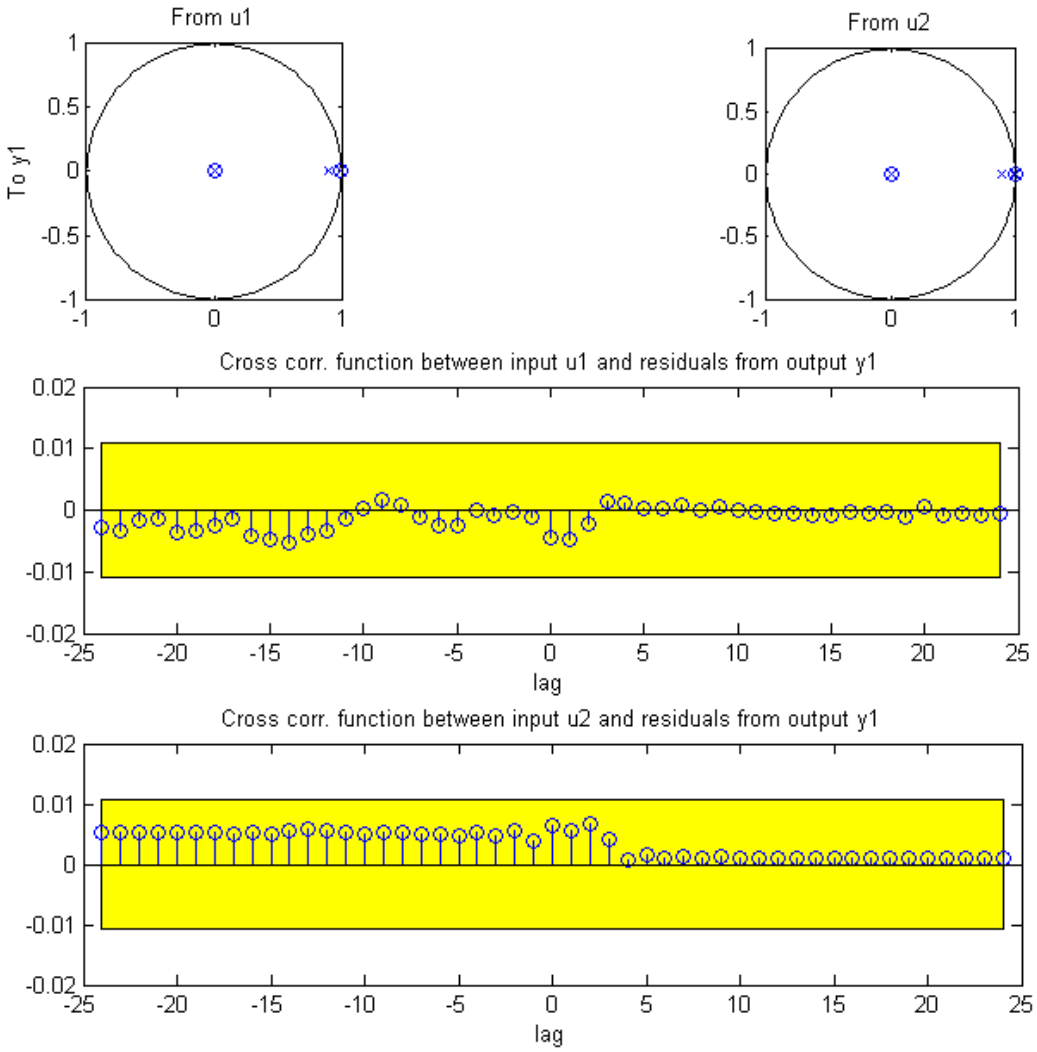

Figure 5.19: Order selection analysis plots for Case 2 - Top: Pole-zero map; Middle: Cross-correlation between u1 and residuals; Bottom: Cross-correlation between u2 and residuals

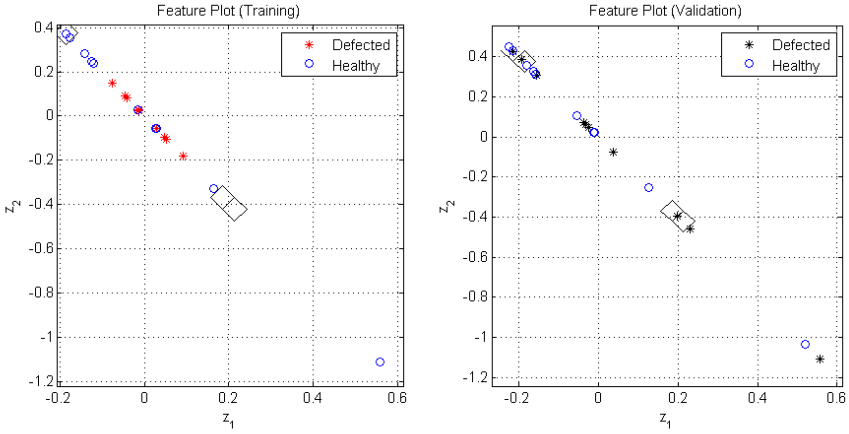

Figure 5.20: Feature plots for Case 2 without PCA - Left: Training Cases; Right: Validation Cases

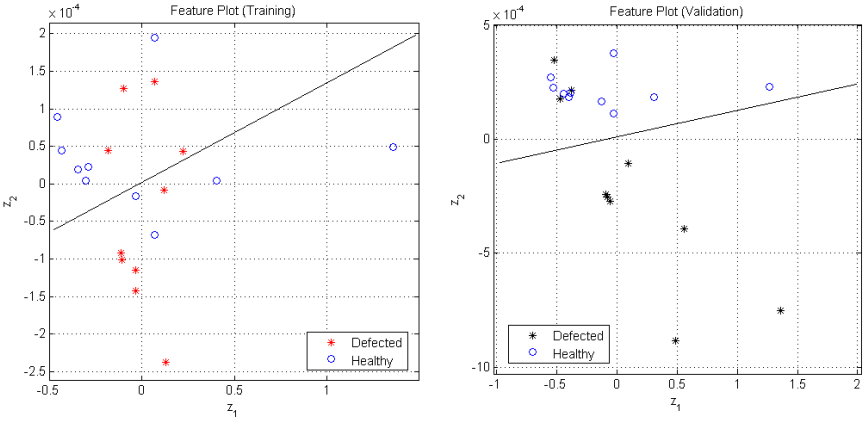

Figure 5.21: Feature plots for Case 2 with PCA - Left: Training Cases; Right: Validation Cases

5.1.4 Condition 4

The time plots for this dataset are shown in Fig. 5.22. It is noted that the dataset does not yield an acceptable model fit when reference position is used as an input (i.e. there are no Case 1 results).

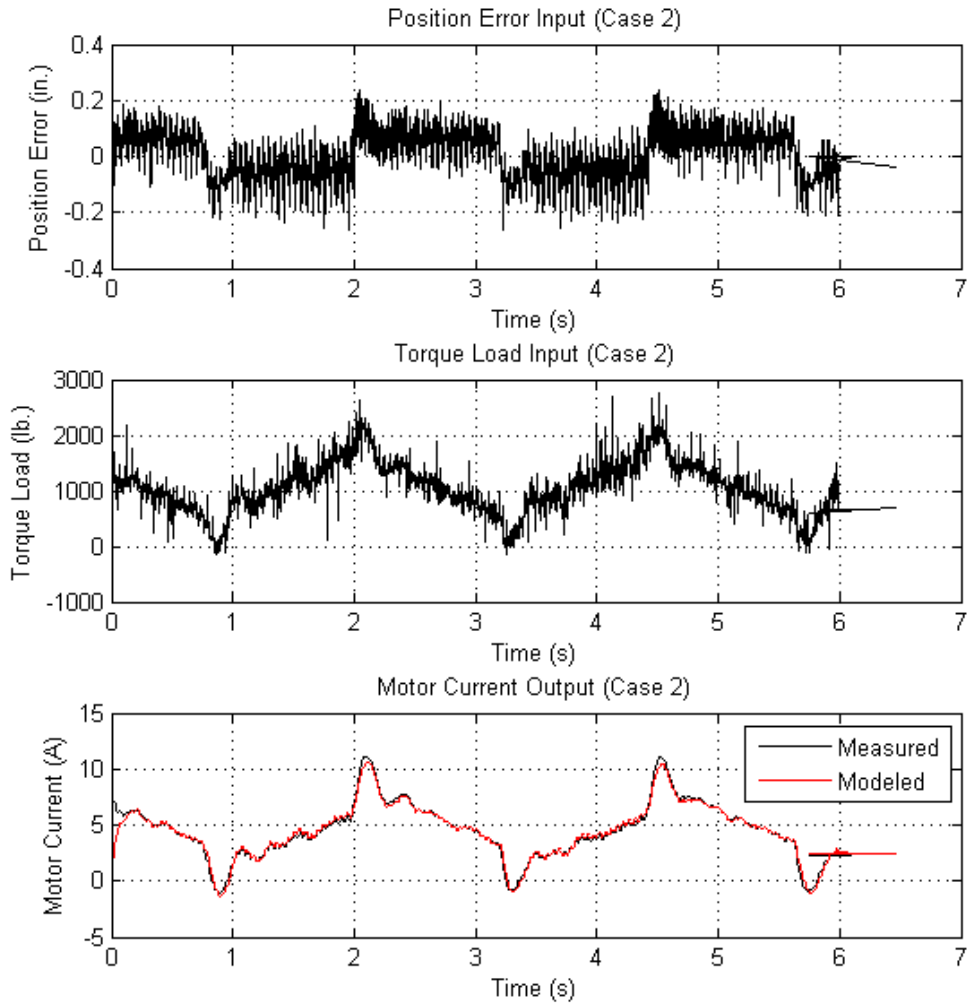

Figure 5.22: Time plots for Condition 4, Case 2: Position error input

Table 5.8: Case 1: Input u1 - Reference Position; Order Vector: [2 2 2 2 1]; Fit: 69.52 %

Associated Plots: None

Case	Feature Extraction	Misclassification %
Training	Without PCA	N/A
Validation	Without PCA	N/A
Training	With PCA	N/A
Validation	With PCA	N/A

Table 5.9: Case 2: Input u1 - Position Error; Order Vector: [3 2 3 1 2]; Fit: 80.37 %

Associated Plots: Figs. 5.23 to 5.25

Case	Feature Extraction	Misclassification %
Training	Without PCA	45
Validation	Without PCA	40
Training	With PCA	30
Validation	With PCA	10

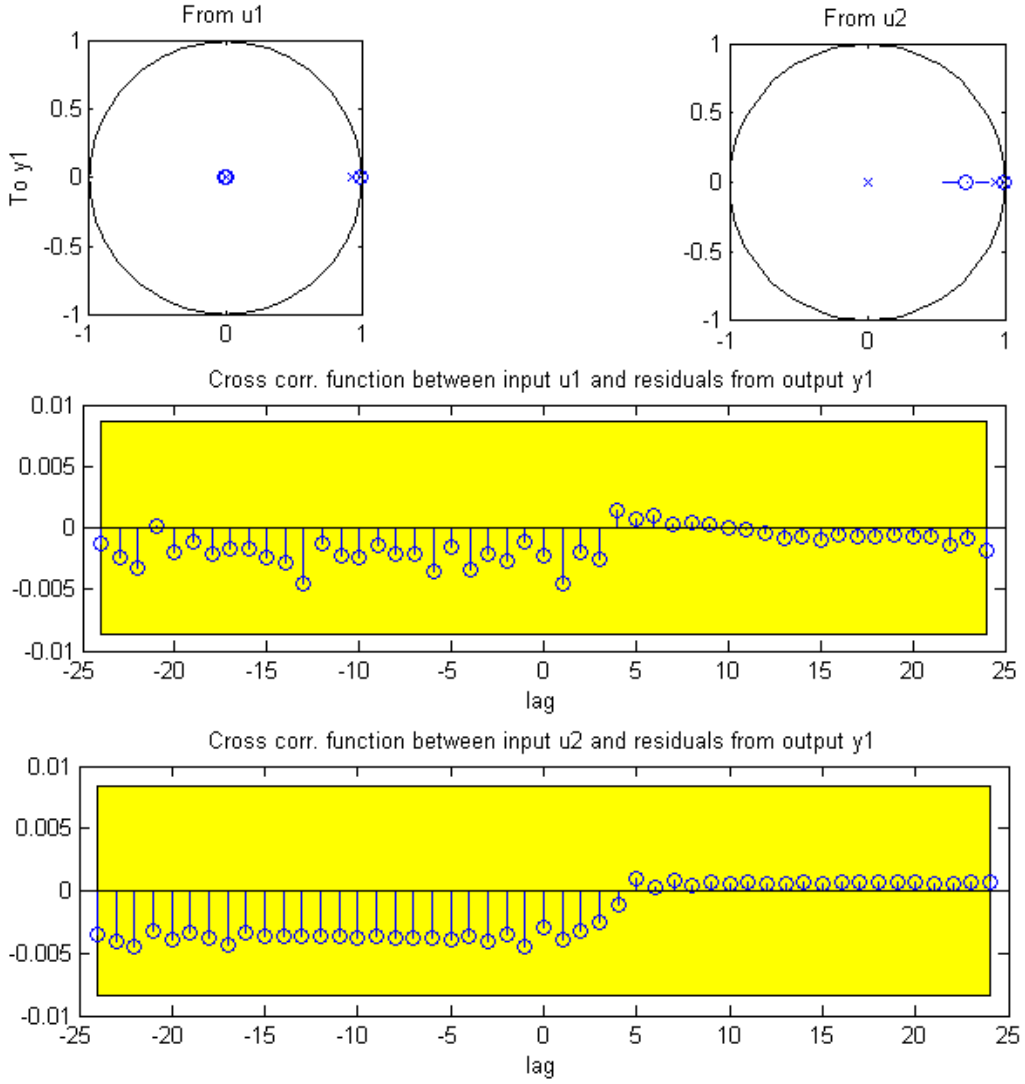

Figure 5.23: Order selection analysis plots for Case 2 - Top: Pole-zero map; Middle: Cross-correlation between u1 and residuals; Bottom: Cross-correlation between u2 and residuals

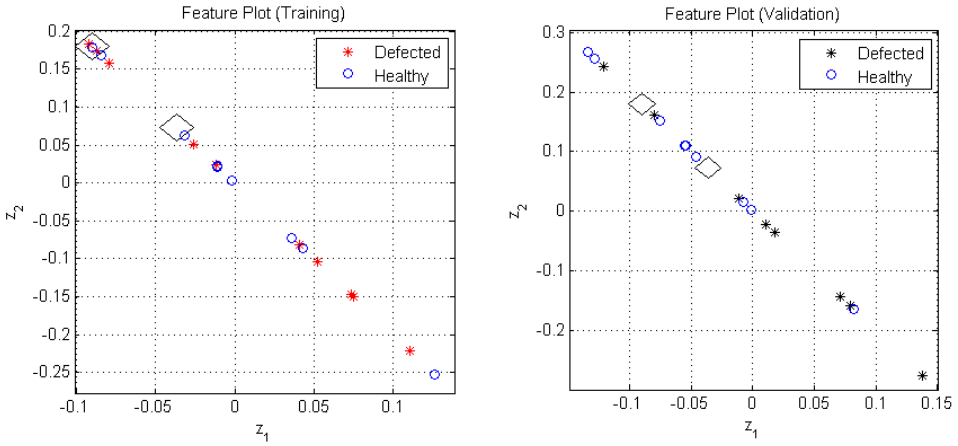

Figure 5.24: Feature plots for Case 2 without PCA - Left: Training Cases; Right: Validation Cases

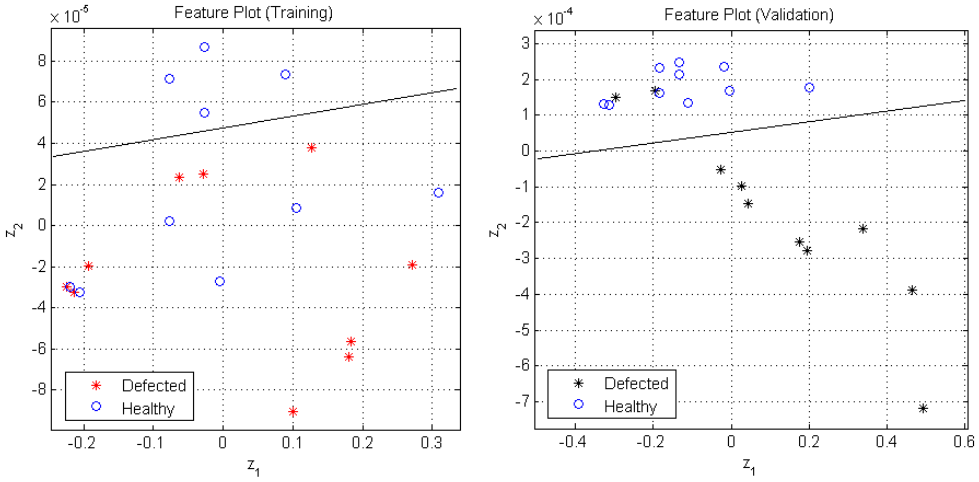

Figure 5.25: Feature plots for Case 2 with PCA - Left: Training Cases; Right: Validation Cases

5.1.5 Condition 5

The time plots for this dataset are shown in Fig. 5.26.

