5-2014

Development of a Method for Incorporating Fault Codes in Prognostic Analysis

Eric Allen Strong
University of Tennessee - Knoxville, estrong3@utk.edu

To the Graduate Council:

I am submitting herewith a dissertation written by Eric Allen Strong entitled "Development of a Method for Incorporating Fault Codes in Prognostic Analysis." I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Nuclear Engineering.

J W. Hines, Major Professor

We have read this dissertation and recommend its acceptance:

Belle Upadhyaya, Lawrence Heilbronn, Mingzhou Jin, Jamie Coble

Accepted for the Council:

Carolyn R. Hodges

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

Development of a Method for Incorporating Fault Codes in Prognostic Analysis

A Dissertation Presented for the
Doctor of Philosophy
Degree
The University of Tennessee, Knoxville

Eric Allen Strong
May 2014

ACKNOWLEDGMENTS

I would like to acknowledge my gratitude to the people who have assisted me in achieving the Doctor of Philosophy degree in Nuclear Engineering. First, I want to thank the members of my doctoral committee for their advice and comments while considering my dissertation: Dr. Hines, Dr. Upadhyaya, Dr. Heilbronn, Dr. Jin, and Dr. Coble. I would especially like to thank Dr. Hines for his continued support throughout the four years that I have performed research under his guidance and for the many topics I have learned while taking his classes. I am also grateful to the many other professors at the University of Tennessee whose classes have increased my knowledge in the fields of nuclear engineering, statistics, computer programming, and mathematics.

I am very appreciative of my family, including my wife Kayla, my parents Allen and Beth, my sister Emily, and my grandparents Carol and Earl, for their love and encouragement while completing this degree. I am particularly thankful for the time Kayla has spent proofreading this document.

Finally, I am also very grateful for my friends, especially those in the nuclear engineering program, who have given me assistance in my research over the last several years. In particular, I would like to thank Michael Sharp and Jamie Coble for their advice and valuable insights about performing research when I first entered the Nuclear Engineering program at the University of Tennessee, and their continued guidance since then.

ABSTRACT

Information from fault codes associated with a component may be used as an indicator of its health. A fault code is defined as a timestamp at which a component is not operating according to recommended guidelines. The type of fault codes which are relevant for this analysis represent mild or moderate deviations from normal behavior, rather than those requiring immediate repair. Potentially, fault codes may be used to determine the Remaining Useful Life (RUL) of a component by predicting its failure time, which will improve safety and reduce maintenance costs associated with the component. In this dissertation, methods have been developed to integrate the degradation information from fault codes into an existing prognostic parameter to improve the estimation of RUL. Optimization methods such as gradient descent were used to weight each fault code based on their relevance to degradation. Furthermore, topic models, a document analysis and clustering technique, were used as both a dimension-reduction method and fault mode isolation. Methods developed for this dissertation were applied to two real-world data sets, an actuator system and monitored signals from a motor accelerated degradation experiment. The best estimation of RUL for the actuator system was a topic model with a mean absolute error of 6.41% of the data received, and the best estimation of RUL for the motor accelerated degradation experiment was 5.7% of the average lifetime of the motors. The primary contributions of this research includes a method to construct a prognostic parameter from fault codes alone, the integration of degradation information from fault codes into an existing prognostic parameter, the use of topic models in reliability analysis of fault codes, and a software suite that performs these functions on generic data sets.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS AND SYMBOLS

AAKR	Auto-Associative Kernel Regression
AC	Alternating Current
BIST	Built-In Self-Test
COLT	Common Organizational Level Tester
FFT	Fast Fourier Transform
GA	Genetic Algorithms
GPM	General Path Model
HP	Horsepower
IC	Integrated Circuits
IEEE	Institute of Electrical and Electronics Engineers
LDA	Latent Dirichlet Allocation
LRU	Line-Replaceable Unit
MATLAB	Matrix Laboratory
MCMC	Monte Carlo Markov Chain
MTTD	Mean Time to Detect
MTTR	Mean Time to Repair
NFF	No Fault Found
NPP	Nuclear Power Plant
OMP	Optimized Merged Parameter
PCA	Principal Component Analysis
PEP	Process and Equipment Prognostics
PHM	Proportional Hazards Model
RAM	Random-Access Memory
RUL	Remaining Useful Life
SPRT	Sequential Probability Ratio Test
VLSI	Very Large Scale Integrated

CHAPTER 1
INTRODUCTION

For power generation, manufacturing, and other commercial applications, the failure of a component can be expensive in terms of safety considerations, lost production, and repair costs. Failure, in this sense, does not merely refer to a catastrophic event but is defined more broadly as the inability of the component to meet its design specifications. As an example, a pump which is designed to draw 10 gallons per minute will be considered to have failed when it is unable to meet that demand. After a failure event has occurred, the component must usually be repaired or replaced to restore operation to design specifications.

The costs associated with failure can be quantified for many systems. As failure data is gathered from a large number of components, the average time that must be spent to restore the component to its design specifications is called the Mean Time to Repair (MTTR). If the average production per unit time of the component is known, it can be multiplied by the MTTR to find the profit lost during the time period of the repair. In addition, there are direct costs accrued from repairing a component, such as the hourly wage of a maintenance worker, or the price of parts within a component or the entire component itself. Together, the repair costs and lost production contribute to the price of failure (Ebeling 2010). However, in many cases, component failure also represents a safety risk for workers or other nearby persons. For these safety-critical components, reliable operation is considered to be extremely important and should not be relegated to simple cost-benefit analysis. Rather, the focus should be on reducing the failure rate of the component and increasing the early detection of impending failure.

As an example of a situation in which the costs of failure might be considered, imagine a machine which produces a particular food item and has a MTTR of 10 hours with an average production of $55 per hour. The lost production associated with a failure event would thus be $550, or $55/hour times 10 hours. The hourly wage of the maintenance worker assigned to repair the machine is $20, and the average cost of replacement parts for the failed components within the machine is $300. The repair costs, then, will be $20/hour times 10 hours plus $300, or a total of $500. Hence, the total cost associated with a failure of the machine will be $1050. In this hypothetical scenario, failure of the machine is not considered to be a safety risk to nearby

workers, so the failure rates of the component will not necessarily need to be minimized, unless the cost of failure is not deemed acceptable. However, if failure of the machine was considered to be a safety risk, it is likely that minimization of failure rates will be necessary, and accurate prediction of the time at which the component will fail is extremely important.

To mitigate the costs of failure, four primary types of maintenance strategies exist. The first type is corrective maintenance, in which the component is repaired only after it has failed. In this case, no unnecessary maintenance is performed, since the component is never serviced before failure occurs. This type reduces maintenance costs but increases the number of failures that will occur. The second type of maintenance strategy is preventative maintenance. For this type, maintenance is performed on the component according to a particular maintenance schedule, such as weekly or monthly, before the component experiences failure. The second type will increase maintenance costs, since unnecessary maintenance will be performed, but the number of failures will be less than corrective maintenance. The third type of maintenance strategy is called condition-based maintenance. For this type of maintenance, monitored signals are used to determine the health of the component. When the component reaches a certain level of degradation, maintenance is performed on the component. The final type of maintenance strategy is proactive maintenance, in which components are redesigned to limit or eliminate common failure modes.

