DETECTION OF BROKEN ROTOR BARS IN INDUCTION MOTORS USING PARAMETER AND STATE ESTIMATION

by

Kyong Rae Cho

BACHELOR OF SCIENCE IN ELECTRICAL ENGINEERING
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
(1987)

Submitted to the Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degree of

MASTERS OF SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June, 1989

Signature of Author _____
Department of Electrical Engineering and Computer Science
June 12, 1989

Certified by ___
Jeffrey H. Lang, Thesis Supervisor
Associate Professor of Electrical Engineer

Certified by ___
Stephen D. Umans, Thesis Supervisor
Principal Research Engineer

Accepted by _____
Arthur C. Smith, Chairman
Departmental Committee on Graduate Students

Detection of Broken Rotor Bars in Induction Motors Using Parameter and State Estimation

by

Kyong Rae Cho

Submitted to the Department of Electrical Engineering on June 12, 1989 in partial fulfillment of the requirements for the Degree of Masters of Science in Electrical Engineering.

Abstract

This thesis studies the detection of broken rotor bars in induction motors using parameter and state estimation techniques. The hypothesis upon which detection is based is that the apparent rotor resistance R_r of an induction motor increases as a result of a rotor-bar breakage. Here, the apparent R_r is that found in the standard single-phase equivalent circuit of an induction motor operating at a constant velocity. The broken-rotor-bar detector is novel in that the ideas and tools of parameter estimation theory are combined with the appropriate electrical and thermal models for the induction motor to produce an estimate of R_r which is in turn compared to its nominal value to detect broken rotor bars.

To detect broken rotor bars, measurements of stator voltage, stator current, and the rotor velocity are taken over a small range of slip. These measurements are processed by three different linear-least-square-error (LLSE) estimations schemes. These include LLSE estimation with estimated stator resistance R_s, iterative LLSE estimation with estimated R_s, and LLSE estimation with measured R_s. The sensitivity of these estimators to the noise in the measurements on which they operate is also analyzed. From the sensitivity analysis, the numerical results predict good estimation of R_r on one hand and poor estimation of R_s on the other. This is confirmed by experimental results of an induction motor operating at a particular steady-state thermal operation. Furthermore, these experiments demonstrate that the LLSE estimation with measured R_s case detects one broken bar out of 45 most clearly among the three types of LLSE estimations.

At the load varies, the temperature profile of the induction motor varies. This can cause a significant variation in R_r, one which could mask the variation of R_r due to a broken rotor bar. To accommodate this problem, a thermal model is developed and combined with a temperature-resistance relation of metallic conductors so that the estimates of R_r at different motor loads are standardized to estimates at a reference load, thereby providing temperature-compensation. The thermal model depends on a good estimate of R_s, which the estimators listed above could not provide. Therefore, the one estimator with measured R_s is the only candidate for thermal compensation. Finally, from the physical experiments at different motor loads, a successful temperature-compensated detection of broken rotor bars via LLSE estimation of R_r with measured R_s is clearly demonstrated.

The experimental results show that the estimates of R_s based on the thermal model at various loads are nearly identical to the measured R_s. Thus, this thesis also provides a highly accurate, yet easily implemented method of estimating R_s. This method replaces the less desirable direct measurement of R_s, and the detection of broken rotor bars using the thermally estimated R_s is equivalent to that with measured R_s.

Thesis Supervisor: Jeffrey H. Lang
Title: Associate Professor of Electrical Engineering
Thesis Supervisor: Stephen D. Umans
Title: Principal Research Engineer

Dedication

To my family: Thank you for always being there for me. I dedicate this thesis to you.

Acknowledgements

Foremost, I would like to thank my advisors, Prof. Jeffrey Lang and Dr. Stephen Umans. Without them I would not be writing this acknowledgement now. Prof. Lang was always understanding and always available. I sincerely doubt I will ever find a better person to work for and work with than Prof. Lang. I thank him from the bottom of my heart for his supervision and for his support and insight. It has been a great experience. I would also like to thank Dr. Umans. I thank him for his guidance and his infectious enthusiasm for the research done in this thesis.

I would like to mention several other people who come to mind as I am feverishly writing the last few sentences of this thesis. I would like to thank Bahman, not only for helping me out with xeroxing - that could not have been too much fun - but also for all the outrageous discussions we had over the two years. I will miss him. I like to thank Chris and Hoa for their support not only for letting me borrow their hard-disk, "Yes, Hoa, I'll return it soon.", but also for encouraging me and even worrying about me on occasions. "Are you done yet?" Also I like to thank everyone in the Bible Study particularly Kyung Tai who is presently fighting like mad to stay awake with me as I attempt the last minute finish. I know that they have been praying for me and believe me, at times, I definitely needed divine help. In any case, my stay at MIT for the last six years - has it been that long? - has been memorable and meaningful because of special friendships that developed over the years. These friends all deserve my acknowledgement since they contributed to my spiritual and emotional health.

Finally, I would like acknowledge the fact that this thesis was partially supported by the contract N00024-87-C-4263 from the United States Navy.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem Statement

Unexpected motor failures, in particular those of large motors can result in costly repairs or replacement, degradation in performance, reduction or loss of safety, and extended downtime. This in turn can lead to large capital losses, serious inconvenience, and perhaps other catastrophic failures. Failure monitoring can be used to predict and detect such motor failures. In this way, failure monitoring increases both the life expectancy and the reliability of electrical motors, which in turn decreases their overall operating cost. Another benefit of failure monitoring is to ensure that the motors have been manufactured properly. This thesis concentrates on induction motors. Specifically, the rotor bar breakage is the targeted failure for detection. The method of detection as will be discussed shortly is based on parameter estimation techniques.

According to recent surveys [1, 2, 3] on the reliability of electric motors, bearing-related failures constitute 41% of all induction motor failure, with stator-related and rotor-related making up the next 37% and 10%, respectively. The remaining failures (12%) are

scattered among a variety of effects. Of the rotor-related failures (10%), there are three main types of failures. These are cage failures (5%), shaft failures (2%), and core failures (1%). The other 2% is categorized under others [1]. The cage faults which constitute half of the rotor related failures, come in the form of broken rotor bars or end rings. The main sources of these cage faults are design errors, manufacturing defects, misoperation or misapplication of the motor, improper maintenance, and aging or fatigue factors. Broken rotor bars increase the vibration of the motor frame, increase the localized rotor temperature, and can cause further breakage in other bars, all contributing to eventual motor failure.

1.2 Comparison of Approaches to Broken Rotor Bar Detection

Two different approaches are prevalent in assessing the condition of operating electric motors [4]. The first method involves the technique of signature trend analysis. Signature trend analysis relies on periodic collection of sensor measurements. These measurements are then processed to identify characteristic "signatures" associated with different types of failures. For instance, vibration data can be collected and processed through a Fourier transform. Each collected data set is processed and compared to previously processed data sets, and some established baseline is used to determine signatures trends. Based on experience, certain trends are correlated with impending or existing failures. In such a way, failures can be predicted or detected. Although this method is widely practiced, the necessity for a historical data and the experience required to associate signature trends to specific types of failures presents serious disadvantages. These methods are not necessarily based on any physical analysis of the failures, and hence do not take advantage of the information inherent in this analysis.

The other method is generally more physically based. It uses mathematical models of the detailed physics of motors to identify fundamental causes of failures and predict their effects. This method offers two distinct advantages over the signature trend analysis. Mathematical models based on the physics of motor operation is absent in the signature trend analysis. Thus, we expect the mathematical models to produce more sensitive and accurate picture of the "internal" details of motors. Also, there is less of a need to depend on empirical intuition or derivation of signature trends for different types of faults.

The broken rotor bar detector developed in this thesis is based on physical models, but it is based also on parameter estimation. In general, a variety of sensors could be used to collect measurements from an electric motor for the purposes of failure monitoring. These sensors might measure stator voltages and currents; air-gap and external magnetic flux densities; rotor position, velocity and torque; internal and external temperatures; and case vibrations, for example. Also in general, a failure monitoring system could monitor a variety of motor failures. These failures might include conductor shorts and opens, demagnetized magnets, bearing failures, and cooling failures. It is apparent then that a failure monitoring system should be capable of extracting in a consistent manner the evidence of many possible failures from measurements from many physically different sensors. The failure monitoring system studied here does so by combining physical models of the motor which include failure mechanisms and symptoms with the estimation and evaluation of the states and parameters within the models. It is through the models that the various sensor measurements are self-consistently analyzed in the process of monitoring failures. Variations of the states and parameters from their norms are used to monitor failures and suggest maintenance. This method of failure monitoring is similar in spirit to that discussed in [5]. Thus, the ultimate goal of this thesis is to illustrate the

usefulness and applicability of this failure monitoring system for electric motors by using induction motor as a specific example.

The research effort of this thesis is a part of the ongoing development of a failure system analysis for electric motors by M.I.T. Laboratory for Electromagnetic and Electronic Systems [6,7]. The concept of failure analysis based on parameter estimation is shown pictorially in Figure 1.1. Note that the foundation of the parameter estimation algorithm rests on the the physical models of the electric motors. The parameter estimation algorithm takes as its input the sensor data and motor models, and produces as its output estimates of the parameters in the models. These values are then evaluated as a whole to determine whether any failure has occurred, and if so, the extent and the type of failure that has occurred. The power of this general scheme lies in the fact that it can be adopted and tailored for any type of electric motors, including an induction motor.

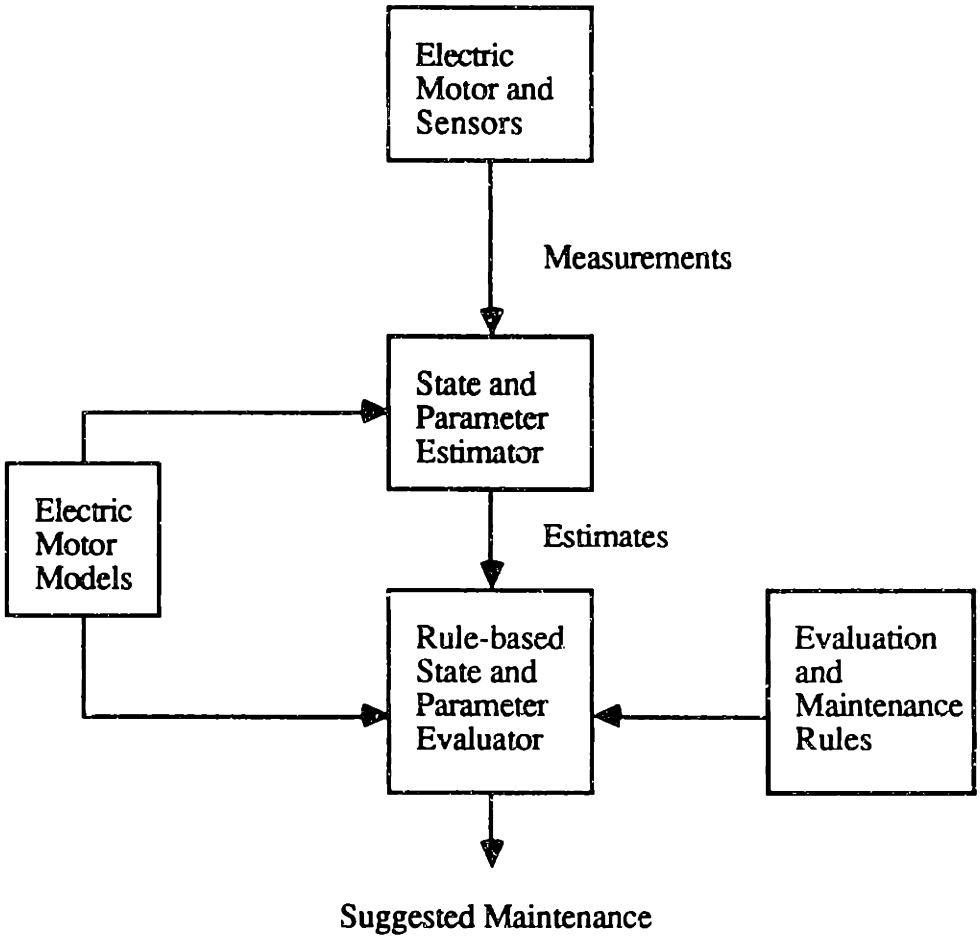


Figure 1.1: The failure analysis system based on parameter estimation.

1.3 Literature Review on Broken-Rotor-Bar Detection and Parameter Estimation

Broken-rotor-bar detection is an old problem, and its study has a substantial history. For many years, visual inspection and bench test methods such as "growlers" and related probe techniques have been applied to disassembled motors [8]. Also, "single phase" tests have been applied to assembled but non-operating motors [8]. A number of recent studies have used the physical-model approach to develop theories of the response motor operation due to broken rotor bars or broken end rings [8,9,10,11,12]. Typical detectors examine the

axial flux [8,9] or rotor velocity fluctuations [10] which result from unbalanced currents caused by a broken rotor bar. Even though they are based on physical models, these studies are limited in that they are specific to rotor cage faults. They cannot be generalized to other types of faults, such as broken fan blade or bearing-related faults.

Unlike the above studies on detection of broken rotor bars, the ability to generalize detection for variety of faults can be accomplished through parameter estimation. As aforementioned, estimation techniques determine the states and the parameter values of physical models of motors. By understanding how these parameters and states change from their norm values as a result of different types of faults, we have the flexibility to use number of estimates to monitor variety of faults.

Numerous studies on parameter estimation of induction motor also exist. Among those with a character similar to this thesis are [13,14,15]. All three papers batch process stator electrical measurements and rotor mechanical measurements. Further, all three papers fit the measurements to models by minimizing error functions over the parameters using numerical gradient methods. Reference [13] additionally augments the single-phase equivalent circuit, the electrical model used in this thesis, to include core losses and leakage, and presents a method for compensating thermal variations in the measurements. Reference [14] discusses thermal issues pertaining to measurement taking. Reference [15] introduces a pseudo-random load torque to guarantee the sufficient richness of the measurements rather than pre-select specific operating points for measurement taking as this thesis and the other two do.

Historically, the development of high performance control algorithms, such as field oriented control [16,17,18] have made induction motors a competitive alternative to dc motors for many applications. The success of these algorithms depend on accurate

knowledge of the motor parameters at all times. Since the parameters can vary significantly during motor operation, the estimates of the parameters must be updated whenever significant variations in parameters are detected. Of the estimated parameters, rotor time constants and rotor resistances are particularly important to the field oriented control [19]. Thus, numerous on-line or non-intrusive static estimators presented in [20,21,22,23] are primarily concerned with estimations of rotor resistances and rotor time constants, which are obtained from the measurements of the stator voltage and current, and the rotor speed. In addition variety of parameter tuning algorithms, based mainly on the theory of Model Reference Adaptive Systems [24,25] are presented in [26,27]. While all these methods produce the estimates of parameters, they make no attempt to explore detection or prediction of motor failures.

1.4 Contributions of Thesis

The main contribution of this thesis lies in the way the ideas and tools of parameter estimation theory are combined with the appropriate physical models for the induction motor in order to detect a failure. The integration of physical models and estimators offers important improvement over previous methods in that the models describe the physics of the response of the motor in presence of a failure, while the estimator allows flexibility to include other types of failures as well. This thesis deals with the estimation of rotor resistance in particular, since the hypothesis is that a broken rotor bar increases the apparent rotor resistance. The apparent rotor resistance is precisely that of the single-phase equivalent circuit of an induction motor. The estimate of this rotor resistance is compared to its nominal value to detect broken rotor bars. In this thesis, one broken rotor bar out of 45 has been clearly detected with estimates at different motor operating conditions as well as at a constant motor operation.

Another significant contribution lies in the development of a thermal model. The fact that an increase in temperature also increases rotor resistance causes a problem in detecting broken rotor bars, since significant temperature variations can occur over normal operations of the motor. Hence, a thermal model is needed to temperature-compensate the estimates of rotor resistances at different operations of the motor to estimates at some reference rotor temperature. In this manner, meaningful comparisons of standardized estimates to the nominal value can be made to determine broken rotor bars. While [13] deals with temperature-compensated measurements it does not deal with temperature-compensated estimates. In fact, none of the previous work developed thermal models to temperature-compensate parameter or state estimates for the purpose of detecting failures.

The development of a thermal model is important in two other ways. One, the thermal model can be used to estimate any temperatures in or on the induction motor. This then can be used to detect broken fan blade or cooling system obstructions, although this is not explored explicitly in this thesis. Thus, the methods developed in this thesis can be applied to detect at least one other failure. Further, the thermal model is used to develop an accurate, non-intrusive method of estimating the stator resistance R_s. The thermal estimation of R_s offers an improvement over existing estimation methods and even non-intrusive measurement of R_s (See Chapter 5).

1.5 Outline of Thesis

This thesis is organized as follows. Chapter 2 introduces the single-phase equivalent circuit of the induction motor and its system transfer function from the stator voltage to the stator current. The rotor resistance R_r is defined in the context of this model. The system

used for the physical experiments are also presented in Chapter 2. Chapter 3 discusses linear-least-square-error (LLSE) estimation schemes used to identify the parameters, including R_r. Included in the chapter is a numerical study of the sensitivity of these estimation schemes to noise in the measurements on which they operate. Chapter 4 presents and discusses constant-temperature experimental results of the parameter estimation schemes discussed in Chapter 3. The results conclusively detect a single broken rotor out of 45 rotor bars for a particular motor loading and its corresponding motor temperature profile. Chapter 5 develops the temperature compensation technique. To support this technique a thermal model for the induction motor is developed. The chapter also presents an alternate, highly accurate estimate of the stator resistance based on the thermal model. In addition, Chapter 5 gives successful experimental results of the temperature-compensated detection of a broken rotor bar. Finally, Chapter 6 concludes by summarizing the thesis and offering future research suggestions.

Chapter 2

The Experimental System and Induction Motor Model

The experimental system for all the physical experiments conducted in this thesis is presented in Section 2.1. This section includes a description of the power supply, the test induction motor, and the measuring equipments. In Section 2.2, the single-phase equivalent circuit of the induction motor and its stator terminal admittance is given in a general form. The single-phase equivalent circuit is important in that for this thesis, the rotor resistance is defined in the context of this model and its behavior. The restrictions and validity of this model are also discussed. Next, experimental information from Section 2.1 is combined with the single-phase equivalent circuit of Section 2.2. In doing so, we can identify which parameters of the single-phase equivalent circuit are known through measurements and which are not. The unknown parameters, including the rotor resistance, must be estimated. Section 2.2 concludes by transforming the stator terminal admittance equation for the single-phase equivalent circuit of an induction motor to a general form that is used by the estimators described in Chapter 3.

2.1 Experimental System

The experimental system for this thesis is shown schematically in Figure 2.1. A nominally-120-V, 60-Hz power supply is connected to a 3-hp, 4-pole squirrel-cage induction motor which in turn is connected to a dynamometer. The nameplate information for the induction motor is given in Table 2.1. All experiments are conducted with one stator, and three identically manufactured rotors. A summary of the rotors is given in Table 2.2. Each rotor has 45 rotor bars. In rotor 2, one bar has been broken deliberately by milling into the rotor surface and through the bar at both of its ends. A Magtrol HD800-8 dynamometer is used to load the motor so that it operates at constant speed. The control section of the HD800-8 also provides digital readout of the measurements given in Table 2.3. Included in Table 2.3 are the errors of each measurement. For the measurements of stator current and the stator voltage, the measurement errors depend on the settings of their range. The measurement errors in Table 2.3 are obtained with the range for the measurement of the stator voltage set at 150V and the range for the measurement of the stator current set at 10A. The stator of the induction motor is fitted with 25 thermocouples. Figure 2.2 and Table 2.4 locate these thermocouples. The temperature readings from the thermocouples are used to verify the steady-state constant-temperature motor operation assumed in Chapters 3 and 4, and to verify the temperature compensation method developed in Chapter 5. The thermocouple readings are processed by a computer and displayed on its screen .

The stator resistance of the motor can be measured. This resistance is measured with the motor temporarily shut down. Since the resistance of the stator is small, on the order of 0.8 ohms, a typical ohmmeter cannot measure this resistance with sufficient accuracy. Therefore, a power supply is connected across two stator lines and measurements from a voltmeter and ammeter are used to calculate the stator resistance. The procedure for

measuring R_s under a steady-state constant-temperature motor operation, is discussed more fully in Chapter 4. A discussion of the motivation behind measuring R_s in this manner, as well as the merits of making such a measurement, is found in Section 3.4.

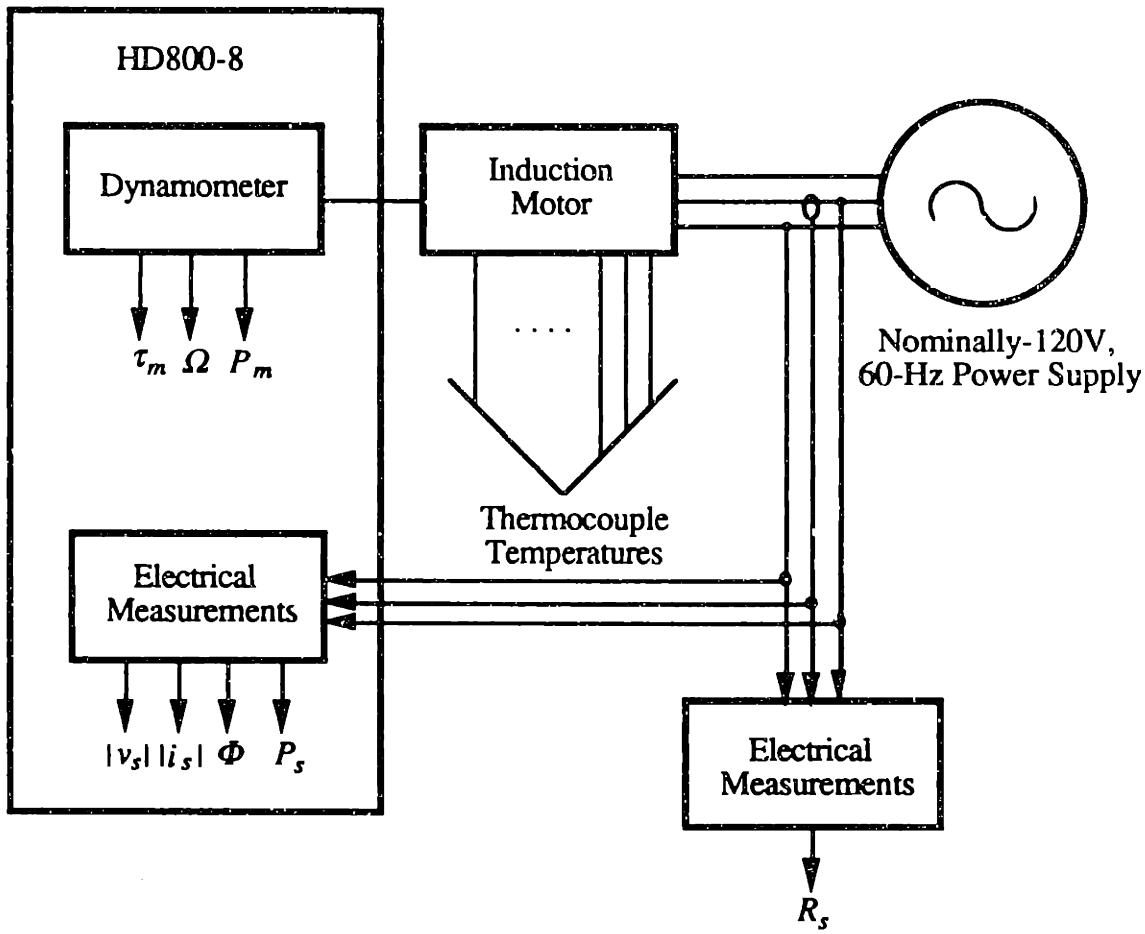


Figure 2.1: Experimental system, see Table 2.3 for the descriptions of HD800-8 measurements.

Westinghouse Life-Long T AC Induction Motor			
3-Phase	230/460 V	3 hp	1730 rpm
60 Hz	2.4/4.7 A	182T Frame	1.0 s.f.
Motor Style: 773B646G41		Nom. Eff. 81.5%	
Catalog No.: 05-3H4SBFC-SKB			

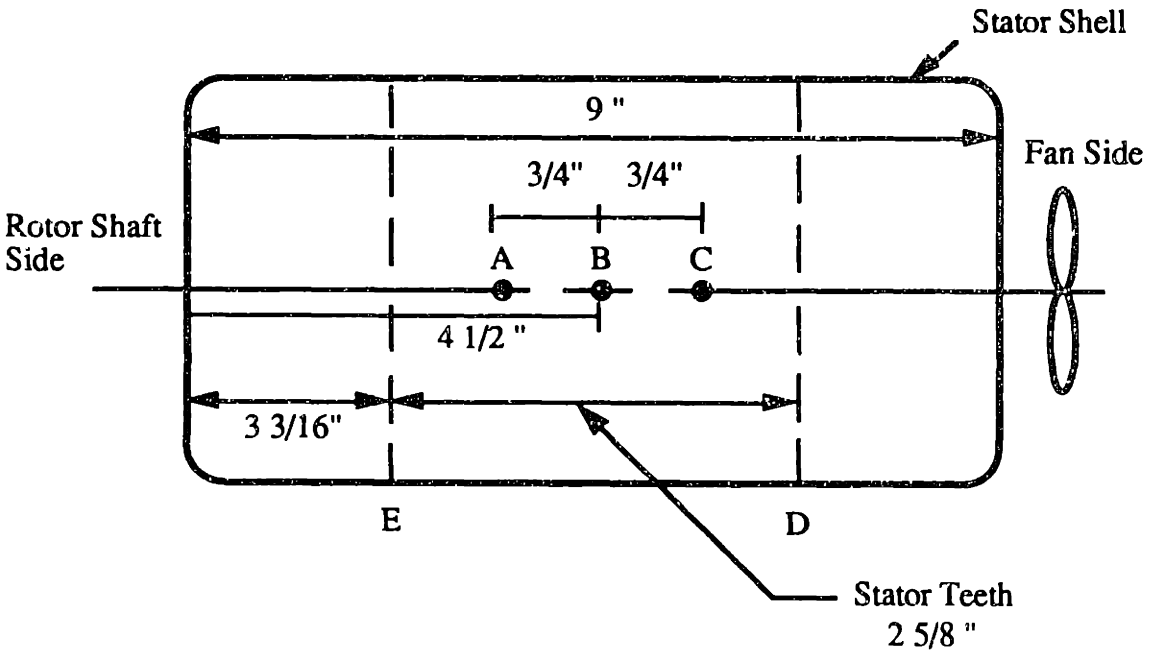
Table 2.1: Nameplate information for the induction motor.

Rotors With 45 Bars Each	
Rotor 1	No Broken Bars
Rotor 2	One Broken Bar With Both Ends Open
Rotor 3	No Broken Bars

Table 2.2: Rotor summary.

HD800-8 Measurements		
Electrical Measurements		
$\lvert v_s \rvert$	Stator Phase Voltage Magnitude	(± 0.3 V rms)
$\lvert i_s \rvert$	Stator Phase Current Magnitude	(± 0.04 A rms)
Φ	Power Factor	(± 0.005)
$P_s = \lvert v_s \rvert \lvert i_s \rvert \Phi$	Stator Phase Power	(± 15 W rms)
Mechanical Measurements		
τ_m	Rotor Shaft Torque	(± 0.01 N-m)
$\Omega = \frac{60\omega_m}{2\pi}$	Rotor Speed	(± 1 rpm)
$P_m = \tau_m \omega_m$	Mechanical Power	(± 0.5 W)

Table 2.3: HD800-8 measurements.



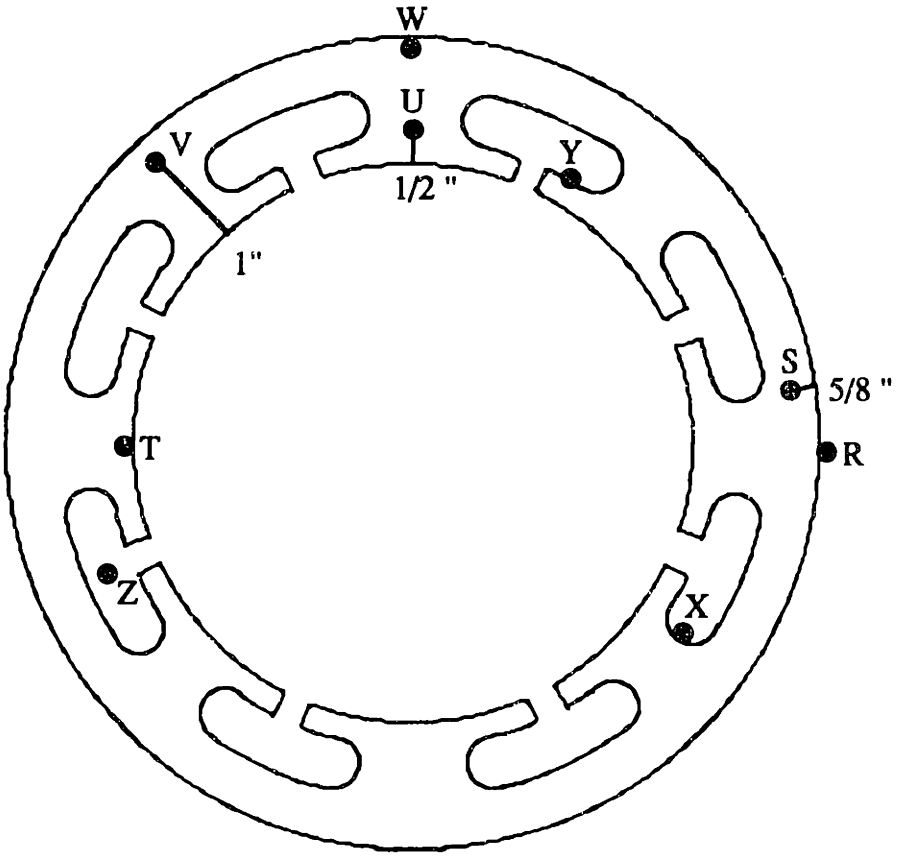


Figure 2.2: Thermocouple locations; see Table 2.4.

Thermocouple	Location	Thermocouple	Location
TC_1	AR*	TC_{14}	EW
TC_2	BR*	TC_{15}	AX
TC_3	CR*	TC_{16}	BX
TC_4	AS	TC_{17}	CX
TC_5	BS	TC_{18}	AY
TC_6	CS	TC_{19}	BY
TC_7	DT	TC_{20}	CY
TC_8	DU	TC_{21}	AZ†
TC_9	DV	TC_{22}	BZ†
TC_{10}	DW	TC_{23}	CZ†
TC_{11}	ET	TC_{24}	End Turn (Fan Side)
TC_{12}	EU	TC_{25}	End Turn (Rotor Shaft Side)
TC_{13}	EV		

* TC_1, TC_2, TC_3 are in holes drilled into the stator shell

† TC_{21}, TC_{22}, TC_{23} are in the middle of stator winding

Table 2.4: Thermocouple locations; see Figure 2.2.

2.2 Single-Phase Equivalent Circuit of an Induction Motor

Induction motors as a class of electrical machines have been modeled extensively. The resulting models can be found in numerous texts [28,29,30]. In this thesis, the single-phase equivalent circuit for an induction motor, shown schematically in Figure 2.3, serves as the definitive model of the induction motor.

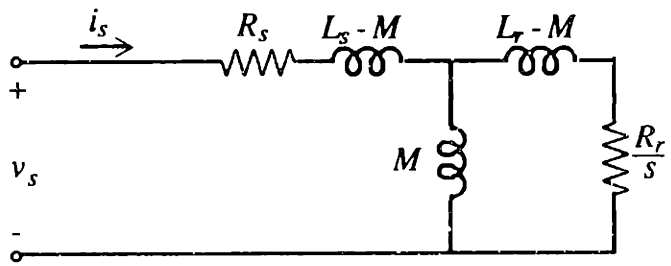


Figure 2.3: Single-phase equivalent circuit of an induction motor.

In the model, v_s is the line-to-neutral stator terminal voltage, i_s is the stator terminal current, R_s is the stator resistance, R_r is the rotor resistance, L_s is the stator inductance, L_r is the rotor inductance, M is the mutual inductance between the stator and the rotor, and s is the slip. The slip is in turn defined by

$$s = (\omega_e - P\omega_m)/\omega_e \tag{2.1}$$

where ω_e is the frequency of electrical stator excitation, ω_m is the mechanical rotor speed, and P is the number of magnetic pole pairs in the induction motor. The importance of this model for broken-rotor-bar detection lies in the rotor resistance, R_r. It is hypothesized that this resistance will increase when a rotor bar breaks.

In general, an induction motor is modeled with different self inductances for the stator and rotor, as shown in Figure 2.1. However, this thesis is concerned with parameter estimation based on electrical measurements of the stator only. With these measurements, it is not possible to uniquely determine the three inductors in Figure 2.1. Therefore, in order to avoid an ill-posed estimation problem, the model in Figure 2.1 must be reduced. The exact form of this reduction, however, is a matter of convenience. Many reductions lead to the same stator terminal behavior in the model. Without loss of generality, then, it is assumed here that

$$L_s = L_r \equiv L \tag{2.2}$$

where L is a self-inductance of both the stator and rotor. Other assumptions such as $L_s = M$ or $L_r = M$ could also be used.

The transfer function of the single-phase equivalent circuit from the stator voltage to the stator current is given by

$$\frac{i_s}{v_s} = \frac{\sqrt{2}|i_s|\,e^{j\theta}}{\sqrt{2}|v_s|} = \frac{1 + j\omega_e s\tau_r}{(R_s + j\omega_e L) + \omega_e s\left(jR_s\tau_r - \omega_e\frac{K}{R_r}\right)} \tag{2.3}$$

where $\tau_r = L/R_r$, $K = L^2 - M^2$, and θ is the phase angle between the stator voltage and current [28,29]. Note that τ_r is the rotor time constant and that $\cos\theta = \Phi$. In (2.3), $|v_s|$ and $|i_s|$ denote the rms magnitudes of the stator voltage and stator current.. Equivalently, (2.3) is the admittance of the induction motor at the stator terminals.

The corresponding mechanical power equation for a sinusoidally-excitated N-phase induction motor is

$$P_m = \tau_m \omega_m = N \frac{(1 - s)}{s} |i_r|^2 R_r \tag{2.4}$$

where τ_m is the mechanical torque and $|i_r|$ is the rms magnitude of the rotor current [28,29]. Thus, the mechanical torque τ_m is

$$\tau_m = N \frac{(1 - s)}{\omega_m s} |i_r|^2 R_r = N \frac{|i_r|^2 R_r}{\omega_e s} \ . \tag{2.5}$$

The single-phase equivalent circuit of an induction motor is strictly valid for balanced excitation and constant motor velocity. Note that the rotor resistance in the model is assumed to be constant. Hence, the model ignores the effect of the magnetic diffusion in the radial direction into the rotor bars. This assumption, however, is only valid over a small range of slip. In reality, the apparent resistance of the rotor bars is a function of slip; as the slip changes, the resulting change in skin depth in the radial direction changes the apparent resistance of the bars [13,28]. Consequently, over a large range of slip, the resistance of the rotor bars is not constant. However, if the range of slip is kept small, this model is valid. For the purpose of this thesis and all the experiments conducted in it, the range of slip is sufficiently small (0.0 to 0.04) to justify the assumption of constant rotor resistance.

In addition, the single-phase equivalent circuit ignores the effect of core loss. The core loss, if considered, can often be modeled as a resistor in parallel with the mutual inductor

[13]. Core loss is important for the thermal model in Chapter 5. However, since the core loss resistance is typically large relative to other electrical parameters in the model, for the purpose of parameter identification in Chapter 3, the core loss resistance is not included.

At this point, it is useful to substitute data from Section 2.1 concerning the experimental system into (2.2). By differentiating the knowns from the unknowns of (2.2), we can properly gather a number of independent equations based on (2.2) into a form to which the estimators of Chapter 3 can be applied. From Section 2.1, the frequency of the stator excitation, ω_e, is known to be 60 Hz. Furthermore, since the test motor is a 4-pole induction motor, (2.1) becomes

$$s = (\omega_e - 2\omega_m)/\omega_e = (1800 \text{ rpm} - \Omega)/1800 \text{ rpm} \ . \tag{2.6}$$

Hence, s is determined by Ω, which is measured by the dynamometer. Finally, since the test motor is a 3-phase motor, (2.4) and (2.5) become

$$P_m = \tau_m \omega_m = 3\frac{(1-s)}{s}|i_r|^2 R_r \tag{2.7}$$

and

$$\tau_m = 3\frac{(1-s)}{\omega_m s}|i_r|^2 R_r = 3\frac{|i_r|^2 R_r}{\omega_e s} \ , \tag{2.8}$$

respectively. In addition to ω_e and s are the measurements $|v_s|$, $|i_s|$, and $\theta = \cos^{-1}(\Phi)$, which are obtained from the dynamometer.

Given the above measurements, the remaining unknown parameters of Equation (2.3) are R_s, R_r, L, and M. Admittedly, R_s can be treated as a known parameter, since Section

2.1 outlines an experimental procedure for measuring R_s. However, to make the analysis most general at this point, R_s is treated as an unknown parameter. The measurement of R_s is important to one of the parameter estimators examined in Chapter 3.

As stated before in this section, the hypothesis upon which broken-rotor-bar detection is based is that R_r will increase when a rotor bar breaks. To monitor the evolution of this failure, R_r is estimated from the measurements ω_e, s, $|v_s|$, $|i_s|$, and Φ. This estimation is based on (2.3) which in turn is based on the electrical model in Figure 2.3 and the definition (2.2). Note that (2.3) is actually two independent equations, one based on the real part of (2.3) and the other on the imaginary part of (2.3). The information that (2.3) provides with one set of measurements is therefore insufficient to uniquely determine the four unknown parameters R_s, R_r, L, and M; we have 2 independent equations to solve for 4 independent unknowns. Additional information is required. Since the measurements ω_e, s, $|v_s|$, $|i_s|$, and Φ are the only ones available, the additional information must come from additional sets of measurements taken with the induction motor at different operating points. An operating point is defined by v_s, ω_e, and the mechanical load on the motor. As presented in Section 2.1, the experimental setup for this thesis is such that the operating point varies as a result of a varying load; $|v_s|$ and ω_e remain fixed, but as the load varies, $|i_s|$, s, and Φ will vary. As a minimum, measurements must be taken at two different operating points. This gives four independent equations for four independent unknowns. However, by using sets of measurements from more than two operating points, immunity to measurement noise is obtained; the parameters become over-specified in this case. A parameter estimator which uses many sets of measurements is therefore desirable. The degree of overspecification is a measure of how well the operating points have been chosen. Equivalently, it is a measure of the richness of the measurements. To determine the degree of overspecification or the richness of the measurements, the linear algebraic concepts of singular value decomposition (SVD) and condition number are introduced.

The relevant discussion of SVD and condition number with respect to the sufficient richness of the measurements is deferred until Section 3.1.

As mentioned, Equation (2.3) is actually two independent equations. The real part of (2.3) yields

$$|i_s|\Phi R_s + \omega_d |i_s|\sqrt{1-\Phi^2}L - \omega_e^2 s |i_s|\Phi\frac{K}{R_r} + \omega_e s |i_s|\sqrt{1-\Phi^2}R_s\tau_r = |v_s| \tag{2.9}$$

and the imaginary part of (2.3) yields

$$-\omega_e s |v_s|\tau_r - R_s |i_s|\sqrt{1-\Phi^2} + \omega_d |i_s|\Phi L + \omega_e^2 s |i_s|\sqrt{1-\Phi^2}\frac{K}{R_r} + \omega_e s |i_s|\Phi R_s\tau_r = 0. \tag{2.10}$$

In general, m sets of independent measurements, presumably obtained at m different mechanical loads or slips, produce $2m$ independent equations, m from (2.9) and m from (2.10). The resulting collection of these equations can be written as

$$\begin{bmatrix} 0 & |i_{s1}|\Phi_1 & \omega_d|i_{s1}|\sqrt{1-\Phi_1^2} & -\omega_e^2 s_1|i_{s1}|\Phi_1 & \omega_e s_1|i_{s1}|\sqrt{1-\Phi_1^2} \\ 0 & |i_{s2}|\Phi_2 & \omega_d|i_{s2}|\sqrt{1-\Phi_2^2} & -\omega_e^2 s_2|i_{s2}|\Phi_2 & \omega_e s_2|i_{s2}|\sqrt{1-\Phi_2^2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & |i_{sm}|\Phi_m & \omega_d|i_{sm}|\sqrt{1-\Phi_m^2} & -\omega_e^2 s_m|i_{sm}|\Phi_m & \omega_e s_m|i_{sm}|\sqrt{1-\Phi_m^2} \\ -\omega_e s_1|v_{s1}| & -|i_{s1}|\sqrt{1-\Phi_1^2} & \omega_d|i_{s1}|\Phi_1 & \omega_e^2 s_1|i_{s1}|\sqrt{1-\Phi_1^2} & \omega_e s_1|i_{s1}|\Phi_1 \\ -\omega_e s_2|v_{s2}| & -|i_{s2}|\sqrt{1-\Phi_2^2} & \omega_d|i_{s2}|\Phi_2 & \omega_e^2 s_2|i_{s2}|\sqrt{1-\Phi_2^2} & \omega_e s_2|i_{s2}|\Phi_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ -\omega_e s_m|v_{sm}| & -|i_{sm}|\sqrt{1-\Phi_m^2} & \omega_d|i_{sm}|\Phi_m & \omega_e^2 s_m|i_{sm}|\sqrt{1-\Phi_m^2} & \omega_e s_m|i_{sm}|\Phi_m \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} |v_{s1}| \\ |v_{s2}| \\ \vdots \\ |v_{sm}| \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{2.11}$$

where the numerical subscripts index the measurement sets, and the unknown parameters are defined according to

$$x_1 = \tau_r \quad x_2 = R_s \quad x_3 = L \quad x_4 = \frac{K}{R_r} \quad x_5 = R_s\tau_r \tag{2.12}$$

where $\tau_r = L/R_r$, and $K = L^2 - M^2$. Equation (2.11) takes the general form

$$\mathbf{A}\bar{x} = \bar{y} \tag{2.13}$$

where A is a matrix based on the measurements, $\bar{x}$ is a vector based on the unknown parameters, and $\bar{y}$ is a vector based on the measurements. At this point, the precise definition of independent measurements used throughout this thesis can be discussed in terms of (2.11) and (2.13). By m independent measurements, we mean m measurements used to produce $2m$ linearly independent rows of A in (2.11). The estimation problem is fully defined by (2.11) from which given A and $\bar{y}$, $\bar{x}$ can be estimated. Various estimators are described in Chapter 3 and experimental evaluation of these estimators are found in Chapter 4. Combinations of measurements and physical models for other types of motors can also be reduced to the general form of (2.13). Therefore, the estimation analysis for the induction motor could just be similarly applied to other types of motor. In this sense, the estimators developed in this thesis need not be motor-specific.