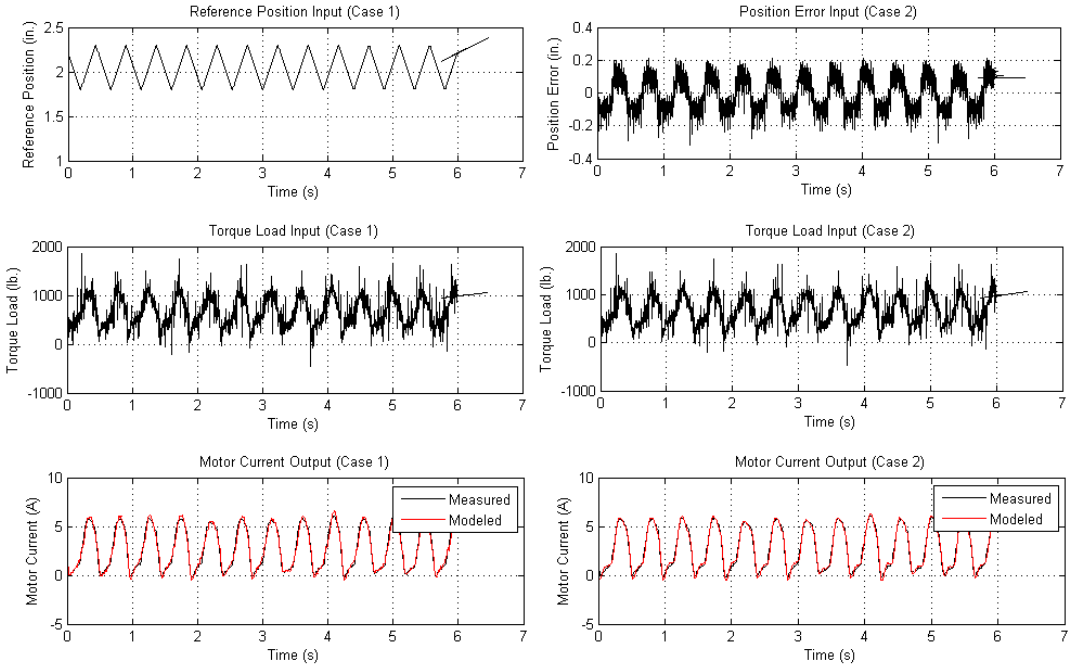

Figure 5.26: Time plots for Condition 5. Left - Case 1: Reference position input; Right - Case 2: Position error input

Table 5.10: Case 1: Input u1 - Reference Position; Order Vector: [1 2 1 1 1]; Fit: 87.52 %

Associated Plots: Figs. 5.27 to 5.29

Case	Feature Extraction	Misclassification %
Training	Without PCA	15
Validation	Without PCA	50
Training	With PCA	10
Validation	With PCA	50

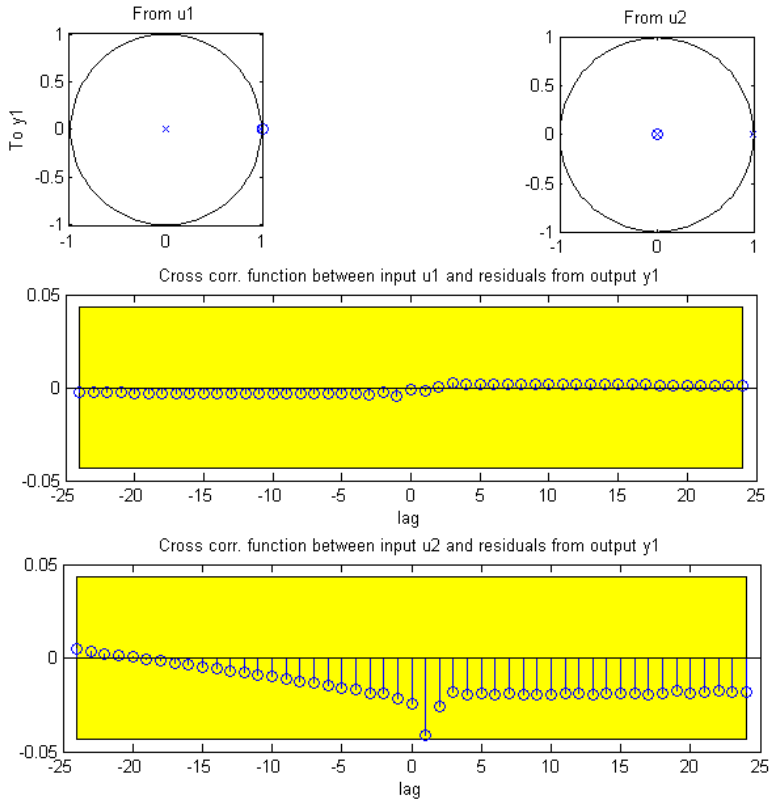

Figure 5.27: Order selection analysis plots for Case 1 - Top: Pole-zero map; Middle: Cross-correlation between u1 and residuals; Bottom: Cross-correlation between u2 and residuals

Table 5.11: Case 2: Input u1 - Position Error; Order Vector: [3 2 1 1 1]; Fit: 89.08 %

Associated Plots: Figs. 5.30 to 5.32

Case	Feature Extraction	Misclassification %
Training	Without PCA	65
Validation	Without PCA	70
Training	With PCA	50
Validation	With PCA	35

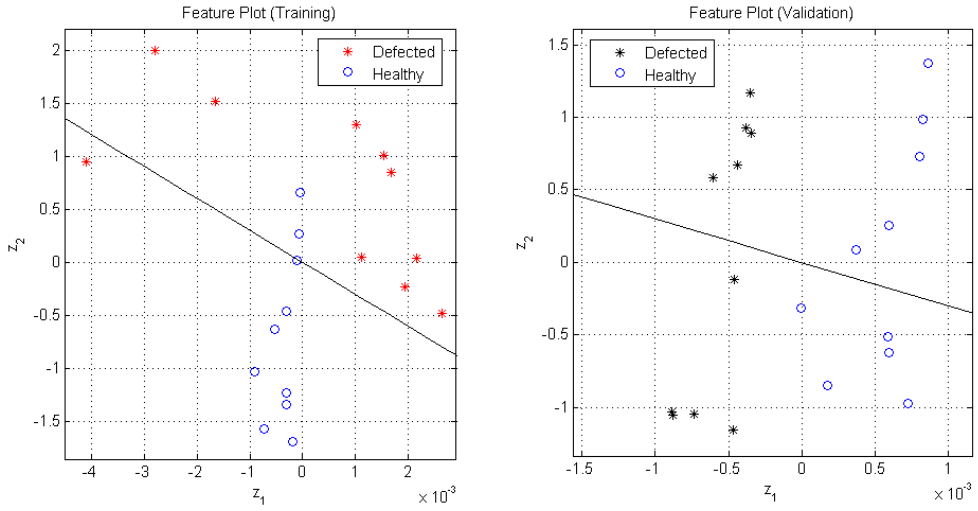

Figure 5.28: Feature plots for Case 1 without PCA - Left: Training Cases; Right: Validation Cases

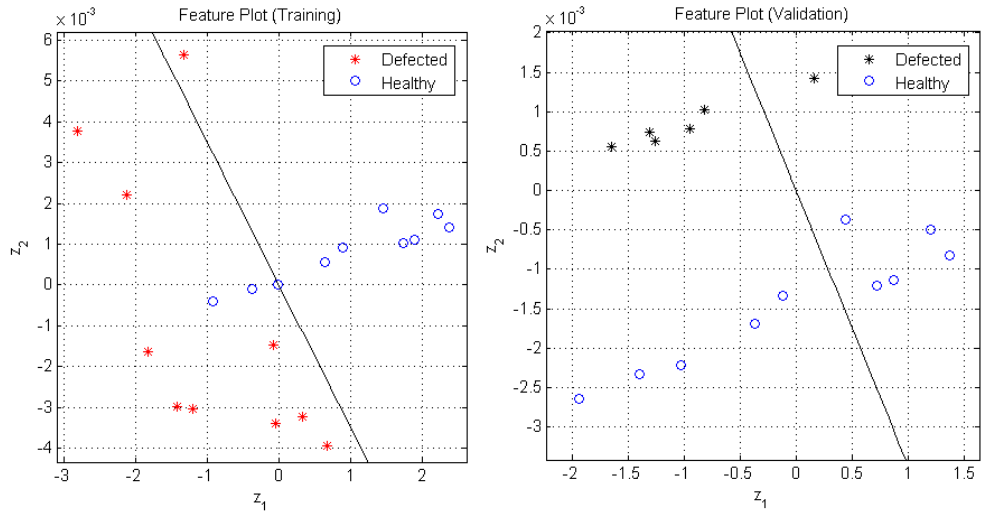

Figure 5.29: Feature plots for Case 1 with PCA - Left: Training Cases; Right: Validation Cases

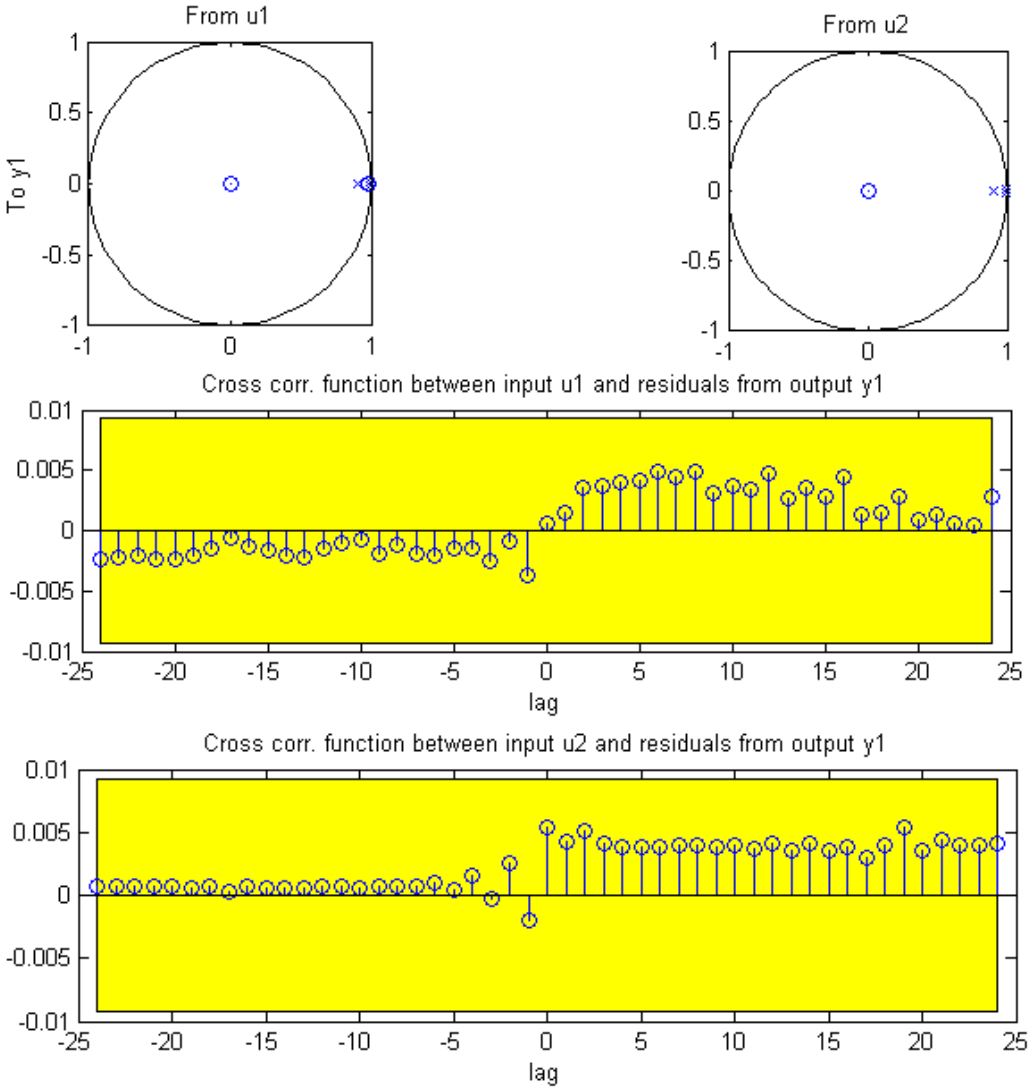

Figure 5.30: Order selection analysis plots for Case 2 - Top: Pole-zero map; Middle: Cross-correlation between u1 and residuals; Bottom: Cross-correlation between u2 and residuals

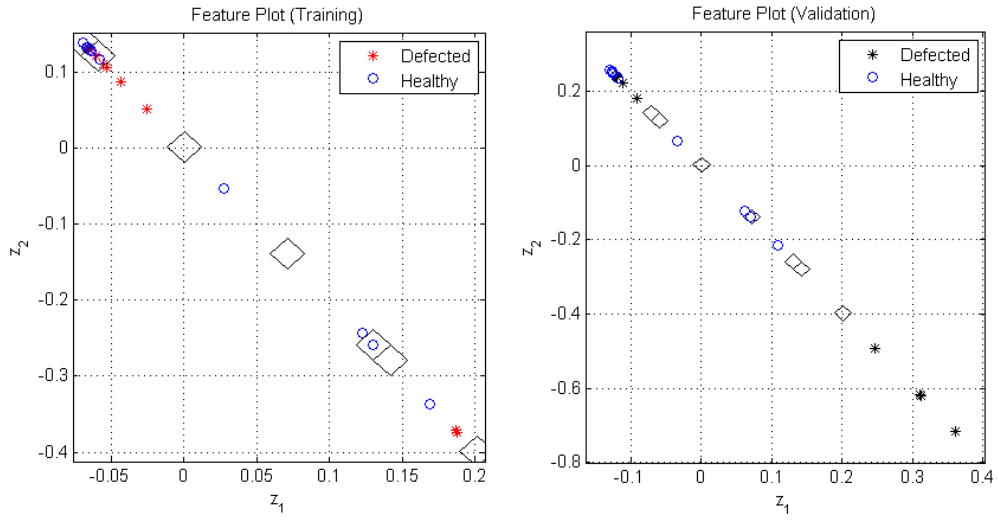

Figure 5.31: Feature plots for Case 2 without PCA - Left: Training Cases; Right: Validation Cases

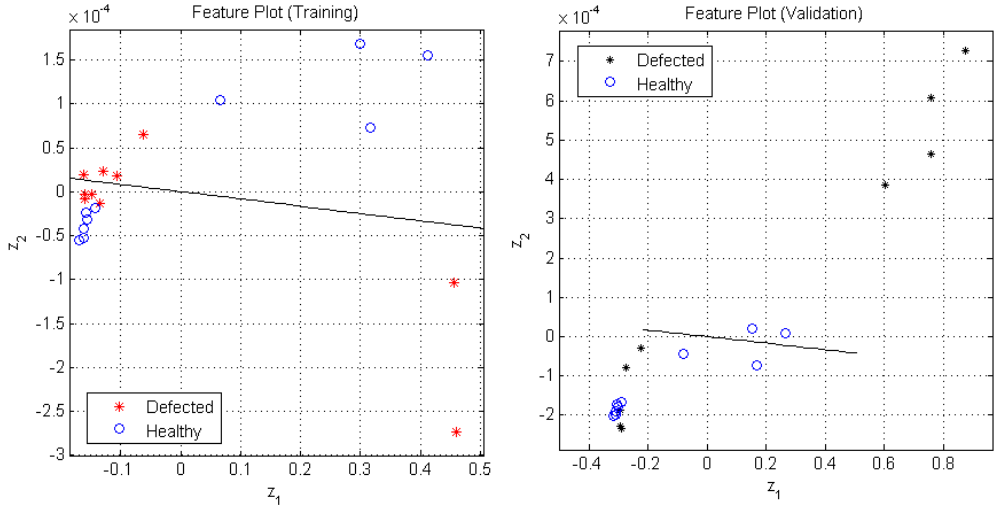

Figure 5.32: Feature plots for Case 2 with PCA - Left: Training Cases; Right: Validation Cases

5.1.6 Inferences for EMA at Moog

The nature of the input signal (i.e. sine wave/triangle wave/square wave) plays an important role in the performance of the algorithm. This is likely because certain types of signals are more or less exciting than others and since parameter estimation is purely a numerical approach, the nature of excitation provided by different types of signals affects the performance of the estimation approach. It is also observed that using the position error as input provides marginally favorable results especially in cases where reference position is unable to even obtain a reasonable fit to the data. This appears to be in conjunction with the discussion on closed-loop fault detection techniques, the theory of which is directly applicable to system identification in closed-loop. It is also noted that in all cases, priority is given to residual analysis since the primary defect type in the EMA from Moog is a single-point defect and it is observed in simulations that the residual analysis plays an important role in determining whether or not such defects are effectively identified.

It is evident that PCA is advantageous for feature extraction thereby resulting in improved classification performance. It is once again concluded that an improvement in the parameter estimation and classification algorithms will further enhance the fault detection results when PCA is employed for feature extraction in the manner discussed in Chapter 3. Although certain cases do not show very good results, they are definitely an improvement over the corresponding results when PCA is not applied. This is shown once again in Section 5.2.

5.2 Results from EMA at RIT

The RIT EMA test rig is shown in Fig. 5.33. The data aquisition is through dSpace. The other components of the rig are similar to the Moog test rig except for the absence of a hydraulic load actuator due to the nature of the applied load on the horizontally mounted EMA. The load is applied via two springs mounted as shown in Fig. 5.33. The springs get compressed and extended with the actuation of the EMA.