Condition-based maintenance will have lower maintenance costs than preventative maintenance and less number of failures than corrective maintenance, but the monitoring system is an added cost that must be considered in a cost-benefit analysis. For an individual system, if the cost of the monitoring equipment is cheaper than the increased maintenance costs of preventative maintenance or the costs associated with failure due to corrective maintenance, then condition-based maintenance is the ideal maintenance strategy. Figure 1-1 shows what a cost-benefit analysis of the three types of maintenance strategies might look like for a hypothetical system. The total cost is minimized for condition-based maintenance, and thus condition-based maintenance would be the ideal choice for this situation. In this situation, of course, it is assumed that safety is not a consideration.

Diagnostics is defined as the assessment of when and how a failure has occurred or will imminently occur in a component, along with root cause analysis of the failure mode involved. Typically, diagnostics is accomplished through condition monitoring, which is the process of collecting information about the health of a component using monitored signals, such as vibration. For diagnostics to be successful, the monitored signal must relate in some way to the normal operation of the component. When the monitored signal is no longer within the bounds of normal operation, a failure has occurred or will imminently occur. Figure 1-2 shows how an electrical motor might be diagnosed based on monitored signals. For example, aberrant behavior in the vibration signal could indicate misalignment or imbalance of the motor, which would be helpful in determining the appropriate maintenance strategy for the machine (Wagner 2013).

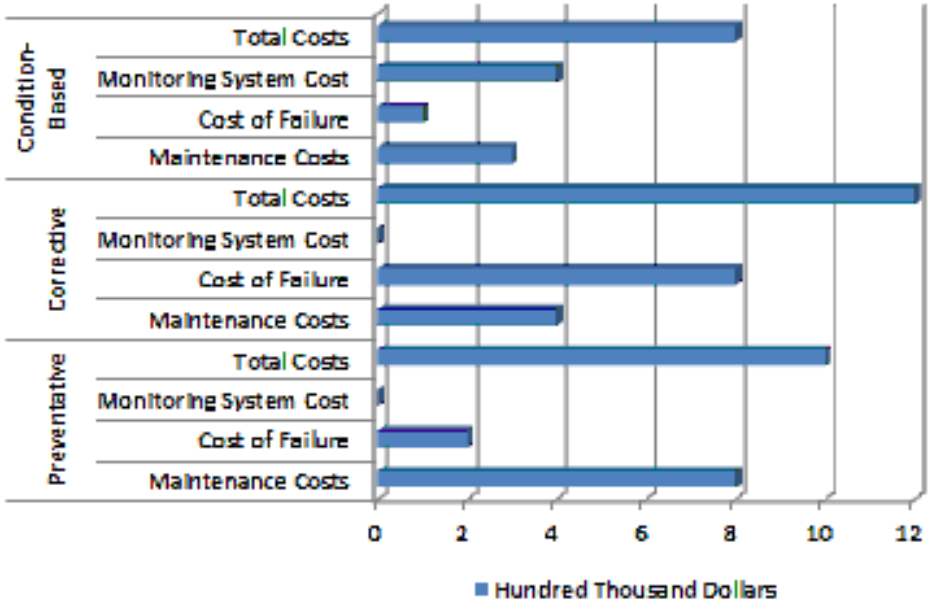

Figure 1-1. Sample Cost-Benefit Analysis of Condition-Based Maintenance.

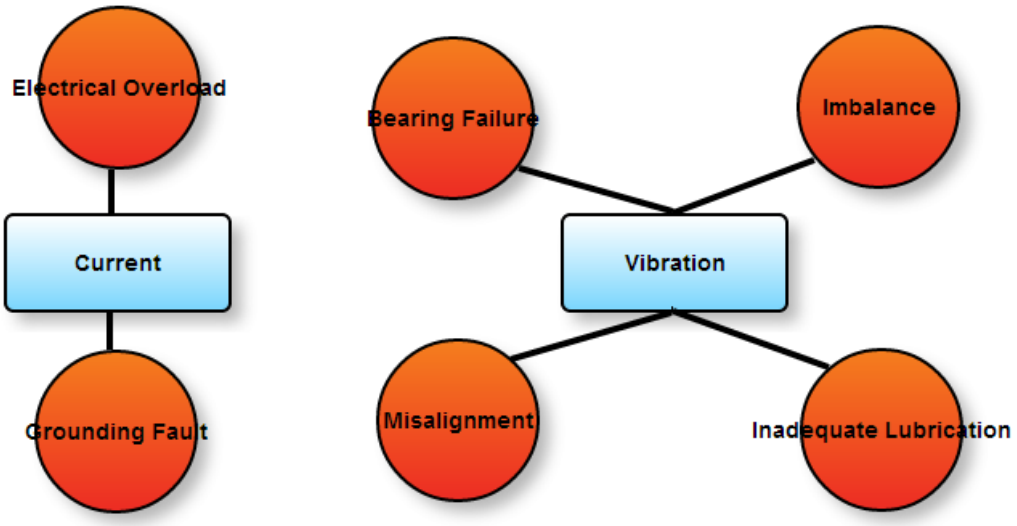

Figure 1-2. Diagnosing Motor Faults Using Monitored Signals.

Usually, a large number of failed components must be observed over time to quantify the bounds of normal operation. Once imminent failure is predicted using the monitored signal, the component may be taken offline for inspection, repair, replacement, or other maintenance actions. Diagnostics will determine which failure mode is responsible for the component failure, especially when there are several competing failure modes for the system. This information will be useful in determining an appropriate maintenance strategy for an individual component and might also be beneficial in deciding the best use of maintenance resources for the system overall.

A system monitored using a diagnostic process has several advantages over an unmonitored system. First, diagnostics coupled with condition monitoring may reduce the Mean Time to Detect (MTTD) of a component, which refers to the average amount of time that a maintenance worker must spend to discover a fault within the component. If diagnostics is able to determine the fault mode using monitored signals, the amount of time needed for a maintenance technician to corroborate the fault is greatly reduced. Second, diagnostics decreases the cost associated with Line-Replaceable Units (LRU). If the component which contains the

fault can be identified, it can be replaced quickly and with minimal identification costs. Finally, diagnostics can reduce the MTTR, which increases the availability of the system (Wheeler 2010).

While diagnostics has been used effectively in many situations to reduce maintenance costs, it is primarily a reactionary strategy that must wait for failure to occur before action can be taken. A more effective method is the prediction of a future failure event before it actually occurs. This method is called prognostics, and it is concerned with finding the Remaining Useful Life (RUL), or time until failure, of a component. Ideally, the component should be taken off-line and repaired or replaced right before it would have experienced failure, maximizing the useful lifetime of the component and minimizing the negative effects of unplanned failure. Maintenance resources are wasted if maintenance is performed more often than needed, but if the component experiences an unplanned failure, expensive outages and replacement costs can occur, especially for critical or non-redundant components. With an accurate estimation of RUL, an optimized maintenance plan can be established that reduces maintenance costs and increases the availability of the system overall.