Finally, it should be emphasized that, for the purpose of estimation, (2.11) is assumed to contain data that is collected at a constant temperature. Since (2.11) contains data obtained from different operating points, it is therefore assumed that the motor is first run at one operating point and allowed to reach steady-state temperature distribution. Following this, it is assumed that the data from the other operating points are obtained very quickly so as not to affect this temperature distribution. The selection of the operating point which determines the temperature distribution is arbitrary.

Chapter 3

Constant-Temperature Estimation Theory

3.1 Introduction

In this chapter, we examine the three different parameter estimators used for estimating the rotor resistance, R_r and the other parameters. These estimators assume a steady-state constant-temperature operating condition of the induction motor. As stressed before, the rotor resistance increases both as the temperature rises and as rotor bars break. Thus, to simplify analyses and gain insight, temperature issues are deferred until Chapter 5 by assuming steady-state constant-temperature operation of the motor. It should be clear that if we cannot develop an estimator which detects broken rotor bars at motor constant-temperature, we certainly cannot develop an estimator which detects broken rotor bars at arbitrary temperatures.

In Chapter 2, given the availability of m independent sets of measurements from an induction motor consisting of ω_e, s, $|v_s|$, $|i_s|$, and Φ, Equation (2.11) is derived from the single-phase equivalent circuit of the induction motor. The general form of (2.11) is given

by (2.13), in which A is a matrix based on the measurements, $\bar{x}$ is a vector based on the unknown parameters, and $\bar{y}$ is a vector based on the measurements. Given A and $\bar{y}$, $\bar{x}$ can be estimated. This is nearly a linear-least-square-error (LLSE) problem [31]; it is not because there is uncertainty in both A and $\bar{y}$ as opposed to in $\bar{y}$ alone. Also, the elements of the vector $\bar{x}$ are not independent. Nonetheless, we might treat it as a LLSE estimation problem with the expectation that the resulting estimator will be an adequate estimator of $\bar{x}$. In this case, the appropriate estimator of $\bar{x}$ is

$$\hat{\bar{x}} = (A^T A)^{-1} A^T \bar{y} \tag{3.1}$$

where a superscript ^ denotes an estimate, and a superscript ^{T} denotes algebraic transposition [31]. In general, given the presence of noise in the measurements, no value of $\bar{x}$ satisfies (2.13) exactly. The value of $\hat{\bar{x}}$ in (3.1) is a compromise in the sense that it minimizes the length of the error vector $\bar{y}$ - $A\bar{x}$. To emphasize this compromise we call $\hat{\bar{x}}$ an estimate. Equation (3.1) can also be used for the discussion of sufficient richness of the measurements. A singular value decomposition can be applied to the matrix A^TA of (3.1) to determine singular values that correspond to each row of A^TA [31]. These singular values in turn determine the degree of overspecification of the parameters in $\bar{x}$. In order for the estimate in (3.1) to be well-behaved, the matrix A^TA must be invertible. The magnitudes of the singular values indicate the invertibility of A^TA. As the magnitude of any singular values becomes progressively smaller than the others, A^TA becomes progressively closer to dropping one in rank. When this happens A^TA no longer invertible as required by (3.1). The condition number, defined to be the magnitude of the ratio of the maximum singular value to the minimum singular value, is used as a convenient quantity to measure the invertibility of A^TA, and hence the sufficient richness of the measurements. As the condition number becomes smaller, the matrix becomes more invertible, and

equivalently the measurements become richer. The three estimators presented in this chapter are all developed in this spirit of LLSE estimation.

The first estimator is simply (3.1) which neglects the dependence of the all five parameters appearing in (2.12). Given the LLSE estimate of $\bar{x}$ to Equation (2.11) by (3.1), the fundamental parameters, R_s, R_r, L, and M can be determined by (2.12), although not uniquely. This estimator thus ignores the fact that the parameter x_5 is the product of the parameters x_1 and x_2. This estimator is discussed in detail in Section 3.2.

The second estimator is an iterative LLSE estimator which follows the first. This estimator recognizes the dependence among the parameters x_1, x_2, and x_5 that the first estimator ignores. By enforcing this dependence on the estimator, we expect that it would produce better estimates over the first estimator; in reality, however, we will see that this expectation is not always met. To begin, one of the parameters, x_1, x_2, or x_5, is selected and evaluated from the other two. LLSE estimation of the other four parameters then proceeds iteratively following a method similar to (3.1). The resulting estimates are used to re-evaluate the evaluated parameter, and the process is repeated until sufficient convergence of all five parameters is obtained. This iterative procedure is developed and discussed for each of parameters x_1, x_2, and x_5 in Section 3.3.

The third estimator also follows the first. However, it assumes knowledge of the stator resistance R_s. Given R_s, the estimation problem underlying the first estimator reduces to the estimation of three parameters, τ_r, L, and K/R_r. The three unknown fundamental parameters L, M, and R_r are then uniquely determined from the estimates of the reduced set of parameters. The main reason for developing this estimator is that R_s cannot be estimated well by either of the two estimators mentioned above. This important result is predicted in Section 3.5 and shown experimentally in Chapter 4. Since all the parameters are estimated

collectively, we expect that inadequate estimation of R_s will have a negative effect on the estimation of R_r as well. Again a LLSE estimator in the spirit of (3.1) is used to estimate the new parameters. This estimator seems to require a measurement beyond ω_e, s, $|v_s|$, $|i_s|$, and Φ, and thus appears to violate the spirit of this thesis. We want to detect broken rotor bars using only the electrical measurements from the stator terminals and the measurement of the rotor speed; we do not want to be intrusive to the operation of the motor. However, it is shown in Chapter 5 that an excellent iterative estimate of R_s can be obtained from only these measurements and an initial measurement of R_s with the help of a thermal model of the induction motor. In practice, this estimate is substituted for a measurement of R_s. The third estimator and measurement issues of R_s are described in detail in Section 3.4.

To summarize, three estimators have been introduced. They use the following estimation methods: (1) LLSE estimation with estimated R_s, (2) iterative LLSE estimation with estimated R_s, and (3) LLSE estimation with known R_s. To determine the merits of each estimator, Section 3.5 analyzes their sensitivities by numerically evaluating the effect of inherent errors in the collected measurements on the estimated parameter values. An important prediction by the sensitivity analysis is that the first two estimators will produce inadequate estimates of R_s. This prediction is confirmed by experimental results in Chapter 4. Consequently, when used in the thermal model developed in Chapter 5, R_s should be either measured or better estimated by a different method such as the non-intrusive measurement of R_s discussed in [32] or temperature-based estimation of R_s described in Chapter 5.

3.2 LLSE Estimation with Estimated R_s

As outlined in the introduction of this chapter, the first estimator independently estimates all five parameters defined by (2.12). Given the collection of equations in the

form of (2.11), which in turn takes the form of (2.13), the estimate of $\bar{x}$ is given by (3.1). This estimator is naive in that it ignores the dependence among the parameters x_1, x_2, and x_5; parameter x_5 is the product of the parameters x_1 and x_2. Since the estimator ignores this dependence, five parameters are independently estimated instead of four. This necessitates the collection of three or more sets of independent measurements, each set consisting of ω_e, s, $|v_s|$, $|i_s|$, and Φ, since as noted in Chapter 2, each set of measurements produces two independent equations. Consequently, the subscript m denoting the number of independent sets of measurements in (2.11) must be 3 or more. If the measurements were perfect, $m = 3$ would yield a perfect estimate of $\bar{x}$. However, the presence of noise in the measurements is unavoidable. Thus, to minimize the effects of error, m should be sufficiently larger than 3 to over-constrain the estimate of $\bar{x}$; larger the m, better the estimate. As more independent measurements are made, the condition number of $\mathbf{A}^T\mathbf{A}$ in (2.11) becomes smaller and hence its invertibility becomes better. This in turn produces better LLSE estimates of $\bar{x}$.

To summarize, the corresponding LLSE estimate of $\bar{x}$ is

$$\hat{\bar{x}} = \begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \\ \hat{x}_3 \\ \hat{x}_4 \\ \hat{x}_5 \end{bmatrix} = \begin{bmatrix} \tau_r \\ R_s \\ L \\ \frac{K}{R_r} \\ R_s\tau_r \end{bmatrix} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\bar{y} \,, \tag{3.2}$$

where the A is the measurement matrix and $\bar{y}$ is the measurement vector of (2.12). Once $\hat{\bar{x}}$ is determined, all the estimates of R_s, R_r, L, and M can be determined. However, these

latter estimates are not unique, since the dependence among the parameters x_1, x_2, and x_5 is not enforced on the estimator. To obtain the estimates of the four fundamental parameters, R_s, R_r, L, and M, only the estimates of any four independent parameters of (2.14) is needed. More specifically, since there are three parameters of the five that share a dependence, namely, x_1, x_2, and x_5, only two of these three are needed. Hence, there are three different ways of obtaining the estimates of R_s, R_r, L, and M, each corresponding to not using different one of the estimates $\hat{x}_1$, $\hat{x}_2$, and $\hat{x}_5$. Again, we do not expect the resulting three sets of estimated values for the fundamental parameters to be equal, since the dependence of x_1, x_2, and x_5 is not taken into account by this estimator.

For the estimator used experimentally in this thesis, $\hat{x}_5$ is not used to determine the four fundamental parameters. For this case, the estimates of the fundamental parameters are as follows. First, the estimates of R_s and L are given by

$$\hat{R}_s = \hat{x}_2, \qquad \hat{L} = \hat{x}_3 \ . \tag{3.3}$$

Using (3.3), the estimate of R_r can then be obtained from

$$\hat{R}_r = \frac{\hat{L}}{\hat{x}_1} \ . \tag{3.4}$$

Finally, the estimate of M uses both (3.3) and (3.4) to give

$$\hat{M} = \sqrt{\hat{L}^2 - \hat{R}_r \hat{x}_4} \ . \tag{3.5}$$

This estimator is a convenient place to start the progression of analysis, but the decision not to use the dependence among the three parameters in shaping the estimator seems

unwarranted. The next section describes an estimator which uses this dependence as a constraint on the estimation process.

3.3 Iterative LLSE Estimation with Estimated R_s

As seen in the last section, an LLSE estimator can be used to estimate all the fundamental parameters. However, as discussed in the last section, $x_5 = R_s\tau_r$, for example, is really a dependent parameter; it is just the product x_1x_2. We can use this extra constraint to iterate on the LLSE estimation. If this extra constraint is forced on the estimator, we expect the iterative LLSE estimates to yield better results than those of the non-iterative LLSE parameter estimator of Section 3.2. This intuition is proven with the experimental results in Chapter 4.

As stated earlier, $x_5 = x_1x_2$. Thus, either x_1, x_2, or x_5 can be considered the extra constraint. This then specifies the three possible iterative schemes. For each of these three possible schemes, the starting point is the estimate of $\bar{x}$ given by (3.2) using the LLSE estimator in Section 3.2. If the measurements were perfect, we could stop there. This single-step LLSE estimation of the parameters will be consistent in that the fifth term will indeed be equal to the product of the first two terms. The measurements, however, are not perfect. So, the problem of matching the dependence of the estimates is necessarily an iterative one. Therefore, one of the parameters x_1, x_2, or x_5, is selected as the extra constraint, and is evaluated in terms of the estimates of the other two parameters. For example, given that we choose x_5, it is evaluated as the product $\hat{x}_1\hat{x}_2$. This new value for the parameter designated as the extra constraint is then treated as a known measurement and LLSE estimation of the other four parameters proceeds iteratively. At each iteration, the evaluated parameter is updated from the new estimates of the other parameters, and the

other parameters are re-estimated using this update. The resulting estimates at the i^{th} iteration are compared to those of $(i-1)^{st}$ iteration, and the process is repeated until sufficient convergence of all the estimates is obtained. Note that a minimum of three sets of measurements are necessary for these iterative estimators, since the initial estimates come from (3.2). An alternative way of iterating is to set the constrained parameter to a nominal value or to zero before the first iteration. This then only requires two independent sets of measurements.

Three possible iterative estimators have been applied to ideal data, that is, noiseless data generated numerically from the single-phase equivalent circuit. Two of the iterative estimators are stable and one is unstable. If $x_1 = \tau_r$ is taken as the dependent variable, then the estimation process diverges rapidly. The rate of convergence for the other two are nearly equal, but the case when $x_2 = R_s$ is taken as the evaluated variable seems to produce slightly better results in terms of estimator performance. Hence, although all three iterative estimators are given here, only the iterative estimator with R_s as the extra constraint is considered for all physical experiments as well as the sensitivity analysis in Section 3.5.

For the first of the three possible iteration schemes, $x_5 = R_s\tau_r$ is chosen as the dependent variable. It is considered as a known measurement and is substituted by the product $\hat{x}_1\hat{x}_2$ in the iterations. At this point, for notational purposes, let A of (2.11) be given by

$$\mathbf{A} = [a_1 \mid a_2 \mid a_3 \mid a_4 \mid a_5] \ , \tag{3.6}$$

and

$$\mathbf{A}_1 = [a_1 \mid a_2 \mid a_3 \mid a_4] \tag{3.7}$$

where a_i is the ith column vector of A. Thus, at the i^{th} iteration, the estimate of x_1, x_2, x_3, and x_4 for this particular iterative estimator is

$$\begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \\ \hat{x}_3 \\ \hat{x}_4 \end{bmatrix} = \begin{bmatrix} \hat{\tau}_r^i \\ \hat{R}_s^i \\ \hat{L}^i \\ \left(\widehat{\frac{K}{R_r}}\right)^i \end{bmatrix} = \left(\mathbf{A}_1^T\mathbf{A}_1\right)^{-1}\mathbf{A}_1^T\left(\bar{y} - \hat{R}_s^{i-1}\hat{\tau}_r^{i-1}a_5\right) \tag{3.8}$$

where the superscripts i and i-1 denote the corresponding iteration step, a_5 is defined in (3.6). The iteration is considered to have converged sufficiently if the following condition is satisfied:

$$\left|\frac{\hat{R}_s^i\hat{\tau}_r^i - \hat{R}_s^{i-1}\hat{\tau}_r^{i-1}}{\hat{R}_s^i\hat{\tau}_r^i}\right| < \delta \ . \tag{3.9}$$

Of course, we can specify how well the terms must converge by picking the value of δ. In testing the iterative estimators, δ is set to 0.001 for all three estimators. As long as the condition (3.9) is not satisfied, the process iterates. When the condition (3.9) is satisfied, the fundamental parameters are determined uniquely from the final result of (3.8). This particular iteration scheme is numerically stable and converges rapidly for the specific induction motor used in this thesis. Hence, the stability is not proven for all other induction motors. The general study of stability is suggested as a topic for future work.

For the second iteration, which is unstable for the motor used in this thesis, $x_1 = \tau_r$ is chosen as the evaluated dependent variable. This extra constraint is evaluated as the

quantity $\hat{x}_5/\hat{x}_2$ in the iterations. Thus, at the i[th] iteration, the estimate of x_2, x_3, x_4, and x_5 is

$$\begin{bmatrix} \hat{x}_2 \\ \hat{x}_3 \\ \hat{x}_4 \\ \hat{x}_5 \end{bmatrix} = \begin{bmatrix} \hat{R}_s^i \\ \hat{L}^i \\ \left(\frac{\hat{K}}{R_r}\right)^i \\ (\widehat{R_s\tau_r})^i \end{bmatrix} = \left(\mathbf{A}_2^T\mathbf{A}_2\right)^{-1}\mathbf{A}_2^T\left(\bar{y} - \frac{(\widehat{R_s\tau_r})^{i-1}}{\hat{R}_s^{i-1}}a_1\right) \tag{3.10}$$

where

$$\mathbf{A}_2 = [a_2 \mid a_3 \mid a_4 \mid a_5] \tag{3.11}$$

with each a_i found again in (3.6). For this second iterative estimator, the iteration is considered to have converged sufficiently if

$$\left|\frac{(\widehat{R_s\tau_r})^i/\hat{R}_s^i - (\widehat{R_s\tau_r})^{i-1}/\hat{R}_s^{i-1}}{(\widehat{R_s\tau_r})^i/\hat{R}_s^i}\right| < \delta \ . \tag{3.12}$$

Unfortunately, when tested with ideal data and perturbed initial guesses, this estimator is numerically unstable and thus the estimation process actually diverges. This means that the convergence condition (3.12) is never satisfied. This estimation scheme should not be used for this motor. For other induction motors, however, the instability has not been proven. This is also left for future work.

The third iterative estimator yields stable estimates and is developed in the same way as for the first two. This estimator uses $x_2 = R_s$ as the dependent variable, which is evaluated as the quantity $\hat{x}_5/\hat{x}_1$. Here, at the i[th] iteration, the estimate of x_1, x_3, x_4, and x_5 is

$$\begin{bmatrix} \hat{x}_1 \\ \hat{x}_3 \\ \hat{x}_4 \\ \hat{x}_5 \end{bmatrix} = \begin{bmatrix} \hat{\tau}_r^i \\ \hat{L}^i \\ \left(\frac{\widehat{K}}{R_r}\right)^i \\ \left(\widehat{R_s\tau_r}\right)^i \end{bmatrix} = \left(\mathbf{A}_3^{\mathrm{T}}\mathbf{A}_3\right)^{-1}\mathbf{A}_3^{\mathrm{T}}\left(\bar{y} - \frac{\left(\widehat{R_s\tau_r}\right)^{i-1}}{\hat{\tau}_r^{i-1}}a_2\right) \tag{3.13}$$

where

$$\mathbf{A}_3 = [a_1 \mid a_3 \mid a_4 \mid a_5] \tag{3.14}$$

with each a_i found again in (3.6). This iteration is considered to have converged sufficiently when

$$\left|\frac{\left(\widehat{R_s\tau_r}\right)^i/\hat{\tau}_r^i - \left(\widehat{R_s\tau_r}\right)^{i-1}/\hat{\tau}_r^{i-1}}{\left(\widehat{R_s\tau_r}\right)^i/\hat{\tau}_r^i}\right| < \delta \tag{3.15}$$

is satisfied. The third iterative estimator is stable like the first, and the four fundamental parameters R_s, R_r, L, and M are determined uniquely once (3.15) is satisfied. Given the same δ and given that both the first and the third iterative estimators are applied to matrices that have been formed by perturbing the ideal data, the estimates of the third iteration scheme are slightly better than those of the first when the convergence conditions of (3.9)

and (3.15) are met for the first time. Thus, only this estimator is actually used out of the three iterative estimators in the remainder of this thesis.

3.4 LLSE Estimation with Known R_s

At this point, another estimator is examined, namely the estimator with an R_s that is assumed to be known. This means that R_s is no longer a parameter to estimate. Since there is one less parameter to estimate, our intuition leads us to believe that the corresponding estimator should do better than the two that also estimate R_s. This primary reason for considering this estimator is the fact that the estimation of R_s turns out be quite unsatisfactory for the previously discussed LLSE and iterative LLSE estimators. This result is predicted by the sensitivity analysis in the next section and shown experimentally in Chapter 4. Furthermore, this result has been observed in other studies [33,34] as well. This is important because for the thermally-compensated detector in Chapter 5 to work, we need good estimates of R_s. Although the inability to estimate R_s well has not been fully explained, it appears related to the fact that core losses are not included in the single-phase equivalent circuit in Figure 2.3. However, preliminary investigation into augmenting the single-phase equivalent circuit by placing a core-loss resistor in parallel with the mutual inductor has not produced any improvements over the existing estimators. To deal with this problem, the unsatisfactory estimation of R_s is replaced by accurate knowledge of R_s.

For the experimental results in Chapter 4, and for much of the experimental results in Chapter 5, the knowledge of R_s comes from an actual measurement of R_s. This seems to violate the spirit of this thesis in that an intrusive measurement is needed beyond ω_e, s, $|v_s|$, $|i_s|$, and Φ. To be a useful tool, the estimation technique should be non-intrusive to the machine operation; the operation of the motor must not be interrupted by the measurement

of R_s. Recently, a non-intrusive method of measuring R_s has been developed [32]. Therefore, the intrusive measurement of R_s given in Chapter 4 can be used to approximate the implementation of the non-intrusive measurement setup described in [32]. More importantly, as stated in the introduction of this chapter an excellent iterative estimate of R_s can be obtained from only the measurements ω_e, s, $|v_s|$, $|i_s|$, and Φ, and an initial measurement of R_s with the help of a thermal model of the induction motor. This is developed in Chapter 5. Since this increases in only a minor way the complexity of the algorithm used to thermally compensated the estimates, instead of the additional hardware setup required by [32], the thermal estimation of R_s is recommended as a standard procedure and substituted for a measurement of R_s in practice. The measurements of the R_s as described in Chapter 4 is then necessary to prove the validity of the thermally-based estimation of R_s, since before it can be used, it must be verified.

If we assume the availability of R_s, then a new estimator can be developed using R_s as a known parameter. In this case, the collection of equations in (2.11) can be reduced. Specifically, $x_2 = R_s$ becomes a known, and x_5 becomes $R_s x_1$ which can then be combined with x_1. Thus, only three parameters need be estimated. The corresponding collection of measurements and equations that is re-derived from (2.11) by using R_s as a known is

$$\begin{bmatrix} \omega_e s_1 R_s |i_{s1}| \sqrt{1-\Phi_1^2} & \omega_e |i_{s1}| \sqrt{1-\Phi_1^2} & -\omega_e^2 s_1 |i_{s1}| \Phi_1 \\ \omega_e s_2 R_s |i_{s2}| \sqrt{1-\Phi_2^2} & \omega_e |i_{s2}| \sqrt{1-\Phi_2^2} & -\omega_e^2 s_2 |i_{s2}| \Phi_2 \\ \vdots & \vdots & \vdots \\ \omega_e s_m R_s |i_{sm}| \sqrt{1-\Phi_m^2} & \omega_e |i_{sm}| \sqrt{1-\Phi_m^2} & -\omega_e^2 s_m |i_{sm}| \Phi_m \\ \omega_e s_1 \left(R_s |i_{s1}| \Phi_1 - |v_{s1}|\right) & \omega_e |i_{s1}| \Phi_1 & \omega_e^2 s_1 |i_{s1}| \sqrt{1-\Phi_1^2} \\ \omega_e s_2 \left(R_s |i_{s2}| \Phi_2 - |v_{s2}|\right) & \omega_e |i_{s2}| \Phi_2 & \omega_e^2 s_2 |i_{s2}| \sqrt{1-\Phi_2^2} \\ \vdots & \vdots & \vdots \\ \omega_e s_m \left(R_s |i_{sm}| \Phi_m - |v_{sm}|\right) & \omega_e |i_{sm}| \Phi_m & \omega_e^2 s_m |i_{sm}| \sqrt{1-\Phi_m^2} \end{bmatrix} \begin{bmatrix} \tau_r \\ L \\ \frac{K}{R_r} \end{bmatrix} = \begin{bmatrix} |v_{s1}| - R_s |i_{s1}| \Phi_1 \\ |v_{s2}| - R_s |i_{s2}| \Phi_2 \\ \vdots \\ |v_{sm}| - R_s |i_{sm}| \Phi_m \\ R_s |i_{s1}| \sqrt{1-\Phi_1^2} \\ R_s |i_{s2}| \sqrt{1-\Phi_2^2} \\ \vdots \\ R_s |i_{sm}| \sqrt{1-\Phi_m^2} \end{bmatrix} \tag{3.16}$$

where the notation of symbols and the subscripts are the same as those of (2.11). Equation (3.16), like (2.11), is in the form of (2.13), with A the reduced measurement matrix and measurement vector $\bar{y}$.

Equation (3.16) is also a near-LLSE estimation problem. Further, the estimator for (3.16) requires no iteration. Consequently, the issues associated with iteration, such as stability and rate of convergence need not be addressed. Yet, since we have more information, namely the value of R_s, this estimator should yield better results. Thus, if we assume a negligible cost of acquiring an accurate value of R_s, which is the case with thermally estimated R_s, we expect the known-R_s estimator to give us the best of both worlds: ease of estimation and more accurate results.

Applying the new estimator, the estimate of x_1, x_3, and x_4 as defined in (2.12) is

$$\begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \\ \hat{x}_3 \end{bmatrix} = \begin{bmatrix} \hat{\tau}_r \\ \hat{L} \\ \left(\widehat{\frac{K}{R_r}}\right) \end{bmatrix} = (\mathbf{A}_4^{\mathrm{T}}\mathbf{A}_4)^{-1}\mathbf{A}_4^{\mathrm{T}}\bar{y}_4 \ , \tag{3.17}$$

where $\mathbf{A}_4$ is the measurement matrix in (3.16) and $\bar{y}_4$ is the measurement vector in (3.16). The estimates of the fundamental parameters R_s, R_r, L, and M are obtained from (3.17) by the following:

$$\hat{L} = \hat{x}_2 \qquad \hat{R}_r = \hat{L}/\hat{x}_1 = \hat{L}/\hat{\tau}_r \qquad \hat{M} = \sqrt{\hat{L}^2 - \hat{R}_r\hat{x}_3} \ . \tag{3.18}$$

The three types of estimators, LLSE with estimated R_s, iterative LLSE with estimated R_s, and LLSE with known R_s, are now fully developed. In the next section we proceed to determine the effect of inherent errors in the measurements on the estimation results given by each of the three estimators developed in this chapter. Also in Chapter 4, these estimators are used in physical experiments to see whether they can in fact detect a broken rotor bar.

3.5 Sensitivity Analysis

In the previous sections, three parameter estimators have been developed. These are the LLSE estimator with estimated R_s, the iterative LLSE estimator with estimated R_s, and the LLSE estimator with known R_s. In this section, we examine the sensitivity of these estimators to noise in the measurements on which they operate. In particular, we bound the variance of the parameter estimates given the variance of the measurements from which the estimates are derived. It must be mentioned that in this section we are only concerned with the sensitivity to measurement error and not to modelling error. The sensitivity analysis presented here is therefore incomplete. As we shall see in this section and in Chapter 4, R_s is not estimated well. While this may result from measurement noise, a phenomenon which is predicted in this section, it may also result from modelling error. Two known sources of modelling error are the omission of core losses and the frequency-dependence of parameters in the electrical model.

It is important to study the sensitivity of the estimators because this sensitivity is a measure of their potential for their success. In particular, if the variance of the estimate of R_r due to measurement noise is larger than the change in R_r that can be expected to come from a broken rotor bar, then it will be very difficult to observe a broken rotor bar via the

estimate of R_r. Since the variance of the estimate of R_r can be reduced with more data, the sensitivity of the estimators can alternatively be used to determine how much data must be collected before a useful estimate of R_r can be made. Finally, since the sensitivities of the three estimators can be expected to differ, a comparison of sensitivities can be used to select one over the other two.

Each of the three estimators is analytically complex. Therefore, it is not possible to determine their sensitivities analytically. Consequently, we employ a numerical method to approximately evaluate the sensitivities. The numerical method begins with ideal data which, when processed by each estimator, yields error-free estimates of the parameters. Next, the ideal data is perturbed and processed again. The sensitivity of an estimator is judged by the change in its parameter estimates which result from the perturbed data. To simplify this analysis, data perturbation and estimation processing is carried out several times. Each time, only one type of measurement is perturbed. In this way, it is also possible to identify the need for particularly precise instrumentation. Section 3.5.1 outlines the method of analysis and Section 3.5.2 presents numerical results of the sensitivity analysis.

3.5.1 Method

The sensitivity method outlined in this section gives upper bounds on the variations of the estimates given the observed error in the measurements by Table 2.3. The method here uses all the relevant data from the experimental system given in Section 2.1. Furthermore, it uses the nominal values of the fundamental parameters, R_s, R_r, L, and M that are averaged from estimates of some preliminary experimental results. This is

done to simulate the experimental conditions as closely as possible, so that the correlations of any analytical results from this section to those of the actual experiments are maximized.

First, ideal data composed of measurement vectors is derived from a set of reference values for the fundamental parameters. These reference values have been obtained experimentally using the procedure described in Chapter 4. A measurement vector of the ideal data is then perturbed by the observed maximum error for that particular measurement. This perturbed data is then used by each one of the estimators to obtain corresponding estimates. This procedure is repeated for each measurement vector of the ideal data. Finally, an upper bound of the variation in each of the parameters is computed from these estimates and the reference parameter values.

To begin, the reference values for the fundamental parameters, R_s, R_r, L, and M, are given in Table 3.1. From these values an ideal data matrix **I** can be constructed. The ideal data **I** must form the following overconstrained matrix,

$$\mathbf{I} = \begin{bmatrix} |i_{s1}| & \Phi_1 & \Omega_1 & |v_{s1}| \\ |i_{s2}| & \Phi_2 & \Omega_2 & |v_{s2}| \\ \vdots & \vdots & \vdots & \vdots \\ |i_{s16}| & \Phi_{16} & \Omega_{16} & |v_{s16}| \end{bmatrix} \tag{3.19}$$

with the numerical subscript indexing the measurement sets and with the measurements as defined in Table 2.3. Note that there are 16 independent sets of measurements, since in Chapter 4, each set of estimates are consistently obtained from 16 sets of independent measurements. The number of independent sets of measurements is important only in that it must guarantee sufficient richness of the measurements; the choice of 16 measurements is arbitrary. Also, since all the rms magnitudes of the stator voltage are nominally 120 volts

from Chapter 2, all the $|v_{si}|$'s of (3.19) are kept constant at 120 volts. Moreover, as is done in the experimental procedure, the rotor speed Ω is decremented by 5 rpm from 1795 to 1720 inclusively. With the two measurement vectors for Ω and $|v_s|$ of (3.19) fixed in this manner, the other two measurement vectors in (3.19) can be determined uniquely by using the reference values and the two equations, one based on the real part and the other based on the imaginary part, provided by Equation (2.3) for each set of measurements. Neglecting the numerical round-off errors, this ideal data is perfect in the sense that applying any of the two estimators, LLSE with estimated R_s and iterative LLSE with estimated R_s, to it produces the reference values for the fundamental parameters R_s, R_r, L, and M exactly without any error. In short, the estimates replicate Table 3.1.

$$\bar{p}^{ref} = \begin{bmatrix} p_1^{ref} \\ p_2^{ref} \\ p_3^{ref} \\ p_4^{ref} \end{bmatrix} = \begin{bmatrix} R_s^{ref} \\ R_r^{ref} \\ L^{ref} \\ M^{ref} \end{bmatrix} = \begin{bmatrix} 0.865\Omega \\ 0.563\Omega \\ 74.13\text{mH} \\ 69.87\text{mH} \end{bmatrix}$$

Table 3.1: Reference Values for the Fundamental Parameters

An ideal data similar to (3.19) can be constructed to test the LLSE estimator with known R_s quite easily. The ideal data in (3.19) must be augmented by one more measurement vector for the measurement of R_s. All the elements of this vector are the same, namely the reference value of R_s found in Table 3.1. Since the sensitivity analysis of the LLSE estimator with known R_s can be done with minimal modifications to the sensitivity analysis of the other two estimators, only the estimators with estimated R_s will be analyzed explicitly. However, appropriate comments for the estimator with known R_s are made whenever necessary.

We can perturb the matrix **I** one measurement column vector at a time by adding to and subtracting from all 16 elements of this vector the maximum observed error in that measurement. The maximum observed measurement errors are taken from Table 2.3. For instance, a set of perturbed matrices formed by perturbing the Ω column of the **I** are

$$\mathbf{P}^{+}(\Delta\Omega) = \begin{bmatrix} |i_{s1}| & \Phi_1 & \Omega_1 + \Delta\Omega & |v_{s1}| \\ |i_{s2}| & \Phi_2 & \Omega_2 + \Delta\Omega & |v_{s2}| \\ \vdots & \vdots & \vdots & \vdots \\ |i_{s16}| & \Phi_{16} & \Omega_{16} + \Delta\Omega & |v_{s16}| \end{bmatrix} \tag{3.20}$$

and

$$\mathbf{P}^{-}(\Delta\Omega) = \begin{bmatrix} |i_{s1}| & \Phi_1 & \Omega_1 - \Delta\Omega & |v_{s1}| \\ |i_{s2}| & \Phi_2 & \Omega_2 - \Delta\Omega & |v_{s2}| \\ \vdots & \vdots & \vdots & \vdots \\ |i_{s16}| & \Phi_{16} & \Omega_{16} - \Delta\Omega & |v_{s16}| \end{bmatrix}. \tag{3.21}$$

Similarly, the matrices $\mathbf{P}^{+}$ and $\mathbf{P}^{-}$ can be formed with errors corresponding to the other measurements. Again, the numerical values of $\Delta|i_s|$, $\Delta\Phi$, $\Delta\Omega$ and $\Delta|v_s|$ are found in Table 2.3, and

$$\Delta R_s = 0.002 \text{ Ohms} \tag{3.22}$$

have also been observed experimentally at constant-temperature operation of the motor. Each of the perturbed matrices gives rise to corresponding data matrix in the form of **A** and $\bar{y}$ in (2.11) and $\mathbf{A}_4$ and $\bar{y}_4$ in (3.16), which in turn can be used to derive estimates via the three estimators. Then, for each of the two estimated-R_s estimators, we define Δp_j the change in the parameter p_j in presence of a $\Delta\mu_i$ a change in the measurement μ_i as

$$\Delta p_j(\Delta\mu_i) = \max \begin{cases} \left\| p_j(\mathbf{P}^+(\Delta\mu_i)) - p_j^{ref} \right\| \\ \left\| p_j(\mathbf{P}^-(\Delta\mu_i)) - p_j^{ref} \right\| \end{cases} \tag{3.23}$$

where the subscript j indexes the fundamental parameters

$$\bar{p} = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{bmatrix} = \begin{bmatrix} R_s \\ R_r \\ L \\ M \end{bmatrix} \tag{3.24}$$

and also the corresponding reference fundamental parameters given by Table 3.1, and where the subscript i indexes the four measurement errors according to

$$\Delta\bar{\mu} = \begin{bmatrix} \Delta\mu_1 \\ \Delta\mu_2 \\ \Delta\mu_3 \\ \Delta\mu_4 \\ \Delta\mu_5 \end{bmatrix} = \begin{bmatrix} \Delta|i_s| \\ \Delta\Phi \\ \Delta\Omega \\ \Delta|v_s| \\ \Delta R_s \end{bmatrix}. \tag{3.25}$$

Notice that the fifth element of (3.25) is excluded since R_s for the two estimators considered is treated as an unknown fundamental parameter, not a measurement. Similarly for the LLSE estimator with known R_s, $\Delta p_j(\Delta\mu_i)$ in (3.23) can be obtained for all the p_j's in (3.24) except for that of the first vector element, since R_s is now a measurement. Correspondingly, $\Delta\bar{\mu}$ extends to include the fifth element of (3.25).

Finally, for the two *Rs*-estimated estimators, an upper bound on the variation of the parameter p_j to the variations in measurement errors is given by

$$\sigma_{P_j}^2 \le \sum_{i=1}^{4} \left[\frac{\Delta p_j(\Delta\mu_i)}{(\Delta\mu_i)} \right]^2 (\Delta\mu_i)^2 = \sum_{i=1}^{4} [\Delta p_j(\Delta\mu_i)]^2 \tag{3.26}$$

where p_j's (j = 1 to 4) are defined in (3.24) and $\Delta\mu_i$'s are defined in (3.25). Likewise for the R_s-known estimator, the upper limit on the summation for (3.26) is changed from 4 to 5 and p_j's are taken for j = 2 to 4 instead. One interpretation of (3.26) is that each $\Delta p_j(\Delta\mu_i)$ can be seen as a distance between a particular estimate of p_j and the expected value of p_j. Thus, each estimate produced from perturbing one measurement vector at a time by the maximum error in that measurement gives rise to one datum point. Equation (3.26) then computes the variation, or the square of the standard deviation, for these data points using p_j^{ref} as the expected value.

Since $\Delta p_j(\Delta\mu_i)$ in (3.23) takes the maximum, this analysis gives a conservative upper bound on standard deviations σ_{P_j}'s. In reality, we do not expect all 16 sets of measurements to read either all up or all down from their expected values by the maximum error values given in Table 2.3 and (3.22). Most likely, we will encounter a mixture of measurements varying in magnitudes and signs, leading to tighter bounds on σ_{P_j}'s than given by (3.26).

3.5.2 Numerical Results

For each of the three estimators studied in this chapter, the numerical results of Equation (3.26) for all the fundamental parameters are given in this section. The results indicate that detection of a broken rotor bar for the experimental motor is possible since the R_r-deviation bounds for two of the three estimators are less than that which corresponds to about 2%

change in rotor resistance due to a single broken rotor bar out of 45 bars. The estimation of R_s on the other hand, is shown to be poor. The implication of the inability to adequately estimate R_s on the thermally-compensated estimation of R_r is then discussed. Lastly, since the bounds and the estimation accuracy itself are functions of measurement errors inherent in the measuring apparatus, we investigate the type of measuring accuracy needed for an acceptable corresponding bound on the variation of R_s.

The numerical results compiled in this chapter and all future chapters are developed using MatrixX; see Appendix C for the MatrixX programs of the various estimators developed in this chapter, and Appendices D and E for the MatrixX programs and their numerical results of the sensitivity analysis. Figure 3.1 shows the upper bound on the standard deviation for each fundamental parameter for all three LLSE estimators. As an alternative presentation, Figure 3.2 gives the bound on the ratio σ_p/p^{ref} in percentage, which is named percent-deviation. Percent-deviation measures the deviation of parameter, p, as a percentage of the reference parameter value, p^{ref}. Given that a broken rotor bar increases the R_r by 2%, an examination of Figure 3.1 and 3.2 show that the LLSE estimator with estimated R_s and the LLSE estimator with known R_s can detect one broken bar, since the upper bounds on their percent-deviations are both less than 2%. Furthermore, the figures indicate that the LLSE estimator with estimated R_s seems to produce the smallest percent-deviation of the rotor resistance.

To begin the comparison of the sensitivity results for the three different types of estimators, we must recognize that by perturbing the ideal data matrix I, we introduce noise into the measurements which subsequently corrupts the estimates generated by all three estimators. The ways in which the three different estimators distribute this noise are used to explain the numerical results of the sensitivity analysis. Among the three estimators, the LLSE estimator with estimated R_s has the most degree of freedom in the unknown

parameters. Thus, for this case, the noise is distributed among five parameters. Apparently, the estimator distributes most of the noise into the estimate of R_s; hence, the bound on the percentage-deviation of R_s is 74.4%. On the other hand, the estimator corrupts the estimates of the other parameters minimally. In particular the bound on the percent-deviation of R_r is 1.4%.

If the degrees of freedom decrease by one as is the case of iterative LLSE estimator with estimated R_s, then the noise is distributed differently. Some of the noise which is present in the estimate of R_s for the first estimator is redistributed into the estimate of R_r. The end result is that while the estimate of R_s becomes better, the estimate of R_r becomes worse. In fact, the 3.9% bound on the percentage-deviation of R_r is well above the required 2% to detect a broken rotor bar for the experimental motor. This result makes sense in that the iterative estimator, by enforcing a dependence among the parameters, in effect balances the overall estimation of R_r and R_s; the estimator, because of the enforced dependence, is concerned more about the estimates taken collectively than it is about any particular estimate. Consequently, the poor estimate of R_s for the first estimator is compensated at a cost of the further corruption of the estimate of R_r. Even so, the bound on the percent-deviation of R_s is 31%. Thus in both of the cases discussed here, the error in the estimate of R_s is too large to support the temperature-compensated estimation of R_r, in which the thermal model requires good estimates R_s. Therefore, even though the 1.4% bound on the percent-deviation of R_r by the first estimator predicts detection of a single broken rotor bar at a constant temperature, this estimator is not useful in implementing the methods developed in Chapter 5. Therefore, we expect that the third estimator will be most successful in Chapter 5.

For the LLSE estimator with known R_s, the degrees of freedom in the unknown parameters decrease to three from the five in the first estimator and from the four in the

second. Since this causes a further restriction in the number of different estimates in which to place the noise, the estimate of R_r is corrupted more than that of the first case. This is seen in the increase of the bound on the percent-deviation of R_r from 1.4 for the first estimator to 1.9 for the third. Note that the 1.9% bound on the percentage-deviation for the third estimator is better than that of the second estimator, and more importantly, it still guarantees detection of a broken rotor bar. Therefore, even though the sensitivity analysis favors the first estimator discussed above for constant-temperature detection of a broken rotor bar, we must choose the LLSE estimator with known R_s, since the first two estimators cannot provide the sufficiently accurate estimates of R_s which are needed for the thermally-compensated estimation of R_r developed in Chapter 5.

Another issue which is not studied in this thesis must be addressed in future work. This is the sensitivity of the estimates to modelling errors. Specifically, core loss is not included in the equivalent circuit model of the induction motor, so the real power into the motor terminals which corresponds to core loss is associated instead with R_s and/or R_r. This may explain the inadequate estimates of R_s as observed in the experimental results in Chapter 4.