Therefore, the load varies directly with position command according to Hooke's law $F = -k_e x$ where k_e is the combined spring constant - since the springs are in parallel, $k_e = 2 \times k$ where k is the spring constant of each spring - and x is the displacement of the the actuator at a given instant of time. Also, while the velocity command

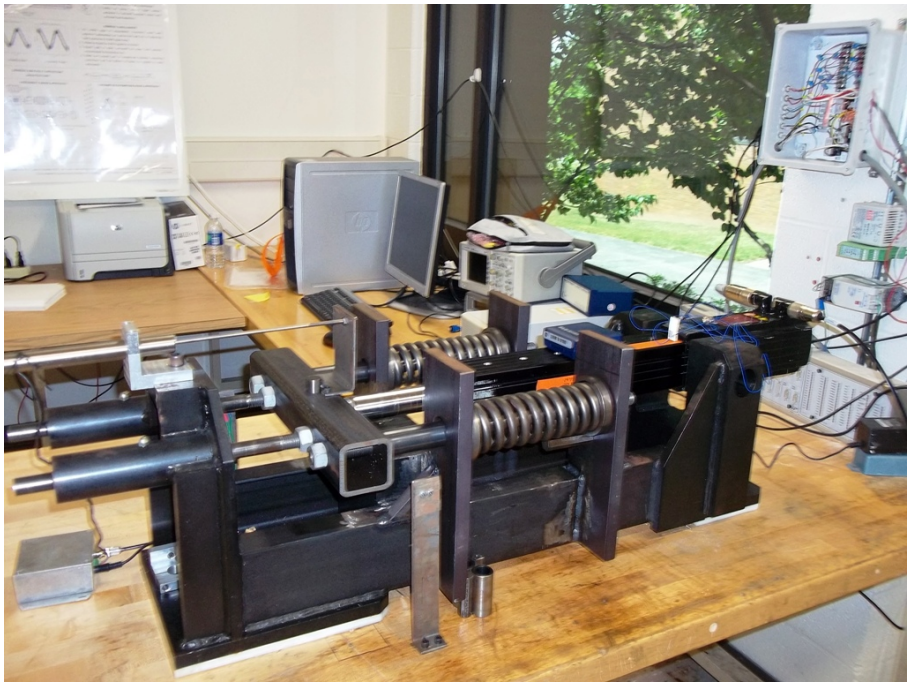

Figure 5.33: RIT Test Rig for Moog MaxForce EMA

signal to the motor controller (see Fig. 5.2) is not available when using the Moog test rig, the signal is available in the RIT test rig, allowing it to be used during the fault detection process.

The lack of the motor quadrature current requires that an approximated value be calculated for use. However, after doing so, it is found that a linear model structure is unable to fit the calculated output due to the errors realised during the approximation process (see Fig. 5.35). Therefore, a few experiments are conducted using the actuator position as the output while using velocity command and position error as possible inputs. This is done because it is hypothesized that since the nature of the defect in the EMA is only due to variations in lubrication, the characteristics of the

unhealthy operation might be noticeable in the motor shaft velocity. Since this signal is unavailable, it is further presumed that any changes in the shaft velocity will affect the actuator linear position.

Amongst the 32 datasets, 16 each are from EMAs having different levels of lubrication. Each of the 16 datasets are further subdivided into 8 each for training and validation. The data aquisition is performed in a manner similar to that of the EMA data collected at Moog with the exception of a few changes in the actual signals captured. More on this is presented in the next section. The only point of importance here is that the sampling interval in this case is appropriate at a value of 0.0001 s.

For the data collected using the RIT test rig, the quadrature motor current is not available and therefore an attempt is made to calculate the current value using the individual phase current measurements and the motor angle obtained by integrating the motor velocity. Should this not work, actuator position is used as the output as it is expected that the reduced lubrication defect will affect the motor shaft speed to some extent which in turn is expected to affect the linear displacement of the actuator. The transformation of the individual phase currents to the motor quadrature current is based on Park's transformation as shown in Eq. (5.1).

$$\begin{bmatrix} I_d \\ I_q \\ I_o \end{bmatrix} = \frac{2}{3} \begin{bmatrix} \cos\theta & \cos\left(\theta - \frac{2}{3}\pi\right) & \cos\left(\theta + \frac{2}{3}\pi\right) \\ \sin\theta & \sin\left(\theta - \frac{2}{3}\pi\right) & \sin\left(\theta + \frac{2}{3}\pi\right) \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} \tag{5.1}$$

where i_a, i_b and i_c are the individual phase currents, I_d, I_q and I_o are the direct, quadrature and zero currents and θ is the motor angle. Park's transformation is a coordinate transformation that converts the three-phase stationary variables into a rotating coordinate system. Applying this transformation is necessary to perform parameter estimation. The nature of the signal in Fig. 5.34 makes this clear.

The drawbacks of calculating the quadrature current in the manner illustrated above are: (i) the transformation is extremely sensitive to variations in the motor angle θ. Typically, fairly accurate results are obtained if the motor angle is recorded

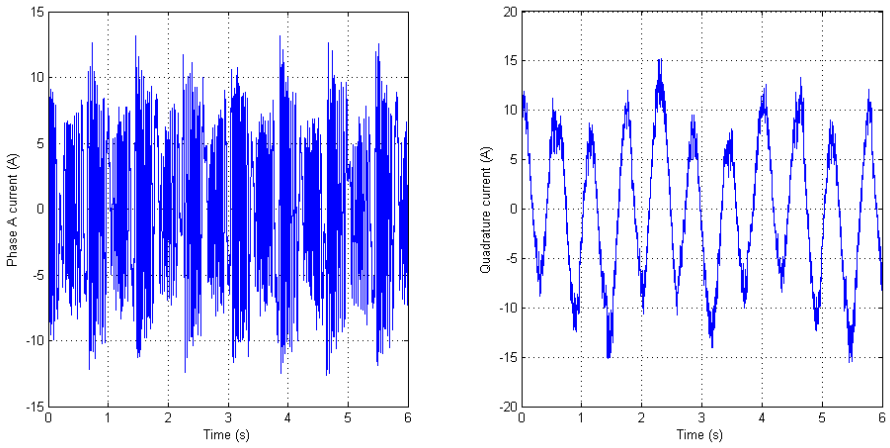

Figure 5.34: Left - EMA Phase A current; Right - EMA quadrature current

instead of calculated by integrating the motor velocity signal. The EMA data collected by Moog provides the quadrature current. Therefore, to prove the above statement, see Fig. 5.35. A blatant difference is observed between the actual quadrature current obtained from the motor controller's accurate calculation based on a recorded value of motor angle and the value calculated using Park's transform and integrating motor velocity to give θ; and (ii) the Park's transformation itself is an approximation. More information about the Park's transformation is available in [43, 44, 45].

Once again, two scenarios are considered - one involving feature extraction without PCA and the other with PCA. Within each scenario, cases for each of the input types are investigated. It is noted that although a total of 80 data sets are available with 20 data sets each from EMAs with 4 different levels of lubrication, only 32 data sets from EMAs with 2 different levels of lubrication are tested. This is done because validation for the use of actuator linear position as an output signal for fault detection is not provided. Therefore, while most of the research in this area constitutes future work, some initial results are presented. The time plots for the datasets used are shown in Figs. 5.36 and 5.37.

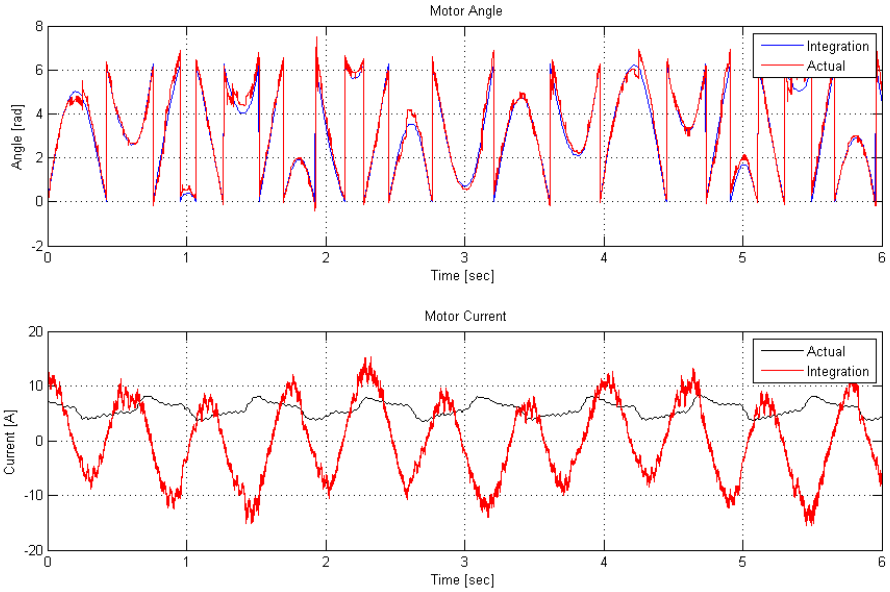

Figure 5.35: Bottom - Comparison of I_q profiles; Top - Comparison of motor angles

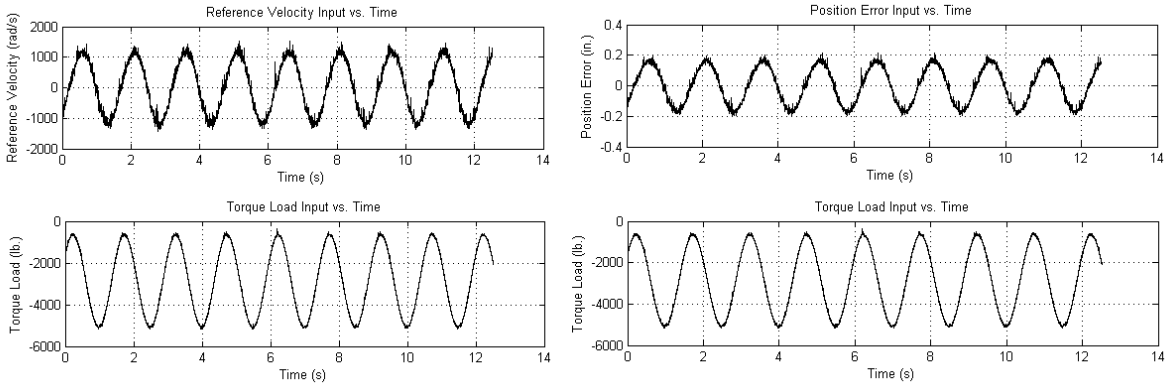

Figure 5.36: Time history of input signals. Left: u_1 - Reference velocity; Right: u_1 - Position error

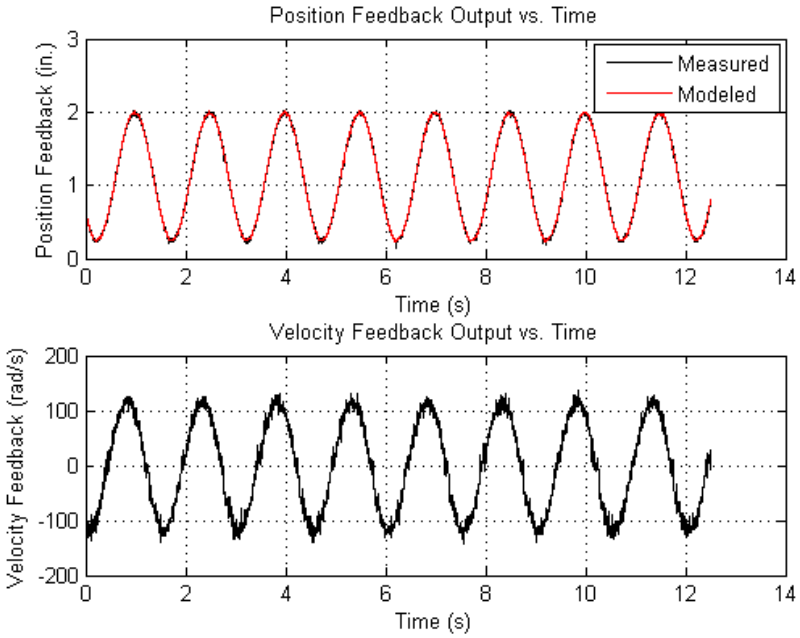

Figure 5.37: Time history of output signals. Top: Position feedback output; Bottom: Velocity feedback output

5.2.1 Scenario 1: Feature Extraction without PCA

Table 5.12: Condition 1: Input u1 - Reference Velocity; Order Vector: [3 1 1 2 2]; Fit: 99.124 %

Associated Plot: Fig. 5.38

Case	Misclassification %
Training	50
Validation	50

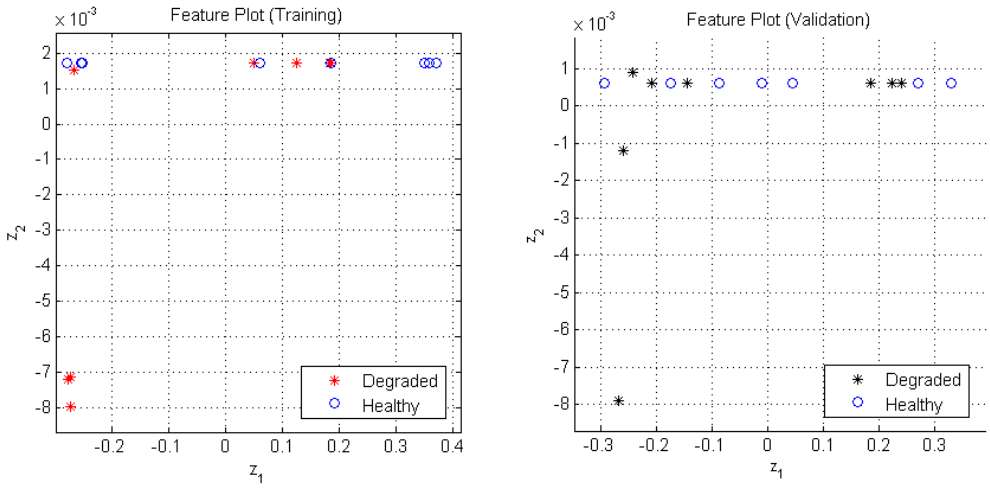

Figure 5.38: Feature plots - Left: Training Cases; Right: Validation Cases

Table 5.13: Condition 2: Input u1 - Position Error; Order Vector: [3 1 1 2 2]; Fit: 99.125 %

Associated Plot: Fig. 5.39

Case	Misclassification %
Training	50
Validation	50

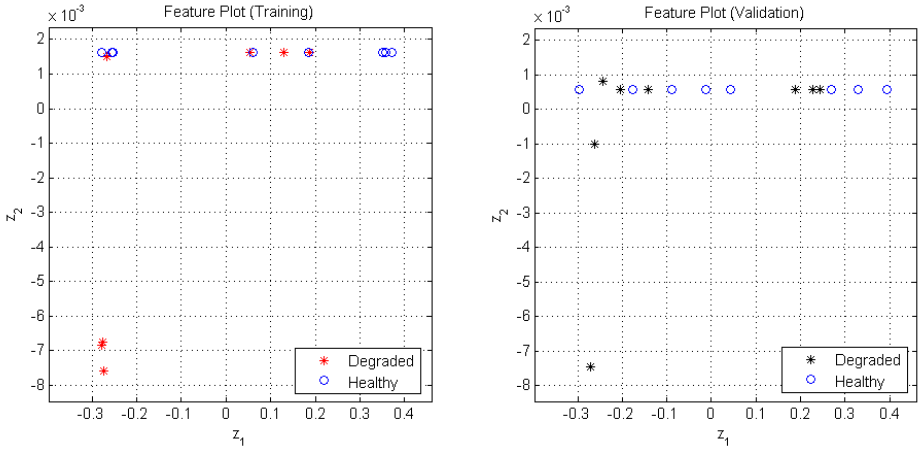

Figure 5.39: Feature plots - Left: Training Cases; Right: Validation Cases

5.2.2 Scenario 2: Feature Extraction with PCA

Table 5.14: Condition 1: Input u1 - Reference Velocity; Order Vector: [3 1 1 2 2]; Fit: 99.124 %

Associated Plot: Fig. 5.40

Case	Misclassification %
Training	18.75
Validation	0

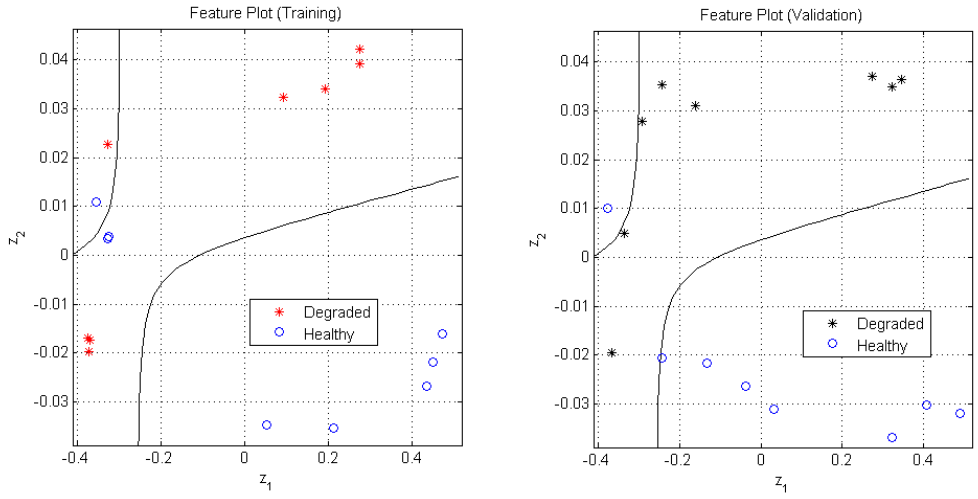

Figure 5.40: Feature plots - Left: Training Cases; Right: Validation Cases

Table 5.15: Condition 2: Input u1 - Position Error; Order Vector: [3 1 1 2 2]; Fit: 99.125 %

Associated Plot: Fig. 5.41

Case	Misclassification %
Training	18.75
Validation	0

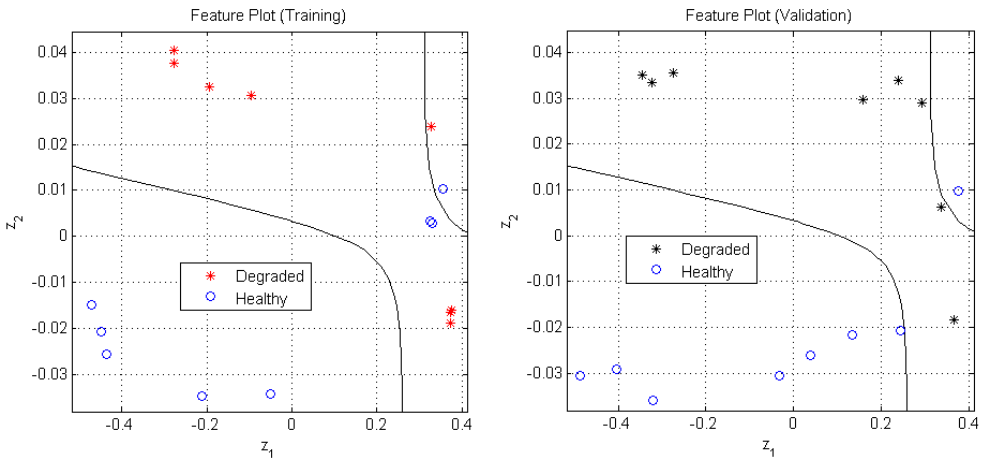

Figure 5.41: Feature plots - Left: Training Cases; Right: Validation Cases

5.2.3 Inferences for EMA at RIT

The following inferences are drawn from the results of Sections 5.2.1 and 5.2.2.