Currently, diagnostic information is used primarily in prognostics to select the type of prognostic model which is most applicable for the situation. However, the central idea of this research is that some diagnostic information can also be used directly in prognostic models, especially when the diagnostic information is non-critical in nature (i.e. the diagnostic code does not require immediate replacement of the component). For instance, the number of times a component has been serviced or its parts have been replaced is likely relevant to the RUL of the entire component, and this information should be included in a prognostic model. Many systems within a component are dependent on one another, and degradation in one component may be correlated with degradation in another related component. Furthermore, increased servicing of a component over the average can also potentially indicate that the component is in a higher-stress environment with factors that may not be quantifiable by other methods. By incorporating this diagnostic information in the form of fault codes, a more accurate estimation of the degradation of the component and its RUL can be obtained.

One of the greatest strengths of incorporating fault codes and diagnostic information into a prognostics model is that this type of information is already being collected by most companies that perform diagnostic monitoring of equipment. Rather than necessitating new sensors to be installed or requiring a change in the normal operation of a component, the method described in this dissertation will be applicable to information that can already be obtained using current practices alone. Section 1.3 gives an overview of fault codes which are already being collected throughout a wide variety of industries for which the methodology described in this dissertation may apply.

1.1 Problem Statement

Fault code data is an often underutilized source of information about the health of a component. A fault code, also known as an error code, refers in general to an indicator at a particular time in which the component is not operating normally. For instance, if a monitored signal shows aberrant behavior during operation, a fault code could be registered at that time stamp for further analysis. For commercial applications, fault codes are typically used as diagnostic indicators. When one or several fault codes are registered, a maintenance action might be scheduled to repair the component or investigate if the component is operating properly. In general, a given system may have hundreds or thousands of potential fault codes that represent many potential types of aberrant behavior.

The central problem addressed in this dissertation is the need for more accurate predictions of RUL for high-value and safety-critical components, and the proposed solution is the incorporation of fault code data to estimate the health of the component. Currently, fault code data are being used primarily for diagnostic and fault detection purposes by determining whether a fault currently exists in the component. However, as described in this dissertation, fault codes can often be indicative of a progressive fault within a component. In other words, as the total number of fault codes increases over time, it is likely that they correspond to decreasing health of the component. As such, the information contained within fault codes might be relevant to a constructed prognostic parameter of the investigated component.

Two primary types of fault code data are considered in this dissertation, although the methods described herein may be applicable to other types of data as well. The first type of fault code data is obtained from a pre-operation test of the component, which may be performed on a regular basis to ensure that the component is operating correctly. This type of data is typically collected from components that are not used continuously but rather have intermittent use, and the analysis will typically be off-line rather than online monitoring. These pre-operation tests might have pass-fail criteria that determine whether the component may be placed back into service, and the pass-fail criteria might depend on a number of fault codes or measurements from monitored signals. In this case, the fault codes or alarms from monitored signals can be used for the fault code analysis described in this dissertation.

The second type of fault code may be obtained from online monitoring of components. When signals such as vibration or current are continually being monitored for a particular component, aberrant behavior such as a sudden spike in the vibration can be recorded as faults. This particular type of data recording might be beneficial when the capacity for data storage is low or the continuous measurement of the signal is not necessary. For this type of analysis, the fault code analysis described in this dissertation can also be performed in an online fashion, with continuous updating of component health.

Information obtained from fault codes may be able to solve the need for increased reliability of plant components for many situations. Monitored signals that are useful for prognostics may be hard to obtain, and fault codes received from maintenance records may be a more prevalent source of information about the condition of a component. When both sources of information are available, fault code information may also be used to increase the accuracy of prognostic models that were previously constructed with monitored signals alone.

1.2 Contributions of this Dissertation

The objective of this dissertation is to provide a means by which fault code data can be used to more accurately estimate the health of a component. Primarily, this objective will be accomplished by using the information contained in fault codes to supplement an existing prognostic parameter. The integration of these two sources of information may provide a more accurate assessment of the health of a component. Two data sets will be analyzed in this dissertation, an actuator system with both monitored signals related to degradation and a large fault code database, and an accelerated degradation experiment involving 5-horsepower (HP) motors in which monitored signals are transformed into fault codes. Topic models will be investigated for the large fault code database in the actuator system as both a dimension-reduction technique and a fault-isolation method.

The expected contributions are as follows:

1. Development of a set of techniques that transforms binary fault code data into a prognostics parameter which is optimized to find ideal weightings for each fault code.
2. Incorporation of fault code information into already-existing prognostic parameters. Gradient descent is used to optimize the weighting for each fault code.
3. Application of topic models as a dimension-reduction and fault mode isolation technique for large fault code databases.
4. Development of MATLAB functions that can be used to apply these techniques to real-world or simulated data.

1.3 Examples of Fault Code Data

Fault code data is collected in a wide variety of fields for many different applications. The methodology described in this dissertation is applicable to reliability analysis of any system that collects fault code data either as a pre-operation self-test or as alarm codes during normal operation. Several diverse fields in which this type of analysis might also be beneficial are: computing and electronics, telecommunications, aviation, instrumentation and control, and national defense. While the fault code data for these fields of study might not necessarily follow the format of the data used in this dissertation, it could potentially be modified or incorporated into an existing prognostic parameter using similar methods to those described herein.

The telecommunications industry employs Very Large Scale Integrated (VLSI) circuits which are chains of many individual Integrated Circuits (IC). Due to the fact that many ICs are LRU, and correct diagnosis of faulty ICs must be accomplished in a large network of VLSI circuits, most ICs are equipped with a localized Built-In Self-Test (BIST). The BIST for telecommunication ICs are designed to diagnose common faults based on direct hardware testing. The most common fault diagnosed by the BIST is a stuck-at fault, where a logic gate is improperly fixed at either a zero or a one; however, BISTs are designed to properly diagnose all ordinary faults within the IC (Mukherjee 1999). While the most common application for BISTs in telecommunications is for diagnostic purposes, it is possible that fault code data could be useful for prognostics in predicting the RUL of ICs as well, using the methods described in this dissertation.

BISTs are commonly used in the computing industry for diagnosing faults in electrical components and verifying the integrity of Random-Access Memory (RAM). In particular, BISTs for RAM result in fault codes for defective addresses of memory which could potentially be employed for the kind of analysis described in this dissertation (Voyiatzis 2005, Bardell 1988, Franklin 1989). Monitoring signals from input/output gate pins in field programmable gate arrays results in diagnostic fault codes that could possibly be applicable to this research (Hofmeister 2010).

Typically, in avionics, an aircraft must be cleared to fly by a pre-flight diagnostic test, which examines relevant condition-based information about the aircraft to determine if a fault exists. Lockheed Martin employs a Common Organizational Level Tester (COLT), which is used to diagnose problems with the weapon systems on their helicopters. The COLT must be run before the helicopter is cleared to fly, and testing results in a number of fault codes for each helicopter (Glickman 2003). The systems which are typically tested by a COLT include control, power, communication, and signal loading. A Verification Test Suite was also developed to ensure that the signals collected by the COLT are accurate and the core systems are not in a faulty state (Glickman 2004). The signals collected by the COLT may be used to generate fault codes and thus may be analyzed using the methods described in this dissertation.