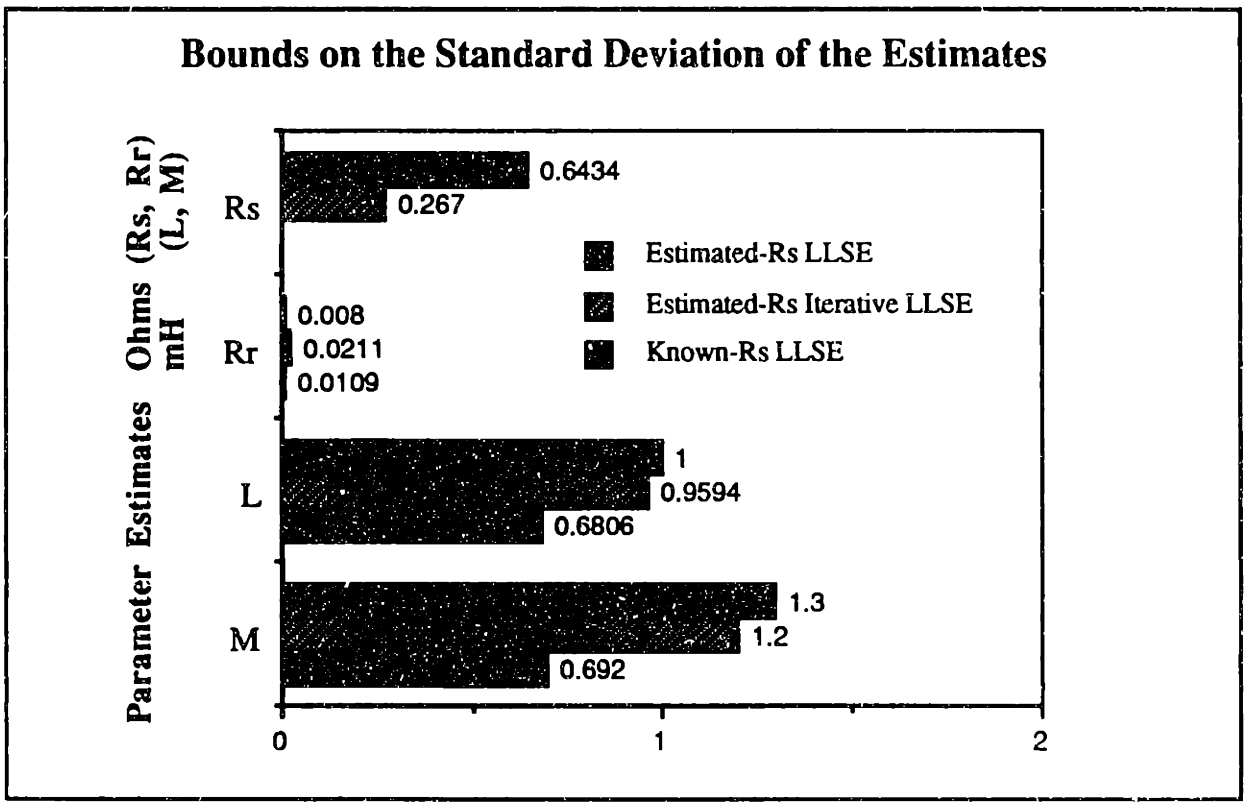


Figure 3.1: Bounds on the standard deviation of the parameter estimates.

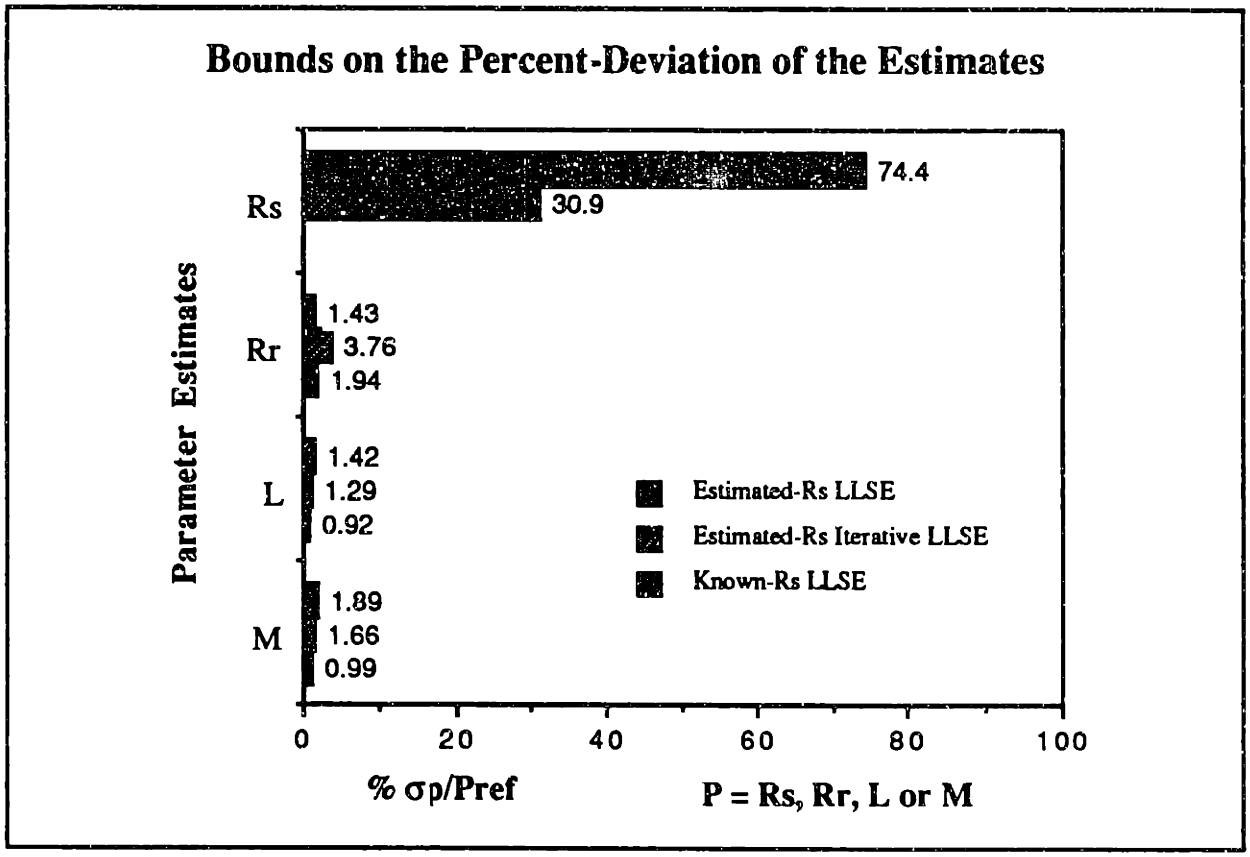


Figure 3.2: Bounds on the percent-deviation of the parameter estimates.

One last interesting analysis that can be performed is to determine the degree of measuring accuracy needed to ensure a reliable estimate of R_s. Note that the error values given in Table 2.3 and (3.22) need not be same for other type of instrumentation. They are only characteristic of the HD800-8 instrumentation used for this thesis. Therefore, as equipment becomes cheaper and more accurate, we can expect the noise limits to be bounded more tightly and subsequently produce better estimates. A few cases of improvements in measurement accuracy have been simulated with MatrixX, and a case which results in much better estimate of R_s is shown in Table 3.2, Figure 3.3, and Figure 3.4. Note from Table 3.2 the improvements necessary in measuring Ω and Φ. Typically, given the accuracy of the HD800-8 instrumentation, the estimation inaccuracy is due mostly to the measurement error in Ω. But by the time the measurement accuracy in Ω is improved

to the value given in Table 3.2, the error in the measurement of Φ starts to dominate. At this point, the bound on the percent-deviation of R_s is still at 15% for the LLSE estimator and 10% for the iterative LLSE estimator, thus the measurement accuracy of Φ has to be improved to the value given in Table 3.2 in order to reduce the percent-deviation of R_s to 2% for the LLSE estimator and 4% for the iterative LLSE estimator. While the desired accuracy of measuring Φ seems to be reasonable, the desired accuracy of measuring Ω seems to unreasonable, since the corresponding rotor speed measuring device must be 50 times more accurate than the one used for this thesis. It needs to resolve 0.02 revolutions per minute. This suggests us that new equipment is not the answer to obtaining reliable estimates.

Figure 3.4 shows that the iterative LLSE estimator does a better job of estimating R_r than the non-iterative LLSE estimator. This opposes the findings in Figure 3.2. Thus, in view of this contradiction, the sensitivity analysis in general is shown to be amplitude dependent. The reason for this is that the noise enters nonlinearly into the estimation process. The results given in Figures 3.1 and 3.2 are then specific only to the reference values and the measurement errors used in the sensitivity analysis. General conclusions cannot yet be drawn and are left for future work, but one point is clear. For the experimental system used in this thesis, we must use the LLSE estimator with known R_s, since the other two estimators produce inadequate estimates of R_s which becomes important in the thermal model.

Having completed the development of the estimators, namely the LLSE estimator with estimated R_s, the iterative LLSE estimator with estimated R_s, and the LLSE estimator with known R_s, and the their sensitivity analysis under the assumption of steady-state constant-temperature operation of the induction motor, we are ready to apply these estimators in

actual constant-temperature experiments. Chapter 4 presents these constant-temperature experimental results.

Original Measurement Errors	Modified Measurement Errors
$\Delta\lvert i_s\rvert = \pm 0.04$ A rms	$\Delta\lvert i_s\rvert = \pm 0.04$ A rms
$\Delta\lvert \Phi\rvert = \pm 0.005$	$\Delta\lvert \Phi\rvert = \pm 0.0005$
$\Delta\lvert \Omega\rvert = \pm 1$ rpm	$\Delta\lvert \Omega\rvert = \pm 0.02$ rpm
$\Delta\lvert v_s\rvert = \pm 0.3$ V rms	$\Delta\lvert v_s\rvert = \pm 0.3$ V rms

Table 3.2: Necessary bounds on the measurement errors for a reliable estimation of R_s.

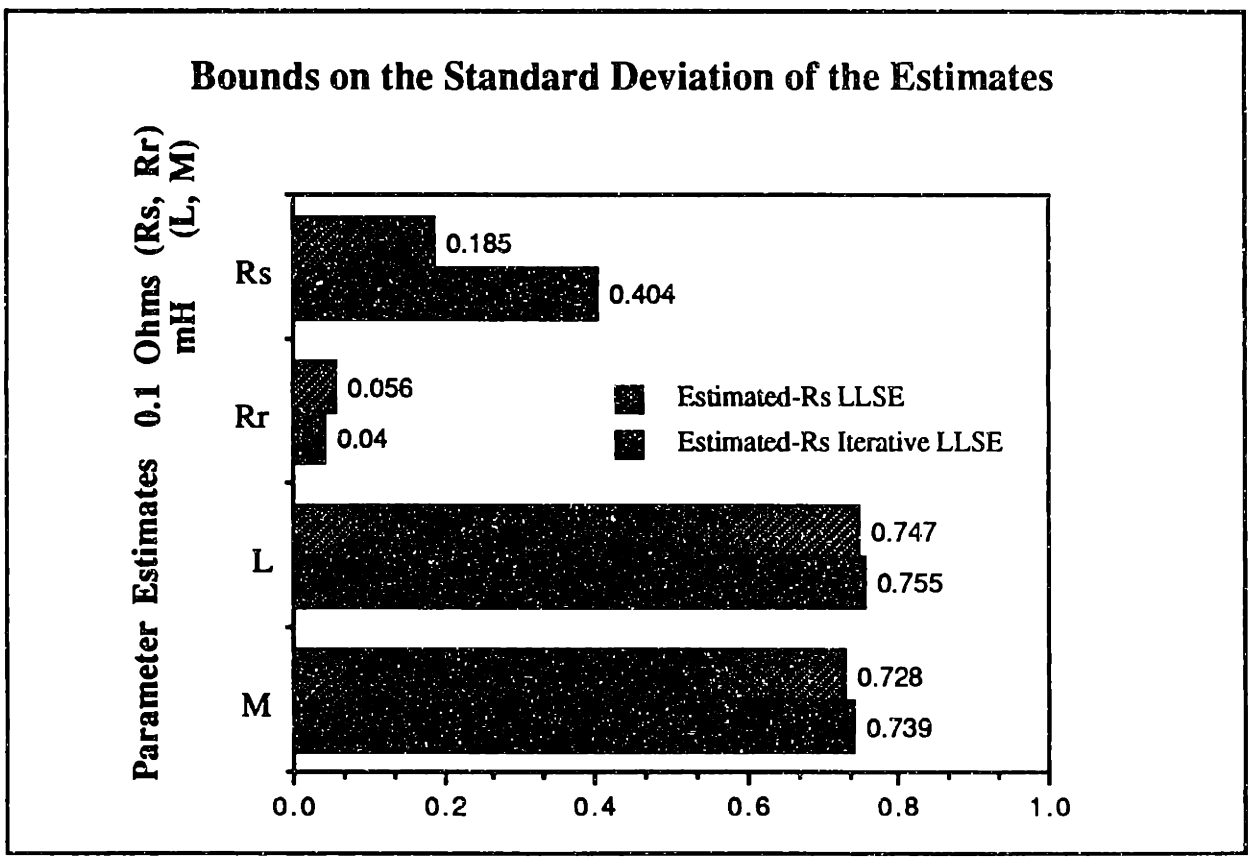


Figure 3.3: Bounds on the standard deviation of the parameter estimates for error deviation in Table 3.2.

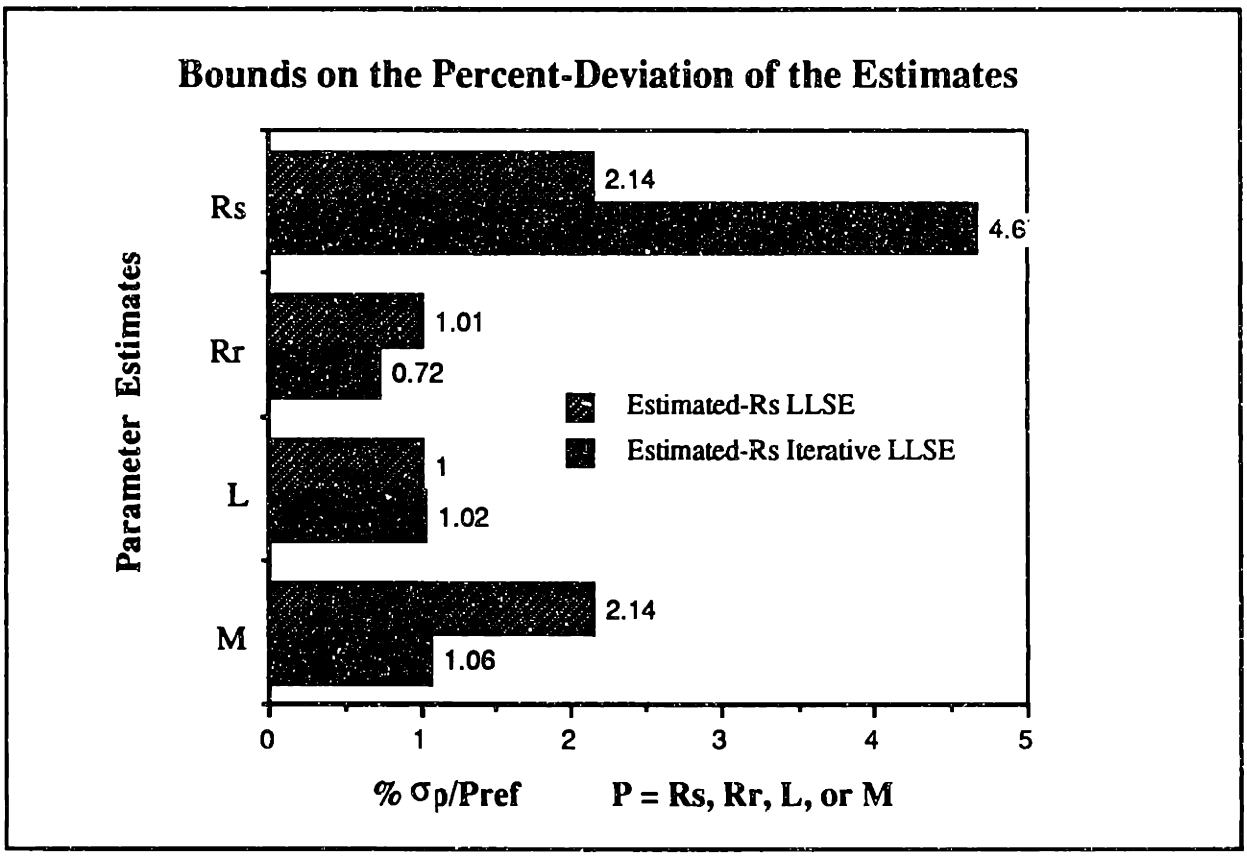


Figure 3.4: Bounds on the percent-deviation of parameter estimates for error deviations in Table 3.2.

Chapter 4

Constant Temperature Experimental Results

4.1 Introduction

In this chapter, we experimentally study the performance of the three estimators developed in Chapter 3. The experimental conditions are restricted to steady-state constant-temperature motor operation. In this chapter we apply the three estimators developed in Chapter 3. The experiments involve three rotors. Rotor 1 and rotor 3 are healthy while rotor 2 has a broken bar. In each of the test experiments, measurement data is collected at 16 different motor slips and then is processed by each of the three estimators to yield the estimates of R_r and R_s. The expected values and the standard deviations of these estimates are computed and examined to determine the success of broken-rotor-bar detection for each estimator.

The experimental results are important, because they justify the theories developed in Chapter 3. In general, the experimental results correspond closely to the results of Chapter 3. From the physical experiments conducted in this chapter, several important conclusions

arise. Firstly, the stator resistance R_s is poorly estimated as is predicted in the sensitivity Section 3.5. Thus R_s must either be measured or be estimated more accurately by some other method. Secondly, LLSE estimator with known R_s and the iterative LLSE estimator with estimated R_s both detect the broken bar in rotor 2, although the estimator with known R_s detects it much more clearly than the other. Finally, for the LLSE estimator with estimated R_s, the detection of a broken rotor bar is not clear; its success is arguable at best.

The chapter is organized as follows. Section 4.2 discusses the experimental procedure for constant temperature tests which have been conducted with the experimental system described in Section 2.1. This section discusses the maintenance of the constant temperature condition, the measurement of R_s, and the collection of the experimental data. Section 4.3 then presents and discusses the experimental results of the three estimators.

4.2 Experimental Procedure

4.2.1 Maintaining a Constant Temperature Operating Condition

To simplify matters, the experiments reported here are set up so that thermal variations of R_r are not an issue. A way to remove the effects of temperature variation is to bring the induction motor to a steady-state temperature for all tests. Since the resistive power dissipations are the main source of temperature for the induction motor, the motor should come to the same steady-state temperature for each experiment as long as the power dissipations are kept constant. This is easily achieved by keeping the stator voltage and the mechanical power, P_m, at some predetermined values until the targeted steady-state

temperature is reached. The thermocouples are used to determine when this condition has been reached.

In this chapter, the stator voltage $|v_s|$ is held constant at 120V rms line-to-neutral and P_m as read by the dynamometer is held at approximately 700W; the actual power is somewhat higher as described in Appendix B. The choice of P_m = 700W is limited in the experimental system by the amount of heat which can be removed from the dynamometer and not by some property of the motor itself. Hence, even though the motor can operate at higher P_m, the mechanical power is limited to 700W when there is no cooling system for the dynamometer. For the experiments conducted in Chapter 5 we needed sufficient richness in the measurements of motor temperatures, so the limit on P_m is subsequently relaxed by installing an air-cooling system for the dynamometer. With the aid of the air-cooling system, P_m is raised safely to 1.7kW. In order to determine when the steady-state temperature condition has been reached, thermocouples TC_2, TC_{16}, and TC_{22} are monitored until they indicate constant temperature readings; see Figure 2.2 and Table 2.4.

To illustrate the thermal time constants of the experimental induction motor, Figure 4.1 shows a graph of the temperatures in degrees Celsius above the ambient temperature at TC_2, TC_{16}, and TC_{22}. For this graph, $|v_s|$ is 120V, P_m is 1500 watts, and the ambient temperature, T_{amb}, is 24 °C. This experiment is the only one in this chapter in which P_m is not 700W. It therefore has been conducted after the installation of the air-cooling system for the dynamometer. Note that although this experiment has been performed with P_m = 1500W, as opposed to the standard of 700W, the thermal time constants should be nearly equal to that of 700W operation. However, because the temperatures at different points of the motor are larger at P_m = 1500W than at P_m = 700W, the thermal time constants can thus be illustrated more clearly with P_m = 1500W. Recall from Chapter 2 that the computer gives a digital display of the thermocouple temperature measurements relative to the

ambient temperature. Notice that all the thermal equilibrium is reached in about 75 minutes. To ensure that the motor reaches constant temperature condition, the induction motor is always left running for 90 minutes before each set of tests are conducted.

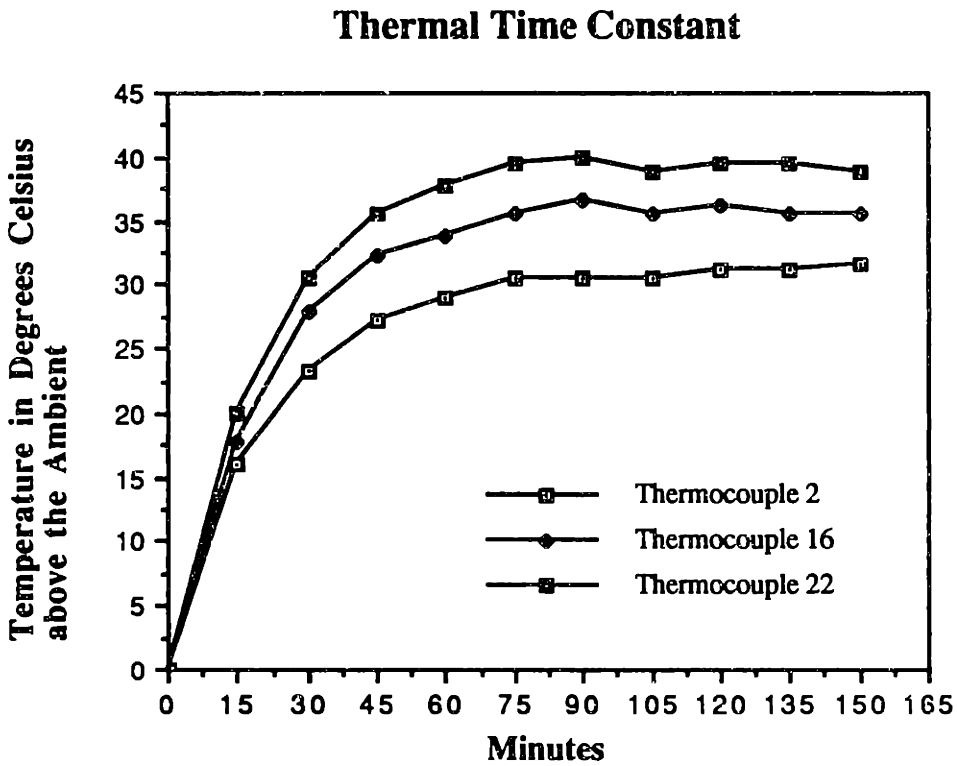


Figure 4.1: Thermocouple temperature versus, $|v_s| = 120V$, $P_m = 1500W$, $T_{amb} = 24°C$.

After 90 minutes, with the mechanical load at 700W, the thermocouple readings are: $TC_2 = 18$ °C, $TC_{16} = 21$ °C, and $TC_{22} = 22$ °C above the ambient temperature of 24 °C. Since it is crucial to be at the same steady-state motor temperature before and during each data collection, the thermocouples TC_2, TC_{16}, and TC_{22} are checked to make sure that they read approximately the same absolute temperature of $TC_2 = 42$ °C, $TC_{16} = 45$ °C, and $TC_{22} = 46$ °C before and throughout the data collection process. As discussed earlier, this requires waiting approximately 75 to 90 minutes after turning the motor on.

4.2.2 Measurement Procedure

Once we are confident that the motor is operating at a steady-state constant-temperature condition, we can proceed with the collection of data. From Chapter 3, we know that for the estimators with estimated R_s, the measurements of $|i_s|$, $|v_s|$, Ω, and Φ must be made, and for the estimator with known R_s, the measurements of $|i_s|$, $|v_s|$, Ω, Φ, and R_s must be made. For the measurement of R_s, we have for the purpose of estimator development assumed that a method of on-line measurement of R_s is available [32]. In Chapter 5, an alternative way of obtaining accurate estimates of R_s based on the thermal model is discussed.

Instead of implementing the instrumentation for the non-intrusive measurement of R_s as it is described in [32], R_s is measured off-line using the setup shown in Section 2.1. This measurement process for R_s required that the motor be temporarily shut down. However, once the motor reaches the constant temperature condition, the measurements of R_s must be made quickly enough, typically within 10 to 20 seconds, so that the thermocouple readings do not indicate a drop in temperature of more than 1°C from their steady-state values. Once the measurements are made, the motor is restarted and left alone until the constant temperature condition prevails. Since the resistance of the stator is small, on the order of 0.8 ohms, a typical ohmmeter cannot provide sufficient accuracy. Therefore, a DC supply is connected to the stator winding and both a voltmeter and an ammeter are used to calculate the stator resistance.

Once the motor reaches its steady-state constant-temperature operation again, sets of measurements ($|i_s|$, $|v_s|$, Ω, Φ) are made at different slips to ensure that these sets of

measurements are independent. Enough independent sets of measurements are collected, so that the measurement matrix, A in Equation (2.11) or A_4 (3.17) is overconstrained; if A or A_4 is not overconstrained, the estimations cannot work. Here, we collect 16 independent sets of measurements. In each of the tests, we decrement the rotor speed Ω from 1795 to 1720 rpm in steps of 5 rpm while holding the stator voltage nominally at 120V. Recall that each set of measurements gives rise to two independent equations, one for the real part of the system function of the single-phase equivalent circuit, and the other for the imaginary part. Therefore, in all, the collection of 16 independent sets of measurements gives rise to 32 independent equations or row vectors for the measurement matrix A or A_4, which more than satisfies the necessary condition of overconstraining A or A_4. Once A or A_4 is overconstrained, the LLSE estimators can be used to estimate the parameter values.

One critical point is that each excursion from the original Ω that corresponds to P_m of 700W to a different Ω must be brief. This is necessary to keep from violating the steady-state temperature condition. The danger is that if the data collection at a different Ω and hence, different P_m, takes too long, the motor will deviate from its original constant temperature state to another constant temperature state. Again, to make sure that the readings are taken at the same constant temperature condition, the thermocouples TC_2, TC_{16}, and TC_{22} are monitored throughout the data collection procedure. The measurements are considered valid if and only if they are obtained at the same thermocouple temperatures as those at the original steady-state constant-temperature condition of the motor. If the thermocouple temperatures deviate from their corresponding steady-state temperatures at any time during the data collection, the mechanical loading is set back to 700W and this point is maintained until the motor returns to the original steady-state temperature.

As long as the other measurements, $|i_s|$, $|v_s|$, Ω, Φ, are collected under the constant temperature condition, the value of R_s is kept constant also. Hence, only a measurement of R_s in the beginning of each test is needed. However, to double-check, at end of each test, R_s must be measured again and compared to the measurement made at the beginning of the test. In general, the two measurement of R_s have been kept close to each other.

4.3 Experimental Results

In order to determine whether a change in the apparent rotor resistance is sufficient to conclude that a rotor bar has been broken, we used a single stator and tested three rotors of the same type. Rotors 1 and 3 have no broken bars while rotor 2 has one broken bar; see Table 2.2. The reason for testing two healthy rotors is to investigate the effects of variations in the rotors due to the manufacturing process.

Each rotor is tested four times. As described in the last section, each test is conducted under the constant temperature condition of $|v_s| = 120$V and $P_m = 700$W, and each test comprises 16 independent sets of measurements taken at 16 different slips. From each test, three estimates for the rotor resistance are obtained numerically using MatrixX programs; see Appendices C, F, and G. These estimates correspond to LLSE estimation with estimated R_s, iterative LLSE estimation with estimated R_s, and LLSE estimation with known R_s. Furthermore, the measurements from each test are used to produce two estimates of R_s that correspond to LLSE and iterative LLSE estimations.

The experimental results of R_r and R_s estimates using different estimators are shown in Figure 4.2, 4.3, and 4.4 for the R_r estimates and Figure 4.5 and 4.6 for the R_s estimates. The results in each figure correspond to a particular estimator. In each figure, the numbers

"1", "2", and "3" locate the rotor or the stator resistance estimated from independent measurements taken with one of three different rotors. The numbers "$\overline{1}$", "$\overline{2}$", and "$\overline{3}$" locate the average estimate of the rotor or the stator resistance for each rotor. The standard deviation of the rotor or the stator resistance estimates for each rotor is indicated by horizontal line segments on both sides of the average of the estimates. Tables 4.1, 4.2, 4.3, 4.4, and 4.5 provide the numerical values for the experimental results shown in Figures 4.2, 4.3, 4.4, 4.5,and 4.6 respectively. In the tables, the average of the estimates is denoted by "Ave$\{\hat{R}_r\}$" or "Ave$\{\hat{R}_s\}$" and the experimental standard deviation of the estimates by the symbol σ_{exp}.

Each table provides two additional pieces of information not found in each figure. An estimate labeled "$\hat{R}_r$ from average data" or "$\hat{R}_s$ from average data" is obtained for each rotor by first averaging all the measurement data matrices for the four tests and then producing estimates of R_r or R_s based on this averaged data matrix. None of averaged-data estimates are shown in the figures, since they closely approximate Ave$\{\hat{R}_r\}$ or Ave$\{\hat{R}_s\}$. The other information is the numerically simulated bound on the standard deviation, labeled σ_{num}, from Figure 3.3 in the last chapter.

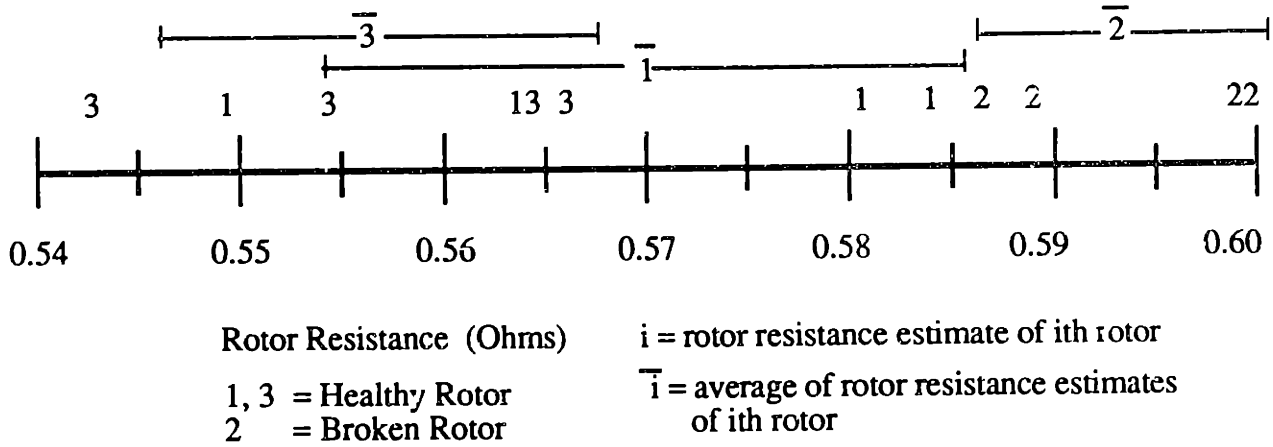


Figure 4.2: Experimental results of R_r estimates, LLSE with estimated R_s.

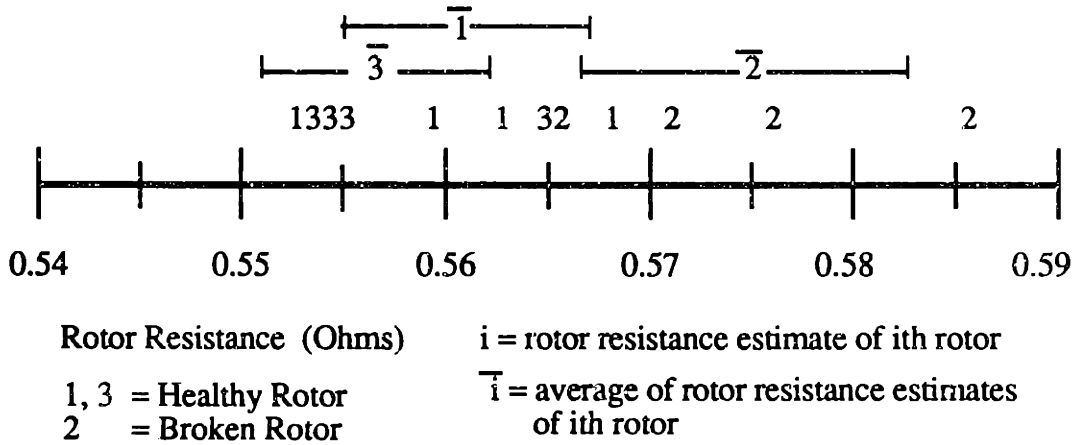


Figure 4.3: Experimental results of R_r estimates, iterative LLSE with estimated R_s.

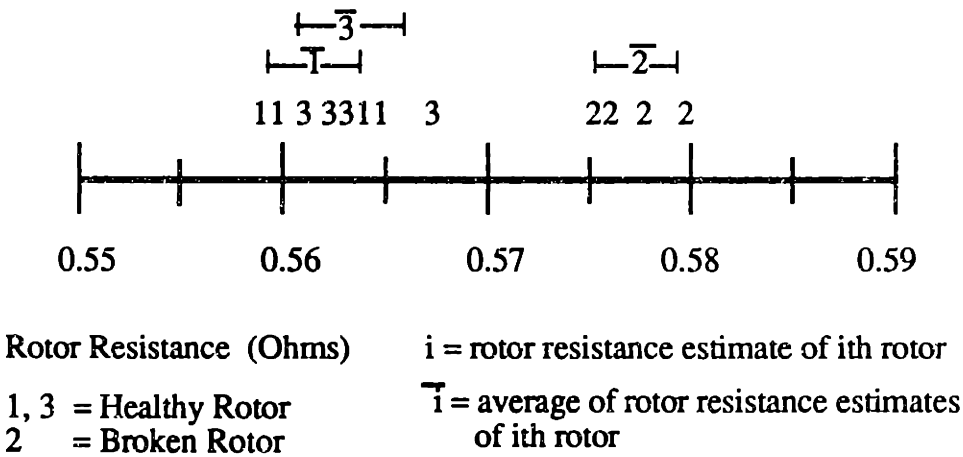


Figure 4.4: Experimental results of R_r estimates, LLSE with known R_s.

Rotor Resistance Estimates (Ohms)			
	Rotor 1	Rotor 3	Rotor 2
Estimate 1	0.565	0.543	0.600
Estimate 2	0.585	0.567	0.588
Estimate 3	0.580	0.554	0.587
Estimate 4	0.549	0.565	0.600
Ave$\{\hat{R}_r\}$	0.570	0.557	0.594
$\hat{R}_r$ from averaged data	0.568	0.556	0.592
σ_{exp}	0.016	0.011	0.007
σ_{num}	0.008	0.008	0.008

Table 4.1: Experimental data for R_r estimates, LLSE with estimated R_s.

Rotor Resistance Estimates (Ohms)			
	Rotor 1	Rotor 3	Rotor 2
Estimate 1	0.553	0.553	0.571
Estimate 2	0.559	0.554	0.576
Estimate 3	0.568	0.555	0.565
Estimate 4	0.563	0.565	0.586
Ave$\{\hat{R}_r\}$	0.561	0.557	0.575
$\hat{R}_r$ from averaged data	0.561	0.557	0.574
σ_{exp}	0.006	0.006	0.009
σ_{num}	0.021	0.021	0.021

Table 4.2: Experimental data for R_r estimates, iterative LLSE with estimated R_s.

Rotor Resistance Estimates (Ohms)			
	Rotor 1	Rotor 3	Rotor 2
Estimate 1	0.559	0.562	0.575
Estimate 2	0.559	0.561	0.580
Estimate 3	0.564	0.563	0.575
Estimate 4	0.564	0.567	0.578
Ave$\{\hat{R}_r\}$	0.562	0.563	0.577
$\hat{R}_r$ from averaged data	0.561	0.563	0.577
σ_{exp}	0.003	0.003	0.002
σ_{num}	0.011	0.011	0.011

Table 4.3: Experimental data for R_r estimates, LLSE with known R_s.

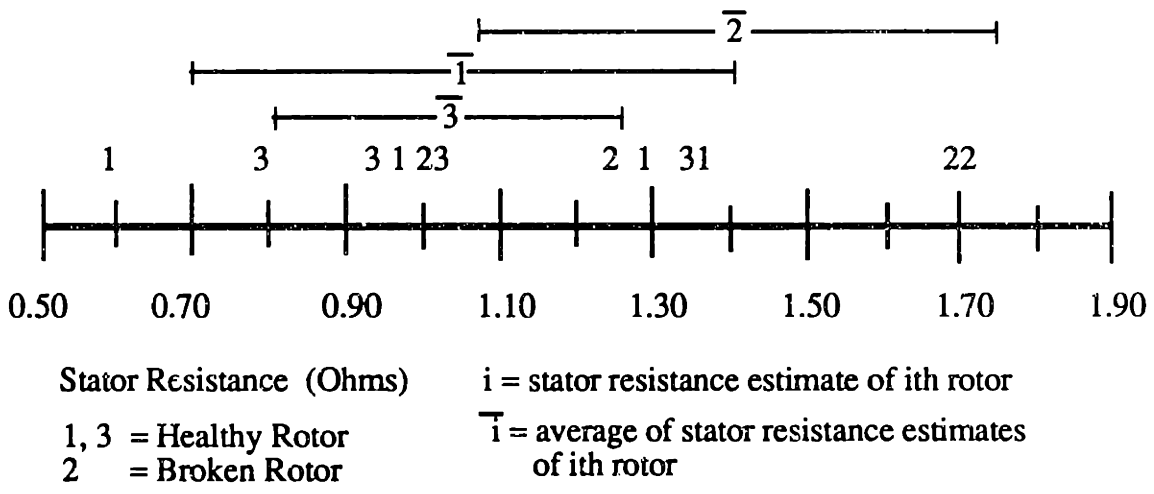


Figure 4.5: Experimental results of R_s estimates, LLSE, measured $R_s = 0.864\ \Omega$.

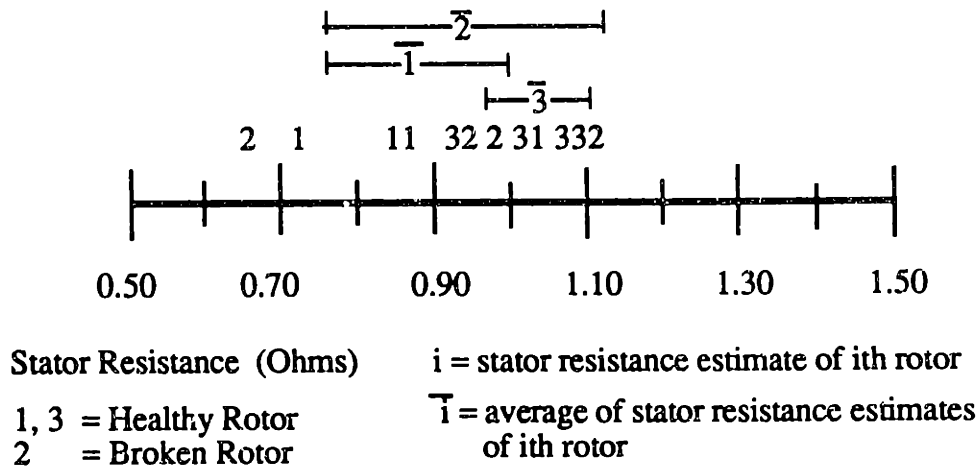


Figure 4.6: Experimental results of R_s estimates, iterative LLSE, measured $R_s = 0.864\ \Omega$.

Stator Resistance Estimates (Ohms)			
	Rotor 1	Rotor 3	Rotor 2
Estimate 1	1.296	0.798	1.691
Estimate 2	1.362	1.346	1.247
Estimate 3	0.960	1.067	1.692
Estimate 4	0.587	0.924	1.001
Ave$\{\hat{R}_s\}$	1.051	1.034	1.408
$\hat{R}_s$ from averaged data	1.020	1.018	1.377
σ_{exp}	0.356	0.235	0.343
σ_{num}	0.643	0.643	0.643

Table 4.4: Experimental data for R_s estimate, LLSE, measured $R_s = 0.864\ \Omega$.

Stator Resistance Estimates (Ohms)			
	Rotor 1	Rotor 3	Rotor 2
Estimate 1	1.044	1.091	0.986
Estimate 2	0.852	1.028	0.953
Estimate 3	0.717	1.072	1.117
Estimate 4	0.880	0.928	0.659
Ave$\{\hat{R}_s\}$	0.873	1.030	0.929
$\hat{R}_s$ from averaged data	0.879	1.032	0.932
σ_{exp}	0.134	0.073	0.193
σ_{num}	0.267	0.267	0.267

Table 4.5: Experimental data for R_s estimate, iterative LLSE, measured $R_s = 0.864\ \Omega$.

The first thing to notice from the experimental results is that, to some degree, all three estimators can detect a broken rotor bar. However, the LLSE with known R_s shows superior broken rotor bar detection over the other two estimation schemes. The others, in particular the iterative LLSE estimator and arguably the LLSE estimator with estimated R_s, seem to detect a broken rotor bar as well, but not as unambiguously as the LLSE estimation with known R_s. Note that by testing rotors 1 and 3 an additional good rotor is tested so that the variation of parameters due to manufacturing process is also taken into account. As we have postulated, all three estimators generally produced increased estimates of R_r for

rotor 2 over those of the other two healthy rotors. The only minor exception is one stray rotor 2 estimate for the iterative LLSE estimated R_s case. Certainly, if only Ave$\{\hat{R}_r\}$ is considered, all three estimators produce increased estimates for the broken rotor, rotor 2, as compared to those of the two healthy rotors, rotor 1 and rotor 3. Note also that the estimated increase is on the order of 2.5%, which is consistent with breaking one bar out of 45. In conclusion though, the LLSE estimator with known R_s is clearly the best choice for detection of broken rotor bars. For the purpose of an unambiguous detection of a broken rotor bar, an important conclusion is that we must either measure R_s or develop an alternate and more accurate way of estimating it.

The tables and the figures indicate that the magnitude of σ_{exp} is on the same order as the magnitude of σ_{num}. However, the order of the estimators in which the standard deviations increase is different for the experimental and numerical cases. This is not surprising, since there are other sources of estimation error in addition to measurement noise that the sensitivity analysis is concerned with. For example, as we have mentioned, modelling error is not taken into account in the sensitivity analysis. One experimental result is clear, though. Note that the lowest σ_{exp} corresponds to the LLSE estimator with measured R_s. Therefore, we again expect the LLSE estimator with known R_s to produce the best R_r estimates.

For both estimators that estimate R_s, the estimates are poor in that they are much larger than the actual measured value of R_s. This demonstrates that R_s cannot be estimated well experimentally as expected in Chapter 3. The experimental and the numerically-bound standard deviation of R_s estimates on the other hand, are in reasonable agreement. Both σ_{exp} and σ_{num} for the R_s estimates decreases by a factor of about three as we go from the LLSE estimator to the iterative LLSE estimator. As hypothesized in Chapter 3, iterative LLSE estimator should produce better estimates as compared to just the LLSE estimator,

since an additional dependence among the parameters x_1, x_2, and x_5 of (2.12) is taken into account by the iterative estimator. Comparing further, the values for σ_{exp} for R_s are approximately half the corresponding values of σ_{num}. This increase in accuracy from the numerically bound case to the experimental case is to be expected because the numerical bounds are conservative.