1. There is no difference when using the reference velocity or position error as inputs. This might be because the two signals are similar or because the output signal dominates both these signals significantly enough that differences in the input signal are not reflected in the extracted features.

2. When PCA is not used for feature extraction, the classification bounds are not noticeable and the classification results are extremely poor. On the other hand, when PCA is used, the results are dramatically improved. The classification bounds are drawn effectively and 100% accuracy in classification is observed. This example further confirms the usefulness of PCA techniques for feature extraction.

3. The model fit in each of the cases is extremely good. This is likely because the signals are not severly corrupted by noise. Furthermore, although it is not shown, the residual analysis plots are satisfactory in all cases with the vector of cross-correlation values between the residuals and inputs generally remaining within the 99% confidence intervals.

4. Since comprehensive testing is not accomplished since the actuator position as an output is not analytically investigated, generalizations about the performance of the algorithm are not made. Therefore, the main conclusion that is drawn from the above results is that the established approach has scope for detecting generalized roughness type defects and its improvement over the previous approach of Chapter 2 is once again noticed.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

Model-based approaches to bearing fault detection in an Electromechanical actuator were investigated. A thorough review of previous related works revealed a large number of capable solutions related to such approaches for fault detection in electric drives. However, no published works regarding the use of basic system identification techniques for fault detection in EMAs in particular was reported.

The first proposed approach therefore attempted to use system identification to detect bearing faults in EMAs. In this approach, linear difference models were selected and various parameter estimation approaches were investigated in order to estimate the model parameters from EMA signal data. Then the estimated parameters were used as features for classification, on the assumption that accurate representations of systems having different health conditions would result in the parameter estimates being different, thereby allowing their comparison to reveal information about the health of the system. The proposed approaches were tested on a PM DC motor system as well as a DC motor control module. While the approach was able to generate fairly accurate models capable of simulating the systems accurately, the parameter estimates from healthy and unhealthy systems differed by an extremely small amount such that effective classification was not possible. Furthermore, it was observed that the use of basic linear system identification and parameter estimation techniques failed to efficiently and effectively isolate unmeasurable disturbances such as sensor noise. Therefore, the noise influenced the parameter estimates and any apparent differences

in the estimates were largely due to the differences in the noise profiles. Additionally, these problems were encountered with simple first or second order systems operating in open-loop. The EMA is a complex closed-loop system consisting of controllers that tend to attenutate any information about system degradation that might be present in the captured signals. Therefore, for effective fault detection, an accruate model of the system coupled with a feature extraction technique that brings out maximum information about the faults from the model parameters was required.

In view of these requirements, a novel fault detection algorithm was developed that combined refined and improved versions of the model parameter estimation techniques already investigated with principal component analysis for feature extraction followed by a Bayesian approach to classification. This modified fault detection scheme for bearing fault detection in EMAs - which has previously not been reported in research publications - was found to provide considerably improved results over the previous approach. Furthermore, its ability to detect both single-point defects and generalized lubrication defects were investigated and shown to be effective. While improvements in the approach are required for more robust fault detection, it is concluded that model parameter estimation when coupled with PCA and Bayesian classification techniques form a potentially effective bearing fault detection scheme for EMAs as well as electric drives in general, with the capability of detecting generalized roughness defects as well as single-point defects.

6.2 Future Work

As indicated earlier, some aspects of the current approach that need improvement include:

1. A more robust, effective and efficient parameter estimation approach - the approach used in this work is the instrumental variables approach along with a linear difference equation model structure, and although this approach has its benefits and is a reliable approach to use for this particular application (see [33]),

improvements related to the ability of the approach to effectively filter unmeasurable disturbances so that the parameter estimates do not differ on account of them can be achieved by conducting more research in this area or using established solutions. Furthermore, the current approach will only work if the same input signals are provided to the system each time during data aquisition. That is, if a sine wave having a specific characteristic is used as an input to aquire the required output data to train the classifier, then the exact same signal must be used to generate data from a system whose health is to be determined. This makes the approach inflexible and therefore inconvenient.

2. A more realistic classification scheme that perhaps utilizes a non-uniform cost function or a different approach (other than Bayesian) altogether.

In addition to improving the proposed approach, it is observed from the review of previous works that a dearth of research in the detection of generalized roughness type defects is often reported. This work shows promise in this regard as evidenced by the results presented in the simulations as well as in the case of the EMA at RIT. Therefore next steps also include refining the proposed algorithm to effectively detect such types of faults.

6.2.1 Piece-wise Parameter Estimation

First steps in this direction are already made as evidenced by the MATLAB programs published in the last two sections of Appendix B. It is proposed that a novel, piece-wise parameter estimation approach be employed to estimate parameters before applying similar feature extraction and classification techniques. To understand the idea behind the approach, assume a sine wave input is provided to a system to extract output data. In the case of EMAs or other DC motor drives that are often accurately represented by first or second order transfer functions, the output waveform will also be sinusoidal. It is then proposed that splitting the output waveform (and the corresponding sections of all the other signal data used during parameter estimation) into linear segments that likely contain the most information about the fault will increase

the magnitude of the fault signature in the output data. For example, if the output signal in Fig. 3.3 is split such that the linear portions of the signal (where the defect appears prominently) are stored and the rest discarded, then (i) more data sets are generated allowing more accurate training of the classifier and (ii) it is expected that the parameter estimates will be able to pick up more information about the fault, thereby allowing the feature extraction process to reveal the differences better, allowing better classification. Figure 6.1 shows a sample feature plot obtained using the current version of the piece-wise approach.

In this approach, the power spectral density of the output signal is computed and

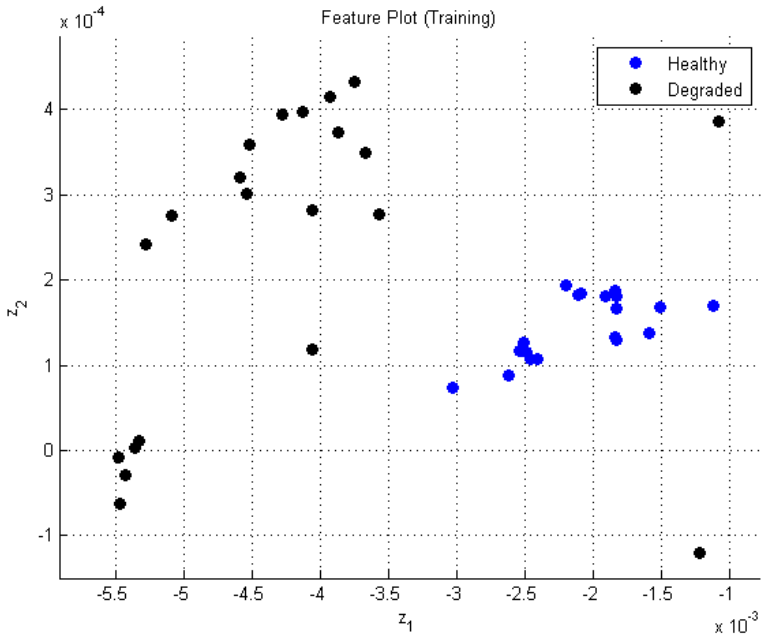

Figure 6.1: Feature plot for detecting generalised roughness type faults using piece-wise parameter estimation

the frequency with the highest power is identified. This is most likely the fundamental frequency of the signal. With this knowledge, a low pass filter is applied on a copy of the output signal to smoothen it completely. Then a step-wise evaluation of the slope of the signal is computed and the linear segments are separated. While all this is done on a copy of the output signal, the same segregation process is mirrored on the actual

data set. This results in multiple data sets consisting of an output that, in this case, is essentially a straight line holding maximum information about the fault. A simple first-order linear difference equation is selected and the fault detection approach devised earlier in this work is applied to each segment. The result in Fig. 6.1 is to be compared directly with Fig. 4.3. The improvement in the isolation of features from the generalised roughness defect (Defect B) is evident. It is believed that refinements to this approach will go a long way in developing an effective model-based approach for efficiently and effectively detecting generalised roughness type faults not only in EMAs but in electric drives in general.

Bibliography

[1] Mark A Davis. High Performance Electromechanical Servoactuation using Brushless DC Motors. In *Motor-Con Conference*, pages 1–12, Atlantic City, 1984.

[2] Edward Balaban, Abhinav Saxena, Kai Goebel, Carl S Byington, Matthew J Watson, Sudarshan Bharadwaj, and Matthew Smith. Experimental Data Collection and Modeling for Nominal and Fault Conditions on Electro-Mechanical Actuators. In *Annual Conference of the Prognostics and Health Management Society*, pages 1–15, 2009.

[3] Lennart Ljung. *System Identification: Theory for the User*. Prentice Hall PTR, 2nd edition, 1999.

[4] Oliver Nelles. *Nonlinear System Identification*. Springer-Verlag, 1st edition, 2001.

[5] LJ Technical Systems. *DC Motor Control Module User Manual*. LJ Group, Holtsville, NY, USA, 1st edition.

[6] Stephen L Botten, Chris R Whitley, and Andrew D King. Flight Control Actuation Technology for Next-Generation All-Electric Aircraft. *Technology Review Journal*, (Millennium):55–68, 2000.

[7] Stephen C Jensen, Gavin D Jenney, Bruce Raymond, and David Dawson. Flight Test Experience with an Electromechanical Actuator on the F-18 Systems Research Aircraft. In *19th Digital Avionics Systems Conference*, pages 1–11, Philadelphia, PA, 2000.

[8] Dominique Van Den Bossche. The A380 Flight Control Electrohydrostatic Actuators, Achievements and Lessons Learnt. In *25th International Congress of the Aeronautical Sciences*, pages 1–8, Hamburg, Germany, 2006.

[9] Edward Balaban, Prasun Bansal, Paul Stoelting, Abhinav Saxena, Kai F Goebel, and S Curran. A Diagnostic Approach for Electro-Mechanical Actuators in Aerospace Systems. In *IEEE Aerospace Conference*, pages 1–13, 2009.

[10] Carl S Byington and Paul Stoelting. A Model-Based Approach to Prognostics and Health Management for Flight Control Actuators. In *IEEE Aerospace Conference*, pages 1–12, 2004.

[11] Sriram Narasimhan, Indranil Roychoudhury, Edward Balaban, and Abhinav Saxena. Combining Model-Based and Feature-Driven Diagnosis Approaches A Case Study on Electromechanical Actuators. In *21st International Workshop on Principles of Diagnosis*, pages 1–8, 2010.

[12] Rolf Isermann and P Balle. Trends in the Application of Model-based Fault Detection and Diagnosis of Technical Processes. *Control Engineering Practice*, 5(5):709–719, 1997.

[13] Rolf Isermann. Model-Based Fault-Detection and Diagnosis Status and Applications. *Annual Reviews in Control*, 29(1):71–85, January 2005.

[14] Wei Zhou, Thomas G. Habetler, and Ronald G. Harley. Stator Current-Based Bearing Fault Detection Techniques: A General Review. In *IEEE International Symposium on Diagnostics for Electric Machines, Power Electronics and Drives*, pages 7–10. Ieee, September 2007.

[15] Mahdi S M Alavi, R Izadi Zamanabadi, and M J Hayes. Robust Fault Detection and Isolation Technique for Single-Input/Single-Output Closed-Loop Control Systems that Exhibit Actuator and Sensor Faults. *IET Control Theory and Applications*, 2(11):951–965, 2008.

[16] C Aubrun, M Robert, and T Cecchin. Fault Detection in a Control Loop. *Control Engineering Practice*, 3(10):1441–1446, 1995.

[17] Sreedhar G Babu, A Lingamurthy, and A S Sekhar. Condition Monitoring of Brushless DC Motor-Based Electromechanical Linear Actuators using Motor Current Signature Analysis. *The International Journal of Condition Monitoring*, 1(1):20–32, 2011.

[18] Fabio Immovilli, Marco Cocconcelli, Alberto Bellini, and Riccardo Rubini. Detection of Generalized-Roughness Bearing Fault by Spectral-Kurtosis Energy of Vibration or Current Signals. *IEEE Transactions on Industrial Electronics*, 56(11):4710–4717, 2009.

[19] Anthony J Chirico III, Jason R Kolodziej, and Larry Hall. A Data Driven Frequency Based Feature Extraction and Classification Method for EMA Fault Detection and Isolation. In *5th Annual Dynamic Systems and Control Conference*, 2012.

[20] Levent Eren and Michael J Devaney. Motor Current Analysis via Wavelet Transform with Spectral Post-Processing for Bearing Fault Detection. In *Instrumentation and Measurement Technology Conference*, number May, pages 20–22, Vail, CO, USA, 2003.

[21] Aitor Isturiz, Javier Vinals, Santiago Fernandez, Rosa Basagoiti, Eduardo De La Torre, and Justo Novo. Development of an Aeronautical Electromechanical Actuator with Real Time Health Monitoring Capability. In *Recent Advances in Aerospace Actuation Systems and Components*, Toulouse, France, 2010.

[22] Satish Rajagopalan, Jose M. Aller, Jose A. Restrepo, Thomas G. Habetler, and Ronald G. Harley. Detection of Rotor Faults in Brushless DC Motors Operating Under Nonstationary Conditions. *IEEE Transactions on Industry Applications*, 42(6):1464–1477, November 2006.

[23] J Royo, R Segui, A Pardina, S Nevot, and F J Arcega. Machine Current Signature Analysis as a Way for Fault Detection in Permanent Magnet Motors in Elevators. In *Proceedings of the 2008 International Conference on Electrical Machines*, pages 1–6, 2008.

[24] Jason R Stack, Thomas G Habetler, and Ronald G Harley. Fault-Signature Modeling and Detection of Inner-Race Bearing Faults. *IEEE Transactions on Industry*, 42(1):61–68, 2006.

[25] J.R. Stack, T.G. Habetler, and R.G. Harley. Fault Classification and Fault Signature Production for Rolling Element Bearings in Electric Machines. *IEEE Transactions on Industry Applications*, 40(3):735–739, May 2004.

[26] C.T. Yiakopoulos, K.C. Gryllias, and I.A. Antoniadis. Rolling Element Bearing Fault Detection in Industrial Environments based on a K-means Clustering Approach. *Expert Systems with Applications*, 38(3):2888–2911, March 2011.

[27] Zhen Dong and Xinjian Jiang. Failure Detection and Diagnosis System of BLDCM with Dynamic Load. In *Prognostics & System Health Management Conference*, Beijing, 2012.

[28] Xiang-Qun Liu, Hong-Yue Zhang, Jun Liu, and Jing Yang. Fault Detection and Diagnosis of Permanent-Magnet DC Motor Based on Parameter Estimation and Neural Network. *IEEE Transactions on Industrial Electronics*, 47(5):1021–1030, 2000.

[29] Kaiping Yu, Fang Yang, Hong Guo, and Jinquan Xu. Fault Diagnosis and Location of Brushless DC Motor System Based on Wavelet Transform and Artificial

Neural Network. In *International Conference on Electrical Machines and Systems*, number 1, pages 2–6, 2010.

[30] Edward Balaban, Abhinav Saxena, Sriram Narasimhan, Indranil Roychoudhury, Kai F Goebel, and Michael T Koopmans. Airborne Electro-Mechanical Actuator Test Stand for Development of Prognostic Health Management Systems. In *Annual Conference of the Prognostics and Health Management Society*, pages 1–13, 2010.

[31] Michael T Koopmans, Rudolph C Hooven, and Irem Y Tumer. Reliability Based Design Recommendations for an Electromechanical Actuator Test Stand. In *Annual Conference of the Prognostics and Health Management Society*, pages 1–15, 2010.