The Apache AH64 helicopter is monitored using a distributed BIST system which incorporates 334 different signals. The output of the BIST is a large number of pass-fail fault codes for each installed component which can be interpreted by an operator (Steinmetz 2002). A "Hardware-In-The-Loop" simulation has been used to provide fault diagnosis resulting in fault codes for helicopter navigational systems and actuators (Yoo 2008). This "Hardware-in-The-Loop" method is an alternate way to generate fault codes with three potential values: normal, warning, and fault. Several value ranges are generated for each of the three potential values of a fault, with previous observations of normal and faulted operation being used to generate the value ranges. However, Yoo (2008) did not use generated fault codes to create a prognostic parameter, but instead recommended maintenance immediately when a warning or fault was generated.

There are numerous other fields in which fault code analysis might be applicable. Fault codes can be obtained from built-in tests of oceanographic sensors using measurements from alternating current (AC) in seawater (Vessey 1994). The U.S. Navy uses diagnostic tests which result in similar fault codes to those described in this dissertation for conventional missiles developed during its Joint Standoff Weapon program (Mitchell 1995) and for submarine-launched missiles (Ma 2010). BISTs for electric vehicles may result in fault codes for the inverter, relay, and motor systems (Xiao 1999).

Although an extensive literature survey was performed, no research was found that employs fault codes to predict the RUL of a component. All condition-assessment of components was purely on a pass-fail basis: either the component requires maintenance or it does not require maintenance. The methodology described in this dissertation is a novel approach that does not appear in any research performed for the literature survey.

1.4 Applications to Nuclear Engineering

Condition-based maintenance is usually considered to be an ideal maintenance strategy for Nuclear Power Plants (NPP). Since many parts of an NPP experience large amounts of stress due to high temperature and many locations in an NPP are difficult for maintenance personnel to reach because of radiation considerations, it is important that components installed in an NPP have high reliability and that failures can be anticipated ahead of time. Furthermore, NPPs have strict safety guidelines, and unplanned component failure may sometimes necessitate a temporary shutdown of the plant, which results in a loss of millions of dollars of profit. Since the kind of fault codes analyzed in this dissertation do not generally refer to a diagnosed fault that would require replacement of the component, NPPs are ideal in that components are often in difficult-to-reach locations and replacement is usually unnecessary due to redundancy, so relevant maintenance records might be collected without replacing the component.

Prognostics and condition-based maintenance are vitally important in ensuring the safe operation of NPPs beyond their current life extensions of 60 years; online monitoring and RUL prediction of components such as valves and rotating machinery will increase safety and reliability of NPPs as they age (Bond 2011, Ramuhalli 2012). NPPs have incorporated condition-monitoring into their maintenance strategy for many different components. The vibration of vane axial fans has been monitored using accelerometers, and the imminent failure of several vane axial fans was predicted during a refueling shutdown of the Limerick Nuclear Power Station (Smith 2007). Core shrouds in reactor pressure vessels sometimes experience cracking as they age, and measurement of the degradation is often difficult due to their location and the ambient radioactive environment. However, nondestructive measurement techniques have been

developed to measure the cracking (Doctor 1997) and the propagation of cracks has a well-established solution in prognostic literature (Lu 1993, Sun 2012 “Health Monitoring for Propagating”).

Maintenance records of components within NPPs could potentially be used for the type of analysis described in this dissertation. A probabilistic risk database for common components in NPPs has been established by Rajasthan Atomic Power Station in India (Rastogi 2010), and the practice is widespread within the industry. The type of data collected by the Rajasthan Atomic Power Station includes information about failures while running, spurious starts, and failure to open/close (for valves), all of which could potentially be used as fault codes in the analysis described in this dissertation. Transient or intermittent faults, such as stuck control valves, pump jams, and actuator faults (Shah 2011) could be construed as fault codes as well.

1.5 Organization of the Dissertation

Chapter 2 of this dissertation gives an overview of relevant methods in prognostics, reliability, and signal analysis. Sequential Probability Ratio Tests are reviewed, as they may be used to create alarm codes in monitored signals. Chapter 3 of this document details the methodology of the primary Original Contributions of this dissertation. In this chapter, simulated data are used to illustrate the methods described therein. Chapter 4 contains validation of the methods established in the Chapter 3, illustrated with two data sets that contain real-world fault or alarm codes. Chapter 5 provides the conclusions that were reached from the application of the methodology to real-world data and recommendations for future work.

CHAPTER 2
LITERATURE REVIEW

2.1 Overview of Prognostics

Minimizing operating and maintenance costs in power generation is often accomplished by implementing an efficient maintenance strategy. Ideally, components should be repaired right before they would have failed. Funding resources are wasted if maintenance is performed more often than is necessary, and expensive outages and replacement costs can arise if maintenance is not performed before the component fails, especially for safety-critical or non-redundant components. For the development of an optimal repair schedule and ensuring that no unnecessary maintenance is performed, an accurate estimation of the component lifetime must be determined. This prediction of the Remaining Useful Life (RUL) of a component is called prognostics.

Figure 2-1 shows a diagram of the three primary types of prognostics and the kinds of information they incorporate to calculate RUL. Type I prognostics uses failure times from multiple historical cases. Type II prognostics combines failure time data and environmental stressors from multiple historical cases. Type III prognostics collects monitored signals from individual components that relate to the health of the component.

The ideal type of prognostics to choose in a particular situation depends strongly on the component and system to which it is applied. Type I and Type II prognostics are generally much cheaper than Type III prognostics, because Type III involves placing monitoring equipment on each individual component. However, in most cases, Type III models will achieve more accurate results than Type I and Type II models. A cost-benefit analysis can be undertaken which estimates the reduction of maintenance costs by using Type III prognostics over Type I or Type II prognostics, minus the cost of monitoring equipment for the fleet of components. If the resulting value is positive, Type III prognostics should be selected rather than Type I or Type II. In many situations, Type I prognostic models are often used as a "proof-of-concept" that prognostics reduces maintenance costs, and a transition to more accurate Type III models can be accomplished over time. (Hines 2008, Coble 2008)

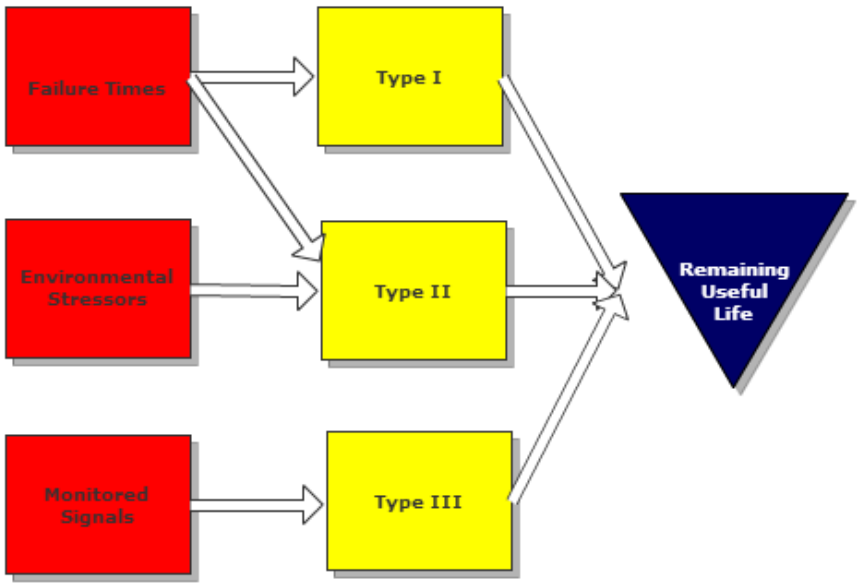

Figure 2-1. Summary of Prognostic Types.