Chapter 5

Temperature Compensation

5.1 Introduction

In Chapter 4, a broken-rotor-bar detector is successfully demonstrated while the induction motor operated at constant temperature. Such a detector has a limited scope; we must detect broken rotor bars over the entire temperature range associated with motor operation. This poses a new problem for the broken-rotor-bar detector described in Chapter 3 because a typical variation in rotor temperature can cause a significant variation in the rotor resistance. A thermally induced variation in rotor resistance can thus be misinterpreted as a broken rotor bar.

One solution to the detection problem posed by thermal variation in rotor resistance is to first determine reference rotor resistances of "healthy" rotors for every operating condition, and then compare all estimates of rotor resistance to these references so as to examine rotor-bar breakage. This solution requires an enormous data base, and seems unnecessarily cumbersome. If we can instead incorporate a thermal model into the detector to thermally compensate its estimates of rotor resistance, then the detection of broken rotor bars at any

arbitrary operation conditions of the induction motor is possible without an extensive data base. The effect of temperature on rotor resistance is thus taken into account and factored out by the thermal model. This chapter then has two goals: to develop a temperature-compensation method for the estimates of R_r and to experimentally demonstrate the thermally-compensated detection of broken rotor bars.

Section 5.2 discusses the foundation of temperature compensation, which is the relation between the resistivity of a material and its temperature. This relation is then generalized to describe the resistance of volumes of materials with non-constant temperature profile. This is necessary, since we are interested in rotor resistance rather than rotor resistivity. Chapter 3 provides a method of estimating the rotor resistance, but the rotor temperature must be known in order to use this method. This problem of circular reasoning is solved in Section 5.3. Section 5.3 outlines a method of obtaining estimates of the rotor temperature. This necessitates the development of a thermal model and the estimation of its parameters. Once the rotor temperature estimates are known, they can be used with the rotor resistance estimates to produce resistance estimates that correspond to a reference rotor temperature. This compensation is carried out via the relation developed in Section 5.2. The experimental thermally-compensated estimates of R_r for rotors 1, 2, and 3 are shown and discussed in Section 5.5. All experiments uses LLSE estimation with known R_s, and all estimates of R_r are transformed to estimates corresponding to the load of 700W as measured by the dynamometer. The thermally compensated estimates of R_r for rotor 2, the broken rotor, are located well above the range of the estimates of R_r for the two healthy rotors, rotor 1 and rotor 3. Thus, a successful thermally-compensated detection of the broken rotor bars is demonstrated.

As a by-product of the development of the thermal model and temperature compensation, an effective method of estimating R_s is also developed and discussed in

Section 5.4. In Section 5.5, the experimental estimates of R_s based on the thermal model are shown to be virtually identical to the measured R_s values. In effect, this demonstrates that in terms of the R_r estimation, there is no difference between using either the temperature-based R_s estimation or measurements of R_s. This is verified experimentally in Section 5.5. The important conclusions are that the thermally-compensated detection of a broken rotor bar is successful and that the recommended estimation scheme for the rotor resistance is that of LLSE estimation with thermally based R_s estimates.

5.2 Temperature-Resistance Relation

This section examines the relationship between the resistivity and the temperature of a conductor. This relationship is then generalized to one between the resistance of a conductor and its temperature, since the single-phase equivalent circuit deals with rotor resistance. For the induction motor used experimentally in this thesis, the rotor bars are made of aluminum. Since these bars give rise to the rotor resistance, knowing the dependence of the resistance of aluminum conductors on the temperature should provide sufficient information to compensate for thermal variations in the estimate of R_r. The stator windings, which are considered later, are made of copper.

In general, the relationship between the resistivity ρ and the temperature T of many conductors is governed by

$$\rho(T_{ref}) = \rho(T) \frac{T_{ref} + T_0}{T + T_0} \tag{5.1}$$

over some range of temperature in the neighborhood of T_{ref} [35,36]. For both aluminum and copper, this range extends beyond the typical temperatures experienced inside induction motors. In (5.1),

$$T_0 = \begin{cases} 228.1\ ^\circ\text{C for aluminum} \\ \\ 234.5\ ^\circ\text{C for copper} \end{cases}, \tag{5.2}$$

T_{ref} is an arbitrary conductor temperature in °C, and T is the conductor temperature also in °C; see Appendix A. Given T_0 and the resistivity of the conductor at temperature T, $\rho(T)$, Equation 5.1 rescales $\rho(T)$ to $\rho(T_{ref})$, the resistivity at temperature T_{ref}. This is the basis for temperature compensation.

Given an aluminum conductor such as the rotor bars, a temperature rise from 20°C to 100°C results in a 34% increase in resistivity. Such an increase dominates the equivalent increase caused by a broken rotor bar. In Chapter 4, it is observed that a broken bar changed the estimated rotor resistance by approximately 2%. Thus, the thermal variation in rotor resistance must be compensated if the broken-rotor-bar detector is to be successful.

Equation 5.1 is concerned with material property of resistivity, rather than resistance. We, therefore, need to generalize Equation 5.1 to resistances, since our single-phase equivalent circuit deals with stator and rotor resistances, not resistivities. In general,

$$R(T(x)) = \int_{C(x)} \frac{\rho(T(x))}{A(x)}\, dx \tag{5.3}$$

where the resistance R is an integral of the temperature-dependent resistivity $\rho(T)$ divided by the cross-section area $A(x)$ of the conductor along the path $C(x)$ which defines the resistor. The variable of integration x is a spatial variable defined along the path C. Note that the resistivity ρ is a function of temperature, which is in turn a function of x. Rearranging (5.1) and substituting it into (5.2) yields

$$R(T(x)) = \int_{C(x)} \frac{\rho(T_{ref})}{A(x)} \left[\frac{T(x) + T_0}{T_{ref} + T_0} \right] dx \ . \tag{5.4}$$

Equation (5.4) can be rewritten as

$$R(T(x)) = \frac{\rho(T_{ref})}{T_{ref} + T_0} \int_{C(x)} \frac{T(x)}{A(x)} \, dx + \frac{T_0}{T_{ref} + T_0} \int_{C(x)} \frac{\rho(T_{ref})}{A(x)} \, dx \ . \tag{5.5}$$

Since

$$\int_{C(x)} \frac{\rho(T_{ref})}{A(x)} \, dx = R(T_{ref}) \ , \tag{5.6}$$

Equation (5.5) becomes

$$R(T(x)) = \frac{\rho(T_{ref})}{T_{ref} + T_0} \int_{C(x)} \frac{T(x)}{A(x)} \, dx + \frac{T_0}{T_{ref} + T_0} R(T_{ref}) \ . \tag{5.7}$$

Finally, rewriting (5.7), yields

$$R(T(x)) = R(T_{ref})\left(\frac{\widetilde{T}}{T_{ref}+T_0} + \frac{T_0}{T_{ref}+T_0}\right) = R(T_{ref})\frac{\widetilde{T}+T_0}{T_{ref}+T_0} , \tag{5.8}$$

or

$$R(T_{ref}) = R(T(x))\frac{T_{ref}+T_0}{\widetilde{T}+T_0} , \tag{5.9}$$

where

$$\widetilde{T} = \frac{\int_{C(x)} \frac{T(x)}{A(x)}\,dx}{\int_{C(x)} \frac{dx}{A(x)}} \equiv T_{ref} + \Delta T . \tag{5.10}$$

Even though the conductor has a distributed temperature profile, for the present purposes, it can be thought of as a conductor with a lumped-parameter temperature $\widetilde{T} \equiv T_{ref} + \Delta T$. This lumped-parameter temperature $\widetilde{T}$ is actually a weighted-average temperature over the conductor. In this way, the thermal equation (5.1) is generalized to resistances.

Equation (5.9) provides a means for compensating for the thermal variation in rotor resistance. If the weighted-average rotor temperature, $\widetilde{T}$, is known, then the estimated rotor resistance can be transformed using (5.9) to an estimated resistance at a reference temperature, thereby providing compensation. To determine $\widetilde{T}$, the broken-rotor-bar detector is expanded to include weighted-average temperature estimation based on a thermal model of the induction motor. The estimated rotor temperature is then used in (5.9). From here on, the weighted-average temperature of the rotor is simply referred to as the rotor

temperature and, likewise, the weighted-average temperature of the stator winding is simply referred to as the stator winding temperature.

5.3 Thermal Model

To use the relation between resistance and temperature derived in Section 5.2 for compensation, the rotor temperature must be known. This chapter develops a means of estimating this rotor temperature using a thermal model. The accuracy of the rotor-temperature estimation is then verified using temperature measurements provided by thermocouples located at various points of the motor; see Chapter 2 for a discussion of the thermocouples. Finally, the temperature-compensation of estimated R_r is described in full. Experimental results of the temperature-compensated R_r estimation are presented in Section 5.5.

The rotor temperature must be estimated rather than measured because it is difficult to implement an effective method of measuring the rotor temperature directly. Even if we could attach a thermocouple, or even several thermocouples, the readings from these only describe point temperatures. It is unclear how these readings can be transformed to yield the lumped temperature $\widetilde{T}$ described in Section 5.2 except through an appropriate estimation scheme. Since the rotor temperature is estimated and not directly measured, the accuracy of the rotor-temperature estimates must be evaluated indirectly using the temperature estimates of thermocouples at various points of the stator. Experimentally, as is discussed later, the thermal model predicts the thermocouple temperatures all to within 2°C of the actual temperature readings. Thus, a successful estimation of the rotor temperature can be inferred. As a final test of the success, Section 5.5 demonstrates both an accurate estimation of the stator resistance based on the thermal model and successful temperature-

compensated detection of broken rotor bars. Both the thermal-based estimation of R_s and temperature-compensated estimation of R_r depend on good estimations of the rotor temperature. Hence, the positive results in Section 5.5 ultimately suggest a successful estimation of the rotor temperature.

In order to estimate the rotor temperature, a thermal model is developed here. The thermal model relates the increase in the rotor temperature to a linear combination of heat sources and their corresponding equivalent thermal resistances. The thermal model allows estimation of the thermal resistances via LLSE estimation operating on data specifically collected for this purpose. Once these thermal resistances are known, the rotor temperature can be estimated at any arbitrary steady-state motor operation.

To simplify the thermal model, linear steady-state thermal operation of the motor is assumed. This assumption of linearity is justified experimentally with thermocouple measurements later in this section. The steady-state assumption is somewhat limiting, but could be lifted with a more involved dynamic model. The development of a dynamic model is left for future work. Three sources of heat are included in the model. These are the total dissipation in the stator winding, P_s, the total dissipation in the rotor bars, P_r, and the total core loss, P_c. Given these assumptions, the temperature rise of the rotor, or the temperature rise at any point in the induction motor, is given by

$$\Delta T = P_s\Theta_s + P_r\Theta_r + P_c\Theta_c \tag{5.11}$$

where ΔT is the temperature rise of the point in question above the ambient temperature, as defined in (5.11), and each Θ is an appropriate thermal resistance. The thermal resistances and ΔT are functions of space for point-wise interpretation of ΔT but are lumped-parameters for the weighted-average interpretation of ΔT. The dissipations are totals, and

so are independent of space. Since the total core loss occurs primarily in the stator, we arbitrarily simplify (5.10) to

$$\Delta T = P_r \Theta_r + (P_s + P_c)\, \Theta_s \ . \tag{5.12}$$

Equation (5.12) is used instead of (5.11) for one main reason. It turns out that the use of (5.11) yields negative values for the estimates of some of the thermal resistances while (5.12) produces all positive values. The troublesome implication of the negative estimates of thermal resistances produced by (5.11) is that under certain conditions, the points in the induction motor that are associated with these negative thermal resistances actually take away heat rather than generate it which is a physical impossibility. Even though this is counter to physical argument, the use of these estimated thermal resistances nevertheless provides estimates of the thermocouple temperatures that are just as good as those estimated from (5.12). Since, (5.11) and (5.12) are equally effective in estimating temperatures in the motor, we choose (5.12) because it also satisfies our physical understanding of the induction motor.

To justify the use of (5.12), the induction motor is studied experimentally. As shown in Figure 2.2 of Chapter 2, the motor is fitted with 25 thermocouples. The induction motor is then operated at constant a $|v_s|$ of 120 volts and ω_e of 120π rad/sec, with loads of 250W, 700W, 1.2kW, and 1.7kW as read by the dynamometer. Next, the motor is allowed to come to thermal equilibrium, and for each operating condition, ΔT from each thermocouple, R_s, $|v_s|$, $|i_s|$, Ω, Φ, and the mechanical torque are measured. Care must be taken to add the pre-measured bearing and windage torques, collectively denoted as $\tau_{m,loss}$ to the measured mechanical torque $\tau_{m,meas}$ to produce the total mechanical torque τ_m. Since the loading P_m measured by the dynamometer is computed from $\tau_{m,meas}$ alone, we must be careful not to associate this measured value to the total mechanical loading. There are other

loads from bearings and windage that are not registered in $\tau_{m,meas}$. Finally, the measurements are used to justify and establish the accuracy of (5.12). By assumption, if the thermal model works well for points in the stator, then it works well for the rotor also.

For each operating condition, ΔT and $P_s = 3|i_s|^2 R_s$ can be computed directly from the measurements. Further, $P_r = 3|i_r|^2 R_r$ can be computed from (2.?) to be

$$P_r = 3|i_r|^2 R_r = \frac{s}{(1-s)}\tau_m \omega_m = \frac{s}{(1-s)}(\tau_{m,meas} + \tau_{m,loss})\omega_m \quad , \tag{5.13}$$

where $\tau_{m,loss}$ = 0.3796 N-m; see Appendix B. P_c is a core loss function, which is dependent on the $|v_s|$ and ω_e. To determine P_c, the peak magnetic flux density in the stator core is approximated given $|v_s|$. Using this flux density, the known ω_e, and the known stator core volume, P_c is determined from core-loss specifications for the steel from which the induction motor is manufactured. Thus, P_c for the induction motor is computed to be 61.1 Watts. In this way, Θ_s and Θ_r are the only unknowns in (5.12).

To justify the linearity of equation (5.12), the thermal resistances Θ_s and Θ_r are first estimated with four sets of measurements at mechanical loads of 250W, 700W, 1.2kW, and 1.7kW as measured by the dynamometer. The estimated thermal resistances are then used to blindly estimate the thermocouple temperatures at three other loads. Finally, the measured and the estimated thermocouple temperatures at each of these three loads are compared to check the validity of (5.12).

To estimate the thermal resistances Θ_{si} and Θ_{ri} for any thermocouple TC_i listed in Table 2.2 and shown in Figure 2.2, (5.12) is transformed to the near-LLSE estimation problem of the form

$$\begin{bmatrix} \Delta T_{i1} \\ \Delta T_{i2} \\ \Delta T_{i3} \\ \Delta T_{i4} \end{bmatrix} = \begin{bmatrix} \frac{s_1}{(1-s_1)}(\tau_{m,meas1}+\tau_{m,loss})\omega_{m1} & 3|i_{s1}|^2R_{s1}+P_c \\ \frac{s_2}{(1-s_2)}(\tau_{m,meas2}+\tau_{m,loss})\omega_{m2} & 3|i_{s2}|^2R_{s2}+P_c \\ \frac{s_3}{(1-s_3)}(\tau_{m,meas3}+\tau_{m,los})\omega_{m3} & 3|i_{s3}|^2R_{s3}+P_c \\ \frac{s_4}{(1-s_4)}(\tau_{m,meas4}+\tau_{m,loss})\omega_{m4} & 3|i_{s4}|^2R_{s4}+P_c \end{bmatrix} \begin{bmatrix} \Theta_{ri} \\ \Theta_{si} \end{bmatrix}. \tag{5.14}$$

The subscripts "1", "2", "3", and "4", correspond to the data collected at the dynamometer loads 250W, 700W, 1.2kW, and 1.7kW, respectively. This can be summarized by $\bar{y}_i = \mathbf{A}\bar{x}_i$ where

$$\mathbf{A} = \begin{bmatrix} \frac{s_1}{(1-s_1)}(\tau_{m,meas1}+\tau_{m,loss})\omega_{m1} & 3|i_{s1}|^2R_{s1}+P_c \\ \frac{s_2}{(1-s_2)}(\tau_{m,meas2}+\tau_{m,loss})\omega_{m2} & 3|i_{s2}|^2R_{s2}+P_c \\ \frac{s_3}{(1-s_3)}(\tau_{m,meas3}+\tau_{m,loss})\omega_{m3} & 3|i_{s3}|^2R_{s3}+P_c \\ \frac{s_4}{(1-s_4)}(\tau_{m,meas4}+\tau_{m,loss})\omega_{m4} & 3|i_{s4}|^2R_{s4}+P_c \end{bmatrix},$$

$$\bar{x}_i = [\Theta_{ri}\ \Theta_{si}]^T,$$

and

$$\bar{y}_i = [\Delta T_{i1}\ \Delta T_{i2}\ \Delta T_{i3}\ \Delta T_{i4}]^T.$$

Again, this estimation problem is only nearly a LLSE estimation problem because of the uncertainty in **A**. Nonetheless, we treat it as one. In this way, the LLSE estimates of the thermal resistance become

$$\begin{bmatrix} \hat{\Theta}_{ri} \\ \hat{\Theta}_{si} \end{bmatrix} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\bar{y}_i, \tag{5.15}$$

for the i^{th} thermocouple. Having obtained the $\hat{\Theta}_{si}$ and $\hat{\Theta}_{ri}$ for each thermocouple, these estimates are then used with (5.12) to estimate each ΔT_i for the dynamometer loads of 500W, 950W, and 1450W. Thus, with the subscripts "5", "6", and "7" denoting the dynamometer loads of 500W, 950W, and 1450W, respectively, the estimated ΔT_i are given by

$$\begin{bmatrix} \Delta\hat{T}_{i5} \\ \Delta\hat{T}_{i6} \\ \Delta\hat{T}_{i7} \end{bmatrix} = \begin{bmatrix} \frac{s_5}{(1-s_5)}(\tau_{m,meas5}+\tau_{m,loss})\omega_{m5} & 3|i_{s5}|^2R_{s5}+P_c \\ \frac{s_6}{(1-s_6)}(\tau_{m,meas6}+\tau_{m,loss})\omega_{m6} & 3|i_{s6}|^2R_{s6}+P_c \\ \frac{s_7}{(1-s_7)}(\tau_{m,meas7}+\tau_{m,loss})\omega_{m7} & 3|i_{s7}|^2R_{s7}+P_c \end{bmatrix} \begin{bmatrix} \hat{\Theta}_{ri} \\ \hat{\Theta}_{si} \end{bmatrix} . \tag{5.16}$$

Given an evaluation of (5.16) and the actual measurement of each ΔT_i, the residuals can be computed, where each residual, DT_i, is defined by

$$\begin{bmatrix} DT_{i5} \\ DT_{i6} \\ DT_{i7} \end{bmatrix} = \begin{bmatrix} \Delta T_{i5} \\ \Delta T_{i6} \\ \Delta T_{i7} \end{bmatrix} - \begin{bmatrix} \Delta\hat{T}_{i5} \\ \Delta\hat{T}_{i6} \\ \Delta\hat{T}_{i7} \end{bmatrix} . \tag{5.17}$$

DT_i has been computed of all 25 thermocouples for rotor 1, rotor 2, and rotor 3. For every case, the maximum magnitude of DT_i never exceeds 2 °C, and generally, the values of each DT_i is under 1 °C. The temperature residuals at the data points that correspond to P_m of 250, 700, 1200 and 1700 watts are even less than for those that correspond to P_m of 500, 950, and 1450 watts. The results justify the use of the linear thermal model in (5.12), since the use of this assumption produced accurate estimates of thermocouple temperatures.

In addition, the above results demonstrate that the thermal resistances are the same for all three rotors, including rotor 2, which has a broken bar. This is to say that in terms of thermal source and conductance, the three rotors are identical. Furthermore, in terms of the resistance of an aluminum conductor such as in the rotor, a worst-case DT_i of 2 °C with T_{ref} of 24.5 °C results in an increase of 0.79% of its reference value. Consequently, if the thermal model works as well for the rotor as it does for the stator, the temperature-compensation of the estimated rotor resistances should work without limiting the accuracy of the estimator developed in Chapter 3. Note that each point temperature which is measured by a thermocouple relates to resistivity. However, as generalized in Section 5.2, the temperature-compensation extends to resistance also. Thus the utility of the thermal model in estimating the rotor temperature is justified.

At this point, the use of (5.12) has been justified. In order to estimate the rotor temperature which is a weighted-average value, we proceed in the same manner as for each thermocouple location. Then, with superscript r denoting a rotor variable, (5.14) becomes

$$\begin{bmatrix} \Delta T_1^r \\ \Delta T_2^r \\ \Delta T_3^r \\ \Delta T_4^r \end{bmatrix} = \begin{bmatrix} \frac{s_1}{(1-s_1)}(\tau_{m,meas1}+\tau_{m,loss})\omega_{m1} & 3|i_{s1}|^2R_{s1}+P_c \\ \frac{s_2}{(1-s_2)}(\tau_{m,meas2}+\tau_{m,loss})\omega_{m2} & 3|i_{s2}|^2R_{s2}+P_c \\ \frac{s_3}{(1-s_3)}(\tau_{m,meas3}+\tau_{m,loss})\omega_{m3} & 3|i_{s3}|^2R_{s3}+P_c \\ \frac{s_4}{(1-s_4)}(\tau_{m,meas4}+\tau_{m,loss})\omega_{m4} & 3|i_{s4}|^2R_{s4}+P_c \end{bmatrix} \begin{bmatrix} \Theta_r^r \\ \Theta_s^r \end{bmatrix} \tag{5.18}$$

and (5.15) becomes

$$\begin{bmatrix} \hat{\Theta}_r^r \\ \hat{\Theta}_s^r \end{bmatrix} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\bar{y}^r \ . \tag{5.19}$$

The matrix A is the same in (5.14) and $\overline{y}^r = [\Delta T_1{}^r \ \Delta T_2{}^r \ \Delta T_3{}^r \ \Delta T_4{}^r]^T$ with subscripts "1", "2", "3", and "4" again corresponding to measurements at dynamometer loads of 250W, 700W, 1.2kW, and 1.7kW.

A problem occurs because the rotor temperatures, $\Delta T_1{}^r$, $\Delta T_2{}^r$, $\Delta T_3{}^r$, and $\Delta T_4{}^r$, cannot be measured directly with thermocouple readings. However, an indirect way to measure them is possible by first estimating the rotor resistance at the ambient temperature. Then, at each load, the rotor resistance is estimated again. Given the ambient temperature, the estimated rotor resistance at ambient temperature, and the estimated rotor resistance at any load, (5.9) produces the necessary rotor temperature at that load from which $\Delta T_i{}^r$ can be computed. Effectively, this method measures the average rotor temperature at various loads. These rotor temperature measurements are then used in (5.18) and (5.19) to estimate the rotor thermal resistances, $\hat{\theta}_r{}^r$ and $\hat{\theta}_s{}^r$.

The experimental procedure for estimating the rotor resistance at room temperature is the same as the one for constant-temperature estimation at an arbitrary load outlined in Chapter 4, with one important exception. Once the motor has started, the measurements used to estimate the rotor resistance at ambient temperature are made quickly before the motor has a chance to heat up, rather than waiting for the motor to reach a steady-state temperature operation. By doing so, we ensure that the estimated rotor resistance corresponds to the ambient temperature. As in Chapter 4, a test consists of collecting R_s, $|i_s|$, $|v_s|$, Φ, and Ω measurements at 16 different slips. Before and during each measurement, the thermocouples are monitored so as to keep the motor approximately at the ambient temperature. If $\hat{R}_r{}^{amb}$ denotes the estimated rotor resistance at the ambient temperature T_{amb}, and $\hat{R}_r{}^j$ denotes the estimated rotor resistance at each load, j = 1 to 4, then by rearranging (5.9) for an aluminum rotor conductor, the estimated rotor temperature $\hat{T}_j{}^r$ becomes

$$\hat{T}_j^r = \frac{\hat{R}_r^j}{\hat{R}_r^{amb}}\left(T_{amb} + 228.1°C\right) - 228.1°C \ . \tag{5.20}$$

Thus,

$$\Delta\hat{T}_j^r = \hat{T}_j^r - T_{amb} \ . \tag{5.21}$$

Notice that the room temperature experiment itself does not contribute directly to (5.18), since this case corresponds to non-operation, but it is necessary in order to determine each $\Delta\hat{T}_j^r$. Once all $\Delta\hat{T}_j^r$ are known, (5.19) can be evaluated to give $\hat{\Theta}_r^r$ and $\hat{\Theta}_s^r$. Experimentally, these two rotor thermal resistances have been determined using rotor 3, a healthy rotor.

The parameters $\hat{\Theta}_r^r$ and $\hat{\Theta}_s^r$ can now be applied to produce the estimated rotor temperatures at any load for any of the three rotors. Given $\hat{\Theta}_r^r$, $\hat{\Theta}_s^r$ and measurements of R_s, $|i_s|$, Ω, and $\tau_{m,meas}$ at an arbitrary load, the corresponding estimate of the rotor temperature $\hat{T}_j^r$ is

$$\hat{T}_j^r = \Delta\hat{T}_j^r + T_{amb} \tag{5.22}$$

where

$$\Delta\hat{T}_j^r = \left[\frac{s_j}{(1-s_j)}\left(\tau_{m,meas_j} + \tau_{m,loss}\right)\omega_{mj} \qquad 3|i_{sj}|^2 R_{sj} + P_c\right]\begin{bmatrix}\hat{\Theta}_r^r \\ \hat{\Theta}_s^r\end{bmatrix} \ . \tag{5.23}$$

Thus by combining the results of Chapter 4 and this chapter, we can estimate the rotor resistance, the rotor temperature, and via (5.9), the estimated rotor resistance can be transformed to an estimated resistance at a reference temperature, for arbitrary operation of the induction motor. This has the desired effect of temperature-compensation. The experimental results of this thermally-compensated broken-rotor bar detector are presented and discussed in Section 5.5.

To summarize, to compensate an estimate of R_r, (5.23) is evaluated with the data of the corresponding operating point, and $\Delta\widehat{T}^r$ is determined. Next (5.22) is evaluated to determine $\widehat{T}^r$. Finally, (5.9) is evaluated with $\widetilde{T} = \widehat{T}^r$ to perform the compensation. Note that at this point the temperature compensation requires a measurement of torque and stator resistance. These requirements are relaxed in Section 5.4.

5.4 Estimation of Stator Resistance Based on the Thermal Model

The stator temperature can be estimated in much the same way that the rotor temperature is. This estimated stator temperature in turn can be used to estimate the stator resistance with (5.9). As is seen experimentally in Section 5.5, this method provides an accurate and readily applicable way to estimate the stator resistance. This method is accurate because it actually approximates closely the values of stator resistance obtained by off-line measurement procedure described in Chapter 4; see Section 5.5 for a comparison. It is readily applicable, since it uses only the same data that the temperature-compensation scheme requires. Moreover, it has the desired characteristic of being non-intrusive to the motor operation. Thus, we no longer need to assume as we have in Chapter 3 and 4 that we can obtain measurements of R_s that are non-intrusive to the motor operation. Finally,

this method offers advantages over that proposed in [32], since this method does away with the need to build the instrumentation necessary in [32] to collect the stator resistance measurements. Basically, an accurate estimate of the stator resistance is obtained with minimal computational cost. In this manner, a highly accurate, yet easily implementable method of estimating the stator resistance is proposed.

Unlike that for obtaining the rotor resistance, the room temperature experiment for obtaining the stator resistance is much simpler. The only requirement is that the measurement of the stator resistance at the room temperature, as opposed to the series of measurements that are needed estimate the rotor resistance. Following the rotor analysis, let R_s^{amb} denote the measured stator resistance at the ambient temperature T_{amb} and R_s^j denote the measured stator resistance at each load j; j = 1,2,3,4 for the dynamometer loads of 250W, 700W, 1.2kW, 1.7kW, respectively. Then, from (5.9) for a copper stator winding, the compensated stator temperature T_j^s becomes

$$T_j^s = \frac{R_s^j}{R_s^{amb}}\left(T_{amb} + 234.5°\text{C}\right) - 234.5°\text{C} \ . \tag{5.24}$$

The thermal model of the stator is given by

$$\begin{bmatrix} \Delta T_1^s \\ \Delta T_2^s \\ \Delta T_3^s \\ \Delta T_4^s \end{bmatrix} = \begin{bmatrix} \frac{s_1}{(1-s_1)}(\tau_{m,meas1}+\tau_{m,loss})\omega_{m1} & 3|i_{s1}|^2R_{s1}+P_c \\ \frac{s_2}{(1-s_2)}(\tau_{m,meas2}+\tau_{m,loss})\omega_{m2} & 3|i_{s2}|^2R_{s2}+P_c \\ \frac{s_3}{(1-s_3)}(\tau_{m,meas3}+\tau_{m,loss})\omega_{m3} & 3|i_{s3}|^2R_{s3}+P_c \\ \frac{s_4}{(1-s_4)}(\tau_{m,meas4}+\tau_{m,loss})\omega_{m4} & 3|i_{s4}|^2R_{s4}+P_c \end{bmatrix} \begin{bmatrix} \Theta_r^s \\ \Theta_s^s \end{bmatrix}, \tag{5.25}$$

where ΔT_j^s is known from (5.24), where

$$\Delta T_j^s = T_j^s - T_{amb} \quad . \tag{5.26}$$

Again, this results in a near-LLSE estimation problem for Θ_r^s and Θ_s^s. Thus, the stator thermal resistances $\hat{\Theta}_s^s$ and $\hat{\Theta}_r^s$ are estimated as

$$\begin{bmatrix} \hat{\Theta}_r^s \\ \hat{\Theta}_s^s \end{bmatrix} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\bar{y}^s \tag{5.27}$$

with **A** the same in (5.14) and $\bar{y}^s = [\Delta T_1^s \ \Delta T_2^s \ \Delta T_3^s \ \Delta T_4^s]^T$. Again, the subscripts "1", "2", "3", and "4" correspond to measurements at the dynamometer loads of 250W, 700W, 1.2kW, and 1.7kW.

Given $\hat{\Theta}_s^s$, $\hat{\Theta}_r^s$ and measurements of R_s, $|i_s|$, Ω, and $\tau_{m,meas}$ at an arbitrary load, the corresponding estimate of the stator temperature $\hat{T}_j^s$ is

$$\hat{T}_j^s = \Delta\hat{T}_j^s + T_{amb} \tag{5.28}$$

where

$$\Delta\hat{T}_j^s = \begin{bmatrix} \frac{s_j}{(1-s_j)}(\tau_{m,meas_j}+\tau_{m,loss})\omega_{mj} & 3|i_{sj}|^2R_{sj}+P_c \end{bmatrix} \begin{bmatrix} \hat{\Theta}_r^s \\ \hat{\Theta}_s^s \end{bmatrix} \quad . \tag{5.29}$$

At the motor load that produces the stator temperature $\hat{T}_j^s$, the corresponding estimate of the stator resistance is

$$\hat{R}_s^j = R_s^{amb}\frac{\hat{T}_j^s + 234.5°C}{T_{amb} + 234.5°C} \quad . \tag{5.30}$$

The accuracy of this estimation method is explored and shown to be highly accurate in the next section. This estimate of the stator resistance works for any of the three rotors, since as far as the stator temperature is concerned, all three rotors are the same.

One problem with estimating the stator resistance thermally as described above, is that we use measurements beyond ω_e, s, $|v_s|$, $|i_s|$, and Φ. In particular, we use torque measurements in place of $3|i_s|^2R_s$. This is limiting for on-line implementation of broken-rotor-bar detection. However, a related estimation method in which only the measurements ω_e, s, $|v_s|$, $|i_s|$, and Φ are needed is presented here. At any arbitrary thermally-steady-state operation of the motor, Equation (5.11) can be written as

$$\Delta T^s = \Theta_s^s(3|i_s|^2R_s + P_c) + \Theta_r^s(3|i_r|^2R_r) \tag{5.31}$$

where $|i_r|$ is determined from the equivalent circuit model to be

$$|i_r| = |i_s| \frac{\omega_e M}{\sqrt{(\omega_e L)^2 + \left(\frac{R_r}{s}\right)^2}} \; . \tag{5.32}$$

Also, by combining (5.24) and (5.26) another expression for ΔT^s is obtained, namely that

$$\Delta T^s = \left(T_{amb} + 234.5°\text{C}\right)\frac{R_s - R_s^{amb}}{R_s^{amb}} \; . \tag{5.33}$$

Equations (5.32) and (5.33) can be combined to solve for the stator resistance R_s by eliminating ΔT^s. Thus, R_s becomes

$$R_s = R_s^{amb}\left[\frac{(T_{amb} + 234.5°C) + 3|i_r|^2 R_r \Theta_r^s + \Theta_s^s P_c}{(T_{amb} + 234.5°C) - 3|i_s|^2 R_s^{amb} \Theta_s^s}\right] . \quad (5.34)$$

Note that (5.34) is an explicit equation for R_s, and we shall use this equation as our thermally-based estimate of R_s, so R_s in (5.34) becomes $\hat{R}_s$. However, note that (5.34) depends on R_r. Note also that only the measurements ω_e, s, $|v_s|$, $|i_s|$, and Φ are needed, since the values for P_c, Θ_s^s and Θ_r^s, R_s^{amb} and T_{amb} can be determined before the motor operates on-line and can thus be treated as known constants. As described in Section 3.4, the thermal estimate of R_s is then used to estimate R_r via (3.17). This appears circular because $\hat{R}_s$ in (5.34) depends on R_r and $\hat{R}_r$ in (3.17) depends on R_s. Therefore, both equations (5.34) and (3.17) must be solved self-consistently with $\hat{R}_r$ substituted for R_r in (5.34) and $\hat{R}_s$ substituted for R_s in (3.17). We use an iterative process to self-consistently solve these equations; and it is found to converge rapidly.

As an example, the particular load of P_m = 500W is examined. The iteration starts by substituting R_s^{amb} = 0.8043 Ω for R_s in (3.17). This yields the estimate $\hat{R}_r$ = 0.5815Ω. Next, the estimated Rr is substituted into (5.34) to yield the estimate $\hat{R}_s$ = 0.8735Ω. The new estimate of $\hat{R}_s$ is substituted into (3.17) and the resulting estimate of $\hat{R}_r$ is substituted (5.34). This round of iteration yields 0.8735Ω for $\hat{R}_s$ and 0.5787Ω for $\hat{R}_r$. The final round of iteration yields 0.8731Ω for $\hat{R}_s$ and 0.5787Ω for $\hat{R}_r$. The estimation process has thus converged in three steps. Note that the estimated value for R_r is very close to 0.5788Ω as estimated in the next section with measured R_s. Note also that the estimated value for R_s is very close to 0.871Ω as measured in the next section. Thus, this method can in practice eliminate the need for torque and stator resistance measurements. Therefore, an important conclusion is that a broken rotor bar can still be detected with the iterative temperature-based estimator presented here.

5.5 Experimental Results

This section presents the experimental results of the temperature-compensated broken-rotor-bar detector described in Sections 5.3. It also presents experimental results of the stator resistance estimation. Measurements at dynamometer loads of 250W, 700W, 1.2kW, and 1.7kW for rotor 3, a healthy rotor, are used to compute the rotor thermal resistances, $\hat{\theta}_r$ and $\hat{\theta}_s$ as discussed in Section 5.3. Using (5.9), all the estimated rotor resistances are then transformed to equivalent estimated resistances at the temperature corresponding to the dynamometer load of 700W. Only the LLSE estimator with known R_s is considered, since it produces the best experimental results in Chapter 4. Later in this section, the experimental results demonstrate that the temperature compensation is successfully implemented to detect a broken rotor bar.

Also, the temperature-based estimation of R_s is compared to the measured R_s at various loads. This comparison shows that the estimation scheme for R_s produces very accurate estimates of R_s. In light of this result, it makes sense to employ the LLSE estimator with known R_s strictly, since in effect, the temperature-based estimation of R_s produces estimates of R_s that are equivalent to measurements of R_s.

Figures 5.1 and Table 5.1 show the results of the temperature-compensated rotor estimates for rotors 1, 2, and 3. Table 5.2 shows the uncompensated rotor estimates for comparison with Table 5.1. The format for the figure is much like the ones presented in Chapter 4. The numbers "1", "2", and "3" locate the thermally-compensated rotor resistance estimated from independent measurements taken with one of the three different rotors. Each estimate corresponds to a different load and hence for a different rotor temperature. The numbers "$\bar{1}$", "$\bar{2}$", and "$\bar{3}$" locate the average estimate of the temperature-

compensated rotor resistance for each rotor. The standard deviation of the rotor resistance estimates for each rotor is indicated by horizontal line segments on both sides of the average of the estimates.

Tables 5.1 provides the numerical values for the experimental results shown in Figures 5.1. In Table 5.1, the average of the estimates is denoted by "Ave{R_r}", the experimental standard deviation of the estimates by the symbol σ_{exp}, and the numerically simulated standard deviation of the estimates by σ_{num}. Temperature-compensated estimates of R_r for rotor 2, as shown by Figure 5.1 and Table 5.1, are located decidedly above the R_r estimates for rotors 1 and 3. Thus, the detection of the broken-rotor-bar in rotor 2 is demonstrated via thermal compensation of the LLSE with known R_s estimates of R_r. Note that all the loads quoted in this section refer to those measured by the dynamometer, the actual loads are somewhat higher.

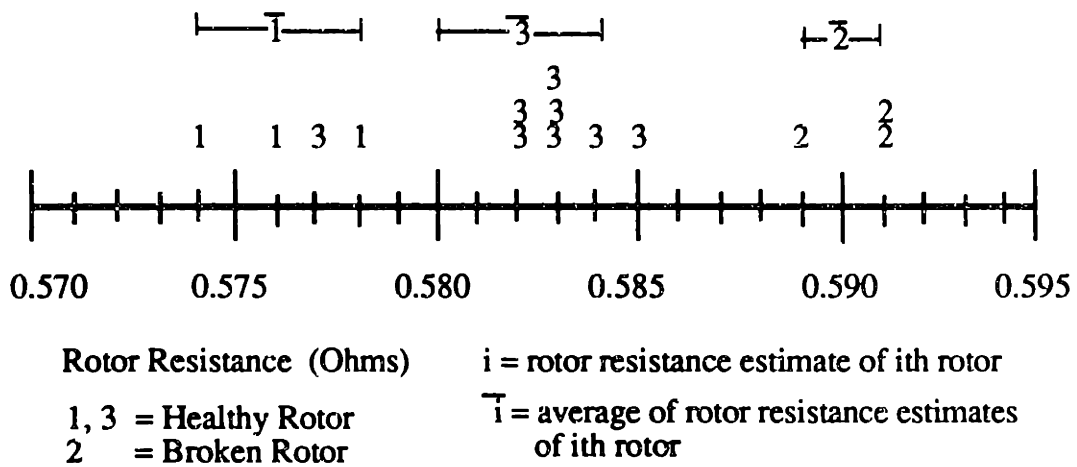


Figure 5.1: Experimental results of temperature-compensated rotor estimates, all the estimates scaled to load of 700W, LLSE estimation with known R_s.

Temperature-Compensated Rotor Resistance Estimates (Ohms) Scaled to the Load of 700W			
Loads	Rotor 3	Rotor 1	Rotor 2
500W	0.585	0.578	0.591
950W	0.577	0.574	0.591
1450W	0.582	0.576	0.589
0W	0.583		
250W	0.583		
700W	0.582		
1200W	0.584		
1700W	0.583		
Ave$\{\hat{R}_r\}$	0.576	0.582	0.590
σ_{exp}	0.002	0.002	0.001
σ_{num}	0.011	0.011	0.011

Table 5.1: Experimental data of temperature-compensated rotor estimates, all the estimates scaled to load of 700W, LLSE estimations with known R_s.

Uncompensated Rotor Resistance Estimates (Ohms)			
Loads	Rotor 3	Rotor 1	Rotor 2
500W	0.579	0.571	0.583
950W	0.586	0.582	0.600
1450W	0.623	0.618	0.632
0W	0.541		
250W	0.573		
700W	0.582		
1200W	0.610		
1700W	0.658		

Table 5.2: Uncompensated estimates of rotor resistance, LLSE estimations with known R_s.

Table 5.3 shows the comparison between the two sets of stator resistance values for all three rotors. For each rotor at various loads, the column marked "Estimated" shows the thermal-based estimated values of the R_s and the other marked "Measured" shows the measured values of R_s. The two values are in good agreement. In fact, there is no significant difference between the two values. Given the previously discussed advantages of temperature-based estimation over the measurement scheme for R_s, Table 5.3 strongly supports adoption of the temperature-based estimation of R_s as a standard procedure. To conclude, the temperature-compensated detection of broken rotor bar via LLSE estimation of R_r with temperature-based estimation of R_s is successfully demonstrated.

Stator Resistance, Estimated and Measured (Ohms)						
Loads	Rotor 1		Rotor 2		Rotor 3	
	Estimated	Measured	Estimated	Measured	Estimated	Measured
500W	0.871	0.867	0.870	0.867	0.874	0.871
950W	0.886	0.883	0.888	0.892	0.888	0.882
1450W	0.916	0.913	0.917	0.914	0.916	0.911
250W					0.870	0.869
700W					0.880	0.882
1200W					0.904	0.902
1700W					0.948	0.948

Table 5.3: Thermal model based estimations of R_s versus measurements of R_s.

Chapter 6

Conclusions and Recommendations for Future Research

6.1 Summary of Thesis

The purpose of this thesis was to study the detection of broken rotor bars in induction motors using state and parameter estimation techniques. The hypothesis upon which the detection was based is that the apparent rotor resistance of an induction motor increases as a result of a rotor-bar breakage. Here, the apparent rotor resistance was that found in the standard single-phase equivalent circuit of an induction motor operating at a constant velocity. To carry out this detection process, it was necessary to develop a thermal model of the induction motor which was used to track variations in rotor and stator resistances. Thus, in this thesis, we have combined the concepts and tools of state and parameter estimation theory with the appropriate electrical and thermal models for the induction motor to derive an estimate of rotor resistance which is in turn compared to its nominal value to detect broken rotor bars.