[32] Michael T Koopmans and Irem Y Tumer. Function-Based Analysis and Redesign of a Flyable Electromechanical Actuator Test Stand. In *Proceedings of the ASME International Design Engineering Technical Conferences & Computers and Information in Engineering Conference*, pages 1–12, Montreal, Quebec, Canada, 2010.

[33] R Dixon and A W Pike. Application of Condition Monitoring to an Electromechanical Actuator: A Parameter Estimation Based Approach. *Computer & Control Engineering Journal*, (April):71–81, 2002.

[34] Olaf Moseler and Rolf Isermann. Application of Model-Based Fault Detection to a Brushless DC Motor. *IEEE Transactions on Industrial Electronics*, 47(5):1015–1020, 2000.

[35] Ravindra Patankar and Liangtao Zhu. Brushless DC Motor Actuator Health Monitoring and Degradation Compensation via Real-Time Multiple Parameter Estimation. *International Journal of Automation and Control*, 1(1):48–63, 2007.

[36] Irina Trendafilova. An Automated Procedure for Detection and Identification of Ball Bearing Damage using Multivariate Statistics and Pattern Recognition. *Mechanical Systems and Signal Processing*, 24(6):1858–1869, August 2010.

[37] Karel J Keesman. *System Identification: An Introduction*. Springer-Verlag, 1st edition, 2011.

[38] Lennart Ljung. System Identification Toolbox User's Guide. Technical report, The MathWorks Inc., Natick,MA, 2012.

[39] Henry a. Sodano, Jae-Sung Bae, Daniel J. Inman, and W. Keith Belvin. Improved Concept and Model of Eddy Current Damper. *ASME Journal of Vibration and Acoustics*, 128(3):294, 2006.

[40] Ramin S Esfandiari and Bei Lu. *Modeling and Analysis of Dynamic Systems*. CRC Press, 1st edition, 2010.

[41] Konstantin P Louganski. *Modeling and Analysis of a DC Power Distribution System in 21st Century Airlifters*. Ms thesis, Virginia Polytechnic Institute and State University, 1999.

[42] Rolf Isermann. *Fault-Diagnosis Systems: An Introduction from Fault Detection to Fault Tolerance*. Springer-Verlag, 1st edition, 2006.

[43] Agustin Vasquez Arvallo. *Condition-Based Maintenance of Actuator Systems using a Model-Based Approach*. Phd dissertation, The University of Texas at Austin, 2000.

[44] Paul Brian Hvass and Delbert Tesar. Condition Based Maintenance for Intelligent Electromechanical Actuators. Technical report, The University of Texas at Austin, Austin, TX, USA, 2004.

[45] Li Liu. *Robust Fault Detection and Diagnosis for Permanent Magnet Synchronous Motors*. Phd dissertation, Florida State University, 2006.

[46] R P W Duin, P Juszczak, P Paclik, E Pekalska, D De Ridder, D M J Tax, and S Verzakov. PRTools4: A Matlab Toolbox for Pattern Recognition. Technical Report August, Delft University of Technology, 2007.

[47] Lindsay I Smith. A Tutorial on Principal Components Analysis. Technical report, 2002.

[48] Jafar Zarei. Induction Motors Bearing Fault Detection using Pattern Recognition Techniques. *Expert Systems with Applications*, 39(1):68–73, January 2012.

[49] Marion Gilson and Paul Van Den Hof. IV Methods for Closed-Loop System Identification. In *13th IFAC Symposium on System Identification*, number 1996, pages 537–542, 2003.

[50] Paul Van Den Hof. Closed-Loop Issues in System Identification. *Annual Reviews in Control*, 22:173–186, 1998.

[51] A. Garinei and R. Marsili. A new Diagnostic Technique for Ball Screw Actuators. *Measurement*, 45(5):819–828, June 2012.

[52] Wei Zhou, T.G. Habetler, and R.G. Harley. Bearing Fault Detection Via Stator Current Noise Cancellation and Statistical Control. *IEEE Transactions on Industrial Electronics*, 55(12):4260–4269, December 2008.

[53] Rolf Isermann and Olaf Moseler. Model-Based Fault Detection for a Brushless DC Motor using Parameter Estimation. In *24th Annual IEEE Conference of the Industrial Electronics Society*, volume 8, pages 1956–1960, 1998.

[54] J. H. Zhou, Z. W. Zhong, M. Luo, and C. Shao. Wavelet-Based Correlation Modelling for Health Assessment of Fluid Dynamic Bearings in Brushless DC Motors. *The International Journal of Advanced Manufacturing Technology*, 41(5-6):421–429, May 2009.

[55] Carl S Byington, Rolf Orsagh, Pattada Kallappa, Jeremy Sheldon, Michael DeChristopher, Sanket Amin, and Jason Hines. Recent Case Studies in Bearing Fault Detection and Prognosis. In *IEEE Aerospace Conference*, pages 1–8, 2006.

[56] Wayne J Dunstan and Robert R Bitmead. Empirical Estimation of Parameter Distributions in System Identification. In *13th IFAC Symposium on System Identification*, pages 1868–1873, Rotterdam, 2003.

Appendix A

Simulink Models

This appendix contains all the Simulink models utilised in this work and referenced in the preceding chapters. The models are presented under sections that have the same title as those under which the model was originally referenced.

A.1 Example: Fault Detection in DC Motor Drives

The first two block diagrams in this Appendix show the DC motor model with a single-point defect seeding mechanism used to test the fault detection algorithm presented in Chapter 2.

A.2 Simulations

The following block diagrams show a DC motor model based on a state-space representation of the standard DC motor equations. It differs from the previous model in the nature of the single point defect generated (it is more realistic in this model), the option of adding a simulated generlised roughness type defect and the important ability to provide two inputs (in this case, speed command and external load). Furthermore, the system shown here operates in closed loop using a PI controller. It is noted that the controller is merely used to obtain effective tracking and that its appropriate design is not of interest.

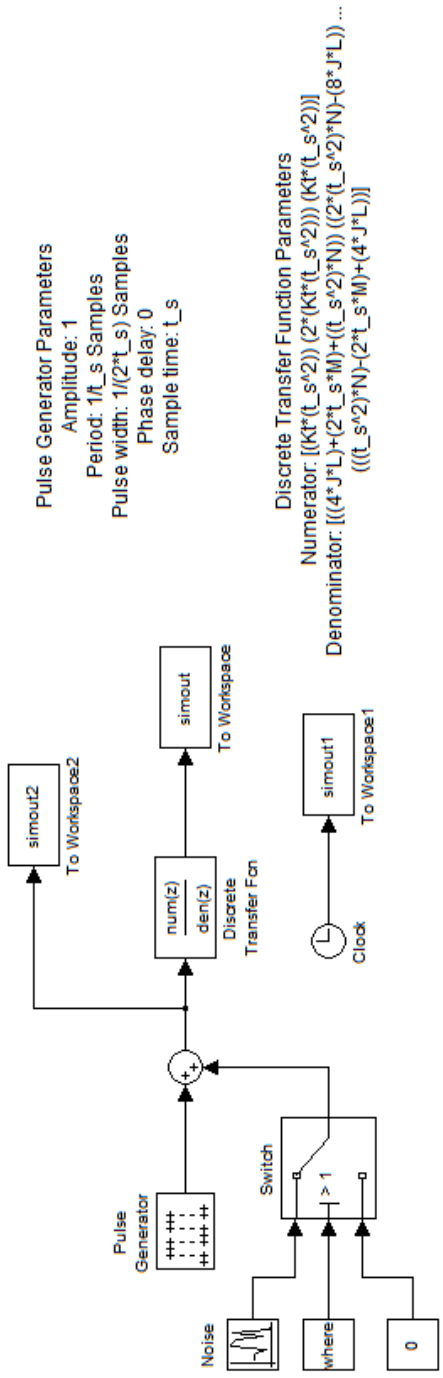

Figure A.1: Simulink block diagram of healthy DC motor

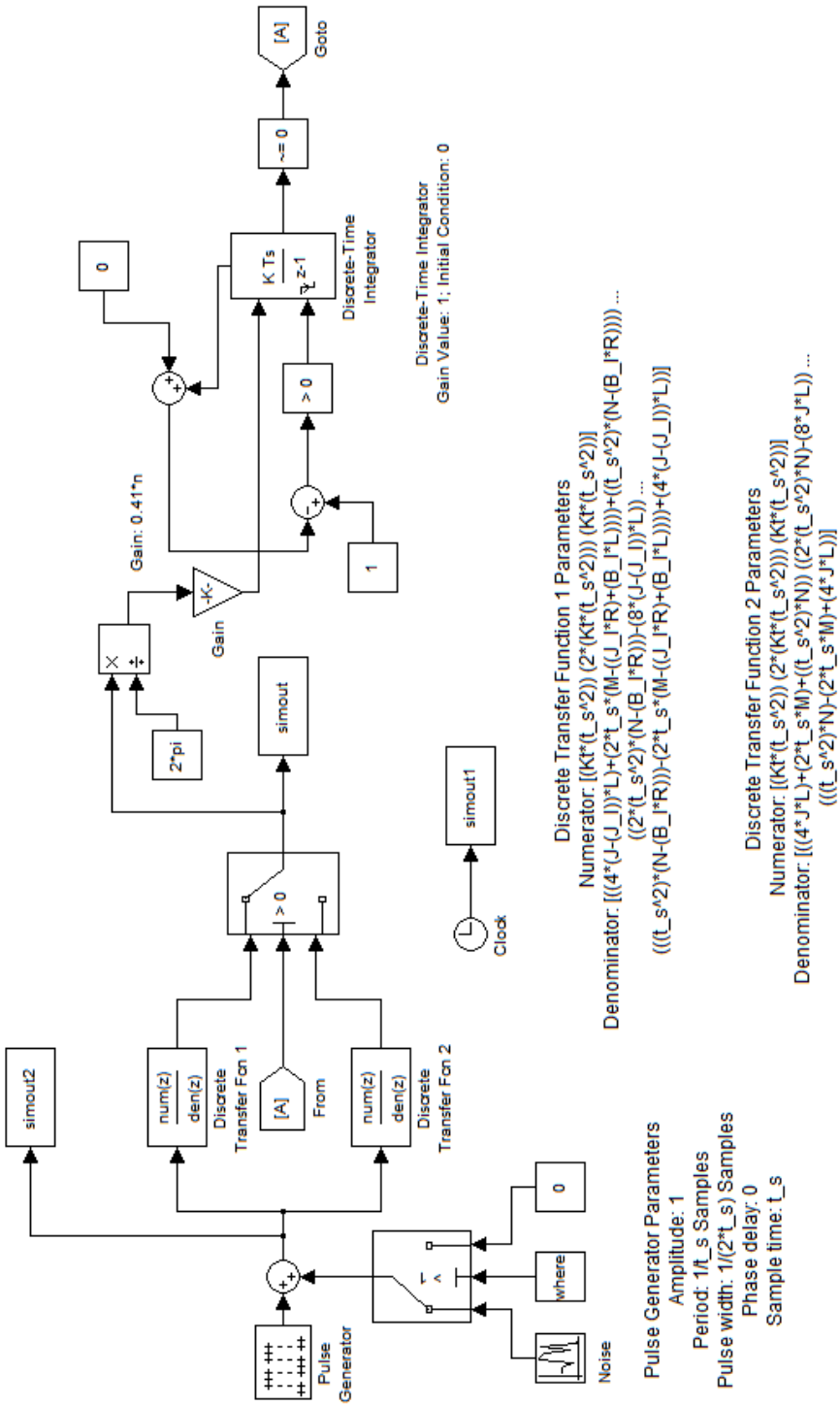

Figure A.2: Simulink block diagram of DC motor seeded with single-point defect

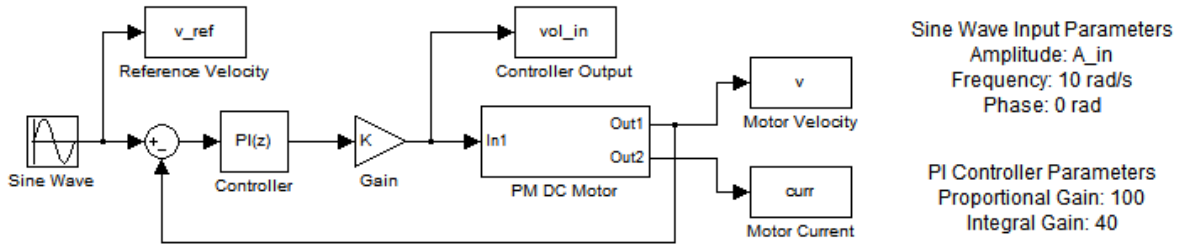

Figure A.3: Simulink block diagram of DC motor seeded with single-point defect and generalised roughess defect used for testing modified fault detection approach via simulation