2.1.1 Type I Prognostics

The simplest type of prognostics, Type I, uses failure data from a number of components which have experienced the same failure mechanism. The distribution of the failure time of these components can be used to find an expected component lifetime, typically by finding the mean of the distribution. Prediction of RUL is accomplished for a currently unfailed component by determining the remaining amount of time until the expected component lifetime, which was generated using the failure time distribution. Type I prognostics is usually inexpensive to perform, and the necessary failure data for Type I prognostics is often already available for many components.

Figure 2-2 shows a histogram of sample failure times which might be used for Type I prognostics. For comparison, a normal distribution curve, denoted by the red line, has been plotted as well. In this example, the component being monitored is non-redundant and a low

probability of failure is preferred, so a predicted 5% chance of failure has been selected. This 5% value is shown as a green line on the figure. In other words, the Type I prognostics model will typically underpredict the RUL. This occurs so that the component does not usually fail before the model's prediction, which would result in unscheduled maintenance downtime and increased operating costs.

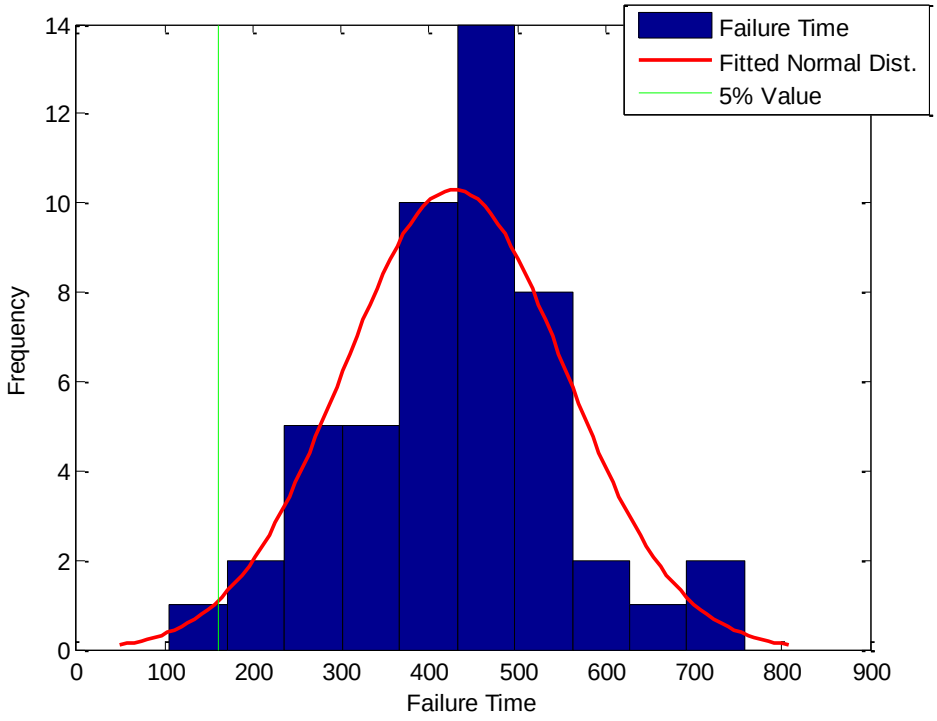

Figure 2-2. Sample Type I Failure Time Histogram.

Table 2-1 gives results from a sample Type I analysis using a model constructed from the data in Figure 2-2. The RUL of five unfailed components at various points in their lifetime are predicted, and the error of the prediction is shown. It can be seen in the table that due to the 5% value selected, the RUL is not overpredicted for any of the components. For components that are not safety-critical or are redundant, the arithmetic mean of the failure distribution might be selected rather than the 5% value.

Table 2-1. Sample Type I Prognostics Results.

Component	Current Time	Predicted RUL	Actual RUL	Error
1	34	395.03	440	44.97
2	53	376.07	435	58.93
3	66	363.13	420	56.87
4	29	400.02	402	1.98
5	51	378.06	510	131.94
			Average	58.938

2.1.2 Type II Prognostics

Type II prognostics takes into account the operating condition of the component. Similarly to Type I prognostics, a failure distribution is established using historical information about component failure, but the distributions are kept separate for different operating conditions. For components under high stress, the total lifetime will typically be shorter than components under low stress. By separating a component into groups under different amounts of stress, a more accurate RUL can usually be assessed for an unfailed component. Type II analysis is most useful when the operating conditions are well-defined and highly separable; for instance, a temperature difference of 40 degrees C between operating conditions will be much more applicable to Type II analysis than a temperature difference of 5 degrees C between operating conditions.

Figure 2-3 shows a sample failure time histogram that might be applicable for Type II analysis. It can be seen in the figure that the histogram appears to be bimodal; two normal distributions could be fitted to each side of the data. This kind of situation likely occurs when there are two operating conditions which differently affect the typical RUL of the component, or potentially when the component has two failure mechanisms. If the operating condition of each failed component is known, they can be separated based on operating condition and Type I analysis can be performed for each condition.

Table 2-2 gives the results for both Type I prognostic analysis performed on five unfailed components without regard to operating condition, and Type II prognostic analysis performed on the same unfailed components which also takes into account operating condition. The current operating time of the component is given in the second column, the operating condition is shown in the third column, and the error from both Type I prognostic analysis and Type II analysis is provided in the fourth and fifth columns. It can be seen that the average error from Type II prognostic analysis is much less than the average error from Type I analysis. For this particular case, separating the components by operating condition increases the accuracy of RUL estimation.

Another category of Type II model can be generated when the operating condition of the component changes over time. This type of analysis might be applicable when the component experiences both high stress and low stress conditions during its operation, perhaps as a result of demand on the component. One algorithm for this situation is called Monte Carlo Markov Chain (MCMC), as it uses the Markov assumption of memoryless operating conditions. In other words, the next operating condition of the component only depends on the current operating condition, and not on any previous operating conditions.