In Chapter 2, the single-phase equivalent circuit of an induction motor was introduced and its system transfer function from the stator voltage to stator current was presented. This model was important for broken-rotor-bar detection in that the rotor resistance R_r was defined by the model. Also in Chapter 2, the physical experimental system used in this thesis was described.

In Chapter 3, we concentrated on the case of constant-temperature induction motor operation. This assumption allowed us to defer until Chapter 5 consideration of thermal effects on the estimation of rotor resistance operating at different motor loads. In Chapter 3, three parameter estimators were derived for estimating R_r based on the single-phase equivalent of Chapter 2. They were all based on linear-least-square-error (LLSE) estimation. Specifically, they were a LLSE estimator with estimated stator resistance R_s, an iterative LLSE estimator with estimated R_s, and a LLSE estimator with known R_s. Four parameters, R_r, the stator resistance R_s, the mutual inductance M, and the self-inductance L of both rotor and stator, were estimated with the first two estimators. The three parameters, R_r, M, and L, were estimated with the third estimator.

The three estimators were driven with measurement sets of ω_e, s, $|i_s|$, $|v_s|$, and Φ for the first two and ω_e, s, $|i_s|$, $|v_s|$, Φ , and R_s for the third. Each independent set of measurements gave rise to two independent equations via the electrical model of Chapter 2. Therefore, the first estimator needed at least three independent sets of measurements while each of the other two estimators needed at least two independent set of measurements to estimate the fundamental parameters. We actually always collected 16 independent sets of measurements to ensure sufficient richness of data.

Finally, Chapter 3 examined the sensitivity of three estimators to the noise in the measurements on which they operate. From the sensitivity analysis, which was performed

numerically, upper bounds on the percent-deviation of R_r were determined to be 1.4%, 3.8%, and 1.9%, for the LLSE estimator with estimated R_s, the iterative LLSE estimator with estimated R_s, and the LLSE estimator with known R_s, respectively. These results indicated that at least the first and the third estimators should successfully detect broken rotor bars in the test motor, since the breakage of one bar in 45 should increase R_r by approximately 2%. The upper bounds on the percent-deviation of R_s, on the other hand, were determined to be 74% for the LLSE estimator and 31% for the iterative-LLSE estimator. To summarize the sensitivity analysis, the numerical results predicted that R_r could be estimated well while R_s could not.

Chapter 4 presented the experimental results based on an induction motor operating at constant temperature. The induction motor was operated at a constant stator voltage of 120 volts and at the load of 700W as measured by the dynamometer. Three rotors of the same type, the two "healthy" rotors, 1 and 3, and the "unhealthy" rotor 2 with one broken bar out of 45, were tested four times each. With each test, the rotor resistance was estimated three times, one time for each of the three types of estimators discussed in Chapter 3. In addition, with each test, the stator resistance was estimated twice, once using the LLSE estimation with estimated R_s, and once, using the iterative-LLSE estimation with estimated R_s. Numerical and graphical results of the all the experimental estimates as well as the average and the standard deviation of these estimates were given in Chapter 4.

In terms of average and standard deviation of the R_r estimates, the detection of a broken rotor bar was demonstrated quite clearly with the LLSE estimator with known R_s. The detection of a broken rotor bar was still possible with the iterative LLSE estimator with estimated R_s, albeit less clearly. Finally, the detection of a broken rotor bar with LLSE estimator with estimated R_s was least clear, and was arguable at best. For the LLSE estimator with known R_s, the ratio of the difference of the averages of the independent R_r

estimates of the "healthy" and "unhealthy" rotors to the average of R_r estimates of the "healthy" rotor was 2.7%, while the ratio of standard deviation of the R_r estimates to the average value of the R_r estimates for each rotor was only 0.3%. The observed change was consistent with what we expected to see when one bar breaks out of 45 bars; and the standard deviation was such that it was seen clearly.

With respect to the stator resistance, the results of sensitivity analysis in Chapter 3 were confirmed. The stator resistance could not be estimated well experimentally by either the LLSE or the iterative-LLSE estimator.

With different steady-state loads, the induction motor exhibits different steady-state temperatures. This poses a problem for broken-rotor-bar detector in that a variation in rotor temperature causes significant variation in the estimate of R_r. In response, Chapter 5 proposed a way to thermally-compensate the estimators of Chapter 3 so that the estimates of R_r at different loads can all be standardized to estimates at a reference load and hence reference temperature. The proposed compensation was based on a temperature-resistance relation for metallic conductors. This temperature-resistance relation required the knowledge of the rotor temperature, which was estimated. The estimation of the rotor temperature in turn was based on a thermal model of the induction motor. Here, a linear model was used and its parameters were determined experimentally using LLSE estimation. The thermal model itself was verified experimentally by comparing point-wise estimates of temperatures with actual thermocouple temperature readings. The two were within 2 °C of each other for all 25 thermocouples over the range of 0 to 1.7kW. This error corresponded to an estimation error in R_r by 0.8% of its reference value if the reference R_r value was 0.565 Ω at 24.5°C, which was the case for the experimental motor. This result indicates that the thermal compensation scheme should work without limiting the estimators developed in Chapter 3.

Next, the temperature-compensated estimator of R_r was examined experimentally. First, the thermal model parameters were estimated at dynamometer-measured loads of 0W, 250W, 700W, 1.2kW, and 1.7kW. Then, using these parameters, the temperature-compensated R_r estimates of rotors 1, 2, and 3 were obtained at loads of 500W, 950W, and 1450W as measured by the dynamometer. Only the LLSE estimator with known R_s was studied. Numerical and graphical results of all experimental estimates as well as the average and the standard deviation of these estimates were given in Chapter 5. The temperature-compensated detection of broken rotor bars via LLSE estimation of R_r with known R_s was demonstrated clearly with the experimental results. The ratio of the difference of the averages of R_r estimates of a "healthy" and "unhealthy" rotors to the average of R_r estimates of the "healthy" rotor was 1.9% while the ratio of standard deviation of R_r estimates to the average value of R_r estimates for each rotor was only 0.2%.

Also in Chapter 5, a study of temperature-compensation led to development of an estimation method for the stator resistance based on the thermal model. Using the thermal model, the stator temperature was estimated using the same technique employed for the rotor temperature. This involved estimating additional model parameters. The stator temperature estimate was then used to estimate the stator resistance. This method of estimating the R_s at various motor loads produced R_s estimates that were very close to the actual measured values of R_s. Typically, the difference between the two were less than 0.5% of their values. Hence, in terms of the detection of broken rotor bars, estimation schemes using estimation of R_s based on thermal model or a measurement of R_s are equivalent. Finally, one problem with thermally estimating the stator resistance as it was originally introduced was that we used measurements beyond ω_e, s, $|v_s|$, $|i_s|$, and Φ. In particular, we used torque measurements in place of $3|i_s|^2R_s$. Hence, an iterative estimation method which used only the measurements ω_e, s, $|v_s|$, $|i_s|$, and Φ was developed

to self-consistently estimate R_r and R_s by iterating between a thermal model and an electrically-based estimator. This iterative procedure produced stable estimates of R_r and R_s whose values were again very close to the estimates produced by the LLSE estimator with measured R_s.

6.2 Conclusion

The temperature-compensated detection of broken rotor bars via LLSE estimation of R_r with estimation of R_s based on a thermal model has been clearly demonstrated. One broken bar out of 45 appears easily detectable. While R_r is easily estimated electrically, R_s is not. Therefore, R_s must be either measured or estimated by some other means; recall that R_s is necessary in the thermal model which is used to compensate R_r. Estimation of R_s based on the thermal model becomes a natural choice, since this method provides an ease of implementation, yet retains the accuracy equal to that of actual measurements of R_s. The detection of a broken rotor bar in an induction motor using a thermal model and the single-phase equivalent circuit as the basis proves to be theoretically satisfying and in the end provides an effective means of detection.

There are two main limitations to the broken-rotor-bar detectors proposed in this thesis. One, the thermal model is a steady-state model and two, we need more than one set of stator terminal electrical measurements to produce estimates. In fact, we use 16 independent sets of measurements. Thus, in terms of practicality, an on-line detector is nearly but not completely possible with the results described in this thesis. The limitations and the possible solutions for them will be discussed more fully in the next section which deals with suggestions for future research.

6.3 Recommendations for Future Work

As mentioned in the conclusion section of this chapter, one problem is that of practical application. We want to monitor broken rotor bar on line, but the need for many independent sets of measurements makes this difficult. We do not want to disturb the normal operation of the motor but this is unavoidable in the process of collecting sufficiently rich steady-state data; in order for the LLSE estimators to work, several independent sets of steady-state data are necessary.

There are two important issues which limit the practicality of the broken rotor bar detector as it is presented in this thesis. The first is that data from different operating points must be collected before estimation can proceed. This data is then organized as in (2.11). In order for R_s and R_r to be factored out of (2.11), they must be the same R_s and R_r for all equations, that is, for all operating points. Therefore, they must be at the same temperature even though they correspond to different loads. This is why the 16 measurements at different operating points, around the operating point which established motor temperature, were taken so quickly. This is the second limitation. To accommodate both limitations requires further study.

To accommodate the first limitation, one should first determine the spread of operating points that is necessary to provide data of sufficient richness for successful estimation. It is possible that this spread of operating points may arise naturally during the operation of the motor. To accommodate the second limitation, it is necessary to combine the thermal and the electrical models so that a thermally-compensated R_s and R_r can be factored out of

(2.11). In doing so, each equation can be evaluated with data which is in thermal as well as electrical steady-state.

There is yet another, although less severe, limitation, namely the need to collect data while the motor is in constant-temperature electrically-steady-state operation. To relax this limitation, dynamic electromechanical and thermal models should be developed and incorporated into the estimators.

The inability to estimate the R_s well using the electrical LLSE estimators was shown by the sensitivity analysis in Chapter 3 and by the experimental results in Chapter 4. This has yet to be explained theoretically, and it should be explained. In addition to the noise studied in Chapter 3, modelling error such as the omission of core loss is suspected of contributing to errors in the estimation of R_s and R_r.

A general stability study should be done for all the iterative estimators. We also stated that the sensitivity analysis is in general amplitude dependent. This should be investigated to draw general conclusions. In addition, we emphasized at various points that the estimation problem addressed in this thesis in nearly LLSE estimation problem. Perhaps a study using other types of estimation schemes, such as total-least-square-error estimation, would be profitable.

The tests were conducted using two healthy rotors and one broken rotor. If all the tests were conducted again, first by testing two healthy rotors and then breaking a rotor bar in one of the two healthy rotors, this would give a better indication of how rotor-bar breakage is observed by the various estimators. In this thesis, we have no estimation results for rotor 2, the broken rotor, prior to its breakage. In addition, this could be followed by the

successive breakage of more bars, so that the effect of continued breakage on the incremental increase of the estimate of R_r could be quantified.

Lastly, the thermal model gave us an effective means of predicting the motor temperature. If, for example, a the fan blade broke, we expect the temperatures at various points of the motor at a given motor load to exceed their nominal values. Thus, a broken fan blade or obstruction of the fan-cooling system should also be detected by all the tools developed in this thesis. Detection of this type of fault could be investigated next.

Bibliography

[1] P. E. Albrecht, et al., "Assessment of the reliability of motors in utility application", *IEE-IAS Annual Meeting Conference Record*, 364-366, Chicago, 1984.

[2] C. R. Heising and P. O'Donnell, "Summary of results from IEEE survey of the reliability of large motors in industrial and commercial installations", *Proceedings of the 12TH INTER-RAM Conference for the Electric Power Industry*, 386-392, 1985.

[3] IEEE Committee Report, P. O'Donnell, coordinating author, "Report of large motor reliability survey of industrial and commercial installations -- parts 1-3", *Industrial and Commercial Power Systems Technical Conference*, May 1983, 84, 85.

[4] J. R. Smith, "Editorial on plant condition monitoring", *IEE Proceedings*, 133, B, 142-148, May 1986.

[5] D. Filbert, "Fault diagnosis in nonlinear electromechanical systems by continuous time parameter estimation", *Proceedings of the ISA International Conference*, Houston, TX, October 1984.

[6] J. H. Lang and G. C. Verghese, "A proposal to study the reliability improvement of electrical machines", *Unpublished research proposal*, MIT Laboratory for Electromagnetic and Electronic Systems, July 1986.

[7] J. L. Kirtley, J. H. Lang, F. C. Schweppe, and S. D. Umans, "Ship service power system development", *Unpublished research proposal*, MIT Laboratory for Electromagnetic and Electronic Systems, June 1986.

[8] G. B. Kliman, R. A. Kogel, J. Stein, R. D. Endicott and M. W. Madden, "Noninvasive detection of broken rotor bars in induction motors", *Proceedings of the IEEE PES Winter Meeting*, New York, NY, February 1988.

[9] J. Penman, M. N. Day, A. J. Tait and W. E. Bryan, "Condition monitoring of electrical drives", *IEE Proceedings*, 133, B, 142-148, May 1986.

[10] C. Hargis, B. G. Gaydon and K. Kamash, "The detection of rotor defects in induction mc rs", *Conference Record of the IEE Electric Machinery Design and Application Conference*, 216-220, London, England, July 1982.

[11] I. Kerszenbaum and C. F. Landy, "The existence of large inter-bar currents in three phase squirrel cage motors with rotor-bar and/or end-ring faults", *IEEE Transaction on Power Apparatus and Systems*, 103, 1854-1862, July 1984.

[12] S. Williamson and A. C. Smith, "Steady-state analysis of 3-phase cage motors with rotor-bar and end-ring faults", *IEE Proceedings*, 129, B, 93-100, May 1982.

[13] J. F. Lindsay and T. H. Barton, "A modern approach to induction machine parameter identification", *IEEE Transactions on Power Apparatus and Systems*, 91, 1493-1500, July 1972.

[14] A. Bellini, A. de Carli, and M. la Cava, "Parameter identification for induction motor simulation", *Proceedings of the IFAC Symposium on Identification and System Parameter Identification*, 195-205, 1977.

[15] A. Consoli, L. Fortuna, and A. Gallo, "Identification of an induction motor by a microcomputer-based structure", *IEEE Transactions on Industrial Electronics*, 34, 422-428, November 1987.

[16] B. K. Bose, *Power Electronics and AC Drives*, Prentice Hall, 1986.

[17] W. Leonhard, *Control of Electrical Drives*, Springer-Verlag, 1985.

[18] W. Leonhard, "Microcomputer control of high dynamic performance ac-drives", *Automatica*, 22, 1,1986.

[19] R. Krishnan and F. C. Doran, "Study on parameter sensitivity in high performance inverter-fed induction motor drive systems", *IEEE-IAS Annual Meeting*, 1984.

[20] T. Irisa, S. Takata, R. Ueda, T. Sonoda, and T. Mochizuki, "A novel approach on parameter self-tuning method in ac servo system", *Proceeding of IFAC Symposium on Control in Power Electronics and Electrical Drives*, 1982.

[21] H. Kubota, K. Matsuse, and T. Fukao, "New control method of inverter-fed induction motor drive by using state observer with rotor resistance identification", *IEEE-IAS Annual Meeting*, 1985.

[22] T. Matsuo and T. A. Lipo, "A rotor identification scheme for vector-controlled induction motor drives", *IEEE Transactions on Industry Applications*, 21, 1985.

[23] Y. Yoshida, R. Ueda, and T. Sonoda, "A new inverter-fed induction motor drive with a function for correcting rotor circuit time constant", *Proceeding of the International Power Electronics Conference*, 1983.

[24] S. Tamai, H. Sugimoto, and M. Yano, "Speed sensor-less vector control of induction motor with model reference adaptive system", *IEEE-IAS Annual Meeting*, 1987.

[25] Y. D. Landau, *Adaptive Control: The Model Reference Approach*, Marcel Dekker, Inc., 1979.

[26] R. Gabriel and W. Leonhard, "Microprocessor control of induction motor", *Proceedings International Semiconductor Power Conference*, 1982.

[27] L. J. Garces, "Parameter adaptation for the speed controlled static ac drive with a squirrel cage induction motor", IEEE Transactions on Industry Applications, 16, 2, 1980.

[28] A. E. Fitzgerald, C. Kingsley, and S. D. Umans, *Electric Machinery*, McGraw-Hill, 1983.

[29] H. H. Woodson, J. Melcher, *Electromechanical Dynamics, Part 1*, John Wiley and Sons, 1968.

[30] P. C. Krause, *Analysis of Electric Machinery*, McGraw-Hill, 1986.

[31] G. Strang, *Linear Algebra and its Applications*, Academic Press, 1980.

[32] D. A. Pace, "Motor thermal protection by continuous monitoring of winding resistance", *IECI*, 27, no. 3, 137-141, August 1980.

[33] K. Minami, *Model-Based Speed and Parameter Tracking for Induction Motors*, S. M. Thesis, Massachusetts Institute of Technology, May 1989.

[34] M. Velez-Reyes, *Speed and Parameter Estimation for Induction Machines*, S. M. Thesis, Massachusetts Institute of Technology, May 1988.

[35] D. Fink, H. Beaty, *Standard Handbook for Electrical Engineers*, McGraw-Hill, 1978.

[36] T. Baumeistor, E. Avallone, T. Baumeistor III, *Mark's Handbook for Mechanical Engineers*, McGraw-Hill, 1978.

Appendix A

Derivation of the Temperature-Resistance Relation

The resistance of metals increases with temperature. For many metals, the resistance at any temperature t in °C over the range of 0°C to 100°C is approximately

$$R = R_0(1 + \alpha t) \tag{A1.1}$$

where R_0 is the resistance at 0°C and α is the temperature coefficient of resistance[36]. Thus, (A1.1) can be expressed at two different temperatures, and the resulting ratio becomes

$$\frac{R_1}{R_2} = \frac{R_0(1 + \alpha t_1)}{R_0(1 + \alpha t_2)} = \frac{\left(\frac{1}{\alpha} + t_1\right)}{\left(\frac{1}{\alpha} + t_2\right)}. \tag{A1.2}$$

The constant $-1/\alpha$ is referred to as the inferred absolute zero, since this value for temperature t in (A1.1) corresponds to the resistance of zero. For copper, this value is -234.5 and for aluminum, it is - 228.1[35]. Therefore, for copper (A1.2) becomes

$$\frac{R_1}{R_2} = \frac{(234.5 + t_1)}{(234.5 + t_2)} \tag{A1.3}$$

and for aluminum (A1.2) becomes

$$\frac{R_1}{R_2} = \frac{(228.1 + t_1)}{(228.1 + t_2)} . \tag{A1.4}$$

Equations (A1.3) and (A1.4) are precisely the Equations (5.1) and (5.2), the temperature-resistance relation for copper and aluminum conductors.

Appendix B

Calculation of $\tau_{m,loss}$

The total mechanical torque τ_m that the test motor produces is not equivalent to $\tau_{m,meas}$, the mechanical torque that the dynamometer measures. Part of τ_m is consumed by windage and bearing losses, and the remaining part drives the load of the dynamometer. Since the dynamometer only reads the part of the total torque that drives it, the windage and bearing torques, collectively called $\tau_{m,loss}$, are not included in $\tau_{m,meas}$. In order to use Equation (5.12) for the thermal model, however, $\tau_{m,loss}$ must be calculated.

To calculate $\tau_{m,loss}$ we use the relation

$$-\tau_{m,loss} = J\frac{d\omega_m}{dt} \tag{A2.1}$$

where J denotes the rotational inertia, ω_m the mechanical speed in radians per second, and the variable t denotes time in seconds. This equation holds for an electrically-undriven motor. The rotational inertia J in (A2.1) for the experimental system is

$$J = J_r + J_d \tag{A2.2}$$

where J_r is the rotational inertia of the rotor and J_d is the rotational inertia of the dynamometer. We can calculate J_r by treating each of the various parts that make up the rotor as either a cylindrical rod or an annular ring. The corresponding mass density and physical dimensions must be taken into account in calculating the rotational inertia of each cylindrical rod or annular ring. Rotational inertias of all the separate parts of the rotor are then summed to produce J_r. The dimensions and the mass densities of all the annular rings and cylindrical rods that make up the rotor are given in a computer program called jrotor.r. This program, which is found at the end of this appendix, uses the method described above to compute J_r. The value for J_r as computed by the program is 8.1547x10-3 kgm^2. The value of J_d which equals 6.005x10-3kgm^2, has been obtained from Magtrol, the manufacturer of the dynamometer. Hence, J which is the sum of J_r and J_d, is equal to

$$J = 0.01416 \text{ kgm}^2. \tag{A2.3}$$

To compute $d\omega_m/dt$ in (A2.1) the test motor is first operated at no load. The motor is then turned off and the rotor is allowed to spin down to a halt. Hence, this experimental procedure is called a "spindown" test. The torque that forces the motor to spin down must be $\tau_{m,loss}$, since the dynamometer does not actively load the motor in this experiment. Figure A2.1 shows three such spindown tests, with data taken at one second intervals. For our purposes we are interested in $d\omega_m/dt$ near the no-load speed. We shall compute it using (A2.1) and (A2.3). The time derivative $d\omega_m/dt$ in (A2.1) is approximated at the no-load speed from Figure A2.1 as -26.808 rad/s^2. By substituting this result we determine that $\tau_{m,loss}$ is given by

$$\tau_{m,loss} = 0.3796. \tag{A2.4}$$

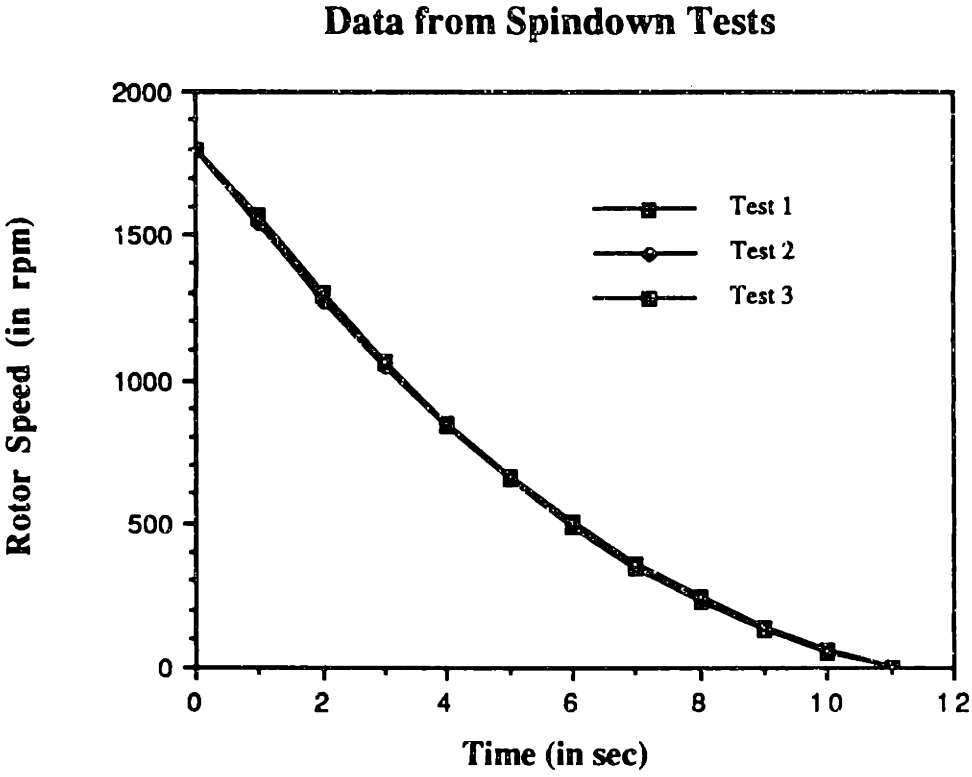


Figure A2.1: Spindown test data

```
////////////////////////////////////////////////
//  Program jrotor.r (in [cho.matfiles.thermo]) //
//                                              //
//  Program to find the rotational inertia      //
//  of the rotor of the induction motor.        //
////////////////////////////////////////////////


clear
diary('djrotor');
long;

//  Constants

//  Mass Densities (kg/m^3)
//       Pss      Stainless Steel
//       Pms      Magnetic Steel
//       Pal      Aluminum

Pss = 7800
Pms = 7700
Pal = 2700

//  inm  (conversion factor multiplied to in to get m)
//  lbkg (conversion factor multiplied to lb to get kg)

inm = 1/39.4
lbkg = 1/2.21

//  Physical Rotor Measurements

//  Lengths in meters

L1 = 3.375*inm
L2 = 8.5*inm
L3 = 1.625*inm
L4 = 2*inm
Lb = 0.3125*inm
Lalring = 0.5*inm
L7 = 2.5*inm

//  Radii in meters

R1 = 1.125*inm/2
R2 = 1.345*inm/2
R3 = 1.140*inm/2
R4 = 0.9*inm/2
R5 = (0.116731 - 2*0.75*inm)/2
R6 = 0.116731/2
Rb = 0.25*inm/2

//  Vwings = volume of each aluminum wings (16 in all)
//  Vb     = volume of each aluminum cylindrical buttons (16 in all)
//  RBarea = cross-section area of rotor bar (45 rotor bars in all)
//  MSarea = cross-section area of magnetic steel
```

```
Vwings = inm**3*((5/8+7/8)*(3/16+3/8)*3/8 + 1/4*1/8*3/8)
Vb = PI*Rb**2*Lb
RBarea = 45*1/8*3/4*inm**2
MSarea = PI*(R6**2-R5**2)-RBarea

//  Mass of each component of the rotor

M(1) = Pss*PI*R1**2*L1;
M(2) = Pss*PI*R2**2*L2;
M(3) = Pss*PI*R3**2*L3;
M(4) = Pss*PI*R4**2*L4;
M(5) = Pms*PI*(R5**2 - R4**2)*L7;
M(6) = Pal*(16*(Vb + Vwings) + PI*(R6**2 - R5**2)*Lalring);
M(7) = L7*(Pal*RBarea + Pms*MSarea);
M

//  Rotor weighed 16 lbs as measured.  The mass we calculate is the mass of
//  of the rotor minus the weight of the 2 bearings.
//      RMass   in kgs
//      RMass   converted to lbs

RMass = sum(M)
RMasslb = RMass/lbkg

//  Rotational Inertia
//  For a solid cylinder J = 1/2*M*R**2
//  For an annulus disk  J = 1/2*M*(Rout**2 + Rin**2)

J(1) = 1/2*M(1)*R1**2;
J(2) = 1/2*M(2)*R2**2;
J(3) = 1/2*M(3)*R3**2;
J(4) = 1/2*M(4)*R4**2;
J(5) = 1/2*M(5)*(R5**2 + R4**2);
J(6) = 1/2*M(6)*(R6**2 + R5**2);
J(7) = 1/2*M(7)*(R6**2 + R5**2);
J

Jrotor = sum(J)

//  Need to solve for constant acceleration
//  since,      Jrotor*Accel = Tf(Torque due to friction)

sp1 = [1798 1570 1293 1058 841 655 490 346 224 123 47];
sp2 = [1798 1544 1272 1041 839 656 494 353 230 129 50];
sp3 = [1798 1564 1293 1060 852 668 505 364 241 137 56];
speed = [sp1;sp2;sp3]


save 'sjrotor';

diary(0);
```

Appendix C

Programs for the Estimators

```
////////////////////////////////////////////////////////////////////
//  Program lsdata.udf (in [cho.scatter.rsparam])                 //
//                                                                //
//  Estimator 1 (LLSE estimator with Estimated Rs)                //
//                                                                //
//  Given matrix datmat, output pv, the vector containing the     //
//  fundamental parameters estimated from datmat                  //
////////////////////////////////////////////////////////////////////


//[pv,x,S,A,WeS] = lsdata(datmat)
We = 376.99;
// We use fake data to check if our least squares algorithm works properly.
// Assume that a data matrix with the rows consisting of
// datmat(T,I,Pe,Pm,pf,rpm) is passed to this function.  The least
// squares estimate x will be returned.  Realizing that matrix
// datmat = | v1 v2 v3 v4 v5 v6 v7| with each vi's corresponding to
//          | T  I  Pe Pm pf rpm V|,
// we can conveniently write A as a linear combo of these vectors.
// Now, we break x into real and imaginary part, forcing x1 to be purely real.
// WeS = We - 2*Wm = We - 4*PI*rpm/60
[nrows, ncols] = size(datmat);
WeS = We*ONES(nrows,1) - (PI/15)*datmat(:,6);
S = WeS/We;
for i=1:nrows,...
pf = datmat(i,5);...
Is = datmat(i,2);...
V = datmat(i,7);...
C(i,1) = 0;...
C(i,2) = Is*pf;...
C(i,3) = Is*sqrt(1-pf*pf);...
C(i,4) = C(i,2)*WeS(i);...
C(i,5) = C(i,3)*WeS(i);...
D(i,1) = -WeS(i)*V;...
D(i,2) = - C(i,3);...
D(i,3) = C(i,2);...
D(i,4) = - C(i,5);...
D(i,5) = C(i,4);...
end
Y1 = datmat(:,7);
Y2 = 0*ONES(nrows,1);
A = [C;D];
Y = [Y1;Y2];
xri = A\Y;
x(1) = xri(1);
x(2) = xri(2) + jay*xri(3);
x(3) = xri(4) + jay*xri(5);
pv = param(x);
retf
```

```
////////////////////////////////////////////////////////////////////
//  Program r2data.udf (in [cho.scatter.rsparam])                 //
//                                                                //
//  Estimator 2 (Iterative LLSE estimator with Estimated Rs)      //
//  Stable, uses x2 = Rs as the dependent variable                //
//                                                                //
//  Given matrix datmat, output pv, the vector containing the     //
//  fundamental parameters estimated from datmat                  //
////////////////////////////////////////////////////////////////////


//[pv,x,S,A,WeS] = r2data(datmat)
We = 376.99;
de = 0.001;
// inquire de 'Enter de: '
// We use fake data to check if our least squares algorithm works properly.
// Assume that a data matrix with the rows consisting of
// datmat(T,I,Pe,Pm,pf,rpm) is passed to this function.  The least
// squares estimate x will be returned.  Realizing that matrix
// datmat = | v1 v2 v3 v4 v5 v6 v7| with each vi's corresponding to
//          | T  I  Pe Pm pf rpm V|,
// we can conveniently write A as a linear combo of these vectors.
// Now, we break x into real and imaginary part, forcing x1 to be purely real.
// WeS = We - 2*Wm = We - 4*PI*rpm/60
[nrows, ncols] = size(datmat);
WeS = We*ONES(nrows,1) - (PI/15)*datmat(:,6);
S = WeS/We;
for i=1:nrows,...
pf = datmat(i,5);...
Is = datmat(i,2);...
V = datmat(i,7);...
C(i,1) = 0;...
C(i,2) = Is*pf;...
C(i,3) = Is*sqrt(1-pf*pf);...
C(i,4) = C(i,2)*WeS(i);...
C(i,5) = C(i,3)*WeS(i);...
D(i,1) = -WeS(i)*V;...
D(i,2) = - C(i,3);...
D(i,3) = C(i,2);...
D(i,4) = - C(i,5);...
D(i,5) = C(i,4);...
end
Y1 = datmat(:,7);
Y2 = 0*ONES(nrows,1);
A = [C;D];
Y = [Y1;Y2];
xri = A\Y;
x(1) = xri(1);
x(2) = xri(2) + jay*xri(3);
x(3) = xri(4) + jay*xri(5);
K = imag(x(3))/x(1);
F = [A(:,1) A(:,3) A(:,4) A(:,5)];
// While x2r does not 'match' x3i/x1, redo ls estimation with
// x3i/x1 as a measurement
ndif = abs((real(x(2)) - K)/real(x(2)));
```

```
while ndif > de,...
K = imag(x(3))/x(1);...
Z = Y -  K*A(:,2);...
xri = F\Z;...
x(1) = xri(1);...
x(3) = xri(3) + jay*xri(4);...
x(2) = imag(x(3))/x(1) + jay*xri(2);...
ndif = abs((real(x(2)) - K)/real(x(2)));...
end
pv = param(x);
retf
```

```
////////////////////////////////////////////////////////////////////
//  Program irecdata.udf (in [cho.matfiles.ideal])                //
//                                                                //
//  Estimator 2 (Iterative LLSE estimator with Estimated Rs)      //
//  Stable, uses x5 = Rs*L/Rr as the dependent variable           //
//                                                                //
//  Given matrix datmat, output pv, the vector containing the     //
//  fundamental parameters estimated from datmat                  //
////////////////////////////////////////////////////////////////////


//[pv,x,A,WeS,S] = irecdata(datmat)
We = 376.99;
V = 80.;
de = 0.001;
// inquire de 'Enter de: '
// We use fake data to check if our least squares algorithm works properly.
// Assume that a data matrix with the rows consisting of
// datmat(T,I,Pe,Pm,pf,rpm) is passed to this function.  The least
// squares estimate x will be returned.  Realizing that matrix
// datmat = | v1 v2 v3 v4 v5 v6 | with each vi's corresponding to
//          | T  I  Pe Pm pf rpm|,
// we can conveniently write A as a linear combo of these vectors.
// Now, we break x into real and imaginary part, forcing x1 to be purely real.
// WeS = We - 2*Wm = We - 4*PI*rpm/60
[nrows, ncols] = size(datmat);
WeS = We*ONES(nrows,1) - (PI/15)*datmat(:,6);
S = WeS/We;
for i=1:nrows,...
pf = datmat(i,5);...
Is = datmat(i,2);...
C(i,1) = 0;...
C(i,2) = Is*pf;...
C(i,3) = Is*sqrt(1-pf*pf);...
C(i,4) = C(i,2)*WeS(i);...
C(i,5) = C(i,3)*WeS(i);...
D(i,1) = -WeS(i)*V;...
D(i,2) = - C(i,3);...
D(i,3) = C(i,2);...
D(i,4) = - C(i,5);...
D(i,5) = C(i,4);...
end
Y1 = V*ONES(nrows,1);
Y2 = 0*ONES(nrows,1);
A = [C;D];
Y = [Y1;Y2];
xri = A\Y;
x(1) = xri(1);
x(2) = xri(2) + jay*xri(3);
x(3) = xri(4) + jay*xri(5);
K = x(1)*real(x(2));
F = [A(:,1) A(:,2) A(:,3) A(:,4)];
// While x3i does not 'match' x1r*x2r, redo ls estimation with
// x1r*x2r as a measurement
ndif = abs((imag(x(3)) - K)/imag(x(3)));
```

```
while ndif > de,...
R = x(1)*real(x(2));...
Z = Y -  R*A(:,S);...
xri = F\Z;...
x(1) = xri(1);...
x(2) = xri(2) + jay*xri(3);...
x(3) = xri(4) + jay*x(1)*real(x(2));...
ndif = abs((imag(x(3)) - R)/imag(x(3)));...
end
pv = param(x);
retf
```

```
////////////////////////////////////////////////////////////////////
//  Program recldata.udf (in [cho.matfiles.ideal])                //
//                                                                //
//  Estimator 2 (Iterative LLSE estimator with Estimated Rs)      //
//  Unstable, uses x1 = L/Rr as the dependent variable            //
//                                                                //
//  Given matrix datmat, output pv, the vector containing the     //
//  fundamental parameters estimated from datmat                  //
////////////////////////////////////////////////////////////////////


//[x,WeS,S] = recldata(datmat)
We = 376.99;
V = 80.;
de = .001;
// We use fake data to check if our least squares algorithm works properly.
// Assume that a data matrix with the rows consisting of
// datmat(T,I,Pe,Pm,pf,rpm) is passed to this function.  The least
// squares estimate x will be returned.  Realizing that matrix
// datmat = | v1 v2 v3 v4 v5 v6 | with each vi's corresponding to
//          | T  I  Pe Pm pf rpm|,
// we can conveniently write A as a linear combo of these vectors.
// Now, we break x into real and imaginary part, forcing x1 t  be purely real.
// WeS = We - 2*Wm = We - 4*PI*rpm/60
[nrows, ncols] = size(datmat);
WeS = We*ONES(nrows,1) - (PI/15)*datmat(:,6);
S = WeS/We;
for i=1:nrows,...
pf = datmat(i,5);...
Is = datmat(i,2);...
C(i,1) = 0;...
C(i,2) = Is*pf;...
C(i,3) = Is*sqrt(1-pf*pf);...
C(i,4) = C(i,2)*WeS(i);...
C(i,5) = C(i,3)*WeS(i);...
D(i,1) = -WeS(i)*V;...
D(i,2) = - C(i,3);...
D(i,3) = C(i,2);...
D(i,4) = - C(i,5);...
D(i,5) = C(i,4);...
end
Y1 = V*ONES(nrows,1);
Y2 = 0*ONES(nrows,1);
A = [C;D]
pause
Y = [Y1;Y2];
xri = A\Y;
x(1) = xri(1);
x(2) = xri(2) + jay*xri(3);
x(3) = xri(4) + jay*xri(5);
x
pause
K = imag(x(3))/real(x(2));
F = [A(:,2) A(:,3) A(:,4) A(:,5)]
pause
```

```
// While x1 does not 'match' x3i/x2r, redo ls estimation with
// x3i/x2r as a measurement
ndif = abs((x(1) - K)/x(1))
while ndif > de,...
K = imag(x(3))/real(x(2));...
Z = Y -  K*A(:,1);...
xri = F\Z;...
x(2) = xri(1) + jay*xri(2);...
x(3) = xri(3) + jay*xri(4);...
x(1) = imag(x(3))/real(x(2));...
ndif = abs((x(1) - K)/x(1)),...
end
retf
```

```
////////////////////////////////////////////////////////////////////////
//  Program param.udf (in [cho.scatter.rsparam])                      //
//                                                                    //
//  Given either the estimated vector x from Estimator 1 or           //
//  Estimator 2 return the fundamental vector pv = [Rs Rs Rr L M]      //
////////////////////////////////////////////////////////////////////////


//pv = param(x)
// pv = [TrRs Rs Rr L M] parameter vector
We = 376.99;
Rs = real(x(2));
L = imag(x(2))/We;
Rr = L/real(x(1));
M = sqrt(Rr*real(x(3))/We + L*L);
pv(1) = real(x(1))*Rs;
pv(2) = Rs;
pv(3) = Rr;
pv(4) = L;
pv(5) = M;
retf
```

```
////////////////////////////////////////////////////////////////////////////
//  Program rsfdata.udf (in [cho.scatter.rsconst])                        //
//                                                                        //
//  Estimator 3 (LLSE estimator with Known Rs)                            //
//                                                                        //
//  Given matrix datmat and Rs, output pv, the estimated fundamental      //
//  parameter vector                                                      //
////////////////////////////////////////////////////////////////////////////


//[pv,x,S,A,WeS] = rsfdata(datmat,Rs)
We = 376.99;
// We use fake data to check if our least squares algorithm works properly.
// Assume that a data matrix with the rows consisting of
// datmat(T,I,Pe,Pm,pf,rpm,V) is passed to this function.  The least
// squares estimate x will be returned.  Realizing that matrix
// datmat = | v1 v2 v3 v4 v5 v6 v7| with each vi's corresponding to
//          | T  I  Pe Pm pf rpm v|,
// we can conveniently write A as a linear combo of these vectors.
// Now, we break x into real and imaginary part, forcing x1 to be purely real.
// WeS = We - 2*Wm = We - 4*PI*rpm/60
[nrows, ncols] = size(datmat);
WeS = We*ONES(nrows,1) - (PI/15)*datmat(:,6);
S = WeS/We;
for i=1:nrows,...
V = datmat(i,7);...
pf = datmat(i,5);...
Is = datmat(i,2);...
Ispf = Is*pf;...
krt = sqrt(1-pf*pf);...
Iskrt = Is*krt;...
C(i,1) = Iskrt*WeS(i)*Rs(i);...
C(i,2) = Iskrt*We;...
C(i,3) = Ispf*WeS(i);...
D(i,1) = WeS(i)*(Ispf*Rs(i) - V);...
D(i,2) = Ispf*We;...
D(i,3) = -Iskrt*WeS(i);...
E(i,1) = V - Ispf*Rs(i);...
F(i,1) = Iskrt*Rs(i);...
end
A = [C;D];
Y = [E;F];
x = A\Y;
pv = param(x);
retf
```

```
////////////////////////////////////////////////////////////
//  Program param.udf (in [cho.scatter.rsconst])          //
//                                                        //
//  Given the estimated vector x from Estimator 3         //
//  return the fundamental vector pv = [Rr L M]           //
////////////////////////////////////////////////////////////


//pv = param(x)
// pv = [Rr L M] parameter vector
We = 376.99;
L = x(2);
Rr = L/x(1);
M = sqrt(x(3)*Rr/We + L*L);
pv(1) = Rr;
pv(2) = L;
pv(3) = M;
retf
```