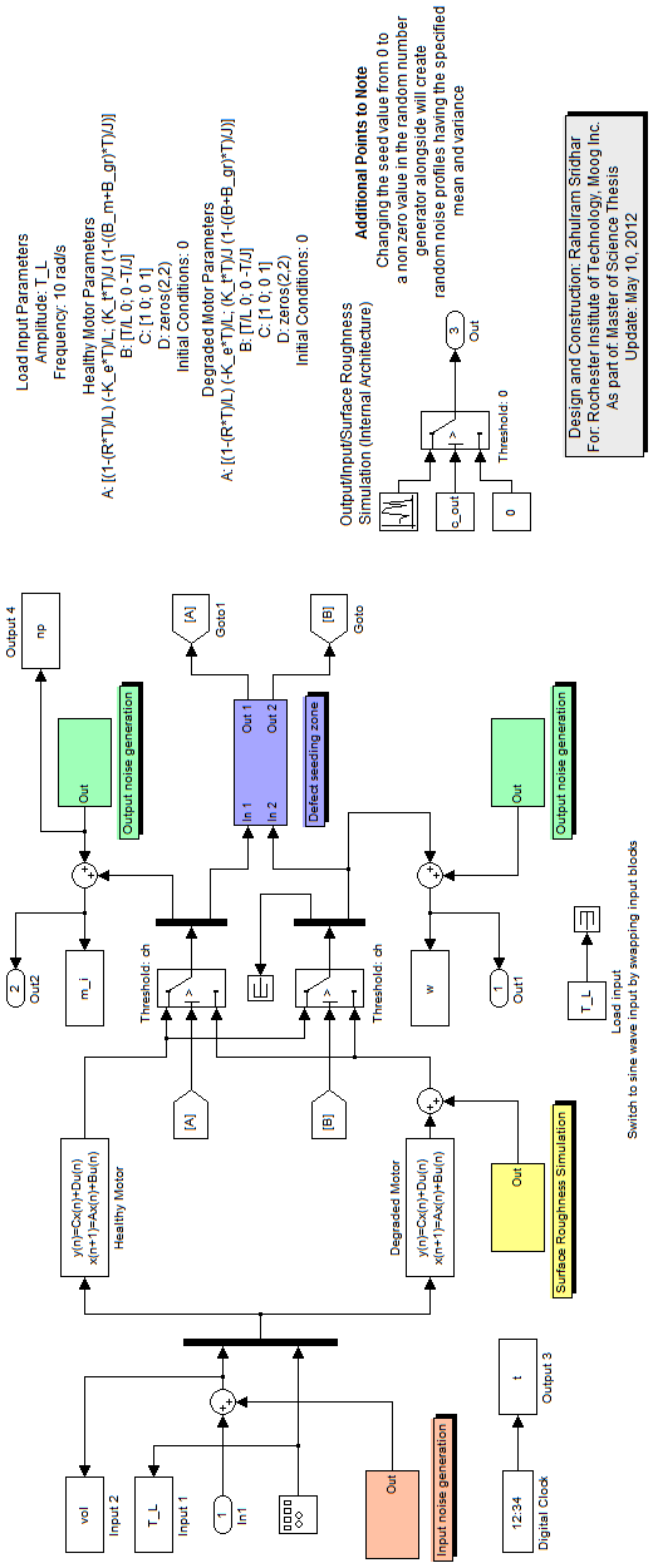

Figure A.4: Internal architecture of PM DC motor shown in Fig. A.2

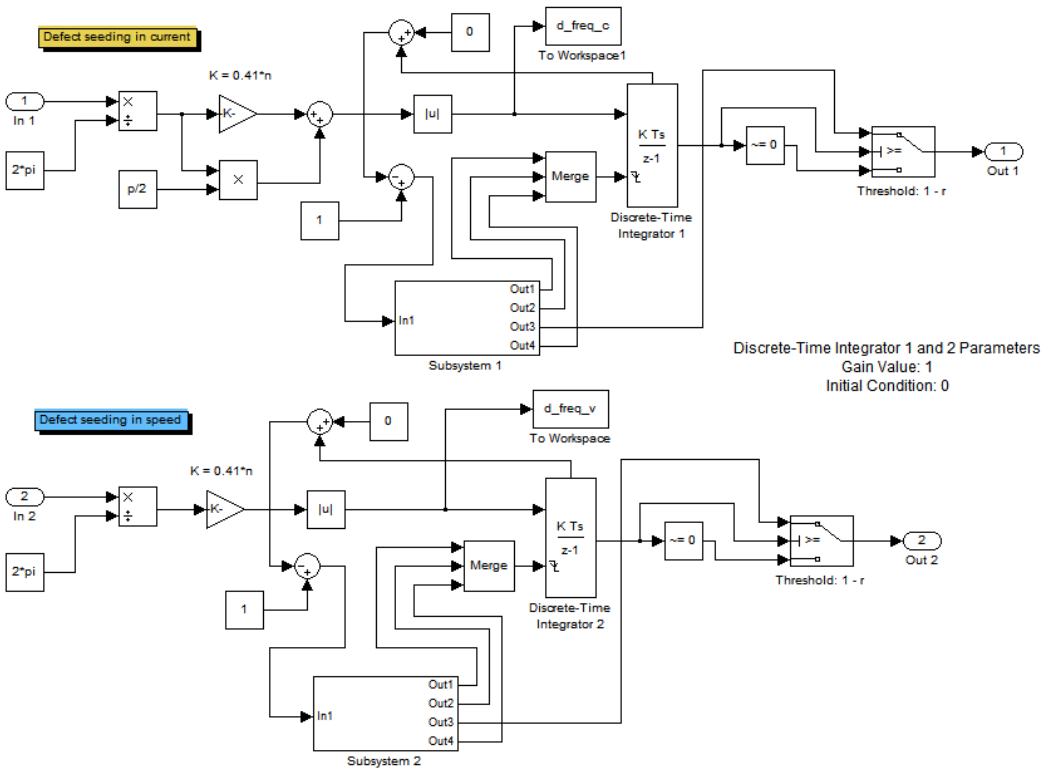

Figure A.5: Internal architecture of defect seeding zone shown in Fig. A.2

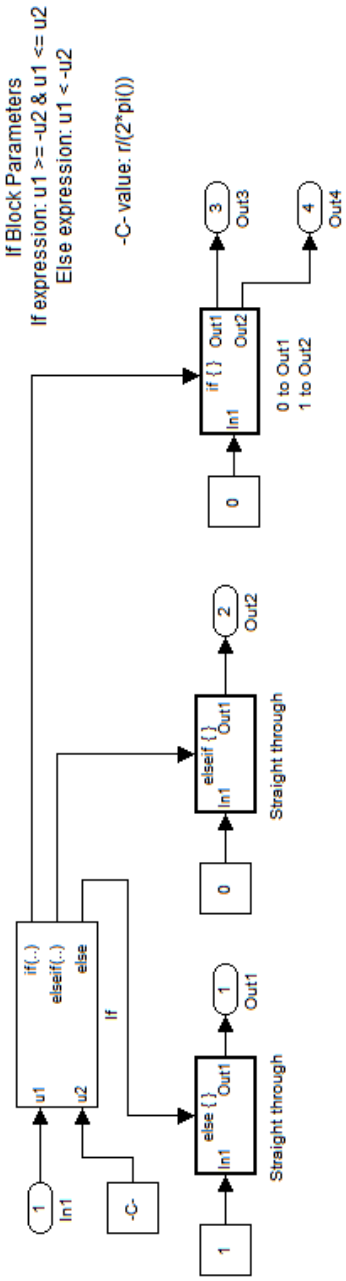

Figure A.6: Internal architecture of Subsystems 1 and 2 shown in Fig. A.2

Appendix B

MATLAB Programs

This appendix contains some of the MATLAB programs utilized in this work and referenced in the preceding chapters. The programs are presented under sections that have the same title as those under which it was originally referenced.

B.1 Estimating Parameters for the ARX Model

The following code estimates the parameter vector for a sample data set using either Eq. (2.17) or (2.21).

```
% ARX Parameter Estimation using LSE Technique %
% By Rahulram Sridhar, ME Department, RIT %

clear all; clc

% Create sample data set
sys_cont = tf([2],[2 1]);
t_s = 0.01; % Sample time
sys_disc = c2d(sys_cont,t_s)
[y t_1]=step(sys_disc,0:t_s:2); [u t_2]=step(c2d(tf(1,1),t_s),0:t_s:2);
data = iddata(y,u,t_s); compdata = data;

% Choose whether or no to add Gaussian white noise
noise_choice = input('Include noise? (1 = yes; 2 = no): ');
if noise_choice == 1
    stnr = 25; % Select signal-to-noise ratio
    y_noisy = awgn(y,stnr,'measured');
    dataset_noisy = iddata(y_noisy,u,t_s); % Create noisy data set
    data = dataset_noisy;
end

lsemethod = input('Choose LSE approach - (1) Matrix or (2) Standard : ');
if lsemethod == 1
% Estimate ARX model parameters using the least squares technique with the
% matrix formulation
p = 1; % Choice of model order (needs to be same for output and input)
for i = 1:p
```

```
        X(:,i) = -(data.y((p+1-i):length(data.y)-i));
        X(:,p+i) = data.u((p+1-i):length(data.u)-i);
end
theta = (X'*X)\(X'*(data.y(p+1:length(data.y))));
% Extract estimated A and B polynomials from theta vector
B = theta(p+1:length(theta))'; A = [1 theta(1:p)'];
else
% Estimate ARX model parameters using the least squares technique with the
% standard formulation
N = length(data.y); sum1 = 0; sum2 = 0;
% Different model orders for input and output is possible
n = 1; % Select number of output terms
m = 1; % Select number of input terms
% The following step of adding zeros for prior time steps is suggested in
% Lennart Ljung's book System Identification: Theory for the User
idataappend = zeros(m,1); odataappend = zeros(n,1);
inputdata = [idataappend' (data.u)'];
outputdata = [odataappend' (data.y)'];
for k = 1:N
    phi1 = outputdata(k:(k+n-1)); phi1 = phi1*-1;
    phi2 = inputdata(k:(k+m-1));
    phi = [phi1 phi2]; % Regression vector
    sum1 = sum1 + (phi'*phi);
    sum2 = sum2 + (phi'*outputdata(k+n));
end
theta = (inv(sum1))*sum2; theta(1:n) = theta(n:-1:1);
theta(n+1:n+m) = theta(n+m:-1:n+1);
% Extract estimated A and B polynomials from theta vector
A = [1 theta(1:n)']; B = theta(n+1:n+m);
end

% Final model structure presentation and brief analysis
fprintf('\nEvaluation by Least Squares\n');
fprintf('Model estimated from data. Ignore comment below.\n');
m_ls = idpoly(A,B,[],[],[],t_s)
fprintf('\n*********************************************\n');
fprintf('************BRIEF MODEL ANALYSIS***************\n');
g = input('Make a choice (1 = Simulation; 2 = Prediction): ');
if g == 1
    figure(1);
    yhat_ls = sim(m_ls,compdata.u,'InitialState','z');
    plot(t_1,compdata.y,t_1,data.y,t_1,yhat_ls);
elseif g == 2
    k = 1; % K-step predictor
    figure(1);
    yhat_ls = predict(m_ls,compdata,k); yhat_ls = yhat_ls.y;
    plot(t_1,compdata.y,t_1,data.y,t_1,yhat_ls);
else
    error('Invalid input. Re-execute code.\n');
end
```

```
xlabel('Time (s)'); ylabel('Signal Amplitude (V)');
title('LS Algorithm Performance with ARX Model');
legend('True Model Simulation','Measured Data','Simulation/Prediction',0);
grid;
fit = 100*(1-norm(compdata.y-yhat_ls)/norm(compdata.y-mean(compdata.y)));
fprintf('\nThe fit is: %5.2f %%.\n', fit);

% End of program
```

The following code estimates the parameter vector for a sample data set using either the advanced IV estimation approach, more information on which is available in [3], or the basic IV approach as described in chapter 2.

```
% ARX Parameter Estimation using IV Technique %
% By Rahulram Sridhar, ME Department, RIT %

clear all; clc

% Create sample data set
sys_cont = tf([2],[2 1]);
t_s = 0.01; % Sample time
sys_disc = c2d(sys_cont,t_s)
[y t_1]=step(sys_disc,0:t_s:2); [u t_2]=step(c2d(tf(1,1),t_s),0:t_s:2);
data = iddata(y,u,t_s); compdata = data;

% Choose whether or no to add Gaussian white noise
noise_choice = input('Include noise? (1 = yes; 2 = no): ');
if noise_choice == 1
    stnr = 25; % Select signal-to-noise ratio
    y_noisy = awgn(y,stnr,'measured');
    dataset_noisy = iddata(y_noisy,u,t_s); % Create noisy data set
    data = dataset_noisy;
end

% Estimate ARX model parameters using the IV technique
p = 1; % Choice of model order (needs to be same for output and input)
for i = 1:p
    X(:,i) = -(data.y((p+1-i):length(data.y)-i));
    X(:,p+i) = data.u((p+1-i):length(data.u)-i);
end
theta = (X'*X)\(X'*(data.y(p+1:length(data.y))));
% Extract estimated A and B polynomials from theta vector
B = theta(p+1:length(theta))'; A = [1 theta(1:p)'];

% Implement IV method
count = input('Enter the number of iterations for IV method: ');
choice2 = input('Choose algorithm type (1 = Advanced; 2 = Basic): ');
for l = 1:count
    y_new = sim(idpoly(A,B,1,1,1,0,t_s),u);
```

```
    data2 = iddata(y_new,u,t_s);
    for i = 1:p
        Z(:,i) = -(data2.y((p+1-i):length(data2.y)-i));
        Z(:,p+i) = data2.u((p+1-i):length(data2.u)-i);
    end
    theta = (Z'*X)\(Z'*(data.y(p+1:length(data2.y))));
    B = theta(p+1:length(theta))'; A = [1 theta(1:p)'];
    if choice2 == 1
        e_IV = resid(idpoly(A,B,1,1,1,0,t_s),data); % Prediction errors
        % Generate AR filter
        for d = 1:(2*p)
            M(:,i) = -(e_IV.y(((2*p)+1-i):length(e_IV.y)-i));
        end
        phi = (M'*M)\(M'*(e_IV.y((2*p)+1:length(e_IV.y))));
        D = [1 theta(1:(2*p))'];
        y_L = sim(idpoly(D,1,1,1,1,0,t_s),data.y); % Filter process output
        % Filter regressor matrix
        for w = 1:(2*p)
            X_L(:,w) = sim(idpoly(D,1,1,1,1,0,t_s),X(:,w));
            Z_L(:,w) = sim(idpoly(D,1,1,1,1,0,t_s),Z(:,w));
        end
        theta_IV = ((Z_L)'*(X_L))\((Z_L)'*(y_L(p+1:length(data2.y))));
        B = theta_IV(p+1:length(theta_IV))'; A = [1 theta_IV(1:p)'];
    elseif choice2 == 2
        continue
    else
        fprintf('\nIncorrect choice. Please re-execute program\n');
        break
    end
end

% Final model structure presentation and brief analysis
fprintf('\nEvaluation by Instrumental Variables\n');
fprintf('Model estimated from data. Ignore comment below.\n');
m_ls = idpoly(A,B,[],[],[],t_s)
fprintf('\n*********************************************\n');
fprintf('************BRIEF MODEL ANALYSIS***************\n');
g = input('Make a choice (1 = Simulation; 2 = Prediction): ');
if g == 1
    figure(1);
    yhat_ls = sim(m_ls,compdata.u,'InitialState','z');
    plot(t_1,compdata.y,t_1,data.y,t_1,yhat_ls);
elseif g == 2
    k = 1; % K-step predictor
    figure(1);
    yhat_ls = predict(m_ls,compdata,k); yhat_ls = yhat_ls.y;
    plot(t_1,compdata.y,t_1,data.y,t_1,yhat_ls);
else
    error('Invalid input. Re-execute code.\n');
end
```

```
xlabel('Time (s)'); ylabel('Signal Amplitude (V)');
title('IV Algorithm Performance with ARX Model');
legend('True Model Simulation','Measured Data','Simulation/Prediction',0);
grid;
fit = 100*(1-norm(compdata.y-yhat_ls)/norm(compdata.y-mean(compdata.y)));
fprintf('\nThe fit is: %5.2f %%.\n', fit);

% End of program
```

B.2 Estimating Parameters for the ARMAX Model

The following code estimates the parameter vector for a sample data set using a multi-stage least squares estimation approach.

```
% ARMAX Parameter Estimation using Multi-Stage LSE Technique %
% By Rahulram Sridhar, ME Department, RIT %

clear all; clc

% Create sample data set
sys_cont = tf([2],[2 1]);
t_s = 0.01; % Sample time
sys_disc = c2d(sys_cont,t_s)
[y t_1]=step(sys_disc,0:t_s:2); [u t_2]=step(c2d(tf(1,1),t_s),0:t_s:2);
data = iddata(y,u,t_s); compdata = data;

% Choose whether or no to add Gaussian white noise
noise_choice = input('Include noise? (1 = yes; 2 = no): ');
if noise_choice == 1
    stnr = 25; % Select signal-to-noise ratio
    y_noisy = awgn(y,stnr,'measured');
    dataset_noisy = iddata(y_noisy,u,t_s); % Create noisy data set
    data = dataset_noisy;
end

% Estimate ARMAX model parameters using the least squares technique
N = length(data.y); sum1 = 0; sum2 = 0;
% Different model orders for input and output is possible
n = 2; % Select number of output terms
m = 1; % Select number of input terms
% The following step of adding zeros for prior time steps is suggested in
% Lennart Ljung's book System Identification: Theory for the User
idataappend = zeros(m,1); odataappend = zeros(n,1);
inputdata = [idataappend' (data.u)'];
outputdata = [odataappend' (data.y)'];
for k = 1:N
    phi1 = outputdata(k:(k+n-1)); phi1 = phi1*-1;
    phi2 = inputdata(k:(k+m-1));
```

```
    phi = [phi1 phi2]; % Regression vector
    sum1 = sum1 + (phi'*phi);
    sum2 = sum2 + (phi'*outputdata(k+n));
end
theta = (inv(sum1))*sum2; param = max([n m]);
e = zeros(1,N+param); l_e = length(e);
theta = [theta' zeros(1,param)]';
count = input('Enter number of iterations of multi-stage LS: ');
for iter = 1:count
% Calculation of prediction errors of the ARX Model
for x = (param+1):(N+param)
    e(x) = (outputdata((x-1):-1:(x-n))*theta(1:n))-...
        (inputdata((x-1):-1:(x-m))*theta(n+1:n+m))-...
        (e((x-1):-1:(x-param))*theta(n+m+1:n+m+param));
end
% Use LS to estimate parameters
sum3 = 0; sum4 = 0;
for y = 1:N
    phi3 = outputdata(y:(y+n-1));  phi3 = phi3*-1;
    phi4 = inputdata(y:(y+m-1));
    phi5 = e(y:(y+param-1));
    phi = [phi3 phi4 phi5]; % Regression vector
    sum3 = sum3 + (phi'*phi);
    sum4 = sum4 + (phi'*outputdata(y+n));
end
theta = (inv(sum3))*sum4;
end
theta(1:n) = theta(n:-1:1);
theta(n+1:n+m) = theta(n+m:-1:n+1);
theta(n+m+1:n+m+param) = theta(n+m+param:-1:n+m+1);
A = [1 theta(1:n)']; B = theta(n+1:n+m);
C = [1 theta(n+m+1:n+m+param)'];

% Final model structure presentation and brief analysis
fprintf('\nEvaluation by Multi-Stage Least Squares\n');
fprintf('Model estimated from data. Ignore comment below.\n');
m_ls = idpoly(A,B,C,[],[],t_s)
fprintf('\n*********************************************\n');
fprintf('*************BRIEF MODEL ANALYSIS***************\n');
g = input('Make a choice (1 = Simulation; 2 = Prediction): ');
if g == 1
    figure(1);
    yhat_ls = sim(m_ls,compdata.u,'InitialState','z');
    plot(t_1,compdata.y,t_1,data.y,t_1,yhat_ls);
elseif g == 2
    k = 1; % K-step predictor
    figure(1);
    yhat_ls = predict(m_ls,compdata,k); yhat_ls = yhat_ls.y;
    plot(t_1,compdata.y,t_1,data.y,t_1,yhat_ls);
else
```

```
    error('Invalid input. Re-execute code.\n');
end
xlabel('Time (s)'); ylabel('Signal Amplitude (V)');
title('Multi-Stage LS Algorithm Performance with ARMAX Model');
legend('True Model Simulation','Measured Data','Simulation/Prediction',0);
grid;
fit = 100*(1-norm(compdata.y-yhat_ls)/norm(compdata.y-mean(compdata.y)));
fprintf('\nThe fit is: %5.2f %%.\n', fit);

% End of program
```