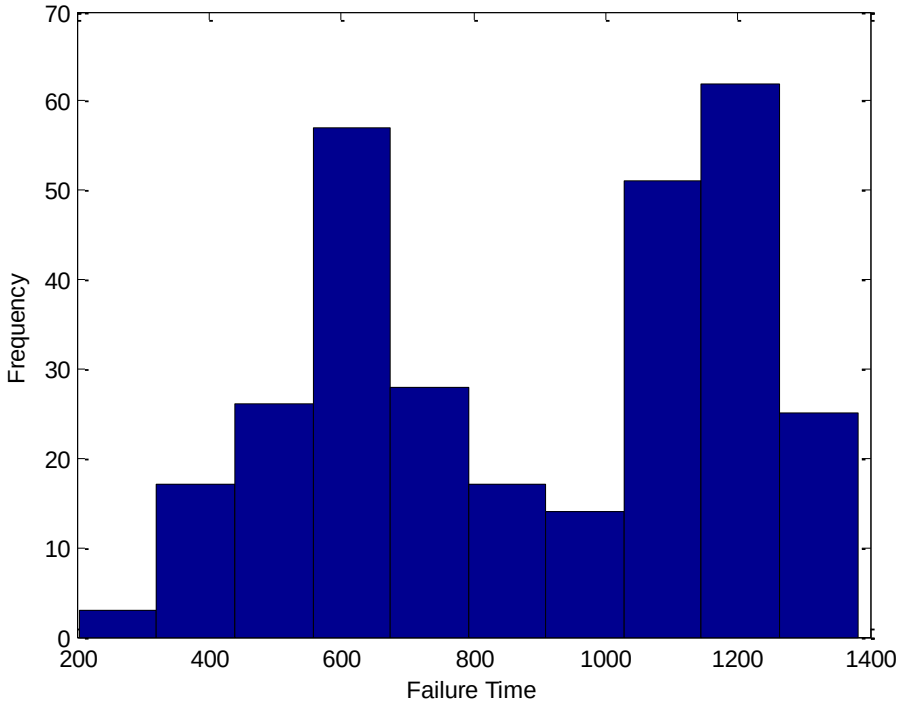

Figure 2-3. Sample Type II Failure Time Histogram.

Table 2-2. Sample Type II Prognostics Results.

Component	Current Time	Operating Condition	Type I Error	Type II Error
1	57	1	234.89	66.78
2	86	2	105.96	101.8
3	95	1	97.01	36.24
4	66	1	125.9	42.93
5	61	2	90.9	10.62
		Average Error	130.932	51.674

For this type of model, each operating condition is assumed to impart degradation to the component at a particular rate, depending on whether it is a high-stress or low-stress condition. The total amount of degradation should be the sum of the each environmental stressor constant times the amount of time that component has spent in that environmental condition. In the form of an equation, this is given by:

$$D(t_n) = \sum_{i=1}^{n} C_{ki} * \Delta t_i$$

Equation 2-1. MCMC Cumulative Degradation.

where $D(t_n)$ is the amount of degradation at time *n*, C_{ki} is the environmental stressor constant for the operating condition at time *i*, and Δt_i is the amount of time spent in the operating condition.

The RUL of an unfailed component can be estimated in MCMC Type II analysis by assembling a transition matrix which indicates the probability that the component will enter a new operating condition based on the current operating condition. An example of such a transition matrix is shown in Figure 2-4 below.

$$\begin{bmatrix} 0.2 & 0.3 & 0.5 \\ 0.8 & 0 & 0.2 \\ 0.1 & 0.1 & 0.8 \end{bmatrix}$$

Figure 2-4. Sample Transition Matrix.

The proper way to read a transition matrix is that the row of the matrix refers to the current operating condition of the component, and the columns refer to the probability of transitioning to a particular state. For example, in the transition matrix shown above, the element [1,2] would show the probability of a component which is currently in operating condition 1 transitioning to operating condition 2. The rows of the matrix should always sum to 1, representing a total of 100% possibility to transition to any state. For the above transition matrix,

the [2,2] value is 0, which means that a component in operating condition 2 has a 0 probability of staying in that operating condition and must transition to either operating condition 1 or 3. However, the [1,1] and [3,3] values are non-zero, which indicates that it is possible that the component may stay in its current state if it is in operating condition 1 or 3.

For an unfailed component, Monte Carlo methods can be used to build a distribution of estimated cumulative degradation for the component. Each Monte Carlo run begins either with the same initial operating condition, or as a random selection of operating condition based on probability. For each subsequent time step, a new operating condition is selected based on the probabilities found from the transition matrix. To estimate the failure time of the component, a failure threshold is selected by determining the typical amount of degradation at which previous components failed. After the cumulative degradation in the Monte Carlo simulation reaches this failure threshold, the component is considered to have failed. Since Monte Carlo methods will typically involve hundreds of thousands of runs, the estimated failure time of the component according to this method will result in a distribution rather than a point-estimate. Usually, the mean of the distribution is selected as the point-estimate failure time of the component, but valuable information about the certainty of the estimate can also be obtained from the standard deviation of the distribution.

A third algorithm of Type II prognostic model is a Proportional Hazards Model (PHM), which was developed by Cox to describe the effects of explanatory covariates on lifetime (Cox 1972). An example of a covariate might be operating temperature. PHMs can be constructed when the covariate, which in this case is operating condition, can be multiplicatively regressed to the hazard rate. This is called the proportional hazards assumption. In other words, when an operating condition can be assigned a numerical value, called a covariate, it must be related multiplicatively to the probability of failure. When the proportional hazard assumption holds, Equation 2-2 can be used to find hazard rates for conditions not previously tested.

$$\lambda(t \mid X) = \lambda_0(t)\exp(B' X)$$ **Equation 2-2. Cox PHM Equation (Cox 1972).**

In this equation, $\lambda(t|X)$ is the hazard rate at time *t* based on covariate value *X*, $\lambda_0(t)$ refers to the baseline hazard rate at time *t*, B is the regression coefficient used to relate hazard rate to covariate values, and X is the value of the covariate.

PHMs are commonly used during accelerated degradation testing, when the component is placed at a higher stress level than normal operation to reduce the testing time involved. The baseline covariate, or normal operating condition, can be found using Equation 2-2. At least seven different generalized proportional hazards models have been developed for accelerated degradation testing when the proportional hazards assumption does not hold (Huang 2009). These models may take into account situations where the hazard rate based on covariate values varies over time. PHMs have successfully been used for RUL estimation of electric mine power cables (Kumar 1996), carbon film resistors (Chen 2010), conveyor pulleys at a mineral processing plant (Ho 2011), and distribution transformers (Prasad 2003).

2.1.3 Type III Prognostics

For Type III prognostics, the degradation of each individual component is considered, not merely the average response to its operating conditions. By associating degradation with a number of measureable signals, such as vibration or current, the change in degradation of the component over time can be found. When performing accelerated degradation testing, the values of these signals at failure can be found, and a failure threshold can be established, which indicates the amount of degradation at which a component will typically fail. This failure threshold can be hard, which means that it will fail at a constant degradation value, or it can be soft, which means that it will fail in a distribution of degradation values. Usually, a Type III prognostics model is the most expensive to implement of the three prognostics types, since it involves monitoring of individual components, but it will typically also give the most accurate results.

When predicting the RUL, a function is fitted to past condition data on an unfailed component, which is then extrapolated until it reaches the failure threshold, giving the RUL of the component. This type of model is called a General Path Model (GPM), and it was developed by Lu and Meeker (Lu 1993, Meeker 1998). A GPM is trained on sufficient amounts of

monitored signal data from failed components that relates in some way to degradation. For example, monitored vibration data might be used to assess the health of a bearing, since bearings often have increases in vibration at characteristic frequencies when they are near failure (Randall 2011). If the monitored signal is not related to degradation of the component, a GPM cannot be used, and Type I or Type II prognostics might be a more appropriate choice.