Appendix D

Programs for the Sensitivity Analysis

```
////////////////////////////////////////////////////////////////////
//  Program compdata.mdf (in [cho.scatter.rsparam] directory)     //
//                                                                //
//  Given reference parameter values, computes ideal data         //
////////////////////////////////////////////////////////////////////


//cdata = compdata(paramet)

//  Computes data matrix given the parameter values

long;

Rs = paramet(2);
Rr = paramet(3);
L  = paramet(4);
M  = paramet(5);

We = 376.99;
V = 120.;
Tr = L/Rr;
K = (L*L - M*M);

x1 = Tr;
x2 = Rs + jay*We*L;
x3 = -We*K/Rr + jay*Rs*Tr;


rpm(1,1) = 1795;
for i=2:16,rpm(i,1) = rpm(i-1,1) - 5;

S = ONES(16,1) - rpm/1800;
WeS = We*S;

for i=1:16,...
Y = V*(1 + jay*WeS(i,1)*x1);...
A = x2 + WeS(i,1)*x3;...
X(i,1) = Y/A;...
end

for i=1:16,...
radic = real(x(i))*real(x(i)) + imag(x(i))*imag(x(i));...
Is(i,1) = sqrt(radic);...
pf(i,1) = real(x(i))/Is(i,1);...
end

zip = 0*ones(16,1);
// We'll round off to 4 significant digits
i = 0;
while Is(8,1) < 1000,...
Is = Is*10;...
i = i + 1;...
end
Is = round(Is);
Is = Is/(10**i);
```

```
i = 0;
while pf(8,1) < 1000,...
pf = pf*10;...
i = i + 1;...
end
pf = round(pf);
pf = pf/(10**i);
i = 0;
while rpm(8,1) < 1000,...
rpm = rpm*10;...
i = i + 1;...
end
rpm = round(rpm);
rpm = rpm/(10**i);
vv = V*ones(16,1);
cdata = [zip Is zip zip pf rpm vv];
retf
```

```
////////////////////////////////////////////////////////////////////////
//  Program scat1.r                                                   //
//                                                                    //
//  Programs scat1.r and scat2.r compute numerically the sensitivity  //
//  of Estimators 1 and 2 to noise.                                   //
////////////////////////////////////////////////////////////////////////


//  This program will first take an ideal data matrix generated from
//  an initial set of electrical parameters of an induction motor such that

//      Rs = 0.865 Ohm
//      Rr = 0.563 Ohm
//      L  = 74.13 mH
//      M  = 69.87 mH
//      V  = 120 volts.
//  Rs value has been measured and the rest of the parameters have been
//  estimated from stator 1 rotor 3, using the rsconst llse method.

//  Once we have generated the data matrix, we will test whether we can
//  get the original parameters back by implementing lsdata.udf and
//  r2data.udf on the idealdata matrix.

//  Next, we will perturb the idealdata by maximum and minimum perturbations
//  of the stator current (I), the stator voltage (V), the power factor (pf),
//  and the rpm measurements that we observe at steady-state constant
//  temperature operation of the induction motor.  From these perturbed
//  matrices, we will determine the scatter or the uncertainty of our
//  measurements in the program scat2.r

define 'param.udf';
define 'compdata.udf';
define 'lsdata.udf';
define 'r2data.udf';

//  Generate the idealdata and check the validity of lsdata and r2data fcns.

Rs = 0.865;
Rr = 0.563;
L  = 74.13;
M  = 69.87;
L  = L*.001;
M  = M*.001;
TrRs = L*Rs/Rr;

paramet = [TrRs Rs Rr L M]';
idealdata = compdata(paramet);

pv = lsdata(idealdata);
rpv = r2data(idealdata);

dpv = pv - paramet;
drpv = rpv - paramet;

//  Generate perturbed matrices
```

```
sing = ones(16,1);
zip = 0*sing;

iuv = 0.04*sing;
ium = [zip iuv zip zip zip zip zip];
piud = idealdata + ium;
pild = idealdata - ium;

pfuv = 0.005*sing;
pfum = [zip zip zip zip pfuv zip zip];
ppfud = idealdata + pfum;
ppfld = idealdata - pfum;

revuv = sing;
revum = [zip zip zip zip zip revuv zip];
prevud = idealdata + revum;
prevld = idealdata - revum;

vuv = 0.3*sing;
vum = [zip zip zip zip zip zip vuv];
pvud = idealdata + vum;
pvld = idealdata - vum;

//  Generate corresponding parameters for each of the perturbed matrices

piup = lsdata(piud);
pilp = lsdata(pild);
ppfup = lsdata(ppfud);
ppflp = lsdata(ppfld);
prevup = lsdata(prevud);
prevlp = lsdata(prevld);
pvup = lsdata(pvud);
pvlp = lsdata(pvld);

rpiup = r2data(piud);
rpilp = r2data(pild);
rppfup = r2data(ppfud);
rppflp = r2data(ppfld);
rprevup = r2data(prevud);
rprevlp = r2data(prevld);
rpvup = r2data(pvud);
rpvlp = r2data(pvld);

save 'sscat1';
```

```
////////////////////////////////////////////////////////////////////////////
//  Program scat2.r                                                       //
//                                                                        //
//  This program is used to evaluate the matrices generated by scat1.r    //
////////////////////////////////////////////////////////////////////////////


diary('dscat2');
load 'sscat1';
short;

paramet
pv
rpv
dpv
drpv

//  Compute percentage dev from the original set of parameters

pim = [pilp piup]
for i=1:2,dpim(:,i) = pim(:,i) - pv;end;
for i=1:5,dppim(i,:) = dpim(i,:)/pv(i);end;

ppfm = [ppflp ppfup]
for i=1:2,dppfm(:,i) = ppfm(:,i) - pv;end;
for i=1:5,dpppfm(i,:) = dppfm(i,:)/pv(i);end;

prevm = [prevlp prevup]
for i=1:2,dprevm(:,i) = prevm(:,i) - pv;end;
for i=1:5,dpprevm(i,:) = dprevm(i,:)/pv(i);end;

pvm = [pvlp pvup]
for i=1:2,dpvm(:,i) = pvm(:,i) - pv;end;
for i=1:5,dppvm(i,:) = dpvm(i,:)/pv(i);end;

dpim
dppfm
dprevm
dpvm

dppim
dpppfm
dpprevm
dppvm

rpim = [rpilp rpiup]
for i=1:2,drpim(:,i) = rpim(:,i) - rpv;end;
for i=1:5,drppim(i,:) = drpim(i,:)/rpv(i);end;

rppfm = [rppflp rppfup]
for i=1:2,drppfm(:,i) = rppfm(:,i) - rpv;end;
for i=1:5,drpppfm(i,:) = drppfm(i,:)/rpv(i);end;

rprevm = [rprevlp rprevup]
for i=1:2,drprevm(:,i) = rprevm(:,i) - rpv;end;
```

```
for i=1:5,drpprevm(i,:) = drprevm(i,:)/rpv(i);end;

rpvm = [rpvlp rpvup]
for i=1:2,drpvm(:,i) = rpvm(:,i) - rpv;end;
for i=1:5,drppvm(i,:) = drpvm(i,:)/rpv(i);end;

drpim
drppfm
drprevm
drpvm

drppim
drpppfm
drpprevm
drppvm

//  Compute the sdev of Rs, Rr, L, and M for the results of lsdata

k(1) = max(abs(dpim(2,:)));
k(1) = k(1)*k(1);
k(2) = max(abs(dppfm(2,:)));
k(2) = k(2)*k(2);
k(3) = max(abs(dprevm(2,:)));
k(3) = k(3)*k(3);
k(4) = max(abs(dpvm(2,:)));
k(4) = k(4)*k(4);
k

sdevRs = k(1) + k(2) + k(3) + k(4);
sdevRs = sqrt(sdevRs)
perdevRs = sdevRs/Rs

k(1) = max(abs(dpim(3,:)));
k(1) = k(1)*k(1);
k(2) = max(abs(dppfm(3,:)));
k(2) = k(2)*k(2);
k(3) = max(abs(dprevm(3,:)));
k(3) = k(3)*k(3);
k(4) = max(abs(dpvm(3,:)));
k(4) = k(4)*k(4);
k

sdevRr = k(1) + k(2) + k(3) + k(4);
sdevRr = sqrt(sdevRr)
perdevRr = sdevRr/Rr

k(1) = max(abs(dpim(4,:)));
k(1) = k(1)*k(1);
k(2) = max(abs(dppfm(4,:)));
k(2) = k(2)*k(2);
k(3) = max(abs(dprevm(4,:)));
k(3) = k(3)*k(3);
k(4) = max(abs(dpvm(4,:)));
k(4) = k(4)*k(4);
k

sdevL = k(1) + k(2) + k(3) + k(4);
sdevL = sqrt(sdevL)
perdevL = sdevL/L
```

```
k(1) = max(abs(dpim(5,:)));
k(1) = k(1)*k(1);
k(2) = max(abs(dppfm(5,:)));
k(2) = k(2)*k(2);
k(3) = max(abs(dprevm(5,:)));
k(3) = k(3)*k(3);
k(4) = max(abs(dpvm(5,:)));
k(4) = k(4)*k(4);
k

sdevM = k(1) + k(2) + k(3) + k(4);
sdevM = sqrt(sdevM)
perdevM = sdevM/M

//  Compute the sdev of Rs, Rr, L, and M for the results of r2data

k(1) = max(abs(drpim(2,:)));
k(1) = k(1)*k(1);
k(2) = max(abs(drppfm(2,:)));
k(2) = k(2)*k(2);
k(3) = max(abs(drprevm(2,:)));
k(3) = k(3)*k(3);
k(4) = max(abs(drpvm(2,:)));
k(4) = k(4)*k(4);
k

rsdevRs = k(1) + k(2) + k(3) + k(4);
rsdevRs = sqrt(rsdevRs)
rperdevRs = rsdevRs/Rs

k(1) = max(abs(drpim(3,:)));
k(1) = k(1)*k(1);
k(2) = max(abs(drppfm(3,:)));
k(2) = k(2)*k(2);
k(3) = max(abs(drprevm(3,:)));
k(3) = k(3)*k(3);
k(4) = max(abs(drpvm(3,:)));
k(4) = k(4)*k(4);
k

rsdevRr = k(1) + k(2) + k(3) + k(4);
rsdevRr = sqrt(rsdevRr)
rperdevRr = rsdevRr/Rr

k(1) = max(abs(drpim(4,:)));
k(1) = k(1)*k(1);
k(2) = max(abs(drppfm(4,:)));
k(2) = k(2)*k(2);
k(3) = max(abs(drprevm(4,:)));
k(3) = k(3)*k(3);
k(4) = max(abs(drpvm(4,:)));
k(4) = k(4)*k(4);
k

rsdevL = k(1) + k(2) + k(3) + k(4);
rsdevL = sqrt(rsdevL)
rperdevL = rsdevL/L

k(1) = max(abs(drpim(5,:)));
k(1) = k(1)*k(1);
```

```
k(2) = max(abs(drppfm(5,:)));
k(2) = k(2)*k(2);
k(3) = max(abs(drprevm(5,:)));
k(3) = k(3)*k(3);
k(4) = max(abs(drpvm(5,:)));
k(4) = k(4)*k(4);
k

rsdevM = k(1) + k(2) + k(3) + k(4);
rsdevM = sqrt(rsdevM)
rperdevM = rsdevM/M

diary(0)
```

```
////////////////////////////////////////////////////////////////////////
//  Program cscat1.r (in [cho.scatter.rsconst])                       //
//                                                                    //
//  Program cscat1.r and cscat2.r compute numerically the sensitivity //
//  of Estimator 3 to noise.                                          //
////////////////////////////////////////////////////////////////////////


//  This program will first take an ideal data matrix generated from an
//  initial set of electrical parameters of an induction motor such that

//       Rs = 0.865 Ohm
//       Rr = 0.563 Ohm
//       L  = 74.13 mH
//       M  = 69.87 mH
//       V  = 120 volts.
//  Rs value has been measured and the rest of the parameters have been
//  estimated from stator 1 rotor 3, using the rsconst llse method.

//  Once we have generated the data matrix, we will test whether we can
//  get the original parameters back by implementing rsfdata.udf on
//  the idealdata matrix.

//  Next, we will perturb the idealdata by maximum and minimum perturbations
//  of the stator current (I), the stator voltage (V), the power factor (pf),
//  stator resistance, and the rpm measurements that we observe at
//  steady-state constant temperature operation of the induction motor.
//  From these perturbed matrices, we will determine the scatter or the
//  uncertainty of our measurements in program cscat2.r

define 'param.udf';
define 'compdata.udf';
define 'rsfdata.udf';

//  Generate the idealdata and check the validity of lsdata and r2data fcns.

Rs = 0.865*ones(16,1);
Rr = 0.563;
L  = 74.13;
M  = 69.87;
L  = L*.001;
M  = M*.001;

paramet = [Rr L M]';

//  Need to augment the parameter vector by two zeroes so that the compdata
//  will work properly.

augparam = [0 0 Rr L M]';
idealdata = compdata(augparam,Rs(1));

pv = rsfdata(idealdata,Rs);

dpv = pv - paramet;
```

```
//  Generate perturbed matrices

sing = ones(16,1);
zip = 0*sing;

iuv = 0.04*sing;
ium = [zip iuv zip zip zip zip zip];
piud = idealdata + ium;
pild = idealdata - ium;

pfuv = 0.005*sing;
pfum = [zip zip zip zip pfuv zip zip];
ppfud = idealdata + pfum;
ppfld = idealdata - pfum;

revuv = sing;
revum = [zip zip zip zip zip revuv zip];
prevud = idealdata + revum;
prevld = idealdata - revum;

vuv = 0.3*sing;
vum = [zip zip zip zip zip zip vuv];
pvud = idealdata + vum;
pvld = idealdata - vum;

rsuv = Rs + 0.002*sing;
rslv = Rs - 0.002*sing;

//  Generate corresponding parameters for each of the perturbed matrices

piup = rsfdata(piud,Rs);
pilp = rsfdata(pild,Rs);
ppfup = rsfdata(ppfud,Rs);
ppflp = rsfdata(ppfld,Rs);
prevup = rsfdata(prevud,Rs);
prevlp = rsfdata(prevld,Rs);
pvup = rsfdata(pvud,Rs);
pvlp = rsfdata(pvld,Rs);
prsup = rsfdata(idealdata,rsuv);
prslp = rsfdata(idealdata,rslv);

save 'scscatl';
```

```
////////////////////////////////////////////////////////////////////////////
//  Program cscat2.r (in [cho.scatter.rsconst])                           //
//                                                                        //
//  This program is used to evaluate the matrices generated by cscat1.r //
////////////////////////////////////////////////////////////////////////////


diary('dcscat2');
load 'scscat1';
short;

paramet
pv
dpv

//  Compute percentage dev from the original set of parameters

pim = [pilp piup]
for i=1:2,dpim(:,i) = pim(:,i) - pv;end;
for i=1:3,dppim(i,:) = dpim(i,:)/pv(i);end;

ppfm = [ppflp ppfup]
for i=1:2,dppfm(:,i) = ppfm(:,i) - pv;end;
for i=1:3,dpppfm(i,:) = dppfm(i,:)/pv(i);end;

prevm = [prevlp prevup]
for i=1:2,dprevm(:,i) = prevm(:,i) - pv;end;
for i=1:3,dpprevm(i,:) = dprevm(i,:)/pv(i);end;

pvm = [pvlp pvup]
for i=1:2,dpvm(:,i) = pvm(:,i) - pv;end;
for i=1:3,dppvm(i,:) = dpvm(i,:)/pv(i);end;

rsm = [prslp prsup]
for i=1:2,drsm(:,i) = rsm(:,i) - pv;end;
for i=1:3,dprsm(i,:) = drsm(i,:)/pv(i);end;

dpim
dppfm
dprevm
dpvm
drsm

dppim
dpppfm
dpprevm
dppvm
dprsm

//  Compute the sdev of Rr, L, and M for the results of rsfdata

k(1) = max(abs(dpim(1,:)));
k(1) = k(1)*k(1);
k(2) = max(abs(dppfm(1,:)));
k(2) = k(2)*k(2);
```

```
k(3) = max(abs(dprevm(1,:)));
k(3) = k(3)*k(3);
k(4) = max(abs(dpvm(1,:)));
k(4) = k(4)*k(4);
k(5) = max(abs(drsm(1,:)));
k(5) = k(5)*k(5);
k

sdevRr = k(1) + k(2) + k(3) + k(4) + k(5);
sdevRr = sqrt(sdevRr)
perdevRr = sdevRr/Rr

k(1) = max(abs(dpim(2,:)));
k(1) = k(1)*k(1);
k(2) = max(abs(dppfm(2,:)));
k(2) = k(2)*k(2);
k(3) = max(abs(dprevm(2,:)));
k(3) = k(3)*k(3);
k(4) = max(abs(dpvm(2,:)));
k(4) = k(4)*k(4);
k(5) = max(abs(drsm(2,:)));
k(5) = k(5)*k(5);
k

sdevL = k(1) + k(2) + k(3) + k(4) + k(5);
sdevL = sqrt(sdevL)
perdevL = sdevL/L

k(1) = max(abs(dpim(3,:)));
k(1) = k(1)*k(1);
k(2) = max(abs(dppfm(3,:)));
k(2) = k(2)*k(2);
k(3) = max(abs(dprevm(3,:)));
k(3) = k(3)*k(3);
k(4) = max(abs(dpvm(3,:)));
k(4) = k(4)*k(4);
k(5) = max(abs(drsm(3,:)));
k(5) = k(5)*k(5);
k

sdevM = k(1) + k(2) + k(3) + k(4) + k(5);
sdevM = sqrt(sdevM)
perdevM = sdevM/M

diary(0);
```

Appendix E

Numerical Results of the Sensitivity Analysis

```
////////////////////////////////////////////////////
//  Numerical results of the sensitivities of     //
//  Estimators 1, 2, and 3                        //
//                                                //
////////////////////////////////////////////////////


//  Sensitivity results for Estimator 1

//  Standard deviations SDEVX, and percentage
//  deviations PERDEVX,
//  where X is Rs, Rr, L, or M

  SDEVRS     = 0.6434
  PERDEVRS   = 0.7438

  SDEVRR     = 0.0080
  PERDEVRR   = 0.0143

  SDEVL      = 0.0010
  PERDEVL    = 0.0142

  SDEVM      = 0.0013
  PERDEVM    = 0.0189


//  Sensitivity results for Estimator 2

//  Standard deviations RSDEVX, and percentage
//  deviations RPERDEVX,
//  where X is Rs, Rr, L, or M


  RSDEVRS    = 0.2670
  RPERDEVRS  = 0.3087

  RSDEVRR    = 0.0211
  RPERDEVRR  = 0.0376

  RSDEVL     = 9.5939D-04
  RPERDEVL   = 0.0129

  RSDEVM     = 0.0012
  RPERDEVM   = 0.0166
```

```
//  Sensitivity results for Estimator 3

//  Standard deviations SDEVX, and percentage
//  deviations PERDEVX,
//  where X is Rs, Rr, L, or M

  SDEVRR      = 0.0109
  PERDEVRR    = 0.0194

  SDEVL       = 6.8063D-04
  PERDEVL     = 0.0092

  SDEVM       = 6.9204D-04
  PERDEVM     = 0.0099
```

Appendix F

Programs for Constant-Temperature Experiments

```
////////////////////////////////////////////////////////
//  Program ctlldata.r (in [cho.matfiles.rrbroken])   //
//                                                    //
//  Computes estimates using Estimator 1              //
//  and Estimator 2 at 700 W                          //
//  Note that lsdata.udf = Estimator 1                //
//  and r2data.udf = Estimator 2                      //
////////////////////////////////////////////////////////


//  Working with real data of healthy motor (stator 1, rotor 1)
//  at constant temp.

//  ri = [T Is Pe Pm pf rpm V]

diary('dctlldata');
long;
define 'param.udf'
define 'lsdata.udf'
define 'r2data.udf'
zip = 0*ones(13,1);

//  real data 1

Is1 = [4.280 4.400 4.613 4.926 5.170 5.526 5.906 6.236]';
Is2 = [6.736 7.056 7.513 7.866 8.393]';
Is = [Is1;Is2];
pf1 = [.1751 .2741 .3776 .4766 .5349 .5928 .6349 .6740]';
pf2 = [.7126 .7312 .7496 .7714 .7809]';
pf = [pf1;pf2];

// Note that the following rpm vector will be used throughout
// as the controlling test input

rpm1 = [1795 1790 1785 1780 1775 1770 1765 1760]';
rpm2 = [1755 1750 1745 1740 1735]';
rpm = [rpm1;rpm2];

v1 = [122.7 122.7 122.5 122.6 122.5 122.5 122.6 122.1]';
v2 = [122.2 122.0 122.1 122.7 122.9]';
v = [v1;v2];
r1 = [zip Is zip zip pf rpm v];

//  real data 2

Is1 = [4.313 4.420 4.630 4.863 5.210 5.523 5.910 6.166]';
Is2 = [6.653 7.076 7.553 7.886 8.340]';
Is = [Is1;Is2];
pf1 = [.1665 .2788 .3784 .4566 .5326 .5889 .6374 .6702]';
pf2 = [.7064 .7302 .7528 .7692 .7785]';
pf = [pf1;pf2];
v1 = [123.1 122.9 123.0 122.7 122.7 122.5 122.6 122.3]';
v2 = [122.2 122.3 122.1 122.0 121.9]';
v = [v1;v2];
r2 = [zip Is zip zip pf rpm v];
```

```
//  real data 3

Is1 = [4.423 4.540 4.740 4.980 5.320 5.636 6.016 6.420]';
Is2 = [6.783 7.256 7.653 8.163 8.463]';
Is = [Is1;Is2];
pf1 = [.1550 .2671 .3676 .4533 .5240 .5829 .6292 .6691]';
pf2 = [.7005 .7314 .7510 .7661 .7812]';
pf = [pf1;pf2];
v1 = [125.4 125.3 125.4 125.3 125.3 125.2 125.2 125.1]';
v2 = [125.1 125.0 124.8 124.9 124.5]';
v = [v1;v2];
r3 = [zip Is zip zip pf rpm v];

//  real data 4

Is1 = [4.450 4.566 4.766 5.020 5.310 5.696 6.026 6.403]';
Is2 = [6.823 7.180 7.653 8.073 8.506]';
Is = [Is1;Is2];
pf1 = [.1513 .2567 .3648 .4525 .5248 .5899 .6366 .6746]';
pf2 = [.7032 .7270 .7494 .7669 .7791]';
pf = [pf1;pf2];
v1 = [125.4 125.3 125.3 125.2 125.2 125.1 124.9 124.8]';
v2 = [124.8 124.7 124.6 124.5 124.6]';
v = [v1;v2];
r4 = [zip Is zip zip pf rpm v];

//  do lsdata and r2data on each of the real data to obtain the least
//  square and recursive least square approximation of the parameters

[pv1,x1,sl1] = lsdata(r1);
[rpv1,rx1,rsl1] = r2data(r1);
[pv2,x2,sl2] = lsdata(r2);
[rpv2,rx2,rsl2] = r2data(r2);
[pv3,x3,sl3] = lsdata(r3);
[rpv3,rx3,rsl3] = r2data(r3);
[pv4,x4,sl4] = lsdata(r4);
[rpv4,rx4,rsl4] = r2data(r4);

//  Take the average of the data, called inave, and perform lsdata and
//  r2data on this inave matrix

inave = 0.25*(r1 + r2 + r3 + r4);
[pvinave,xinave,sinave] = lsdata(inave);
[rpvinave,rxinave,rsinave] = r2data(inave);

//  Print out the lse results

pv = [pv1 pv2 pv3 pv4]
pvinave

//  Give statistics on the lse output results compared to itself; namely,
//  the average, the minimum, and the maximum of each parameter.

pvoutave = 0.25*(pv1 + pv2 + pv3 + pv4);
for i=1:5, llim(i,1) = min(pv(i,:));end;
for i=1:5, ulim(i,1) = max(pv(i,:));end;
outstat = [pvoutave llim ulim]

//  Print out the rlse results
```

```
rpv = [rpv1 rpv2 rpv3 rpv4]
rpvinave

//  Give statistics on the rlse output results; namely, the average, the
//  minimum, and the maximum of each parameter

rpvoutave = 0.25*(rpv1 + rpv2 + rpv3 + rpv4);
for i=1:5, llim(i,1) = min(rpv(i,:));end;
for i=1:5, ulim(i,1) = max(rpv(i,:));end;
routstat = [rpvoutave llim ulim]

//  Compute the normalized pvinave sensitivity and statistics

for i=1:4, dpvin(:,i) = pv(:,i) - pvinave;end;
for i=1:5, dpvin(i,:) = dpvin(i,:)/pvinave(i);end;
dpvin
dpvinave = 0.25*(dpvin(:,1) + dpvin(:,2) + dpvin(:,3) + dpvin(:,4));
adpvinave=(abs(dpvin(:,1))+abs(dpvin(:,2))+abs(dpvin(:,3))+abs(dpvin(:,4)))/4.;
for i=1:5, llim(i,1) = min(dpvin(i,:));end;
for i=1:5, ulim(i,1) = max(dpvin(i,:));end;
dpvinstat = [dpvinave adpvinave llim ulim]

//  Compute the normalized rpvinave sensitivity and statistics

for i=1:4, drpvin(:,i) = rpv(:,i) - rpvinave;end;
for i=1:5, drpvin(i,:) = drpvin(i,:)/rpvinave(i);end;
drpvin
drpvinave = 0.25*(drpvin(:,1)+drpvin(:,2)+drpvin(:,3)+drpvin(:,4));
adrpvinave=(abs(drpvin(:,1))+abs(drpvin(:,2))+abs(drpvin(:,3)));
adrpvinave = (adrpvinave + abs(drpvin(:,4)))*0.25;
for i=1:5, llim(i,1) = min(drpvin(i,:));end;
for i=1:5, ulim(i,1) = max(drpvin(i,:));end;
drpvinstat = [drpvinave adrpvinave llim ulim]

//  Compute the normalized pvoutave sensitivity and statistics

for i=1:4, dpvout(:,i) = pv(:,i) - pvoutave;end;
for i=1:5, dpvout(i,:) = dpvout(i,:)/pvoutave(i);end;
dpvout;
dpvoutave = 0.25*(dpvout(:,1)+dpvout(:,2)+dpvout(:,3)+dpvout(:,4));
adpvoutave=(abs(dpvout(:,1))+abs(dpvout(:,2))+abs(dpvout(:,3)));
adpvoutave = (adpvoutave + abs(dpvout(:,4)))*0.25;
for i=1:5, llim(i,1) = min(dpvout(i,:));end;
for i=1:5, ulim(i,1) = max(dpvout(i,:));end;
dpvoutstat = [dpvoutave adpvoutave llim ulim];

//  Compute the normalized rpvoutave sensitivity and statistics

for i=1:4, drpvout(:,i) = rpv(:,i) - rpvoutave;end;
for i=1:5, drpvout(i,:) = drpvout(i,:)/rpvoutave(i);end;
drpvout;
drpvoutave = 0.25*(drpvout(:,1)+drpvout(:,2)+drpvout(:,3)+drpvout(:,4));
adrpvoutave = (abs(drpvout(:,1))+abs(drpvout(:,2))+abs(drpvout(:,3)));
adrpvoutave = (adrpvoutave + abs(drpvout(:,4)))*0.25;
for i=1:5, llim(i,1) = min(drpvout(i,:));end;
for i=1:5, ulim(i,1) = max(drpvout(i,:));end;
drpvoutstat = [drpvoutave adrpvoutave llim ulim];

save 'sctl1data';
```

```
diary(0);
```

```
////////////////////////////////////////////////////////////
//  Program ctb21data2.r (in [cho.matfiles.rrbroken])     //
//                                                        //
//  Computes estimates using Estimator 1                  //
//  and Estimator 2 at 700 W                              //
//  Note that 1sdata.udf = Estimator 1 and                //
//  r2data.udf = Estimator 2)                             //
////////////////////////////////////////////////////////////


//  Working with broken bar data (rotor 2, stator 1) at const. temp.
//  ri = [T Is Pe Pm pf rpm V]
//  Date:  April 30, 1988.  Dynamometer fixed, Bar cut at the other end
//  on April 15, 1988.

diary('dctb21data2');
long;
define 'param.udf'
define '1sdata.udf'
define 'r2data.udf'
zip = 0*ones(16,1);

//  real data 1

Is1 = [4.406 4.501 4.713 4.976 5.205 5.563 5.990 6.325]';
Is2 = [6.646 7.073 7.478 7.948 8.436 8.821 9.226 9.610]';
Is = [Is1;Is2];
pf1 = [.1810 .2745 .3887 .4604 .5269 .5919 .6392 .6745]';
pf2 = [.7020 .7255 .7459 .7643 .7768 .7912 .8002 .8072]';
pf = [pf1;pf2];

// Note that the following rpm vector will be used throughout
// as the controlling test input

rpm1 = [1795 1790 1785 1780 1775 1770 1765 1760]';
rpm2 = [1755 1750 1745 1740 1735 1730 1725 1720]';
rpm = [rpm1;rpm2];

v1 = [125.6 125.6 125.5 125.4 125.4 125.3 125.3 125.2]';
v2 = [125.0 124.9 124.9 124.8 124.5 124.5 124.5 124.3]';
v = [v1;v2];
r1 = [zip Is zip zip pf rpm v];

//  real data 2

Is1 = [4.403 4.548 4.631 4.931 5.225 5.556 5.983 6.346]';
Is2 = [6.668 7.081 7.451 7.940 8.320 8.821 9.180 9.620]';
Is = [Is1;Is2];
pf1 = [.1692 .2877 .3584 .4510 .5219 .5846 .6363 .6730]';
pf2 = [.6995 .7282 .7474 .7646 .7836 .7909 .7994 .8056]';
pf = [pf1;pf2];
v1 = [125.7 125.6 125.6 125.5 125.4 125.3 125.2 125.1]';
v2 = [125.0 125.0 124.8 124.7 124.7 124.7 124.7 124.7]';
v = [v1;v2];
r2 = [zip Is zip zip pf rpm v];
```

```
//  real data 3

Is1 = [4.215 4.333 4.520 4.760 5.041 5.456 5.785 6.148]';
Is2 = [6.531 6.950 7.358 7.805 8.228 8.603 9.018 9.535]';
Is = [Is1;Is2];
pf1 = [.1880 .2905 .3886 .4697 .5412 .6058 .6505 .6874]';
pf2 = [.7139 .7340 .7516 .7700 .7829 .7923 .8027 .8115]';
pf = [pf1;pf2];
v1 = [123.9 123.7 123.5 123.6 123.6 123.5 123.6 123.3]';
v2 = [123.3 123.3 123.4 123.5 123.5 123.4 123.3 123.3]';
v = [v1;v2];
r3 = [zip Is zip zip pf rpm v];

//  real data 4

Is1 = [4.338 4.460 4.688 4.910 5.235 5.560 5.876 6.240]';
Is2 = [6.640 7.126 7.545 7.991 8.385 8.795 9.246 9.688]';
Is = [Is1;Is2];
pf1 = [.1451 .2706 .3733 .4529 .5273 .5782 .6195 .6595]';
pf2 = [.6961 .7248 .7463 .7639 .7795 .7908 .7973 .8069]';
pf = [pf1;pf2];
v1 = [125.4 125.5 125.4 125.1 125.1 125.2 125.2 125.1]';
v2 = [125.2 125.1 125.1 125.0 124.5 124.8 124.6 124.6]';
v = [v1;v2];
r4 = [zip Is zip zip pf rpm v];

//  do lsdata and r2data on each of the real data to obtain the least
//  square and recursive least square approximation of the parameters

[pv1,x1] = lsdata(r1);
[rpv1,rx1] = r2data(r1);
[pv2,x2] = lsdata(r2);
[rpv2,rx2] = r2data(r2);
[pv3,x3] = lsdata(r3);
[rpv3,rx3] = r2data(r3);
[pv4,x4] = lsdata(r4);
[rpv4,rx4] = r2data(r4);

//  Take the average of the data, called inave, and perform lsdata and
//  r2data on this inave matrix

inave = (r1 + r2 + r3 + r4)/4.;
[bpvinave,bxinave] = lsdata(inave);
[brpvinave,brxinave] = r2data(inave);

//  Print out the lse results

bpv = [pv1 pv2 pv3 pv4]
bpvinave

//  Give statistics on the lse output results compared to itself; namely,
//  the average, the minimum, and the maximum of each parameter.

bpvoutave = (pv1 + pv2 + pv3 + pv4)/4.;
for i=1:5, llim(i,1) = min(bpv(i,:));end;
for i=1:5, ulim(i,1) = max(bpv(i,:));end;
boutstat = [bpvoutave llim ulim]

//  Print out the rlse results
```

```
brpv = [rpv1 rpv2 rpv3 rpv4]
brpvinave

//  Give statistics on the rlse output results; namely, the average, the
//  minimum, and the maximum of each parameter

brpvoutave = (rpv1 + rpv2 + rpv3 + rpv4)/4.;
for i=1:5, llim(i,1) = min(brpv(i,:));end;
for i=1:5, ulim(i,1) = max(brpv(i,:));end;
broutstat = [brpvoutave llim ulim]

//  Compute the normalized bpvinave sensitivity and statistics

for i=1:4, dbpvin(:,i) = bpv(:,i) - bpvinave;end;
for i=1:5, dbpvin(i,:) = dbpvin(i,:)/bpvinave(i);end;
dbpvin
dbpvinave = (dbpvin(:,1)+dbpvin(:,2)+dbpvin(:,3)+dbpvin(:,4))/4.;
adbpvinave=abs(dbpvin(:,1))+abs(dbpvin(:,2))+abs(dbpvin(:,3));
adbpvinave = (adbpvinave + abs(dbpvin(:,4)))/4.;
for i=1:5, llim(i,1) = min(dbpvin(i,:));end;
for i=1:5, ulim(i,1) = max(dbpvin(i,:));end;
dbpvinstat = [dbpvinave adbpvinave llim ulim]

//  Compute the normalized brpvinave sensitivity and statistics

for i=1:4, dbrpvin(:,i) = brpv(:,i) - brpvinave;end;
for i=1:5, dbrpvin(i,:) = dbrpvin(i,:)/brpvinave(i);end;
dbrpvin
dbrpvinave = (dbrpvin(:,1)+dbrpvin(:,2)+dbrpvin(:,3)+dbrpvin(:,4))/4.;
adbrpvinave=abs(dbrpvin(:,1))+abs(dbrpvin(:,2))+abs(dbrpvin(:,3));
adbrpvinave = (adbrpvinave + abs(dbrpvin(:,4)))/4.;
for i=1:5, llim(i,1) = min(dbrpvin(i,:));end;
for i=1:5, ulim(i,1) = max(dbrpvin(i,:));end;
dbrpvinstat = [dbrpvinave adbrpvinave llim ulim]

//  Compute the normalized bpvoutave sensitivity and statistics

for i=1:4, dbpvout(:,i) = bpv(:,i) - bpvoutave;end;
for i=1:5, dbpvout(i,:) = dbpvout(i,:)/bpvoutave(i);end;
dbpvout;
dbpvoutave = (dbpvout(:,1)+dbpvout(:,2)+dbpvout(:,3)+dbpvout(:,4))/4.;
adbpvoutave=abs(dbpvout(:,1))+abs(dbpvout(:,2))+abs(dbpvout(:,3));
adbpvoutave = (adbpvoutave + abs(dbpvout(:,4)))/4.;
for i=1:5, llim(i,1) = min(dbpvout(i,:));end;
for i=1:5, ulim(i,1) = max(dbpvout(i,:));end;
dbpvoutstat = [dbpvoutave adbpvoutave llim ulim];

//  Compute the normalized brpvoutave sensitivity and statistics

for i=1:4, dbrpvout(:,i) = brpv(:,i) - brpvoutave;end;
for i=1:5, dbrpvout(i,:) = dbrpvout(i,:)/brpvoutave(i);end;
dbrpvout;
dbrpvoutave = (dbrpvout(:,1)+dbrpvout(:,2)+dbrpvout(:,3)+dbrpvout(:,4))/4.;
adbrpvoutave = abs(dbrpvout(:,1))+abs(dbrpvout(:,2))+abs(dbrpvout(:,3));
adbrpvoutave = (adbrpvoutave + abs(dbrpvout(:,4)))/4;
for i=1:5, llim(i,1) = min(dbrpvout(i,:));end;
for i=1:5, ulim(i,1) = max(dbrpvout(i,:));end;
dbrpvoutstat = [dbrpvoutave adbrpvoutave llim ulim];
```

```
//  Load healthy rotor data from sctlldata.dat

load 'sct31data2'

inm = [bpv bpvinave];
rinm = [brpv brpvinave];

//  Compute the normalized inm sensitivity and statistics

for i=1:5, dinm(:,i) = inm(:,i) - pvinave;end;
for i=1:5, dinm(i,:) = dinm(i,:)/pvinave(i);end;
dinm
dinmave = (dinm(:,1)+dinm(:,2)+dinm(:,3)+dinm(:,4))/4.;
adinmave = (abs(dinm(:,1))+abs(dinm(:,2))+abs(dinm(:,3))+abs(dinm(:,4)))/4.;
for i=1:5, llim(i,1) = min(dinm(i,:));end;
for i=1:5, ulim(i,1) = max(dinm(i,:));end;
dinmstat = [dinmave adinmave llim ulim]

//  Compute the normalized inm sensitivity and statistics

for i=1:5, drinm(:,i) = rinm(:,i) - rpvinave;end;
for i=1:5, drinm(i,:) = drinm(i,:)/rpvinave(i);end;
drinm
drinmave = (drinm(:,1)+drinm(:,2)+drinm(:,3)+drinm(:,4))/4.;
adrinmave = (abs(drinm(:,1))+abs(drinm(:,2))+abs(drinm(:,3)));
adrinmave = (adrinmave+abs(drinm(:,4)))/4.;
for i=1:5, llim(i,1) = min(drinm(i,:));end;
for i=1:5, ulim(i,1) = max(drinm(i,:));end;
drinmstat = [drinmave adrinmave llim ulim]

save 'sctb21data2';
diary(0);
```

```
////////////////////////////////////////////////////////////
//  Program ct31data2.r (in [cho.matfiles.rrbroken])      //
//                                                        //
//  Computes estimates using Estimator 1                  //
//  and Estimator 2 at 700 W                              //
//  Note that lsdata.udf = Estimator 1 and                //
//  r2data.udf = Estimator 2)                             //
////////////////////////////////////////////////////////////


//  Working with real data of healthy motor (stator 1, rotor 3)
//  at constant temp.
//  Date:  April 30, 1988.  Dynamometer fixed.

//  ri = [T Is Pe Pm pf rpm V]

diary('dct31data2');
long;
define 'param.udf'
define 'lsdata.udf'
define 'r2data.udf'
zip = 0*ones(16,1);

//  real data 1

Is1 = [4.323 4.431 4.663 4.943 5.231 5.628 5.913 6.353]';
Is2 = [6.748 7.130 7.536 7.965 8.433 8.848 9.230 9.656]';
Is = [Is1;Is2];
pf1 = [.1642 .2727 .3785 .4735 .5375 .6050 .6405 .6841]';
pf2 = [.7147 .7351 .7575 .7740 .7871 .7980 .8048 .8141]';
pf = [pf1;pf2];

// Note that the following rpm vector will be used throughout
// as the controlling test input

rpm1 = [1795 1790 1785 1780 1775 1770 1765 1760]';
rpm2 = [1755 1750 1745 1740 1735 1730 1725 1720]';
rpm = [rpm1;rpm2];

v1 = [123.5 123.4 123.6 123.5 123.5 123.4 123.1 123.2]';
v2 = [123.2 123.0 123.0 122.9 123.0 122.6 122.9 122.6]';
v = [v1;v2];
r1 = [zip Is zip zip pf rpm v];

//  real data 2

Is1 = [4.388 4.485 4.720 4.980 5.263 5.650 5.995 6.375]';
Is2 = [6.816 7.226 7.660 8.055 8.488 8.860 9.385 9.768]';
Is = [Is1;Is2];
pf1 = [.1774 .2832 .3872 .4710 .5362 .5985 .6427 .6777]';
pf2 = [.7116 .7370 .7565 .7708 .7855 .7976 .8078 .8145]';
pf = [pf1;pf2];
v1 = [124.2 123.9 124.3 124.1 124.1 124.2 124.1 124.0]';
v2 = [124.0 124.0 123.8 123.8 123.6 123.5 123.4 123.4]';
v = [v1;v2];
```

```
r2 = [zip Is zip zip pf rpm v];

//  real data 3

Is1 = [4.526 4.650 4.830 5.106 5.391 5.768 6.156 6.558]';
Is2 = [6.908 7.358 7.763 8.163 8.690 9.005 9.525 9.875]';
Is = [Is1;Is2];
pf1 = [.1704 .2844 .3747 .4562 .5242 .5898 .6414 .6774]';
pf2 = [.7057 .7341 .7544 .7689 .7845 .7934 .8041 .8126]';
pf = [pf1;pf2];
v1 = [126.0 126.0 125.9 126.0 125.8 125.8 125.6 125.6]';
v2 = [125.5 125.3 125.4 125.4 125.3 125.3 125.3 125.1]';
v = [v1;v2];
r3 = [zip Is zip zip pf rpm v];

//  real data 4

Is1 = [4.571 4.681 4.861 5.118 5.386 5.751 6.105 6.570]';
Is2 = [6.903 7.341 7.701 8.235 8.625 9.028 9.455 9.943]';
Is = [Is1;Is2];
pf1 = [.1574 .2693 .3650 .4519 .5264 .5849 .6293 .6751]';
pf2 = [.7028 .7306 .7468 .7670 .7807 .7941 .8029 .8104]';
pf = [pf1;pf2];
v1 = [126.4 126.5 126.3 126.3 126.0 126.2 126.1 126.0]';
v2 = [125.9 125.9 125.9 125.8 125.7 125.7 125.4 125.5]';
v = [v1;v2];
r4 = [zip Is zip zip pf rpm v];

//  do lsdata and r2data on each of the real data to obtain the least
//  square and recursive least square approximation of the parameters

[pv1,x1,sl1] = lsdata(r1);
[rpv1,rx1,rsl1] = r2data(r1);
[pv2,x2,sl2] = lsdata(r2);
[rpv2,rx2,rsl2] = r2data(r2);
[pv3,x3,sl3] = lsdata(r3);
[rpv3,rx3,rsl3] = r2data(r3);
[pv4,x4,sl4] = lsdata(r4);
[rpv4,rx4,rsl4] = r2data(r4);

//  Take the average of the data, called inave, and perform lsdata and
//  r2data on this inave matrix

inave = 0.25*(r1 + r2 + r3 + r4);
[pvinave,xinave,sinave] = lsdata(inave);
[rpvinave,rxinave,rsinave] = r2data(inave);

//  Print out the lse results

pv = [pv1 pv2 pv3 pv4]
pvinave

//  Give statistics on the lse output results compared to itself; namely,
//  the average, the minimum, and the maximum of each parameter.

pvoutave = 0.25*(pv1 + pv2 + pv3 + pv4);
for i=1:5, llim(i,1) = min(pv(i,:));end;
for i=1:5, ulim(i,1) = max(pv(i,:));end;
outstat = [pvoutave llim ulim]
```

```
//  Print out the rlse results

rpv = [rpv1 rpv2 rpv3 rpv4]
rpvinave

//  Give statistics on the rlse output results; namely, the average, the
//  minimum, and the maximum of each parameter

rpvoutave = 0.25*(rpv1 + rpv2 + rpv3 + rpv4);
for i=1:5, llim(i,1) = min(rpv(i,:));end;
for i=1:5, ulim(i,1) = max(rpv(i,:));end;
routstat = [rpvoutave llim ulim]

//  Compute the normalized pvinave sensitivity and statistics

for i=1:4, dpvin(:,i) = pv(:,i) - pvinave;end;
for i=1:5, dpvin(i,:) = dpvin(i,:)/pvinave(i);end;
dpvin
dpvinave = 0.25*(dpvin(:,1) + dpvin(:,2) + dpvin(:,3) + dpvin(:,4));
adpvinave=(abs(dpvin(:,1))+abs(dpvin(:,2))+abs(dpvin(:,3))+abs(dpvin(:,4)))/4.;
for i=1:5, llim(i,1) = min(dpvin(i,:));end;
for i=1:5, ulim(i,1) = max(dpvin(i,:));end;
dpvinstat = [dpvinave adpvinave llim ulim]

//  Compute the normalized rpvinave sensitivity and statistics

for i=1:4, drpvin(:,i) = rpv(:,i) - rpvinave;end;
for i=1:5, drpvin(i,:) = drpvin(i,:)/rpvinave(i);end;
drpvin
drpvinave = 0.25*(drpvin(:,1)+drpvin(:,2)+drpvin(:,3)+drpvin(:,4));
adrpvinave=(abs(drpvin(:,1))+abs(drpvin(:,2))+abs(drpvin(:,3)));
adrpvinave = (adrpvinave + abs(drpvin(:,4)))*0.25;
for i=1:5, llim(i,1) = min(drpvin(i,:));end;
for i=1:5, ulim(i,1) = max(drpvin(i,:));end;
drpvinstat = [drpvinave adrpvinave llim ulim]

//  Compute the normalized pvoutave sensitivity and statistics

for i=1:4, dpvout(:,i) = pv(:,i) - pvoutave;end;
for i=1:5, dpvout(i,:) = dpvout(i,:)/pvoutave(i);end;
dpvout;
dpvoutave = 0.25*(dpvout(:,1)+dpvout(:,2)+dpvout(:,3)+dpvout(:,4));
adpvoutave=(abs(dpvout(:,1))+abs(dpvout(:,2))+abs(dpvout(:,3)));
adpvoutave = (adpvoutave + abs(dpvout(:,4)))*0.25;
for i=1:5, llim(i,1) = min(dpvout(i,:));end;
for i=1:5, ulim(i,1) = max(dpvout(i,:));end;
dpvoutstat = [dpvoutave adpvoutave llim ulim];

//  Compute the normalized rpvoutave sensitivity and statistics

for i=1:4, drpvout(:,i) = rpv(:,i) - rpvoutave;end;
for i=1:5, drpvout(i,:) = drpvout(i,:)/rpvoutave(i);end;
drpvout;
drpvoutave = 0.25*(drpvout(:,1)+drpvout(:,2)+drpvout(:,3)+drpvout(:,4));
adrpvoutave = (abs(drpvout(:,1))+abs(drpvout(:,2))+abs(drpvout(:,3)));
adrpvoutave = (adrpvoutave + abs(drpvout(:,4)))*0.25;
for i=1:5, llim(i,1) = min(drpvout(i,:));end;
for i=1:5, ulim(i,1) = max(drpvout(i,:));end;
drpvoutstat = [drpvoutave adrpvoutave llim ulim];
```

```
save 'sct31data2';
diary(0);
```

```
////////////////////////////////////////////////////////////////////
//  Program ct1ldata.r (in [cho.matfiles.rrbroken.rsconst])       //
//                                                                //
//  Computes estimates using Estimator 3 at 700W                  //
//  Note that rsfdata.udf = Estimator 3                           //
////////////////////////////////////////////////////////////////////


//  Working with real data of healthy motor (stator 1, rotor 2)
//  at constant temp.

//  ri = [T Is Pe Pm pf rpm V]

diary('dctrs1ldata');
long;
define 'param.udf'
define 'rsfdata.udf'
zip = 0*ones(13,1);

//  real data 1

Is1 = [4.280 4.400 4.613 4.926 5.170 5.526 5.906 6.236]';
Is2 = [6.736 7.056 7.513 7.866 8.393]';
Is = [Is1;Is2];
pf1 = [.1751 .2741 .3776 .4766 .5349 .5928 .6349 .6740]';
pf2 = [.7126 .7312 .7496 .7714 .7809]';
pf = [pf1;pf2];

// Note that the following rpm vector will be used throughout
// as the controlling test input

rpm1 = [1795 1790 1785 1780 1775 1770 1765 1760]';
rpm2 = [1755 1750 1745 1740 1735]';
rpm = [rpm1;rpm2];

v1 = [122.7 122.7 122.5 122.6 122.5 122.5 122.6 122.1]';
v2 = [122.2 122.0 122.1 122.7 122.9]';
v = [v1;v2];
r1 = [zip Is zip zip pf rpm v];

//  real data 2

Is1 = [4.313 4.420 4.630 4.863 5.210 5.523 5.910 6.166]';
Is2 = [6.653 7.076 7.553 7.886 8.340]';
Is = [Is1;Is2];
pf1 = [.1665 .2788 .3784 .4566 .5326 .5889 .6374 .6702]';
pf2 = [.7064 .7302 .7528 .7692 .7785]';
pf = [pf1;pf2];
v1 = [123.1 122.9 123.0 122.7 122.7 122.5 122.6 122.3]';
v2 = [122.2 122.3 122.1 122.0 121.9]';
v = [v1;v2];
r2 = [zip Is zip zip pf rpm v];

//  real data 3
```

```
Is1 = [4.423 4.540 4.740 4.980 5.320 5.636 6.016 6.420]';
Is2 = [6.783 7.256 7.653 8.163 8.463]';
Is = [Is1;Is2];
pf1 = [.1550 .2671 .3676 .4533 .5240 .5829 .6292 .6691]';
pf2 = [.7005 .7314 .7510 .7661 .7812]';
pf = [pf1;pf2];
v1 = [125.4 125.3 125.4 125.3 125.3 125.2 125.2 125.1]';
v2 = [125.1 125.0 124.8 124.9 124.5]';
v = [v1;v2];
r3 = [zip Is zip zip pf rpm v];

//  real data 4

Is1 = [4.450 4.566 4.766 5.020 5.310 5.696 6.026 6.403]';
Is2 = [6.823 7.180 7.653 8.073 8.506]';
Is = [Is1;Is2];
pf1 = [.1513 .2567 .3648 .4525 .5248 .5899 .6366 .6746]';
pf2 = [.7032 .7270 .7494 .7669 .7791]';
pf = [pf1;pf2];
v1 = [125.4 125.3 125.3 125.2 125.2 125.1 124.9 124.8]';
v2 = [124.8 124.7 124.6 124.5 124.6]';
v = [v1;v2];
r4 = [zip Is zip zip pf rpm v];

//  do rsfdata on each of the real data with known, measured Rs

Rs1 = .859*ones(13,1);
Rs2 = .859*ones(13,1);
Rs3 = .858*ones(13,1);
Rs4 = .858*ones(13,1);

[pv1,x1,s1] = rsfdata(r1,Rs1);
[pv2,x2,s2] = rsfdata(r2,Rs2);
[pv3,x3,s3] = rsfdata(r3,Rs3);
[pv4,x4,s4] = rsfdata(r4,Rs4);

rinave = 0.25*(r1 + r2 + r3 + r4);
Rsave = 0.25*(Rs1 + Rs2 + Rs3 + Rs4);
[pvinave,xinave,sinave] = rsfdata(rinave,Rsave);

//  Print out the results

pv = [pv1 pv2 pv3 pv4]
pvinave

//  Give statistics on the output results compared to itself; namely,
//  the average, the minimum, and the maximum of each parameter.

pvoutave = 0.25*(pv1 + pv2 + pv3 + pv4);
for i=1:3, llim(i,1) = min(pv(i,:));end;
for i=1:3, ulim(i,1) = max(pv(i,:));end;
outstat = [pvoutave llim ulim]

//  Compute the normalized pvinave sensitivity and statistics

for i=1:4, dpvin(:,i) = pv(:,i) - pvinave;end;
for i=1:3, dpvin(i,:) = dpvin(i,:)/pvinave(i);end;
dpvin
dpvinave = 0.25*(dpvin(:,1) + dpvin(:,2) + dpvin(:,3) + dpvin(:,4));
adpvinave=(abs(dpvin(:,1))+abs(dpvin(:,2))+abs(dpvin(:,3))+abs(dpvin(:,4)))/4.;
```

```
for i=1:3, llim(i,1) = min(dpvin(i,:));end;
for i=1:3, ulim(i,1) = max(dpvin(i,:));end;
dpvinstat = [dpvinave adpvinave llim ulim]

//  Compute the normalized pvoutave sensitivity and statistics

for i=1:4, dpvout(:,i) = pv(:,i) - pvoutave;end;
for i=1:3, dpvout(i,:) = dpvout(i,:)/pvoutave(i);end;
dpvout;
dpvoutave = 0.25*(dpvout(:,1)+dpvout(:,2)+dpvout(:,3)+dpvout(:,4));
adpvoutave=(abs(dpvout(:,1))+abs(dpvout(:,2))+abs(dpvout(:,3)));
adpvoutave = (adpvoutave + abs(dpvout(:,4)))*0.25;
for i=1:3, llim(i,1) = min(dpvout(i,:));end;
for i=1:3, ulim(i,1) = max(dpvout(i,:));end;
dpvoutstat = [dpvoutave adpvoutave llim ulim];

save 'sctrsl1data';
diary(0);
```

```
////////////////////////////////////////////////////////////////
//  Program ctb21data.r (in [cho.matfiles.rrbroken.rsconst])  //
//                                                            //
//  Computes estimates using Estimator 3 at 700W              //
//  Note that rsfdata.udf = Estimator 3                       //
////////////////////////////////////////////////////////////////


//  Working with broken bar data (stator 1, rotor 2)
//  at constant temp.
//  Date:
//  ri = [T Is Pe Pm pf rpm V]

diary('dctrsb21data2');
long;
define 'param.udf'
define 'rsfdata.udf'
zip = 0*ones(16,1);

//  real data 1

Is1 = [4.406 4.501 4.713 4.976 5.205 5.563 5.990 6.325]';
Is2 = [6.646 7.073 7.478 7.948 8.436 8.821 9.226 9.610]';
Is = [Is1;Is2];
pf1 = [.1810 .2745 .3887 .4604 .5269 .5919 .6392 .6745]';
pf2 = [.7020 .7255 .7459 .7643 .7768 .7912 .8002 .8072]';
pf = [pf1;pf2];

// Note that the following rpm vector will be used throughout
// as the controlling test input

rpm1 = [1795 1790 1785 1780 1775 1770 1765 1760]';
rpm2 = [1755 1750 1745 1740 1735 1730 1725 1720]';
rpm = [rpm1;rpm2];

v1 = [125.6 125.6 125.5 125.4 125.4 125.3 125.3 125.2]';
v2 = [125.0 124.9 124.9 124.8 124.5 124.5 124.5 124.3]';
v = [v1;v2];
r1 = [zip Is zip zip pf rpm v];

//  real data 2

Is1 = [4.403 4.548 4.631 4.931 5.225 5.556 5.983 6.346]';
Is2 = [6.668 7.081 7.451 7.940 8.320 8.821 9.180 9.620]';
Is = [Is1;Is2];
pf1 = [.1692 .2877 .3584 .4510 .5219 .5846 .6363 .6730]';
pf2 = [.6995 .7282 .7474 .7646 .7836 .7909 .7994 .8056]';
pf = [pf1;pf2];
v1 = [125.7 125.6 125.6 125.5 125.4 125.3 125.2 125.1]';
v2 = [125.0 125.0 124.8 124.7 124.7 124.7 124.7 124.7]';
v = [v1;v2];
r2 = [zip Is zip zip pf rpm v];

//  real data 3
```

```
Is1 = [4.215 4.333 4.520 4.760 5.041 5.456 5.785 6.148]';
Is2 = [6.531 6.950 7.358 7.805 8.228 8.603 9.018 9.535]';
Is = [Is1;Is2];
pf1 = [.1880 .2905 .3886 .4697 .5412 .6058 .6505 .6874]';
pf2 = [.7139 .7340 .7516 .7700 .7829 .7923 .8027 .8115]';
pf = [pf1;pf2];
v1 = [123.9 123.7 123.5 123.6 123.6 123.5 123.6 123.3]';
v2 = [123.3 123.3 123.4 123.5 123.5 123.4 123.3 123.3]';
v = [v1;v2];
r3 = [zip Is zip zip pf rpm v];

//  real data 4

Is1 = [4.338 4.460 4.688 4.910 5.235 5.560 5.876 6.240]';
Is2 = [6.640 7.126 7.545 7.991 8.385 8.795 9.246 9.688]';
Is = [Is1;Is2];
pf1 = [.1451 .2706 .3733 .4529 .5273 .5782 .6195 .6595]';
pf2 = [.6961 .7248 .7463 .7639 .7795 .7908 .7973 .8069]';
pf = [pf1;pf2];
v1 = [125.4 125.5 125.4 125.1 125.1 125.2 125.2 125.1]';
v2 = [125.2 125.1 125.1 125.0 124.5 124.8 124.6 124.6]';
v = [v1;v2];
r4 = [zip Is zip zip pf rpm v];

//  do rsfdata on each of the real data with known, measured Rs

Rs1 = .869*ones(16,1);
Rs2 = .869*ones(16,1);
Rs3 = .865*ones(16,1);
Rs4 = .863*ones(16,1);

[pv1,x1] = rsfdata(r1,Rs1);
[pv2,x2] = rsfdata(r2,Rs2);
[pv3,x3] = rsfdata(r3,Rs3);
[pv4,x4] = rsfdata(r4,Rs4);

rinave = (r1 + r2 + r3 + r4)/4.;
Rsave = (Rs1 + Rs2 + Rs3 + Rs4)/4.;
[bpvinave,bxinave] = rsfdata(rinave,Rsave);

//  Print out the results

bpv = [pv1 pv2 pv3 pv4]
bpvinave

//  Give statistics on the output results compared to itself; namely,
//  the average, the minimum, and the maximum of each parameter.

bpvoutave = (pv1 + pv2 + pv3 + pv4)/4.;
for i=1:3, llim(i,1) = min(bpv(i,:));end;
for i=1:3, ulim(i,1) = max(bpv(i,:));end;
boutstat = [bpvoutave llim ulim]

//  Compute the normalized bpvinave sensitivity and statistics

for i=1:4, dbpvin(:,i) = bpv(:,i) - bpvinave;end;
for i=1:3, dbpvin(i,:) = dbpvin(i,:)/bpvinave(i);end;
dbpvin
dbpvinave = (dbpvin(:,1)+dbpvin(:,2)+dbpvin(:,3)+dbpvin(:,4))/4.;
adbpvinave=abs(dbpvin(:,1))+abs(dbpvin(:,2))+abs(dbpvin(:,3));
```

```
adbpvinave = (adbpvinave + abs(dbpvin(:,4)))/4.;
for i=1:3, llim(i,1) = min(dbpvin(i,:));end;
for i=1:3, ulim(i,1) = max(dbpvin(i,:));end;
dbpvinstat = [dbpvinave adbpvinave llim ulim]

//  Compute the normalized bpvoutave sensitivity and statistics

for i=1:4, dbpvout(:,i) = bpv(:,i) - bpvoutave;end;
for i=1:3, dbpvout(i,:) = dbpvout(i,:)/bpvoutave(i);end;
dbpvout;
dbpvoutave = (dbpvout(:,1)+dbpvout(:,2)+dbpvout(:,3)+dbpvout(:,4))/4.;
adbpvoutave=abs(dbpvout(:,1))+abs(dbpvout(:,2))+abs(dbpvout(:,3));
adbpvoutave = (adbpvoutave + abs(dbpvout(:,4)))/4.;
for i=1:3, llim(i,1) = min(dbpvout(i,:));end;
for i=1:3, ulim(i,1) = max(dbpvout(i,:));end;
dbpvoutstat = [dbpvoutave adbpvoutave llim ulim];

//  Load healthy rotor data from sctrs11data.dat

load 'sctrs31data2'

inm = [bpv bpvinave];

//  Compute the normalized inm sensitivity and statistics

for i=1:5, dinm(:,i) = inm(:,i) - pvinave;end;
for i=1:3, dinm(i,:) = dinm(i,:)/pvinave(i);end;
dinm
dinmave = (dinm(:,1)+dinm(:,2)+dinm(:,3)+dinm(:,4))/4.;
adinmave = (abs(dinm(:,1))+abs(dinm(:,2))+abs(dinm(:,3))+abs(dinm(:,4)))/4.;
for i=1:3, llim(i,1) = min(dinm(i,:));end;
for i=1:3, ulim(i,1) = max(dinm(i,:));end;
dinmstat = [dinmave adinmave llim ulim]

save 'sctrsb21data2';
diary(0);
```

```
////////////////////////////////////////////////////////////////////
//  Program ct31data.r (in [cho.matfiles.rrbroken.rsconst])       //
//                                                                //
//  Computes estimates using Estimator 3 at 700W                  //
//  Note that rsfdata.udf = Estimator 3                           //
////////////////////////////////////////////////////////////////////


//  Working with real data of healthy motor (stator 1, rotor 3)
//  at constant temp.
//  Date:  April 30, 1988.  Dynamometer fixed.
//  ri = [T Is Pe Pm pf rpm V]

diary('dctrs31data2');
long;
define 'param.udf'
define 'rsfdata.udf'
zip = 0*ones(16,1);

//  real data 1

Is1 = [4.323 4.431 4.663 4.943 5.231 5.628 5.913 6.353]';
Is2 = [6.748 7.130 7.536 7.965 8.433 8.848 9.230 9.656]';
Is = [Is1;Is2];
pf1 = [.1642 .2727 .3785 .4735 .5375 .6050 .6405 .6841]';
pf2 = [.7147 .7351 .7575 .7740 .7871 .7980 .8048 .8141]';
pf = [pf1;pf2];

// Note that the following rpm vector will be used throughout
// as the controlling test input

rpm1 = [1795 1790 1785 1780 1775 1770 1765 1760]';
rpm2 = [1755 1750 1745 1740 1735 1730 1725 1720]';
rpm = [rpm1;rpm2];

v1 = [123.5 123.4 123.6 123.5 123.5 123.4 123.1 123.2]';
v2 = [123.2 123.0 123.0 122.9 123.0 122.6 122.9 122.6]';
v = [v1;v2];
r1 = [zip Is zip zip pf rpm v];

//  real data 2

Is1 = [4.388 4.485 4.720 4.980 5.263 5.650 5.995 6.375]';
Is2 = [6.816 7.226 7.660 8.055 8.488 8.860 9.385 9.768]';
Is = [Is1;Is2];
pf1 = [.1774 .2832 .3872 .4710 .5362 .5985 .6427 .6777]';
pf2 = [.7116 .7370 .7565 .7708 .7855 .7976 .8078 .8145]';
pf = [pf1;pf2];
v1 = [124.2 123.9 124.3 124.1 124.1 124.2 124.1 124.0]';
v2 = [124.0 124.0 123.8 123.8 123.6 123.5 123.4 123.4]';
v = [v1;v2];
r2 = [zip Is zip zip pf rpm v];

//  real data 3
```

```
for i=1:3, llim(i,1) = min(dpvin(i,:));end;
for i=1:3, ulim(i,1) = max(dpvin(i,:));end;
dpvinstat = [dpvinave adpvinave llim ulim]

//  Compute the normalized pvoutave sensitivity and statistics

for i=1:4, dpvout(:,i) = pv(:,i) - pvoutave;end;
for i=1:3, dpvout(i,:) = dpvout(i,:)/pvoutave(i);end;
dpvout;
dpvoutave = 0.25*(dpvout(:,1)+dpvout(:,2)+dpvout(:,3)+dpvout(:,4));
adpvoutave=(abs(dpvout(:,1))+abs(dpvout(:,2))+abs(dpvout(:,3)));
adpvoutave = (adpvoutave + abs(dpvout(:,4)))*0.25;
for i=1:3, llim(i,1) = min(dpvout(i,:));end;
for i=1:3, ulim(i,1) = max(dpvout(i,:));end;
dpvoutstat = [dpvoutave adpvoutave llim ulim];

save 'sctrs31data2';
diary(0);
```

```
Is1 = [4.526 4.650 4.830 5.106 5.391 5.768 6.156 6.558]';
Is2 = [6.908 7.358 7.763 8.163 8.690 9.005 9.525 9.875]';
Is = [Is1;Is2];
pf1 = [.1704 .2844 .3747 .4562 .5242 .5898 .6414 .6774]';
pf2 = [.7057 .7341 .7544 .7689 .7845 .7934 .8041 .8126]';
pf = [pf1;pf2];
v1 = [126.0 126.0 125.9 126.0 125.8 125.8 125.6 125.6]';
v2 = [125.5 125.3 125.4 125.4 125.3 125.3 125.3 125.1]';
v = [v1;v2];
r3 = [zip Is zip zip pf rpm v];

//  real data 4

Is1 = [4.571 4.681 4.861 5.118 5.386 5.751 6.105 6.570]';
Is2 = [6.903 7.341 7.701 8.235 8.625 9.028 9.455 9.943]';
Is = [Is1;Is2];
pf1 = [.1574 .2693 .3650 .4519 .5264 .5849 .6293 .6751]';
pf2 = [.7028 .7306 .7468 .7670 .7807 .7941 .8029 .8104]';
pf = [pf1;pf2];
v1 = [126.4 126.5 126.3 126.3 126.0 126.2 126.1 126.0]';
v2 = [125.9 125.9 125.9 125.8 125.7 125.7 125.4 125.5]';
v = [v1;v2];
r4 = [zip Is zip zip pf rpm v];

//  do rsfdata on each of the real data with known, measured Rs

Rs1 = .863*ones(16,1);
Rs2 = .863*ones(16,1);
Rs3 = .865*ones(16,1);
Rs4 = .865*ones(16,1);

[pv1,x1,s1] = rsfdata(r1,Rs1);
[pv2,x2,s2] = rsfdata(r2,Rs2);
[pv3,x3,s3] = rsfdata(r3,Rs3);
[pv4,x4,s4] = rsfdata(r4,Rs4);

rinave = 0.25*(r1 + r2 + r3 + r4);
Rsave = 0.25*(Rs1 + Rs2 + Rs3 + Rs4);
[pvinave,xinave,sinave] = rsfdata(rinave,Rsave);

//  Print out the results

pv = [pv1 pv2 pv3 pv4]
pvinave

//  Give statistics on the output results compared to itself; namely,
//  the average, the minimum, and the maximum of each parameter.

pvoutave = 0.25*(pv1 + pv2 + pv3 + pv4);
for i=1:3, llim(i,1) = min(pv(i,:));end;
for i=1:3, ulim(i,1) = max(pv(i,:));end;
outstat = [pvoutave llim ulim]

//  Compute the normalized pvinave sensitivity and statistics

for i=1:4, dpvin(:,i) = pv(:,i) - pvinave;end;
for i=1:3, dpvin(i,:) = dpvin(i,:)/pvinave(i);end;
dpvin
dpvinave = 0.25*(dpvin(:,1) + dpvin(:,2) + dpvin(:,3) + dpvin(:,4));
adpvinave=(abs(dpvin(:,1))+abs(dpvin(:,2))+abs(dpvin(:,3))+abs(dpvin(:,4)))/4.;
```