B.3 Estimating Parameters for the OE Model

The following code estimates the parameter vector for a sample data set using a multi-stage least squares estimation approach.

```
% OE Parameter Estimation using Multi-Stage LSE Technique %
% By Rahulram Sridhar, ME Department, RIT %

clear all; clc

% Create sample data set
sys_cont = tf([2],[2 1]);
t_s = 0.01; % Sample time
sys_disc = c2d(sys_cont,t_s)
[y t_1]=step(sys_disc,0:t_s:2); [u t_2]=step(c2d(tf(1,1),t_s),0:t_s:2);
data = iddata(y,u,t_s); compdata = data;

% Choose whether or no to add Gaussian white noise
noise_choice = input('Include noise? (1 = yes; 2 = no): ');
if noise_choice == 1
    stnr = 25; % Select signal-to-noise ratio
    y_noisy = awgn(y,stnr,'measured');
    dataset_noisy = iddata(y_noisy,u,t_s); % Create noisy data set
    data = dataset_noisy;
end

% Estimate OE model parameters using multi-stage least squares estimation
N = length(data.y);

m = input('Enter number of terms to be considered: ');
n = m; % For initial ARX model approximation

% Create new data vectors compatible with program
% Values outside the measured range are assumed zero

idataappend = zeros(m,1); odataappend = zeros(n,1);
inputdata = [idataappend' (data.u)'];
```

```
outputdata = [odataappend' (data.y)'];

% Estimation of OE Model
sum1 = 0; sum2 = 0;
for k = 1:N
    phi1 = outputdata(k:(k+n-1)); phi1 = phi1*-1;
    phi2 = inputdata(k:(k+m-1));
    phi = [phi1 phi2]; % Regression vector
    sum1 = sum1 + (phi'*phi);
    sum2 = sum2 + (phi'*outputdata(k+n));
end
theta = (inv(sum1))*sum2; A = [1 theta(1:n)'];
count = input('Enter number of iterations of multi-stage LSE: ');
for iter = 1:count
    if iter == 1
        m_filt = idpoly(A,1,[],[],[],t_s);
        y_filt = sim(m_filt,data.y,'InitialState','z');
        u_filt = sim(m_filt,data.u,'InitialState','z');
        idataappend = zeros(m,1); odataappend = zeros(n,1);
        u_filt = [idataappend' (u_filt)'];
        y_filt = [odataappend' (y_filt)'];
        sum3 = 0; sum4 = 0;
        for k2 = 1:N
            phi3 = y_filt(k:(k+n-1)); phi3 = phi3*-1;
            phi4 = u_filt(k:(k+m-1));
            phi_filt = [phi3 phi4]; % Regression vector
            sum13 = sum3 + (phi_filt'*phi_filt);
            sum4 = sum4 + (phi_filt'*y_filt(k+n));
        end
        theta2 = (inv(sum3))*sum4;
        B = theta(m+1:length(theta))'; F = [1 theta(1:n)'];
    else
        m_filt = idpoly(F,1,[],[],[],t_s);
        y_filt = sim(m_filt,data.y,'InitialState','z');
        u_filt = sim(m_filt,data.u,'InitialState','z');
        idataappend = zeros(m,1); odataappend = zeros(n,1);
        u_filt = [idataappend' (u_filt)'];
        y_filt = [odataappend' (y_filt)'];
        sum3 = 0; sum4 = 0;
        for k2 = 1:N
            phi3 = y_filt(k:(k+n-1)); phi3 = phi3*-1;
            phi4 = u_filt(k:(k+m-1));
            phi_filt = [phi3 phi4]; % Regression vector
            sum13 = sum3 + (phi_filt'*phi_filt);
            sum4 = sum4 + (phi_filt'*y_filt(k+n));
        end
        theta2 = (inv(sum3))*sum4;
        B = theta(m+1:length(theta))'; F = [1 theta(1:n)'];
    end
end
```

```
% Final model structure presentation and brief analysis
fprintf('\nEvaluation by Multi-Stage Least Squares\n');
fprintf('Model estimated from data. Ignore comment below.\n');
m_ls = idpoly(1,B,[],[],F,t_s)
fprintf('\n*********************************************\n');
fprintf('************BRIEF MODEL ANALYSIS***************\n');
g = input('Make a choice (1 = Simulation; 2 = Prediction): ');
if g == 1
    figure(1);
    yhat_ls = sim(m_ls,compdata.u,'InitialState','z');
    plot(t_1,compdata.y,t_1,data.y,t_1,yhat_ls);
elseif g == 2
    k = 1; % K-step predictor
    figure(1);
    yhat_ls = predict(m_ls,compdata,k); yhat_ls = yhat_ls.y;
    plot(t_1,compdata.y,t_1,data.y,t_1,yhat_ls);
else
    error('Invalid input. Re-execute code.\n');
end
xlabel('Time (s)'); ylabel('Signal Amplitude (V)');
title('Multi-Stage LS Algorithm Performance with OE Model');
legend('True Model Simulation','Measured Data','Simulation/Prediction',0);
grid;
fit = 100*(1-norm(compdata.y-yhat_ls)/norm(compdata.y-mean(compdata.y)));
fprintf('\nThe fit is: %5.2f %%.\n', fit);

% End of program
```

B.4 Example: Fault Detection in DC Motor Drives

The following program is the fault detection algorithm used in the example on fault detection in DC motor drives. It is used in conjunction with the Simulink models shown in Section A.1 of Appendix A. Some form of this program is also used to generate the results in Chapters 4 and 5. A section of the code also contains an initial attempt of the piece-wise parameter estimation approach discussed under future work.

```
% Example: Fault Detection in DC Motor Drives using System Identification
% Techniques
% By Rahulram Sridhar, ME Department, RIT %

% Seeded Fault Type: Single-point Fault

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
clear all; clc
% clearvars -except net1 net2; clc % use when accumulating data
mainchoice = 1; ctr = 1; figctr = 2; h_uh = 1; t_s = 0.002; % Sample time
var = input('Enter noise variance: ');
fprintf('Healthy data set.\n');
while mainchoice == 1
    where = 1;
    % Create sample data set
    if h_uh == 1
        % ***DC Motor Parameters***
        Kt = 0.05; Ke = 0.05;
        L = 4.5e-7; R = 0.5;
        J_m = 0.00025; J_l = 0;
        B_m = 0.0001; B_l = 0;
        J = J_m+J_l; B = B_m+B_l;
        M = (B*L)+(J*R); N = (B*R)+(Ke*Kt);
        % ***Simulate***
        simdata = sim('Healthy',1);
        x2(:,1) = simout; % store data for plotting in future
    else
        % ***DC Motor Parameters***
        Kt = 0.05; Ke = 0.05;
        L = 4.5e-7; R = 0.5;
        J_m = 0.00025; J_l = 0.3e-4;
        B_m = 0.0001; B_l = 0.3e-4;
        J = J_m+J_l; B = B_m+B_l;
        M = (B*L)+(J*R); N = (B*R)+(Ke*Kt);
        n = 9;
        % ***Simulate***
        simdata = sim('Degraded',1);
        x2(:,2) = simout; % store data for plotting
        subplot(2,3,1);
        plot(simout1,simout2,simout1,x2(:,2),simout1,x2(:,1));
        xlabel('Time (s)'); ylabel('Amplitude');
        title('(a)');
        legend('Input (V)','Degraded (rad/s)','Healthy (rad/s)');
        subplot(2,3,4);
        plot(simout1,simout2,simout1,simout); grid;
        xlabel('Time (s)'); ylabel('Amplitude');
        title('(d)');
        legend('Input (V)','Degraded (rad/s)');
    end
    y = simout; u = simout2; t_1 = simout1;
    clear simdata simout simout1 simout2
    data = iddata(y,u,t_s); compdata = data; % noiseless data
    noise_choice = input('Include noise? (1 = yes; 2 = no): ');
    if noise_choice == 1
        where = input('Include noise at 1. Output or 2. Input?: ');
        if where == 1
```

```
        y_noisy = y + sqrt(var).*randn(length(y),1);
        dataset_noisy = iddata(y_noisy,u,t_s); % Create noisy data set
        data = dataset_noisy;
    elseif where == 2
        if h_uh == 1
            simdata = sim('Healthy',1);
            y_noisy = simout;
        else
            simdata = sim('Degraded',1);
            y_noisy = simout;
        end
        % Create noisy data set
        dataset_noisy = iddata(y_noisy,simout2,t_s);
        data = dataset_noisy;
    else
        disp('Incorrect choice.\n'); break;
    end
end

% Estimate model parameters
m = 1; % Model order (validated based on L value - do not change)
for i = 1:m
    X(:,i) = -(data.y((m+1-i):length(data.y)-i));
    X(:,m+i) = data.u((m+1-i):length(data.u)-i);
end
theta = (X'*X)\(X'*(data.y(m+1:length(data.y))));
B = theta(m+1:length(theta))'; A = [1 theta(1:m)'];
count = input('Number of estimation iterations: ');
choice = input('Choice of IV algorithm (1. Advanced; 2. Basic): ');
for l = 1:count
    y_new = sim(idpoly(A,B,1,1,1,0,t_s),u);
    data2 = iddata(y_new,u,t_s);
    for i = 1:m
        Z(:,i) = -(data2.y((m+1-i):length(data2.y)-i));
        Z(:,m+i) = data2.u((m+1-i):length(data2.u)-i);
    end
    theta = (Z'*X)\(Z'*(data.y(m+1:length(data2.y))));
    B = theta(m+1:length(theta))'; A = [1 theta(1:m)'];
    if choice == 1
        % Calculate prediction errors
        e_IV = resid(idpoly(A,B,1,1,1,0,t_s),data);
        % Generate AR filter
        for d = 1:(2*m)
            Q(:,i) = -(e_IV.y(((2*m)+1-i):length(e_IV.y)-i));
        end
        phi = (Q'*Q)\(Q'*(e_IV.y((2*m)+1:length(e_IV.y))));
        D = [1 theta(1:(2*m))'];
        % Filter process output
        y_L = sim(idpoly(D,1,1,1,1,0,t_s),data.y);
        % Filter regressor matrix
```

```
        for w = 1:(2*m)
            X_L(:,w) = sim(idpoly(D,1,1,1,1,0,t_s),X(:,w));
            Z_L(:,w) = sim(idpoly(D,1,1,1,1,0,t_s),Z(:,w));
        end
        theta_IV = ((Z_L)'*(X_L))\((Z_L)'*(y_L(m+1:length(data2.y))));
        B = theta_IV(m+1:length(theta_IV))'; A = [1 theta_IV(1:m)'];
    elseif choice == 2
        continue
    else
        fprintf('\nIncorrect choice. Please re-execute program\n');
        break
    end
end
fprintf('\nEstimation Results\n');
fprintf('Model estimated from data. Ignore comment below.\n');
m_iv = idpoly(A,B,1,1,1,0,t_s)
fprintf('\n*********************************************\n');
fprintf('************BRIEF MODEL ANALYSIS***************\n');
yhat_iv = sim(m_iv,compdata.u);
subplot(2,3,figctr); plot(t_1,compdata.y,t_1,data.y,t_1,yhat_iv);
xlabel('Time (s)'); ylabel('Amplitude (rad/s)');
legend('Truth','Data','Model',0);
figctr = figctr+1;
fit_iv = 100*(1-norm(compdata.y-yhat_iv)/norm(compdata.y...
    -mean(compdata.y)));
fprintf('\nThe fit is: %5.2f %%.\n', fit_iv);

% Feature Extraction and Plotting of Distributions
[cm_iv ss_iv var_vec_iv] = health_class(A,B,Z,data,t_s,m);
[f x] = gaussplot(var_vec_iv,m,A,B);
pdfdata(:,ctr:ctr+(2*m)-1) = f;
t_pdf(:,ctr:ctr+(2*m)-1) = x;
if figctr < 4
    fprintf('Degraded data set.\n'); h_uh = 2;
else
    mainchoice = 2;
end
if mainchoice == 1
    ctr = ctr+(2*m);
else
    % Uncomment below to store data (manually change index in steps of
    % 4 with every run of the code)
    % net1(:,1:4) = t_pdf; net2(:,1:4) = pdfdata;
    figctr = figctr + 1;
    for h = 1:m:(2*m)
        subplot(2,3,figctr);
        plot(t_pdf(:,h),pdfdata(:,h),'-r'); hold on;
        plot(t_pdf(:,(2*m+h)),pdfdata(:,(2*m+h)),'-k'); hold off;
        legend('Healthy','Degraded',0);
        figctr = figctr+1;
```

```
        end
    end
end
```

The following function files generate the individual parameter distributions used for fault detection.

```
function [cov_mat sigma_square var_vec] = health_class(A,B,Y,data,t_s,m)
% health_class: Evaluates the covariance matrix of the estimated parameters
%   Inputs: Estimated parameters A and B, Regression matrices X and Z,
%           iddata object, sample time and choice variable to choose
%           between X and Z.
%   Return: cov_mat - the covariance matrix of the estimated parameters

% By Rahulram Sridhar, ME Department, RIT %

e_id = resid(idpoly(A,B,1,1,1,0,t_s),data); e = e_id.y; clear e_id
sigma_square = (1/(length(Y(:,1))-2))*sum(e(1:length(Y(:,1))).^2);
cov_mat = inv(Y'*Y)*sigma_square; var_vec = diag(cov_mat);
fprintf('\nThe covariance matrix of the estimator is:\n');
disp(cov_mat);
fprintf('The estimated variance in the noise is: %f.\n', sigma_square);
fprintf('The estimated variances in the A parameters are:\n');
for z = 1:m
    fprintf('a_1: %6.4f\n',var_vec(z))
end
fprintf('The estimated variances in the B parameters are:\n');
for z = 1:m
    fprintf('b_1: %6.4f\n',var_vec(z+m))
end
end



function [f x] = gaussplot(var_vec,m,A,B)
% gaussplot: Plots the distributions of the various model parameters
%   Inputs: Vector of variances

% By Rahulram Sridhar, ME Department, RIT %

mean_vec = [A(2:(m+1)) B(1:m)];
for w = 1:2*m
    s = var_vec(w); mu = mean_vec(w); i = (mu-(5*s)); j = (mu+(5*s));
    x(:,w) = [i:(j-i)/1000:j];
    f(:,w) = pdf('Normal',x(:,w),mu,s);
end
end
```