If the component has several different failure modes that it can experience throughout its lifetime, there must be ways of distinguishing them, usually through forensic analysis of failed components. Ideally, data from different failure modes should be modeled separately, because degradation in the monitored variables might not manifest itself in the same way, depending on the failure mode involved. For instance, bearings undergoing cracking might behave completely differently than bearings that do not have enough lubrication. Diagnosis of the fault can depend on selection of monitored variables. For instance, a particular variable such as vibration might be used to diagnose bearing cracking, while another variable such as bearing temperature might be used to diagnose a lack of lubrication. After large amounts of training data with monitored signals have been collected, the GPM is trained by fitting the monitored signals, or degradation parameters, to a particular function, such as a linear or quadratic fit.

Figure 2-5 shows a monitored signal from ten simulated components plotted over time, with the last data point being the failure point. It can be seen that the signal is likely related to degradation, as it increases fairly steadily over time, and the component is accruing damage over time. Each of the ten failed components has been fitted to a linear function, and the slope has been determined. The mean slope and y-intercept of the ten failed components has been calculated, and the red line on the figure refers to this typical, or average, signal path. The failure threshold, or the mean signal value at which a component fails, is plotted as a green line on the figure.

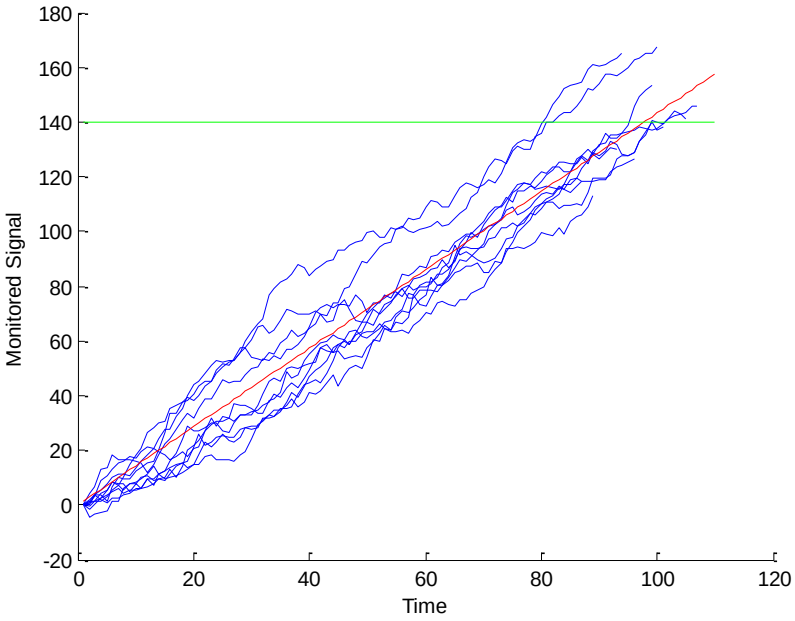

Figure 2-5. Sample Degradation Paths.

Prediction of RUL from an unfailed component is accomplished by extrapolating the typical fit from the training data to the selected failure threshold. Figure 2-6 gives an example of how this might be accomplished, using the same data as from Figure 2-5. The blue data points are historical data collected from the unfailed component, which is then fitted to the same type of distribution as the training data, shown with the red line. The black line, or the extrapolated path, is the same as the typical or average path calculated from the training data, and it is concatenated on the last data point of the red line. The failure threshold, shown by the green line, is the same value as it was in Figure 2-5, calculated by the mean value of the monitored signal at failure for the historical components. The point at which the extrapolated path, or the black line, reaches the failure threshold, or the green line, will give the estimated failure time of the component, shown as a red point on the figure. However, there is some error in this estimation, since the actual failure time of the component, given as a cyan point, is several time steps away from the estimated failure time.

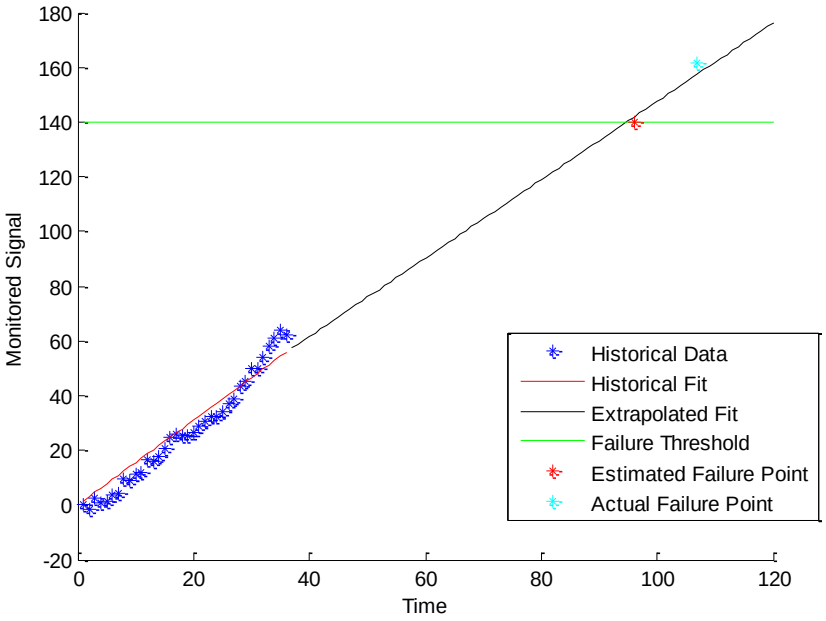

Figure 2-6. Sample Extrapolated Path.

GPMs have been successfully used to predict the RUL of many different types of systems. One of the first applications of extrapolating degradation paths from monitored signals was the prediction of the residual life of induction motors using current and temperature signals (Upadhyaya 1994). GPMs have also been useful for RUL prediction of drilling tools for the petroleum industry (Garvey 2010), bearings (Zhang 2005, Lybeck 2007, Boskoski 2012), and avionics (Kirkland 2004, Zhenhua 2011, Xu 2011).

2.2 Prognostic Parameter Construction

A prognostic parameter, often created by a linear combination of signals relating to degradation, is a measurement of the health of an individual component, which may be trended in a GPM to determine RUL. In the previous section, GPMs were constructed using a single monitored signal. However, in many cases, there are several monitored signals which might be relevant to component health. For example, when using a GPM to predict a bearing fault in a motor, the magnitude of characteristic frequencies in the vibration spectrum and the bearing noise might both be related to the degradation of the bearing. Typically, one signal will be more related to degradation than another signal; in this type of situation, the signals might be weighted in a linear combination based on how correlated they are with degradation of the component.

Especially for components where many different kinds of signals are being monitored, it might be difficult to determine which signals are most related to degradation. An ideal weighting of the monitored signals should be established to best relate the prognostic parameter to the health of the component. One way to accomplish this goal is to maximize performance metrics which are related to an ideal prognostic parameter. Three prognostics performance metrics have been developed for this purpose: monotonicity, trendability, and prognosability (Coble 2009). Equations and explanations for these three performance metrics are given below, and more information about the performance metrics can be found in (Coble 2009) or (Coble 2010a).