Appendix G

Results from Constant-Temperature Experiments

```
////////////////////////////////////////////////////////////
//  Experimental estimates for rotors 1, 2, and 3         //
//  using Estimator 1                                     //
//                                                        //
//  For each rotor                                        //
//  PV = |pv(r1) pv(r2) pv(r3) pv(r4)|                    //
//  where pv(ri) = parameter vector estimated from        //
//  data matrix ri found in ct11data.r, ctb21data2.r,     //
//  and ct31data2.r                                       //
//  pv = [Rs Rr L M]', where ' denotes transposition      //
//  and pvinave is x(E(A)), where x = Rs, Rr, L, or M     //
////////////////////////////////////////////////////////////


//  Rotor 1 experimental estimates

  PV =
        1.2955    1.3619    0.9596    0.5867
        0.5648    0.5846    0.5804    0.5488
        0.0751    0.0755    0.0750    0.0744
        0.0702    0.0707    0.0706    0.0698

  PVINAVE    =

        1.0195
        0.5675
        0.0750
        0.0703


//  Rotor 2 experimental estimates


  BPV        =

        1.6914    1.2474    1.6917    1.0008
        0.5995    0.5882    0.5868    0.5995
        0.0752    0.0752    0.0772    0.0759
        0.0703    0.0706    0.0722    0.0715

  BPVINAVE   =

        1.3774
        0.5919
        0.0759
        0.0712
```

```
//  Rotor 3 experimental estimates

  PV          =

        0.7977    1.3461    1.0671    0.9236
        0.5426    0.5666    0.5544    0.5648
        0.0748    0.0742    0.0729    0.0730
        0.0703    0.0697    0.0685    0.0687

  PVINAVE     =

        1.0178
        0.5561
        0.0737
        0.0692
```

```
////////////////////////////////////////////////////////////////
//  Experimental estimates for rotors 1, 2, and 3             //
//  using Estimator 2                                         //
//                                                            //
//  For each rotor                                            //
//  RPV = |pv(r1) pv(r2) pv(r3) pv(r4)|                       //
//  where pv(ri) = parameter vector estimated from            //
//  data matrix ri found in ct11data.r, ctb21data2.r,         //
//  and ct31data2.r                                           //
//  pv = [Rs Rr L M]', where ' denotes transposition          //
//  and rpvinave is x(E(A)), where x = Rs, Rr, L, or M         //
////////////////////////////////////////////////////////////////


//  Rotor 1 experimental estimates

  RPV =

        1.0438    0.8515    0.7169    0.8800
        0.5527    0.5593    0.5682    0.5630
        0.0751    0.0755    0.0750    0.0744
        0.0702    0.0708    0.0706    0.0698

  RPVINAVE    =

        0.8791
        0.5605
        0.0750
        0.0703


//  Rotor 2 experimental estimates


  BRPV        =

        0.9861    0.9530    1.1166    0.6588
        0.5705    0.5762    0.5651    0.5855
        0.0754    0.0753    0.0775    0.0760
        0.0707    0.0707    0.0726    0.0717

  BRPVINAVE   =

        0.9320
        0.5741
        0.0761
        0.0714
```

```
//  Rotor 3 experimental estimates

  RPV          =

        1.0910    1.0281    1.0722    0.9275
        0.5534    0.5543    0.5546    0.5649
        0.0747    0.0744    0.0729    0.0730
        0.0701    0.0699    0.0685    0.0687

  RPVINAVE     =

        1.0324
        0.5567
        0.0737
        0.0693
```

```
////////////////////////////////////////////////////////////
//  Experimental estimates for rotors 1, 2, and 3         //
//  using Estimator 3                                     //
//                                                        //
//  For each rotor                                        //
//  PV = |pv(r1) pv(r2) pv(r3) pv(r4)|                    //
//  where pv(ri) = parameter vector estimated from        //
//  data matrix ri found in ct11data.r, ctb21data2.r,     //
//  and ct31data2.r                                       //
//  pv = [Rr L M]', where ' denotes transposition         //
//  and pvinave is x(E(A)), where x = Rr, L, or M         //
////////////////////////////////////////////////////////////


//  Rotor 1 experimental estimates

  PV          =

        0.5589    0.5591    0.5635    0.5638
        0.0755    0.0755    0.0747    0.0744
        0.0709    0.0708    0.0701    0.0699

  PVINAVE     =

        0.5612
        0.0750
        0.0704


//  Rotor 2 experimental estimates

  BPV         =

        0.5751    0.5795    0.5750    0.5775
        0.0757    0.0755    0.0781    0.0755
        0.0712    0.0710    0.0735    0.0710

  BPVINAVE    =

        0.5767
        0.0762
        0.0716
```

```
//  Rotor 3 experimental estimates

  PV          =

        0.5624    0.5608    0.5628    0.5674
        0.0752    0.0748    0.0734    0.0732
        0.0709    0.0705    0.0692    0.0689

  PVINAVE     =

        0.5633
        0.0741
        0.0699
```