B.5 Simulations

The following code is the main program. A similar program is also used for validation and is not shown here.

```
% Fault Detection Simulation using PM DC Motor Model
% Training Command File for PM DC Motor Simulation
% By Rahulram Sridhar, ME Dept., RIT

% PR Tools (courtesy TU Delft) is used to draw the Bayesian Classification
% Bounds
addpath('C:\Program Files (x86)\MATLAB\R2011a Student\toolbox\prtools');
clear all; clc

% Choice of input and output vectors
fprintf('Feedback load is a mandatory input. Choose 1 additional');
fprintf(' input and 1 output\nfrom the choices for parameter ');
fprintf('estimation.\n\n');
choice1 = input('Input 1 or 2? (1 - Ref. Velocity; 2 - Velocity Error): ');
choice2 = input('Output 1 or 2? (1 - Feedback Velocity; 2 - Current): ');
choice3 = input('Attempt piecewise approach? ("y" or "n"): ','s');

% Order selection
if strcmp(choice3,'n') == 1 || strcmp(choice3,'N') == 1
    [orderset,na,nb,nc] = orderselection(choice1,choice2,choice3);
end

validatechoice = 'y';
while strcmp(validatechoice,'y') == 1 || strcmp(validatechoice,'Y') == 1
    if strcmp(choice3,'n') == 1 || strcmp(choice3,'N') == 1
        % Standard Model (no piece-wise approach)
        % Parameter Estimation
        for K = 1:3
            maincount = 1;
            for k1 = 1:20
                % Load Data Set
                clc; fprintf('Data Set: Healthy/Unhealthy%d\n\n',k1);
                if K == 1
                    load(sprintf('healthy%d',k1));
                elseif K == 2
                    load(sprintf('defectA%d',k1));
                else
                    load(sprintf('defectB%d',k1));
                end
                clc
                % Dataset creation based on above inputs
                u2 = FinalData.Torque_Load; t = FinalData.Time;
                T = t(10)-t(9); N = length(t);
                if choice1 == 1
```

```
            u1 = FinalData.Velocity_Reference;
        elseif choice1 == 2
            u1 = FinalData.Velocity_Reference-...
                FinalData.Motor_Velocity;
        else
            error('Invalid input during choice of input vector');
        end
        if choice2 == 1
            y1 = FinalData.Motor_Velocity;
        elseif choice2 == 2
            y1 = FinalData.Motor_Current;
        else
            error('Invalid input during choice of output vector');
        end
        data = iddata(y1,[u1 u2],T); clear u1 u2 y1 B A
        % Perform Parameter Estimation
        m = iv4(data,orderset,'Focus','Simulation');
        [a,b] = polydata(m); A_D(maincount,:) = a;
        [m_m,m_n] = size(b); starta = 1; startb = 1;
        for ra = 1:m_n
            if b(1,ra) ~= 0
                b_mod(starta) = b(1,ra); starta = starta + 1;
            end
            if b(2,ra) ~= 0
                c(startb) = b(2,ra); startb = startb + 1;
            end
        end
        B_D(maincount,:) = b_mod; C_D(maincount,:) = c;
        maincount = maincount + 1;
    end
    % Training Set Matrix Formation
    clc;
    if K == 1
        A_h = A_D(:,2:(na+1)); B_h = B_D(:,1:nb); C_h = C_D(:,1:nc);
    elseif K == 2
        A_d1 = A_D(:,2:(na+1)); B_d1 = B_D(:,1:nb); C_d1 = C_D(:,1:nc);
    else
        A_d2 = A_D(:,2:(na+1)); B_d2 = B_D(:,1:nb); C_d2 = C_D(:,1:nc);
        break;
    end
    disp('To train using next set of data, press a key'); pause; clc
end
A = [A_h; A_d1; A_d2]; B = [B_h; B_d1; B_d2]; C = [C_h; C_d1; C_d2];
% Centering feature vectors by subtracting means
for k1 = 1:na
    A_dt(:,k1) = A(:,k1)-mean(A(:,k1));
end
for k1 = 1:nb
    B_dt(:,k1) = B(:,k1)-mean(B(:,k1));
end
```

```
    for k1 = 1:nc
        C_dt(:,k1) = C(:,k1)-mean(C(:,k1));
    end
else
    % Piece-wise Model
    % Parameter Estimation
    for K = 1:3
        maincount = 1;
        for k1 = 1:20
            % Load Data Set
            clc; fprintf('Data Set: Healthy/Unhealthy%d\n\n',k1);
            if K == 1
                load(sprintf('healthy%d',k1));
            elseif K == 2
                load(sprintf('defectA%d',k1));
            else
                load(sprintf('defectB%d',k1));
            end
            clc
            % Dataset creation based on above inputs
            u2 = FinalData.Torque_Load; t = FinalData.Time;
            T = t(10)-t(9); N = length(t);
            if choice1 == 1
                u1 = FinalData.Velocity_Reference;
            elseif choice1 == 2
                u1 = FinalData.Velocity_Reference-...
                    FinalData.Motor_Velocity;
            else
                error('Invalid input during choice of input vector');
            end
            if choice2 == 1
                y1 = FinalData.Motor_Velocity;
            elseif choice2 == 2
                y1 = FinalData.Motor_Current;
            else
                error('Invalid input during choice of output vector');
            end
            [PSD, STD, Fvec] = psdestimate(y1,length(y1)-1,0,1/T);
            for k5 = 1:length(PSD)
                if PSD(k5) >= 0.2
                    fundfreq = Fvec(k5);
                end
            end
            data = iddata(y1,[u1 u2],T);
            filtdata = idfilt(data,[0 fundfreq*2*pi]);
            ytemp = filtdata.y;
            % Determine splitting points
            counter1 = 1; % Initial number of split data sets
            slope(1) = 0; counter2 = 2; t_new(1) = T; check2 = 1;
            for k2 = 2:length(ytemp)
```

```
            slope(k2) = (ytemp(k2)-ytemp(k2-1))/(t(k2)-t(k2-1));
        end
        for k2 = 2:length(ytemp)
            if abs(slope(k2)) > abs(min(slope))*0.8
                t_new((counter2),1) = t_new((counter2-1),1)+T;
                y1_new((counter2-1),1) = y1(k2-1);
                u1_new((counter2-1),1) = u1(k2-1);
                u2_new((counter2-1),1) = u2(k2-1);
                counter2 = counter2+1; check2 = check2+1;
            else
                if check2 > 200 % Value is minimum length of dataset
                    % Perform Parameter Estimation
                    if length(t_new) > length(y1_new)
                        t_new(end) = [];
                    elseif length(t_new) < length(y1_new)
                        t_new(end+1) = t_new(end)+T;
                    end
                    data = iddata(y1_new,[u1_new u2_new], T);
                    m = iv4(data,[1 [1 1] [1 1]],'Focus','Simulation');
                    ytemp2 = sim(m,[u1_new u2_new]);
                    fit = 100*(1 - norm(ytemp2 - y1_new)/norm(y1_new...
                        -mean(y1_new)));
                    [a,b] = polydata(m); A_D(maincount,:) = a;
                    B_D(maincount,:) = b(1,2);
                    C_D(maincount,:) = b(2,2);
                    maincount = maincount + 1;
                    check2 = 1; counter2 = 2; counter1 = counter1+1;
                else
                    check2 = 1; counter2 = 2;
                end
            end
        end
    end
    if counter1 == 1
        error('File Splitting Unsuccessful. No Datasets Generated');
    end
    % Training Set Matrix Formation
    clc;
    if K == 1
        A_h = A_D(:,2); B_h = B_D(:,1); C_h = C_D(:,1);
    elseif K == 2
        A_d1 = A_D(:,2); B_d1 = B_D(:,1); C_d1 = C_D(:,1);
    else
        A_d2 = A_D(:,2); B_d2 = B_D(:,1); C_d2 = C_D(:,1);
        break;
    end
    disp('To train using next set of data, press a key'); pause; clc
end
A = [A_h; A_d1; A_d2]; B = [B_h; B_d1; B_d2]; C = [C_h; C_d1; C_d2];
% Centering feature vectors by subtracting means
```

```
    A_dt(:,1) = A(:,1)-mean(A(:,1));
    B_dt(:,1) = B(:,1)-mean(B(:,1));
    C_dt(:,1) = C(:,1)-mean(C(:,1));
end

pca_choice = input('Use PCA? ("y" or "n"): ','s');
if strcmp(pca_choice,'y') == 1 || strcmp(pca_choice,'Y') == 1
    % Principal Component Analysis
    fprintf('Beginning Principal Component Analysis\n\n');
    if (strcmp(choice3,'n') == 1 || strcmp(choice3,'N') == 1) && na > 1
        PM = [A_dt];
    else
        PM = [A_dt B_dt];
    end
    C_mat = cov(PM); % Determine covariance matrix
    [V,D] = eigs(C_mat); % Find normalized eigenvalues and vectors of C
    FV = [V(:,1) V(:,2)]; % Find feature vector (consisting of 2 features)
    A_final = FV'*PM'; A_final_1 = A_final(1,:); A_final_2 = A_final(2,:);
    fprintf('End of Principal Component Analysis\n');
else
    % No PCA
    if (strcmp(choice3,'n') == 1 || strcmp(choice3,'N') == 1) && na > 1
        A_final_1 = (A_dt(:,1))'; A_final_2 = (A_dt(:,2))';
    else
        A_final_1 = (A_dt(:,1))'; A_final_2 = (B_dt(:,1))';
    end
    FV = 0; % dummy
end

if strcmp(choice3,'n') == 1 || strcmp(choice3,'N') == 1
    % Generate Bayesian Classification Bounds
    % Cost Function: Uniform; Prior Probabilities: Equal
    % Bound Type: Quadratic
    for k3 = 1:length(A_final_1)/3
        n1{k3} = num2str('Healthy'); n2{k3} = num2str('Defect A');
        n3{k3} = num2str('Defect B');
    end
    dat1 = [(A_final_1(1:end/3))' (A_final_2(1:end/3))'];
    dat2 = [(A_final_1((end/3)+1:2*end/3))' (A_final_2((end/3)+1:...
        2*end/3))'];
    dat3 = [(A_final_1((2*end/3)+1:end))' (A_final_2((2*end/3)+1:end))'];
    dat = [dat1; dat2; dat3]; lab = [n1'; n2'; n3'];
    dat_a = [dat1; dat2]; lab1 = [n1'; n2']; dat_b = [dat1; dat3];
    lab2 = [n1'; n3']; s1 = num2str('bo'); s2 = num2str('r*');
    s3 = num2str('k+'); sa = [s2; s1]; sb = [s3; s1];
    subplot(1,2,1);
    xmin1 = min(dat_a(:,1))-abs(0.1*(min(dat_a(:,1))-mean(dat_a(:,1))));
    xmax1 = max(dat_a(:,1))+abs(0.1*(max(dat_a(:,1))-mean(dat_a(:,1))));
    ymin1 = min(dat_a(:,2))-abs(0.1*(min(dat_a(:,2))-mean(dat_a(:,2))));
    ymax1 = max(dat_a(:,2))+abs(0.1*(max(dat_a(:,2))-mean(dat_a(:,2))));
```

```
    z1 = dataset(dat_a,lab1); z1 = setprior(z1,0);
    cbound_1 = ldc(z1); scatterd(z1,2,sa,[],11,'legend');
    plotc(cbound_1,'k',1); grid; xlabel('z_1'); ylabel('z_2');
    title('Feature Plot (Training)'); axis([xmin1 xmax1 ymin1 ymax1]);
    hold on; subplot(1,2,2);
    xmin2 = min(dat_b(:,1))-abs(0.1*(min(dat_b(:,1))-mean(dat_b(:,1))));
    xmax2 = max(dat_b(:,1))+abs(0.1*(max(dat_b(:,1))-mean(dat_b(:,1))));
    ymin2 = min(dat_b(:,2))-abs(0.1*(min(dat_b(:,2))-mean(dat_b(:,2))));
    ymax2 = max(dat_b(:,2))+abs(0.1*(max(dat_b(:,2))-mean(dat_b(:,2))));
    z2 = dataset(dat_b,lab2); z2 = setprior(z2,0);
    cbound_2 = ldc(z2); scatterd(z2,2,sb,[],11,'legend');
    plotc(cbound_2,'k',1); grid; xlabel('z_1'); ylabel('z_2');
    title('Feature Plot (Training)'); axis([xmin2 xmax2 ymin2 ymax2]);
    hold on;
else
    % Generate Scatter Diagram of Training Data without Bayesian Bounds
    scatter(A_final_1(1:(end/3)),A_final_2(1:(end/3)),'b'); hold on;
    scatter(A_final_1(((end/3)+1):(2*end/3)),A_final_2(((end/3)+1):...
        (2*end/3)),'r');
    scatter(A_final_1(((2*end/3)+1):end),A_final_2(((2*end/3)+1):end),'k');
    grid; xlabel('z_1'); ylabel('z_2'); title('Feature Plot (Training)');
end

if strcmp(choice3,'n') == 1 || strcmp(choice3,'N') == 1
    validatechoice = input('Validate? ("y" or "n"): ','s');
    if strcmp(validatechoice,'y') == 1 || strcmp(validatechoice,'Y') == 1
        % Perform Validation
        V_final = validate(FV,orderset,choice1,choice2,choice3,na,...
            nb,nc,pca_choice);
        V_final = V_final';
        for k4 = 1:length(V_final)/3
            n1v{k4} = num2str('Healthy'); n2v{k4} = num2str('Defect A');
            n3v{k4} = num2str('Defect B');
        end
        dat1v = [(V_final((1:end/3),1)) (V_final((1:end/3),2))];
        dat2v = [(V_final(((end/3)+1:2*end/3),1)) (V_final(((end/3)+1:...
            2*end/3),2))];
        dat3v = [(V_final(((2*end/3)+1:end),1)) (V_final(((2*...
            end/3)+1:end),2))];
        datv = [dat1v; dat2v; dat3v]; labV = [n1v'; n2v'; n3v'];
        dat_av = [dat1v; dat2v]; lab1v = [n1v'; n2v']; dat_bv = ...
            [dat1v; dat3v];
        lab2v = [n1v'; n3v'];
        s1v = num2str('mo'); s2v = num2str('k*'); s3v = num2str('r+');
        sav = [s2v; s1v]; sbv = [s3v; s1v];
        subplot(1,2,1);
        z1v = dataset(dat_av,lab1v); z1v = setprior(z1v,0);
        scatterd(z1v,2,sav,[],11,'legend');
        xmin1v = min(dat_av(:,1))-abs(0.1*(min(dat_av(:,1))-...
            mean(dat_av(:,1))));
```

```
            xmax1v = max(dat_av(:,1))+abs(0.1*(max(dat_av(:,1))-...
                mean(dat_av(:,1))));
            ymin1v = min(dat_av(:,2))-abs(0.1*(min(dat_av(:,2))-...
                mean(dat_av(:,2))));
            ymax1v = max(dat_av(:,2))+abs(0.1*(max(dat_av(:,2))-...
                mean(dat_av(:,2))));
            title('Feature Plot (Validation)'); axis([xmin1v xmax1v...
                ymin1v ymax1v]); hold off;
            healthclass1 = dat_av*cbound_1*labeld; fp1 = 0; fn1 = 0;
            for j1 = 1:40
                if strcmp(healthclass1(j1),'H') == 0 && j1 < 21
                    fp1 = fp1 + 1;
                elseif strcmp(healthclass1(j1),'D') == 0 && j1 > 20
                    fn1 = fn1 + 1;
                end
            end
            clc;
            fprintf...
                ('Case 1: False positives - %d; False negatives - %d\n',...
                fp1,fn1); subplot(1,2,2);
            z2v = dataset(dat_bv,lab2v); scatterd(z2v,2,sbv,[],11,'legend');
            xmin2v = min(dat_bv(:,1))-abs(0.1*(min(dat_bv(:,1))-...
                mean(dat_bv(:,1))));
            xmax2v = max(dat_bv(:,1))+abs(0.1*(max(dat_bv(:,1))-...
                mean(dat_bv(:,1))));
            ymin2v = min(dat_bv(:,2))-abs(0.1*(min(dat_bv(:,2))-...
                mean(dat_bv(:,2))));
            ymax2v = max(dat_bv(:,2))+abs(0.1*(max(dat_bv(:,2))-...
                mean(dat_bv(:,2))));
            title('Feature Plot (Validation)'); axis([xmin2v xmax2v ...
                ymin2v ymax2v]); hold off;
            healthclass2 = dat_bv*cbound_2*labeld; fp2 = 0; fn2 = 0;
            for j2 = 1:40
                if strcmp(healthclass2(j2),'H') == 0 && j2 < 21
                    fp2 = fp2 + 1;
                elseif strcmp(healthclass2(j2),'D') == 0 && j2 > 20
                    fn2 = fn2 + 1;
                end
            end
            fprintf('Case 2: False positives - %d; False negatives - %d\n',...
                fp2,fn2);
            break
        end
    end
end
disp('Press a key to end program'); pause
clc; disp('End of program');

% End of program
```

The following code is for the order selection process.

```
function [orderset,na,nb,nc] = orderselection(choice1,choice2,choice3)
%ORDERSELECTION loads selected data files from the set to allow choice of
%an model structure order for parameter estimation
% By Rahulram Sridhar

clc; fit = 0; count2 = 0;

% Load Data Set 1
load healthy1

% Data of Interest
t = FinalData.Time; curr = FinalData.Motor_Current; T = t(10)-t(9);
v = FinalData.Motor_Velocity; T_L = FinalData.Torque_Load;
p_ref = FinalData.Velocity_Reference; err = p_ref-v;
if choice2 == 1
    if choice1 == 1, z1 = iddata(v,[p_ref T_L],T);
    elseif choice1 == 2, z1 = iddata(v,[err T_L],T); end
elseif choice2 == 2
    if choice1 == 1, z1 = iddata(curr,[p_ref T_L],T);
    elseif choice1 == 2, z1 = iddata(curr,[err T_L],T);  end
end

% Load Data Set Validation
load healthy2

% Data of Interest
t = FinalData.Time; curr = FinalData.Motor_Current; T = t(10)-t(9);
v = FinalData.Motor_Velocity; T_L = FinalData.Torque_Load;
p_ref = FinalData.Velocity_Reference; err = p_ref-v;
if choice2 == 1
    if choice1 == 1, zv = iddata(v,[p_ref T_L],T);
    elseif choice1 == 2, zv = iddata(v,[err T_L],T); end
elseif choice2 == 2
    if choice1 == 1, zv = iddata(curr,[p_ref T_L],T);
    elseif choice1 == 2, zv = iddata(curr,[err T_L],T);  end
end

clearvars -except z1 zv fit count2 choice3

while fit < 85
    if strcmp(choice3,'y') == 1 || strcmp(choice3,'Y') == 1
        if count2 == 0
            V1 = ivstruc(z1,zv,struc(1:2,1:2,1:2,1:2,1:2));
            orderset = selstruc(V1,0);
            fprintf('\nThe selected temporary order is: '); disp(orderset);
        else
            clc; fprintf('Temporary order selection failed to provide ');
            fprintf('a good fit.\n'); fprintf('Change order till a good');
            fprintf(' fit is obtained or change data set.\n');
            orderset = input...
```

```
            ('Enter orders as a row vector in brackets [na nb ...]: ');
    end
    na = orderset(1); nb = orderset(2); nc = orderset(3);
    m1 = iv4(z1,orderset,'Focus','Simulation'); ytemp = sim(m1,z1.u);
    fit = 100*(1 - norm(ytemp - z1.y)/norm(z1.y-mean(z1.y)));
    fprintf('The fit generated with the above order is %f.\n', fit);
    fprintf('Press a key.\n'); pause;
else
    if count2 > 0
        clc; fprintf('Chosen order does not give satisfactory fit.');
        fprintf(' Redo.\n\n');
    else
        clc
    end
    % Estimate model order
    na = input('Select maximum order for ''a'' parameter (>= 1): ');
    nb = input('Select maximum order for ''b'' parameter (>= 1): ');
    nc = input('Select maximum order for ''c'' parameter (>= 1): ');
    nkb = input('Select maximum delay for input ''b'' (>= 1): ');
    nkc = input('Select maximum delay for input ''c'' (>= 1): ');
    V1 = ivstruc(z1,zv,struc(1:na,1:nb,1:nc,1:nkb,1:nkc));
    nn1 = selstruc(V1,0);
    clear V1 nkb nkc % Clear unnecessary information

    % Attempt manual model order change
    m1 = iv4(z1,nn1,'Focus','Simulation'); ytemp = sim(m1,z1.u);
    fit = 100*(1 - norm(ytemp - z1.y)/norm(z1.y-mean(z1.y)));
    fprintf('The fit percentage is %f.\n',fit);
    orderset = nn1; check = 'y'; change = 'n'; count1 = 1;
    disp('Press key to view cross correlation b/w input 2 and output');
    figure(1); pzmap(m1,'sd',3);
    figure(2); resid(m1,z1,'Corr');
    fprintf('Current order vector:\n'); disp(orderset);
    while strcmp(check,'y') == 1 || strcmp(check,'Y') == 1
        if strcmp(change,'y') == 1 || strcmp(change,'Y') == 1
            new_nn = input('Enter orders into new row vector: ');
            newmodel = iv4(z1,new_nn,'Focus','Simulation');
            close all
            ytemp = sim(newmodel,z1.u);
            fit = 100*(1 - norm(ytemp - z1.y)/norm(z1.y-mean(z1.y)));
            fprintf('The fit percentage is %f.\n',fit);
            disp('Press for cross correlation b/w input 2 and output');
            figure(3); pzmap(newmodel,'sd',3);
            figure(4); resid(newmodel,z1,'Corr');
            count1 = count1 + 1;
        end
        check = input('Change order vector? (y/n): ','s');
        change = check;
    end
    close all
```

```
        if count1 > 1
            revert = input('Revert to original order? (y/n): ','s');
            if strcmp(revert,'n') == 1 || strcmp(revert,'N') == 1
                orderset = new_nn;
            end
        end
    end
    count2 = count2 + 1;
end
clc; fprintf('All criteria for order selection are satisfied.');
fprintf(' Press a key.\n');
pause
% Save order set
na = orderset(1); nb = orderset(2); nc = orderset(3);
clearvars -except orderset na nb nc
end
```

B.6 Experiments

Refer to the MATLAB programs presented earlier under 'Simulations'. The programs require some minor modifications but are essentially the same.