Appendix H

Programs for the Thermally-Compensated Detector of Broken Rotor Bar

```
////////////////////////////////////////////////////////////
//  Program t31.r (in [cho.matfiles.thermo.rdata])        //
//                                                        //
//  Experimental data at 250W, 700W, 1.2kW, 1.7kW         //
//  to be used to determine the thermal resistances       //
//  of all the thermocouples as well as for the           //
//  rotor and stator weighted-average temperatures        //
//                                                        //
//  Note that lsdata.udf in this directory is             //
//  Estimator 3                                           //
////////////////////////////////////////////////////////////


//  Working with data (stator 1, rotor 3)
//  at constant temps.

define 'param.udf'
define 'lsdata.udf'

//  ri = [Is V pf Pm Torque rpm]

//  real data matrix (at room temp)

Is1 = [3.720 3.990 4.101 4.455 4.791 5.378 5.896 6.166]';
Is2 = [6.400 6.965 7.541 8.040 8.378 8.845 9.423 9.495]';
Is  = [Is1;Is2];
v1  = [122.3 123.5 121.5 122.2 122.1 124.2 125.7 124.7]';
v2  = [121.4 121.6 125.0 124.6 122.2 123.1 123.9 119.9]';
v   = [v1;v2];
pf1 = [.2247 .3605 .4528 .5367 .6078 .6605 .6991 .7311]';
pf2 = [.7559 .7784 .7877 .8017 .8146 .8249 .8287 .8371]';
pf  = [pf1;pf2];
Pm1 = [159.90 371.44 505.10 679.04 859.44 1085.1 1302.2 1412.9]';
Pm2 = [1478.8 1639.6 1872.7 2027.8 2045.6 2230.1 2379.1 2339.8]';
Pm  = [Pm1;Pm2];
Torque1  = [0.850 1.980 2.700 3.640 4.620 5.850 7.040 7.660]';
Torque2  = [8.040 8.940 10.24 11.12 11.25 12.30 13.16 12.98]';
Torque   = [Torque1;Torque2];
rpm1 = [1795 1790 1785 1780 1775 1770 1765 1760]';
rpm2 = [1755 1750 1745 1740 1735 1730 1725 1720]';
rpm = [rpm1;rpm2];
rmtemp = [Is V pf Pm Torque rpm];

//  At power of 250 watts.

Is1 = [4.066 4.216 4.416 4.796 5.073 5.373 5.783 6.153]';
Is2 = [6.536 6.920 7.360 7.760 8.116 8.636 9.003 9.383]';
Is  = [Is1;Is2];
v1  = [124.3 124.2 124.2 124.4 124.2 124.2 124.3 124.0]';
v2  = [124.0 124.1 123.9 123.9 123.9 123.8 123.7 123.7]';
v   = [v1;v2];
pf1 = [.2323 .3511 .4308 .5311 .5869 .6297 .6771 .7088]';
pf2 = [.7331 .7571 .7742 .7892 .7999 .8122 .8198 .8254]';
pf  = [pf1;pf2];
Pm1 = [129.80 285.15 443.36 611.88 775.73 914.53 1108.0 1222.9]';
```

```
Pm2 = [1359.2 1491.1 1624.0 1745.1 1860.1 1969.0 2075.4 2170.3]';
Pm  = [Pm1;Pm2];
Torque1  = [0.690 1.5[illegible] 2.370 3.280 4.170 4.930 5.990 6.630]';
Torque2  = [7.390 8.130 8.880 9.570 10.23 10.86 11.48 12.04]';
Torque   = [Torque1;Torque2];
temp1 = [Is V pf Pm Torque rpm];

//  At power of 1200 watts

Is1 = [4.410 4.620 4.946 5.183 5.570 5.936 6.296 6.660]';
Is2 = [7.033 7.373 7.786 8.196 8.623 9.040 9.366 9.713]';
Is  = [Is1;Is2];
v1  = [124.5 124.2 124.0 123.9 124.0 123.8 123.8 123.8]';
v2  = [123.6 123.6 123.3 123.3 123.5 123.4 123.2 123.1]';
v   = [v1;v2];
pf1 = [.4371 .5088 .5810 .6197 .6657 .7023 .7296 .7492]';
pf2 = [.7688 .7840 .7957 .8064 .8176 .8251 .8300 .8343]';
pf  = [pf1;pf2];
Pm1 = [169.31 303.91 460.20 626.81 768.29 901.54 1024.7 1176.8]';
Pm2 = [1307.7 1426.9 1549.0 1677.7 1778.3 1892.8 1979.6 2101.8]';
Pm  = [Pm1;Pm2];
Torque1  = [0.900 1.620 2.460 3.360 4.130 4.860 5.540 6.380]';
Torque2  = [7.110 7.780 8.470 9.200 9.780 10.44 10.95 11.66]';
Torque   = [Torque1;Torque2];
rpmn = rpm - 10*ones(16,1);
temp3 = [Is V pf Pm Torque rpmn];

//  At power of 700 watts

Is1 = [4.130 4.296 4.493 4.783 5.073 5.376 5.773 6.106]';
Is2 = [6.546 6.956 7.326 7.703 8.133 8.506 8.886 9.270]';
Is  = [Is1;Is2];
v1  = [125.2 125.3 125.1 125.1 125.0 125.0 124.9 124.8]';
v2  = [124.6 124.6 124.5 124.5 124.4 124.2 124.2 124.1]';
v   = [v1;v2];
pf1 = [.2281 .3443 .4382 .5195 .5809 .6262 .6750 .7046]';
pf2 = [.7360 .7560 .7735 .7867 .8010 .8094 .8167 .8238]';
pf  = [pf1;pf2];
Pm1 = [124.16 296.40 462.07 600.69 751.55 907.11 1050.6 1173.1]';
Pm2 = [1311.4 1447.0 1576.4 1679.5 1811.0 1936.4 2041.0 2125.3]';
Pm  = [Pm1;Pm2];
Torque1  = [0.660 1.580 2.470 3.220 4.040 4.890 5.680 6.360]';
Torque2  = [7.130 7.890 8.620 9.210 9.960 10.68 11.29 11.79]';
Torque   = [Torque1;Torque2];
temp2 = [Is V pf Pm Torque rpm];

//  At power of 1700 watts

Is1 = [5.330 5.693 6.026 6.330 6.676 7.076 7.396 7.773]';
Is2 = [8.140 8.510 9.273 9.636 9.966 10.33 10.70 4.820]';
Is  = [Is1;Is2];
v1  = [123.6 123.5 123.4 123.2 123.0 123.2 123.0 122.9]';
v2  = [122.9 122.9 122.8 122.8 122.8 122.8 122.8 123.9]';
v   = [v1;v2];
pf1 = [.6518 .6889 .7189 .7389 .7610 .7780 .7903 .8014]';
pf2 = [.8128 .8262 .8324 .8360 .8415 .8449 .8459 .5708]';
pf  = [pf1;pf2];
Pm1 = [141.09 243.87 432.14 600.69 714.34 881.14 1021.0 1130.7]';
Pm2 = [1276.4 1384.7 1505.1 1619.3 1716.5 1840.3 1938.0 2017.1]';
Pm  = [Pm1;Pm2];
```

```
Torque1  = [0.750 1.300 2.130 3.220 3.840 4.750 5.520 6.130]';
Torque2  = [6.940 7.550 8.230 8.880 9.440 10.15 10.72 11.19]';
Torque   = [Torque1;Torque2];
rpmn1 = [1765 1760 1755 1750 1745 1740 1735 1730]';
rpmn2 = [1725 1720 1710 1705 1700 1695 1690 1775]';
rpmn = [rpmn1;rpmn2];
temp4 = [Is V pf Pm Torque rpmn];

//  Measured Rs values

Rsrt  = (0.805 + 0.804 + 0.804)/3;
Rs(1) = (0.871 + 0.872 + 0.865)/3;
Rs(2) = (0.883 + 0.883 + 0.879)/3;
Rs(3) = (0.903 + 0.903 + 0.899)/3;
Rs(4) = (0.951 + 0.949 + 0.945)/3;

//  We need to obtain estimates of each Rri's.  (We measure Rs's)

[pvrt,errrt] = lsdata(rmtemp,Rsrt);
[pv1,err1] = lsdata(temp1,Rs(1));
[pv2,err2] = lsdata(temp2,Rs(2));
[pv3,err3] = lsdata(temp3,Rs(3));
[pv4,err4] = lsdata(temp4,Rs(4));

Rrrt  = pvrt(1);
Rr(1) = pv1(1);
Rr(2) = pv2(1);
Rr(3) = pv3(1);
Rr(4) = pv4(1);

//  We need stdata in the form |Is V pf Pm Torque rpm|
//  and TempM which is converted from TempF
//  where TempF = |dT1 dT2 ... dT26 Tamb|
//  The delta Ti's are in F, but Tamb is in C.  Hence, we
//  need to convert the TempF so that it reads in C (the
//  transformed temp matrix will be called TempM.

//  TempM = |tr1(250W) tr2(700W) tr3(1200W) tr4(1700W)|'
//  tri = |t1 ... t9 Tamb|
//  t1 = |tc1  tc2   tc3|, t2 = |tc4 tc5   tc6|, t3 = |tc11 tc12 tc21|
//  t4 = |tc10 tc18 tc19|, t5 = |tc9 tc14 tc20|, t6 = |tc13 tc25 tc26|
//  t7 = |tc7  tc23 tc24|, t8 = |tc8      tc17|, t9 = |tc15 tc16 tc22|

t1 = [30.0 30.0 28.0];
t2 = [21.5 22.5 22.5];
t3 = [32.5 32.5 31.5];
t4 = [27.0 32.5 34.0];
t5 = [29.0 26.5 32.5];
t6 = [31.0 35.5 32.0];
t7 = [31.5 35.5 35.0];
t8 = [31.0 34.5];
t9 = [34.5 35.0];
Tamb = 24.6;
tr1 = [t1 t2 t3 t4 t5 t6 t7 t8 t9 Tamb];

t1 = [35.5 35.5 33.5];
t2 = [26.0 27.5 27.5];
t3 = [40.0 40.0 38.5];
t4 = [33.0 39.5 40.5];
t5 = [35.5 32.0 40.0];
```

```
t6 = [37.0 44.0 39.5];
t7 = [38.5 43.5 43.5];
t8 = [38.5 42.5];
t9 = [41.5 43.0];
Tamb = 24.4;
tr2 = [t1 t2 t3 t4 t5 t6 t7 t8 t9 Tamb];

t1 = [45.0 44.5 43.0];
t2 = [33.0 35.0 34.5];
t3 = [51.0 51.0 48.5];
t4 = [42.5 50.5 52.5];
t5 = [45.5 40.5 51.0];
t6 = [47.0 58.0 52.0];
t7 = [49.5 56.5 56.5];
t8 = [48.5 52.5];
t9 = [53.0 55.5];
Tamb = 25.2;
tr3 = [t1 t2 t3 t4 t5 t6 t7 t8 t9 Tamb];

t1 = [64.0 64.0 61.5];
t2 = [46.0 49.5 49.0];
t3 = [75.0 75.0 72.5];
t4 = [60.5 73.5 76.0];
t5 = [66.5 58.0 73.0];
t6 = [68.0 86.5 77.0];
t7 = [72.5 82.5 82.5];
t8 = [71.5 76.0];
t9 = [76.0 82.0];
Tamb = 25.6;
tr4 = [t1 t2 t3 t4 t5 t6 t7 t8 t9 Tamb];

TempF = [tr1; tr2; tr3; tr4];

//  Delta of temp in c = 5[(dF + Famb) -32]/9 - Camb
//                     = 5*dF/9 + 5(Famb - 32)/9 - Camb = 5dF/9
//  Note tempF(:,26) = Ambient Temp, is in degrees Celsius

for i = 1:4,...
for j = 1:25,...
TempM(i,j) = 5*tempF(i,j)/9;...
end
end

TempM = [TempM TempF(:,26)];

//  Recall that stdata = |I V pf Pm Torque rpm|

std1 = [4.153 124.2 0.3041 253.54 1.350 1792];
std2 = [4.840 124.8 0.5428 700.64 3.760 1778];
std3 = [5.936 123.8 0.7023 1197.1 6.490 1760];
std4 = [7.513 123.2 0.7924 1704.6 9.380 1734];
stdata = [std1; std2; std3; std4];

save 'st31';
```

```
////////////////////////////////////////////////////////
//  Program rthest.r (in [cho.matfiles.thermo.rdata])  //
//                                                     //
//  Takes the data generated from t31.r and computes   //
//  the thermal resistances of all the thermocouples   //
//  and of the rotor and the stator weighted-average   //
//  temperatures                                       //
////////////////////////////////////////////////////////


//  Working with stator 1 and rotor 3, the thermal resistances will be
//  estimated.

clear
diary('drthest');
long;

//  First, retrieve the necessary data, assuming that the data are stored
//  in the file ct31.dat.  Three principle matrices t1 t2 t3 t4 are recalled.
//  Mdata is the data matrix which has for each of its rows the steady state
//  measurements for each set of temp .

load 'st31' stdata TempM Rsrt Rs Rrrt Rr;

//  Get the dimension for the matrix mdata

[nrows, ncols] = size(stdata);

//  Assuming that stdata is of the form
//  stdata = |Is V pf Psh Torque rpm|,
//             1  2  3  4    5     6
//  we can manipulate this data to obtain slips and Pe.

//  S = 1 - rpm/1800

for i=1:nrows,...
S(i) = 1 - stdata(i,6)/1800;...
end

S

//  Now the payoff:  We obtain the thermal resistance parameters.

//  The following are thermal sources for each matrix
//     Pir = slip*Pm/(1-slip), where Pm = (Torq,meas + Torq,loss)*Wm
//     = (2*PI*rpm/60)*(Torq,meas + Torq,loss) where Torq,loss = 0.3796
//     Pis = 3*Is*Is*Rsi
//  T = Pr*Rthr + Ps*Rths + Pc*Rthc, Pc*Rthc = const = Pc*Rthc
//     which is approx. Pc*Rths, with Pc = 23.51b*2.6W/1b

Torql = 0.3796;
Pc = 23.5*2.6;
for i=1:nrows,...
Pm = PI*(stdata(i,5) + Torql)*stdata(i,6)/30;...
slip = S(i);...
```

```
Is = stdata(i,1);...
Pr(i) = slip*Pm/(1-slip);...
Ps(i) = 3*Is*Is^Rs(i) + Pc;...
end

A = [Pr Ps]

for i=1:25,...
y = TempM(:,i);...
Tparam = A\y;...
Rth(2*i-1) = Tparam(1);...
Rth(2*i) = Tparam(2);...
Teddy(i) = Tparam(2)*Pc;...
end

Rth
Teddy
maxTeddy = max(Teddy)
minTeddy = min(Teddy)

//  We also know Tsw, Tro (inferred from the R(T,material) equation).
//  Here is the first time we needed the room temp. experiment.
//  The room temp exp. gave us the Rrrt estimate which is used to infer
//  the rotor temps at different set of heat input values.

//  Tambrt = ambient temp. for room temp experiments

Tambrt = 24.2;
Tambv = TempM(:,26);

for i = 1:nrows,...
Tsw(i) = Rs(i)*(234.4 + Tambrt)/Rsrt - 234.4;   .
Tro(i) = Rr(i)*(228.1 + Tambrt)/Rrrt - 228.1;...
end

//  Let's check to see how well our equation works for at least the Rs
//  values. 1 is the shaft side, 2 is the middle, 3 is the fan side.

dTsw1 = Tsw - TempM(:,21) - Tambv
dTsw2 = Tsw - TempM(:,20) - Tambv
dTsw3 = Tsw - TempM(:,25) - Tambv

//  The only time the result of the room experiment effects any of the
//  other results.  Scale such that Tsw and Tro are incremental temp from
//  the Tambient.

Tsw = Tsw - Tambv
Tro = Tro - Tambv

Rthsw = A\Tsw
Rthro = A\Tro

TswEst = A*Rthsw;
TroEst = A*Rthro;

dTsw = Tsw - TswEst
dTro = Tro - TroEst

for i = 1:nrows,...
RsEst(i) = Rsrt*(TswEst(i) + Tambv(i) + 234.4)/(Tambrt+ 234.4);...
```

```
RrEst(i) = Rrrt*(TroEst(i) + Tambv(i) + 228.1)/(Tambrt + 228.1);...
end

Rs
RsEst
Rr
RrEst

dRs = Rs - RsEst
dRr = Rr - RrEst

for i = 1:nrows,...
perRs(i) = dRs(i)/Rs(i);...
perRr(i) = dRr(i)/Rr(i);...
end

perRs
perRr

//  We want to scale all the estimated resistances back to a reference
//  temperature and compare the deviance from each other.  In this case,
//  the reference Temp is the temperature at which the Pm was 700W.

Tswnorm = TswEst(2) + Tambv(2) + 234.4;
Tronorm = TroEst(2) + Tambv(2) + 228.1;

for i=1:nrows,...
Rssc(i) = Rs(i)*Tswnorm/(TswEst(i)+Tambv(i)+234.4);...
Rrsc(i) = Rr(i)*Tronorm/(TroEst(i)+Tambv(i)+228.1);...
end

Rssc(nrows+1) = Rsrt*Tswnorm/(Tambrt + 234.4);
Rrsc(nrows+1) = Rrrt*Tronorm/(Tambrt + 228.1);

Rssc
Rrsc
Rsnorm = (Rssc(1)+Rssc(2)+Rssc(3)+Rssc(4)+Rssc(5))/5
Rrnorm = (Rrsc(1)+Rrsc(2)+Rrsc(3)+Rrsc(4)+Rrsc(5))/5

for i = 1:nrows+1,...
dRssc(i) = Rssc(i) - Rsnorm;...
dRrsc(i) = Rrsc(i) - Rrnorm;...
perRssc(i) = dRssc(i)/Rsnorm;...
perRrsc(i) = dRrsc(i)/Rrnorm;...
end

dRssc
dRrsc
perRssc
perRrsc

save 'srthest';

diary(0);
```

```
////////////////////////////////////////////////////////////
//  Program ctldata.r (in [cho.matfiles.thermo.rdata])    //
//                                                        //
//  Experimental data from rotor 1 at                     //
//  loads of 500W, 950W,and 1450W                         //
//  Used by Program anl.r                                 //
////////////////////////////////////////////////////////////


//  Working with real data of healthy motor (stator 1, rotor 1)
//  at constant temp.
//  Date:  April 30, 1988.  Dynamometer fixed.
//  ri = [T Is Pe Pm pf rpm V]

clear;
long;
define 'param.udf'
define 'lsdata.udf'
zip = 0*ones(16,1);

//  real data 1, Pm = 501.36 W

Is1 = [4.018 4.176 4.408 4.668 4.988 5.376 5.745 6.093]';
Is2 = [6.435 6.913 7.253 7.636 8.155 8.555 8.951 9.361]';
Is = [Is1;Is2];
pf1 = [.2601 .3647 .4632 .5385 .6035 .6568 .6924 .7195]';
pf2 = [.7433 .7654 .7808 .8004 .8086 .8183 .8237 .8316]';
pf = [pf1;pf2];
rpm1 = [1795 1790 1785 1780 1775 1770 1765 1760]';
rpm2 = [1755 1750 1745 1740 1735 1730 1725 1720]';
rpm = [rpm1;rpm2];
v1 = [124.3 124.3 124.2 124.2 124.1 124.0 124.1 124.0]';
v2 = [123.7 123.7 123.7 123.6 123.5 123.5 123.5 123.3]';
v = [v1;v2];
r1 = [Is v pf zip zip rpm];

//  real data 2, Pm = 951.63 W

Is1 = [3.990 4.146 4.316 4.610 4.863 5.200 5.656 6.001]';
Is2 = [6.335 6.730 7.173 7.590 7.951 8.323 8.823 9.175]';
Is = [Is1;Is2];
pf1 = [.2558 .3706 .4469 .5378 .5884 .6494 .6931 .7194]';
pf2 = [.7437 .7659 .7841 .7974 .8080 .8189 .8205 .8306]';
pf = [pf1;pf2];
v1 = [124.4 124.4 124.2 124.1 123.9 123.4 123.8 123.7]';
v2 = [123.6 123.4 123.5 123.6 123.4 123.4 123.2 123.3]';
v = [v1;v2];
r2 = [Is v pf zip zip rpm];

//  real data 3, Pm = 1453.5 W

Is1 = [4.280 4.555 4.788 5.098 5.423 5.735 6.151 6.523]';
Is2 = [6.871 7.283 7.695 8.083 8.486 8.886 9.266 9.715]';
Is = [Is1;Is2];
pf1 = [.4294 .5165 .5785 .6280 .6735 .7012 .7360 .7538]';
```

```
pf2 = [.7722 .7894 .8022 .8134 .8215 .8270 .8325 .8375]';
pf = [pf1;pf2];
v1 = [124.8 124.8 124.6 124.4 124.3 124.3 124.1 124.0]';
v2 = [123.8 123.8 124.0 123.9 123.8 124.0 123.9 123.8]';
v = [v1;v2];
rpm1 = rpm - 10*ones(16,1);
r3 = [Is v pf zip zip rpm1];

//  We need stdata in the form |Is V pf Pm Torque rpm|
//  and TempM which is converted from TempF
//  where TempF = |dT1 dT2 ... dT26 Tamb|
//  The delta Ti's are in F, but Tamb is in C.  Hence, we
//  need to convert the TempF so that it reads in C (the
//  transformed temp matrix will be called TempM.

//  TempM = |tr1(501W) tr2(951W) tr3(1451)|'
//  tri = |t1 ... t9 Tamb|
//  t1 = |tc1  tc2   tc3|, t2 = |tc4 tc5   tc6|, t3 = |tc11 tc12 tc21|
//  t4 = |tc10 tc18 tc19|, t5 = |tc9 tc14 tc20|, t6 = |tc13 tc25 tc26|
//  t7 = |tc7  tc23 tc24|, t8 = |tc8       tc17|, t9 = |     tc16 tc22|

t1 = [31.5 31.5 29.5];
t2 = [23.5 24.5 24.5];
t3 = [35.5 35.5 33.5];
t4 = [29.0 35.5 36.5];
t5 = [31.0 29.5 35.0];
t6 = [33.0 38.5 35.0];
t7 = [33.5 38.5 38.0];
t8 = [33.5 37.0];
t9 = [37.0 38.0];
Tamb = 24.0;
tr1 = [t1 t2 t3 t4 t5 t6 t7 t8 t9 Tamb];

t1 = [39.0 38.5 36.5];
t2 = [28.5 30.5 30.5];
t3 = [44.5 44.5 42.0];
t4 = [36.0 43.0 46.0];
t5 = [39.0 36.0 44.0];
t6 = [41.0 48.5 44.0];
t7 = [42.5 48.0 48.0];
t8 = [42.5 45.5];
t9 = [45.5 47.5];
Tamb = 24.2;
tr2 = [t1 t2 t3 t4 t5 t6 t7 t8 t9 Tamb];

t1 = [52.5 52.5 49.5];
t2 = [39.0 41.0 41.0];
t3 = [61.0 61.0 57.5];
t4 = [49.0 59.5 62.5];
t5 = [54.0 49.0 59.5];
t6 = [55.5 69.5 62.5];
t7 = [59.0 67.5 67.5];
t8 = [58.5 62.5];
t9 = [62.5 67.0];
Tamb = 24.2;
tr3 = [t1 t2 t3 t4 t5 t6 t7 t8 t9 Tamb];

TempF = [tr1; tr2; tr3];

[Trows,Tcols] = size(TempF);
```

```
//  Delta of temp in c = 5[(dF   Famb) -32]/9 - Camb
//                     = 5*dF/9 + 5(Famb - 32)/9 - Camb = 5dF/9
//  Note tempF(:,26) = Ambient Temp, is in degrees Celsius

for i = 1:Trows,...
for j = 1:25,...
TempM(i,j) = 5*TempF(i,j)/9;...
end;...
end

TempM = [TempM TempF(:,26)];

//  Recall that stdata = |I V pf Pm Torque rpm|

std1 = [4.328 123.1 0.4826 501.36 2.680 1785];
std2 = [5.293 125.5 0.6352 949.24 5.120 1769];
std3 = [6.573 123.8 0.7606 1450.9 7.920 1748];

stdata = [std1; std2; std3];

//  do lsdata on each of the real data with known, measured Rs

Rs1 = (.867 + .868 + .865)/3;
Rs2 = (.895 + .895 + .885)/3;
Rs3 = (.915 + .914 + .912)/3;

pv1 = lsdata(r1,Rs1);
pv2 = lsdata(r2,Rs2);
pv3 = lsdata(r3,Rs3);

rinave = (r1 + r2 + r3)/3;
Rsave  = (Rs1 + Rs2 + Rs3)/3;
pvinave = lsdata(rinave,Rsave);

pv = [pv1 pv2 pv3];

Rs = [Rs1 Rs2 Rs3]';
Rr = [pv1(1) pv2(1) pv3(1)]';

save 'sct2data';
```

```
////////////////////////////////////////////////////////
//  Program anl.r (in [cho.matfiles.thermo.rdata])    //
//                                                    //
//  Use the rotor 1 data collected in ct1data.r and   //
//  Computes DT's of the thermocouples and gives the  //
//  thermally-compensated estimates of Rr             //
////////////////////////////////////////////////////////


//  Take data from ct3data.r and do various analyses

clear;
diary('dan1');
long;

load 'srthest' Rth Rthsw Rthro Rsnorm Rrnorm Tswnorm Tronorm;
load 'sct1data' pv1 pv2 pv3 pv stdata TempM Rs Rr;
load 'st31' Rsrt Rrrt;

//  Get the dimension for the matrix mdata

[nrows, ncols] = size(stdata);

//  Assuming that stdata is of the form
//  stdata = |Is V pf Psh Torque rpm|,
//             1  2  3  4    5     6
//  we can manipulate this data to obtain slips and Pe.

//  S = 1 - rpm/1800

for i=1:nrows,...
S(i) = 1 - stdata(i,6)/1800;...
end

//  The following are thermal sources for each matrix
//     Pir = slip*Pm/(1-slip), where Pm = (Torq,meas + Torq,loss)*Wm
//     = (2*PI*rpm/60)*(Torq,meas + Torq,loss) where Torq,loss = 0.3796
//     Pis = 3*Is*Is*Rsi
//  T = Pr*Rthr + Ps*Rths + Pc*Rthc, Pc*Rthc = const = Pc*Rthc
//     which is approx. Pc*Rths, with Pc = 23.51b*2.5W/lb

Torq1 = 0.3796;
Pc = 23.5*2.6;
for i=1:nrows,...
Pm = PI*(stdata(i,5) + Torq1)*stdata(i,6)/30;...
slip = S(i);...
Is = stdata(i,1);...
Pr(i) = slip*Pm/(1-slip);...
Ps(i) = 3*Is*Is*Rs(i) + Pc;...
end

A = [Pr Ps]

//  Thermal stuff
```

```
for i=1:nrows,...
for j=1:25,...
EstTempM(i,j) = A(i,1)*Rth(2*j-1) + A(i,2)*Rth(2*j);...
end,...
end
EstTempM = [EstTempM TempM(:,26)];

Tambrt = 24.2;
Tambv = TempM(:,26);

for i = 1:nrows,...
Tsw(i) = Rs(i)*(234.4 + Tambrt)/Rsrt - 234.4;...
Tro(i) = Rr(i)*(228.1 + Tambrt)/Rrrt - 228.1;...
end

TswEst = A*Rthsw;
TroEst = A*Rthro;

DT = TempM - EstTempM
DTsw = Tsw - TswEst - Tambv
DTro = Tro - TroEst - Tambv

//  Scale all the rotor resistances back to the 700W case

for i = 1:nrows,...
RsEst(i) = Rsrt*(TswEst(i) + Tambv(i) + 234.4)/(Tambrt+ 234.4);...
RrEst(i) = Rrrt*(TroEst(i) + Tambv(i) + 228.1)/(Tambrt + 228.1);...
end

Rs
Rr
RsEst
RrEst
Rsnorm
Rrnorm
dRs = Rs - RsEst
dRr = Rr - RrEst

for i = 1:nrows,...
perRs(i) = dRs(i)/Rs(i);...
perRr(i) = dRr(i)/Rr(i);...
end

perRs
perRr

//  We want to scale all the estimated resistances back to a reference
//  temperature and compare the deviance from each other.  In this case,
//  the reference Temp is the temperature at which the Pm was 700W.

for i=1:nrows,...
Rssc(i) = Rs(i)*Tswnorm/(TswEst(i)+Tambv(i)+234.4);...
Rrsc(i) = Rr(i)*Tronorm/(TroEst(i)+Tambv(i)+228.1);...
end

Rssc
Rrsc

for i = 1:nrows,...
dRssc(i) = Rssc(i) - Rsnorm;...
```

```
dRrsc(i) = Rrsc(i) - Rrnorm;...
perRssc(i) = dRssc(i)/Rsnorm;...
perRrsc(i) = dRrsc(i)/Rrnorm;...
end

dRssc
dRrsc
perRssc
perRrsc

save 'san1';

diary(0);
```

```
////////////////////////////////////////////////////////
//  Program ct2data.r (in [cho.matfiles.thermo.rdata])  //
//                                                      //
//  Experimental data from rotor 2 at                   //
//  loads of 500W, 950W,and 1450W                       //
//  Used by Program an2.r                               //
////////////////////////////////////////////////////////


//  Working with real data of healthy motor (stator 1, rotor 2)
//  at constant temp.
//  Date:  April 30, 1988.  Dynamometer fixed.
//  ri = [T Is Pe Pm pf rpm V]

clear;
long;
define 'param.udf'
define 'lsdata.udf'
zip = 0*ones(16,1);

//  real data 1, Pm = 501.36 W

Is1 = [3.856 4.036 4.328 4.483 4.773 5.120 5.541 5.915]';
Is2 = [6.321 6.755 7.098 7.506 7.861 8.258 8.626 9.100]';
Is = [Is1;Is2];
pf1 = [.2546 .3696 .4826 .5339 .5951 .6572 .6932 .7211]';
pf2 = [.7506 .7684 .7831 .7970 .8060 .8145 .8238 .8269]';
pf = [pf1;pf2];
rpm1 = [1795 1790 1785 1780 1775 1770 1765 1760]';
rpm2 = [1755 1750 1745 1740 1735 1730 1725 1720]';
rpm = [rpm1;rpm2];
v1 = [123.7 123.6 123.1 123.4 123.3 123.2 123.3 123.3]';
v2 = [123.2 123.2 122.9 122.9 122.7 122.7 122.7 123.1]';
v = [v1;v2];
r1 = [Is v pf zip zip rpm];

//  real data 2, Pm = 949.24 W

Is1 = [4.083 4.250 4.450 4.663 4.970 5.283 5.606 5.986]';
Is2 = [6.343 6.730 7.126 7.533 7.880 8.313 8.710 9.080]';
Is = [Is1;Is2];
pf1 = [.2479 .3653 .4473 .5298 .5887 .6329 .6748 .7087]';
pf2 = [.7333 .7595 .7719 .7879 .7997 .8044 .8165 .8258]';
pf = [pf1;pf2];
v1 = [125.9 125.9 125.8 125.6 125.6 125.4 125.4 125.3]';
v2 = [125.3 125.3 125.1 125.2 125.2 124.9 124.9 124.8]';
v = [v1;v2];
r2 = [Is v pf zip zip rpm];

//  real data 3, Pm = 1450.7 W

Is1 = [4.196 4.481 4.676 5.081 5.351 5.656 6.016 6.418]';
Is2 = [6.783 7.155 7.576 7.975 8.326 8.640 8.998 9.446]';
Is = [Is1;Is2];
pf1 = [.4285 .5175 .5663 .6328 .6691 .6963 .7254 .7494]';
```

```
pf2 = [.7677 .7841 .7975 .8072 .8154 .8218 .8289 .8331]';
pf = [pf1;pf2];
v1 = [124.4 124.3 124.2 124.3 124.1 124.0 124.0 123.9]';
v2 = [123.9 123.6 123.8 123.7 123.6 123.5 123.5 123.5]';
v = [v1;v2];
rpm1 = rpm - 10*ones(16,1);
r3 = [Is v pf zip zip rpm1];

//  We need stdata in the form |Is V pf Pm Torque rpm|
//  and TempM which is converted from TempF
//  where TempF = |dT1 dT2 ... dT26 Tamb|
//  The delta Ti's are in F, but Tamb is in C.  Hence, we
//  need to convert the TempF so that it reads in C (the
//  transformed temp matrix will be called TempM.

//  TempM = |tr1(501W) tr2(951W) tr3(1451)|'
//  tri = |t1 ... t9 Tamb|
//  t1 = |tc1  tc2   tc3|, t2 = |tc4 tc5   tc6|, t3 = |tc11 tc12 tc21|
//  t4 = |tc10 tc18 tc19|, t5 = |tc9 tc14 tc20|, t6 = |tc13 tc25 tc26|
//  t7 = |tc7  tc23 tc24|, t8 = |tc8      tc17|, t9 = |     tc16 tc22|

t1 = [31.5 31.5 29.0];
t2 = [23.0 24.5 23.5];
t3 = [35.0 34.5 33.5];
t4 = [29.5 34.5 35.5];
t5 = [31.5 28.5 34.5];
t6 = [33.0 37.5 34.0];
t7 = [33.5 38.0 37.5];
t8 = [33.5 36.5];
t9 = [36.5 37.5];
Tamb = 24.0;
tr1 = [t1 t2 t3 t4 t5 t6 t7 t8 t9 Tamb];

t1 = [40.0 40.0 37.5];
t2 = [29.5 30.5 30.5];
t3 = [45.5 45.5 43.0];
t4 = [37.5 45.0 46.5];
t5 = [40.0 36.0 45.0];
t6 = [42.5 50.0 45.5];
t7 = [43.5 49.5 49.5];
t8 = [43.0 47.0];
t9 = [46.5 49.0];
Tamb = 24.2;
tr2 = [t1 t2 t3 t4 t5 t6 t7 t8 t9 Tamb];

t1 = [52.5 52.5 50.0];
t2 = [38.5 39.5 39.5];
t3 = [61.5 60.5 57.5];
t4 = [49.5 60.0 62.5];
t5 = [54.5 48.5 59.5];
t6 = [56.0 69.5 62.5];
t7 = [59.5 68.0 67.5];
t8 = [57.5 62.5];
t9 = [62.0 68.0];
Tamb = 24.4;
tr3 = [t1 t2 t3 t4 t5 t6 t7 t8 t9 Tamb];

TempF = [tr1; tr2; tr3];

[Trows,Tcols] = size(TempF);
```

```
//  Delta of temp in c = 5[(dF + Famb) -32]/9 - Camb
//                     = 5*dF/9 + 5(Famb - 32)/9 - Camb = 5dF/9
//  Note tempF(:,26) = Ambient Temp, is in degrees Celsius

for i = 1:Trows,...
for j = 1:25,...
TempM(i,j) = 5*TempF(i,j)/9;...
end;...
end

TempM = [TempM TempF(:,26)];

//  Recall that stdata = |I V pf Pm Torque rpm|

std1 = [4.408 124.2 0.4632 501.36 2.680 1785];
std2 = [5.200 123.4 0.6494 951.63 5.130 1770];
std3 = [6.590 124.0 0.7584 1453.5 7.930 1749];

stdata = [std1; std2; std3];

//  do lsdata on each of the real data with known, measured Rs

Rs1 = (.867 + .867 + .866)/3;
Rs2 = (.884 + .884 + .881)/3;
Rs3 = (.914 + .914 + .912)/3;

pv1 = lsdata(r1,Rs1);
pv2 = lsdata(r2,Rs2);
pv3 = lsdata(r3,Rs3);

rinave = (r1 + r2 + r3)/3;
Rsave  = (Rs1 + Rs2 + Rs3)/3;
pvinave = lsdata(rinave,Rsave);

pv = [pv1 pv2 pv3];

Rs = [Rs1 Rs2 Rs3]';
Rr = [pv1(1) pv2(1) pv3(1)]';

save 'sctldata';
```

```
////////////////////////////////////////////////////////////
//  Program an2.r (in [cho.matfiles.thermo.rdata])        //
//                                                        //
//  Use the rotor 2 data collected in ctldata.r and       //
//  Computes DT's of the thermocouples and gives the      //
//  thermally-compensated estimates of Rr                 //
////////////////////////////////////////////////////////////


//  Take data from ct3data.r and do various analyses

clear;
diary('dan2');
long;

load 'srthest' Rth Rthsw Rthro Rsnorm Rrnorm Tswnorm Tronorm;
load 'sct2data' pv1 pv2 pv3 pv stdata TempM Rs Rr;
load 'st31' Rsrt Rrrt;

//  Get the dimension for the matrix mdata

[nrows, ncols] = size(stdata);

//  Assuming that stdata is of the form
//  stdata = [Is V pf Psh Torque rpm],
//             1  2  3  4    5     6
//  we can manipulate this data to obtain slips and Pe.

//  S = 1 - rpm/1800

for i=1:nrows,...
S(i) = 1 - stdata(i,6)/1800;...
end

//  The following are thermal sources for each matrix
//     Pir = slip*Pm/(1-slip), where Pm = (Torq,meas + Torq,loss)*Wm
//     = (2*PI*rpm/60)*(Torq,meas + Torq,loss) where Torq,loss = 0.3796
//     Pis = 3*Is*Is*Rsi
//  T = Pr*Rthr + Ps*Rths + Pc*Rthc, Pc*Rthc = const = Pc*Rthc
//     which is approx. Pc*Rths, with Pc = 23.5lb*2.5W/lb

Torql = 0.3796;
Pc = 23.5*2.6;
for i=1:nrows,...
Pm = PI*(stdata(i,5) + Torql)*stdata(i,6)/30;...
slip = S(i);...
Is = stdata(i,1);...
Pr(i) = slip*Pm/(1-slip);...
Ps(i) = 3*Is*Is*Rs(i) + Pc;...
end

A = [Pr Ps]

//  Thermal stuff
```

```
for i=1:nrows,...
for j=1:25,...
EstTempM(i,j) = A(i,1)*Rth(2*j-1) + A(i,2)*Rth(2*j);...
end,...
end
EstTempM = [EstTempM TempM(:,26)];

Tambrt = 24.2;
Tambv = TempM(:,26);

for i = 1:nrows,...
Tsw(i) = Rs(i)*(234.4 + Tambrt)/Rsrt - 234.4;...
Tro(i) = Rr(i)*(228.1 + Tambrt)/Rrrt - 228.1;...
end

TswEst = A*Rthsw;
TroEst = A*Rthro;

DT = TempM - EstTempM
DTsw = Tsw - TswEst - Tambv
DTro = Tro - TroEst - Tambv

//  Scale all the rotor resistances back to the 700W case

for i = 1:nrows,...
RsEst(i) = Rsrt*(TswEst(i) + Tambv(i) + 234.4)/(Tambrt+ 234.4);...
RrEst(i) = Rrrt*(TroEst(i) + Tambv(i) + 228.1)/(Tambrt + 228.1);...
end

Rs
Rr
RsEst
RrEst
Rsnorm
Rrnorm
dRs = Rs - RsEst
dRr = Rr - RrEst

for i = 1:nrows,...
perRs(i) = dRs(i)/Rs(i);...
perRr(i) = dRr(i)/Rr(i);...
end

perRs
perRr

//  We want to scale all the estimated resistances back to a reference
//  temperature and compare the deviance from each other.  In this case,
//  the reference Temp is the temperature at which the Pm was 700W.

for i=1:nrows,...
Rssc(i) = Rs(i)*Tswnorm/(TswEst(i)+Tambv(i)+234.4);...
Rrsc(i) = Rr(i)*Tronorm/(TroEst(i)+Tambv(i)+228.1);...
end

Rssc
Rrsc

for i = 1:nrows,...
dRssc(i) = Rssc(i) - Rsnorm;...
```

```
dRrsc(i) = Rrsc(i) - Rrnorm;...
perRssc(i) = dRssc(i)/Rsnorm;...
perRrsc(i) = dRrsc(i)/Rrnorm;...
end

dRssc
dRrsc
perRssc
perRrsc

save 'san2';

diary(0);
```

```
////////////////////////////////////////////////////////////
//  Program ct3data.r (in [cho.matfiles.thermo.rdata])    //
//                                                        //
//  Experimental data from rotor 1 at                     //
//  loads of 500W, 950W,and 1450W                         //
//  Used by Program an3.r                                 //
////////////////////////////////////////////////////////////


//  Working with real data of healthy motor (stator 1, rotor 3)
//  at constant temp.
//  Date:  April 30, 1988.  Dynamometer fixed.
//  ri = [T Is Pe Pm pf rpm V]

clear;
long;
define 'param.udf'
define 'lsdata.udf'
zip = 0*ones(16,1);

//  real data 1, Pm = 501.36 W

Is1 = [4.100 4.253 4.483 4.726 5.046 5.343 5.753 6.090]';
Is2 = [6.453 6.796 7.240 7.650 8.146 8.456 8.926 9.373]';
Is = [Is1;Is2];
pf1 = [.2448 .3455 .4485 .5187 .5879 .6328 .6758 .7050]';
pf2 = [.7308 .7550 .7742 .7884 .8031 .8134 .8180 .8249]';
pf = [pf1;pf2];
rpm1 = [1795 1790 1785 1780 1775 1770 1765 1760]';
rpm2 = [1755 1750 1745 1740 1735 1730 1725 1720]';
rpm = [rpm1;rpm2];
v1 = [124.5 124.5 124.3 124.1 124.0 124.0 124.2 124.1]';
v2 = [123.9 123.4 123.6 123.6 123.6 123.4 123.6 123.5]';
v = [v1;v2];
r1 = [Is v pf zip zip rpm];

//  real data 2, Pm = 952.94 W

Is1 = [4.100 4.226 4.443 4.683 4.970 5.326 5.663 5.996]';
Is2 = [6.380 6.790 7.213 7.593 8.036 8.340 8.830 9.153]';
Is = [Is1;Is2];
pf1 = [.2516 .3517 .4457 .5196 .5812 .6387 .6751 .7072]';
pf2 = [.7366 .7548 .7736 .7888 .8036 .8100 .8205 .8236]';
pf = [pf1;pf2];
v1 = [124.5 124.4 124.2 124.1 124.1 123.8 123.9 123.6]';
v2 = [123.7 123.7 123.7 123.7 123.5 123.3 123.3 123.4]';
v = [v1;v2];
r2 = [Is v pf zip zip rpm];

//  real data 3, Pm = 1450.7 W

Is1 = [4.396 4.621 4.873 5.223 5.493 5.870 6.240 6.598]';
Is2 = [6.971 7.386 7.806 8.148 8.523 8.888 9.306 9.738]';
Is = [Is1;Is2];
pf1 = [.4247 .5099 .5638 .6266 .6657 .7023 .7312 .7517]';
```

```
pf2 = [.7727 .7880 .8000 .8095 .8181 .8257 .8310 .8376]';
pf = [pf1;pf2];
v1 = [125.8 125.6 125.6 125.7 125.3 125.4 125.3 125.3]';
v2 = [125.1 125.1 125.2 125.1 124.8 124.8 124.9 124.8]';
v = [v1;v2];
rpm1 = rpm - 10*ones(16,1);
r3 = [Is v pf zip zip rpm1];

//  We need stdata in the form |Is V pf Pm Torque rpm|
//  and TempM which is converted from TempF
//  where TempF = |dT1 dT2 ... dT26 Tamb|
//  The delta Ti's are in F, but Tamb is in C.  Hence, we
//  need to convert the TempF so that it reads in C (the
//  transformed temp matrix will be called TempM.

//  TempM = |tr1(501W) tr2(950W) tr3(1451)|'
//  tri = |t1 ... t9 Tamb|
//  t1 = |tc1  tc2   tc3|, t2 = |tc4 tc5   tc6|, t3 = |tc11 tc12 tc21|
//  t4 = |tc10 tc18 tc19|, t5 = |tc9 tc14 tc20|, t6 = |tc13 tc25 tc26|
//  t7 = |tc7  tc23 tc24|, t8 = |tc8      tc17|, t9 = |     tc16 tc22|

t1 = [32.0 32.0 30.5];
t2 = [23.5 24.5 24.5];
t3 = [35.5 35.5 34.0];
t4 = [30.5 36.5 36.5];
t5 = [31.5 29.0 36.5];
t6 = [34.0 39.5 35.5];
t7 = [34.5 39.0 38.5];
t8 = [34.0 37.5];
t9 = [37.5 38.5];
Tamb = 24.6;
tr1 = [t1 t2 t3 t4 t5 t6 t7 t8 t9 Tamb];

t1 = [38.5 38.5 36.5];
t2 = [28.5 29.5 29.5];
t3 = [43.5 43.5 41.5];
t4 = [36.5 43.5 44.5];
t5 = [39.5 35.0 43.5];
t6 = [40.5 48.5 44.0];
t7 = [42.5 48.5 47.5];
t8 = [41.5 45.5];
t9 = [45.5 47.5];
Tamb = 24.2;
tr2 = [t1 t2 t3 t4 t5 t6 t7 t8 t9 Tamb];

t1 = [52.0 52.0 48.0];
t2 = [38.5 41.0 40.5];
t3 = [60.5 60.5 56.5];
t4 = [48.5 59.5 59.5];
t5 = [51.5 47.5 58.0];
t6 = [55.5 69.0 61.5];
t7 = [57.5 66.5 66.5];
t8 = [57.5 62.0];
t9 = [62.5 66.0];
Tamb = 24.2;
tr3 = [t1 t2 t3 t4 t5 t6 t7 t8 t9 Tamb];

TempF = [tr1; tr2; tr3];

[Trows,Tcols] = size(TempF);
```

```
//  Delta of temp in c = 5[(dF + Famb) -32]/9 - Camb
//                     = 5*dF/9 + 5(Famb - 32)/9 - Camb = 5dF/9
//  Note tempF(:,26) = Ambient Temp, is in degrees Celsius

for i = 1:Trows,...
for j = 1:25,...
TempM(i,j) = 5*TempF(i,j)/9;...
end;...
end

TempM = [TempM TempF(:,26)];

//  Recall that stdata = |I V pf Pm Torque rpm|

std1 = [4.483 124.3 0.4485 501.36 2.680 1785];
std2 = [5.330 123.9 0.6375 952.94 5.140 1769];
std3 = [6.598 125.3 0.7517 1450.7 7.910 1750];

stdata = [std1; std2; std3];

//  do lsdata on each of the real data with known, measured Rs

Rs1 = (.872 + .873 + .869)/3;
Rs2 = (.882 + .882 + .881)/3;
Rs3 = (.912 + .912 + .910)/3;

pv1 = lsdata(r1,Rs1);
pv2 = lsdata(r2,Rs2);
pv3 = lsdata(r3,Rs3);

rinave = (r1 + r2 + r3)/3;
Rsave  = (Rs1 + Rs2 + Rs3)/3;
pvinave = lsdata(rinave,Rsave);

pv = [pv1 pv2 pv3];

Rs = [Rs1 Rs2 Rs3]';
Rr = [pv1(1) pv2(1) pv3(1)]';

save 'sct3data';
```

```
////////////////////////////////////////////////////////////
//  Program an3.r (in [cho.matfiles.thermo.rdata])        //
//                                                        //
//  Use the rotor 3 data collected in ct1data.r and       //
//  Computes DT's of the thermocouples and gives the      //
//  thermally-compensated estimates of Rr                 //
////////////////////////////////////////////////////////////


//  Take data from ct3data.r and do various analyses

clear;
diary('dan3');
long;

load 'srthest' Rth Rthsw Rthro Rsnorm Rrnorm Tswnorm Tronorm;
load 'sct3data' pv1 pv2 pv3 pv stdata TempM Rs Rr;
load 'st31' Rsrt Rrrt;

//  Get the dimension for the matrix mdata

[nrows, ncols] = size(stdata);

//  Assuming that stdata is of the form
//  stdata = [Is V pf Psh Torque rpm],
//             1  2  3  4    5     6
//  we can manipulate this data to obtain slips and Pe.

//  S = 1 - rpm/1800

for i=1:nrows,...
S(i) = 1 - stdata(i,6)/1800;...
end

//  The following are thermal sources for each matrix
//     Pir = slip*Pm/(1-slip), where Pm = (Torq,meas + Torq,loss)*Wm
//     = (2*PI*rpm/60)*(Torq,meas + Torq,loss) where Torq,loss = 0.3796
//     Pis = 3*Is*Is*Rsi
//  T = Pr*Rthr + Ps*Rths + Pc*Rthc, Pc*Rthc = const = Pc*Rthc
//     which is approx. Pc*Rths, with Pc = 23.5lb*2.5W/lb

Torql = 0.3796;
Pc = 23.5*2.6;
for i=1:nrows,...
Pm = PI*(stdata(i,5) + Torql)*stdata(i,6)/30;...
slip = S(i);...
Is = stdata(i,1);...
Pr(i) = slip*Pm/(1-slip);...
Ps(i) = 3*Is*Is*Rs(i) + Pc;...
end

A = [Pr Ps]

//  Thermal stuff

for i=1:nrows,...
```

```
for j=1:25,...
EstTempM(i,j) = A(i,1)*Rth(2*j-1) + A(i,2)*Rth(2*j);...
end,...
end
EstTempM = [EstTempM TempM(:,26)];

Tambrt = 24.2;
Tambv = TempM(:,26);

for i = 1:nrows,...
Tsw(i) = Rs(i)*(234.4 + Tambrt)/Rsrt - 234.4;...
Tro(i) = Rr(i)*(228.1 + Tambrt)/Rrrt - 228.1;...
end

TswEst = A*Rthsw;
TroEst = A*Rthro;

DT = TempM - EstTempM
DTsw = Tsw - TswEst - Tambv
DTro = Tro - TroEst - Tambv

//  Scale all the rotor resistances back to the 700W case

for i = 1:nrows,...
RsEst(i) = Rsrt*(TswEst(i) + Tambv(i) + 234.4)/(Tambrt+ 234.4);...
RrEst(i) = Rrrt*(TroEst(i) + Tambv(i) + 228.1)/(Tambrt + 228.1);...
end

Rs
Rr
RsEst
RrEst
Rsnorm
Rrnorm
dRs = Rs - RsEst
dRr = Rr - RrEst

for i = 1:nrows,...
perRs(i) = dRs(i)/Rs(i);...
perRr(i) = dRr(i)/Rr(i);...
end

perRs
perRr

//  We want to scale all the estimated resistances back to a reference
//  temperature and compare the deviance from each other.  In this case,
//  the reference Temp is the temperature at which the Pm was 700W.

for i=1:nrows,...
Rssc(i) = Rs(i)*Tswnorm/(TswEst(i)+Tambv(i)+234.4);...
Rrsc(i) = Rr(i)*Tronorm/(TroEst(i)+Tambv(i)+228.1);...
end

Rssc
Rrsc

for i = 1:nrows,...
dRssc(i) = Rssc(i) - Rsnorm;...
dRrsc(i) = Rrsc(i) - Rrnorm;...
```

```
perRssc(i) = dRssc(i)/Rsnorm;...
perRrsc(i) = dRrsc(i)/Rrnorm;...
end

dRssc
dRrsc
perRssc
perRrsc

save 'san3';

diary(0);
```

Appendix I

Results for the Thermally-Compensated Detector of Broken Rotor Bar

```
//////////////////////////////////////////
//  Experimental Results for Rotor 1    //
//////////////////////////////////////////


//////////////////////////////////////////
//  DTi's (in Celsius) of thermocouple  //
//  TCi computed such that              //
//                                      //
//  DTi = |500W 950W 1450W| and         //
//  DT  = |DT1|                         //
//        |DT2|                         //
//         ...                          //
//        |DT25|                        //
//////////////////////////////////////////
```

DT =

-0.2208	0.5169	0.3297
-0.1557	0.3060	0.4017
-0.2686	0.2401	-0.1024
0.1431	0.4730	0.8143
0.0450	0.6796	0.4691
0.0840	0.7753	0.6879
0.1907	0.9204	0.5548
0.1907	0.9204	0.5548
-0.1497	0.4453	-0.1548
-0.1759	0.3520	0.0570
0.2867	0.3254	0.2684
0.1216	1.1402	0.7884
-0.1850	0.4863	0.3944
0.5743	1.0237	1.1666
-0.1913	0.7578	0.3085
-0.0151	0.6948	0.3933
-0.0724	0.3231	0.5054
0.1205	0.5689	0.7104
-0.2797	0.5893	0.5345
0.0235	0.5613	0.8189
-0.1301	0.6514	0.8374
-0.0776	0.8707	0.7128
-0.1617	0.4320	0.6312
-0.0082	0.5352	0.6285
0.0830	0.6213	0.8851

```
////////////////////////////////////////////////
//  Thermally Compensated and Thermally       //
//  Uncompensated Estimates of Rr             //
//  and Measured vs Thermally Estimated       //
//  of Rs                                     //
//                                            //
//  Rr = |Load Uncompensated Compensated|     //
//       and                                  //
//  Rs = |Load Measured  Estimated|           //
////////////////////////////////////////////////
```

RR	=	
500W	0.5705	0.5782
950W	0.5819	0.5737
1450W	0.6184	0.5764

RS	=	
500W	0.8667	0.8713
950W	0.8830	0.8857
1450W	0.9133	0.9164

```
//////////////////////////////////////////
//  Experimental Results for Rotor 2    //
//////////////////////////////////////////


//////////////////////////////////////////
//  DTi's (in Celsius) of thermocouple  //
//  TCi computed such that              //
//                                      //
//  DTi = |500W 950W 1450W| and         //
//  DT  = |DT1|                         //
//        |DT2|                         //
//         ...                          //
//        |DT25|                        //
//////////////////////////////////////////
```

DT =

0.0672	0.5437	0.4145
0.1311	0.6121	0.4854
-0.2767	0.2932	0.2437
0.0755	0.6450	0.6020
0.2650	0.2723	-0.3046
-0.2519	0.3709	-0.0825
0.2270	0.8762	0.8906
-0.0508	0.8762	0.3350
0.1518	0.4239	-0.1010
0.3651	0.6922	0.3976
0.0445	0.8439	0.6126
-0.1089	0.8041	0.8589
0.3729	0.5084	0.7262
0.2750	0.5484	0.9572
-0.1517	0.7188	0.3843
0.2819	0.9743	0.7456
-0.2860	0.4812	0.5352
-0.1266	0.7975	0.7441
0.0241	0.5647	0.8683
0.0888	0.7362	1.1551
-0.0672	0.8280	0.8911
0.2232	0.5758	0.2152
-0.1036	0.6431	0.7217
0.0468	0.4704	0.4348
0.1424	0.8039	1.4927

```
////////////////////////////////////////////////
//  Thermally Compensated and Thermally       //
//  Uncompensated Estimates of Rr             //
//  and Measured vs Thermally Estimated       //
//  of Rs                                     //
//                                            //
//  Rr = |Load Uncompensated Compensated|     //
//       and                                  //
//  Rs = |Load Measured  Estimated|           //
////////////////////////////////////////////////
```

RR	=	
500W	0.5828	0.5911
950W	0.6003	0.5905
1450W	0.6325	0.5886

RS	=	
500W	0.8667	0.8702
950W	0.8917	0.8877
1450W	0.9137	0.9167

```
//////////////////////////////////////////
//  Experimental Results for Rotor 3    //
//////////////////////////////////////////


//////////////////////////////////////////
//  DTi's (in Celsius) of thermocouple  //
//  TCi computed such that              //
//                                      //
//  DTi = |500W 950W 1450W| and         //
//  DT  = |DT1|                         //
//        |DT2|                         //
//          ..                          //
//        |DT25|                        //
//////////////////////////////////////////
```

DT =

-0.2623	-0.3216	0.0540
-0.1961	-0.2531	0.1269
-0.0120	-0.2932	-0.9206
-0.0899	0.0664	0.5338
-0.1989	-0.3081	0.4770
-0.1596	-0.2092	0.4138
-0.1576	-0.2725	0.3207
-0.1576	-0.2725	0.3207
-0.2062	-0.4457	-0.6663
0.3655	0.1062	-0.2012
0.4948	-0.0263	0.3016
0.2390	-0.3451	-0.8456
-0.2179	0.1972	-0.9583
0.0123	-0.0362	0.3437
0.2900	-0.1512	-0.5014
0.2112	-0.1709	0.4110
0.1040	-0.3958	0.3150
0.0561	-0.0746	0.2256
-0.0610	-0.0272	-0.2565
-0.0792	0.1391	0.3169
-0.2302	-0.3247	0.3403
-0.1333	-0.2934	0.1961
-0.2563	-0.2282	0.3659
-0.0996	-0.1233	0.6478
-0.0131	-0.0706	0.3885

```
////////////////////////////////////////////
//  Thermally Compensated and Thermally   //
//  Uncompensated Estimates of Rr         //
//  and Measured vs Thermally Estimated   //
//  of Rs                                 //
//                                        //
//  Rr = |Load Uncompensated Compensated|  //
//       and                              //
//  Rs = |Load Measured  Estimated|       //
////////////////////////////////////////////
```

RR		
500W	0.5788	0.5847
950W	0.5862	0.5766
1450W	0.6234	0.5817
Ambient	0.5413	0.5830
250W	0.5725	0.5831
700W	0.5820	0.5820
1.2kW	0.6101	0.5841
1.7kW	0.6576	0.5826

RS		
500W	0.8713	0.8744
950W	0.8817	0.8878
1450W	0.9113	0.9163
250W	0.8693	0.8696
700W	0.8817	0.8798
1.2kW	0.9017	0.9040
1.7kW	0.9483	0.9476