Monotonicity, shown in Equation 2-3, refers to the tendency of the prognostic parameter to regularly trend upward or downwards. Essentially, the monotonicity metric counts the number of times the slope is negative between two points, subtracted from the number of times the slope is positive between two points. If the data is fairly random and each data point is equally likely to be above or below the previous data point, the monotonicity metric will be close to 0. However, if the data is consistently trending either upwards or downwards, the monotonicity metric will likely achieve a value close to 1. High values for the monotonicity metric are favorable because it is assumed that components do not self-heal in most situations.

$$Monotonicity = \left(\frac{\#of\ \frac{d}{dx} > 0}{n-1} - \frac{\#of\ \frac{d}{dx} < 0}{n-1}\right)$$ **Equation 2-3. Monotonicity. (Coble 2009)**

The equation for trendability is given in Equation 2-4. First, the training data is resampled as a percent of life. This resampling is performed so that the length of each training vector will be equal; otherwise, the correlation coefficient could not be found. Next, the correlation coefficient is used to find the linear correlations in the data. In this equations, the variable *i* refers to the row of the correlation coefficient matrix, while *j* refers to the column of the correlation coefficient matrix. The minimum absolute value of the correlation coefficient for each index is considered to be the value of the trendability metric for the prognostic parameter. If the prognostic parameter tends to follow the same functional fit for all or most examples of the training components, the trendability metric will have a high value near 1. However, if the functional fit differs between training components, the trendability will take a value near 0.

$$Trendability = \min_{i=1:n,\, j=i:n}(|\, corrcoef_{i,j}\,|)$$ **Equation 2-4. Trendability. (Coble 2010a)**

Prognosability is the measurement of how distinguishable the distribution of values at failure will be from the initial distribution of the parameter. The equation for this performance metric is given below in Equation 2-5. If the initial values of the parameter are very spread out and overlap with the distribution of degradation at failure, an accurate RUL cannot be determined. The smaller the standard deviation of the values at failure of the prognostic parameter, the more easily extrapolation to a critical threshold value can be accomplished. Ideally, a highly prognosable parameter should minimize the standard deviation of the distribution of values at failure of the parameter while also maximizing the range between the initial distribution and the faulted distribution.

$$\Pr ognosability = \exp\left(\frac{-std(\deg\_at\_failure)}{mean(\deg\_at\_failure - initial\_\deg)}\right)$$ **Equation 2-5. Prognosability.**

(Coble 2009)

Figure 2-7 shows a graphical representation using simulated data of how the prognosability metric works. The green points are the values of the prognostic parameter at the beginning of life, the red points are the values of the prognostic parameter at failure, the blue lines are the prognostic parameter values over time, and the black lines are the distributions of values at baseline and failure. To achieve a high prognosability metric, the distribution of values at failure should have a low standard deviation, which appears to occur in this figure. For each component, the difference between the mean parameter value at the baseline and the value at failure should be large, which also occurs in this figure. The prognostic parameter shown in this figure would likely have a high prognosability due to these two characteristics.

Each of the three metrics mentioned above will take values in the interval between [0,1]. Practically, good prognostic parameters will have values greater than 0.7, while poor prognostic parameters will have values less than 0.3. The most important of the three metrics is likely prognosability, because without a significant difference between the initial condition and the faulted condition, prognostics will not be able to determine an accurate RUL. The Process and Equipment Prognostics (PEP) toolbox, developed by Jamie Coble and others at the University of Tennessee (Coble 2010b), will be used in this dissertation to calculate the values of these performance metrics. Furthermore, an ideal weighting function was developed in PEP which employs genetic algorithms as an optimization process. This weighting function will be employed in this dissertation to construct the prognostic parameters built from monitored signals.

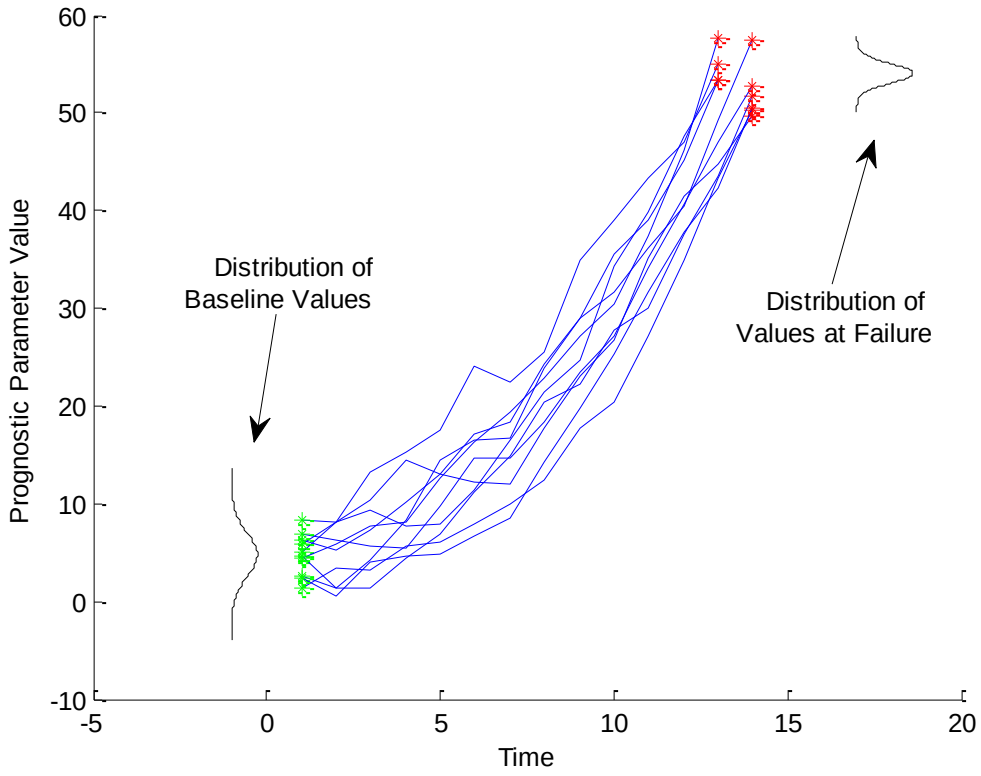

Figure 2-7. Demonstration of Prognosability.

Genetic algorithms work in a manner similar to the way that evolution and natural selection work in nature. To begin with, genetic algorithms require a fitness function which describes optimal values of candidate solutions. Fitness functions are used to evaluate the performance of each solution, and candidates with higher fitness values are selected for. The fitness function used in this dissertation for prognostic parameter construction is shown in Equation 2-6. The fitness function is the sum of the monotonicity, trendability, and prognosability performance metrics. Each performance metric may be weighted to give stronger influence to one performance metric over another; however, in this particular case, all of the weights w_m, w_t, and w_p were chosen to be 1, which will give equal influence to all three performance metrics.

$$Fitness = w_m * monotonicity + w_t * trendability + w_p * prognosability$$ **Equation 2-6. Fitness Function. (Coble 2010a)**

An initial population is required to begin the genetic algorithm process; this initial population may be constructed randomly from possible solutions to the optimization problem, or expert knowledge about good candidates may be used for the initial population. Each candidate in the population is comprised of a number of chromosomes, or properties, that may be used in the fitness function. For this case, the chromosomes involved in the genetic algorithm are the weighting of the signals in the prognostic parameter. The values of the signals themselves are not allowed to change, but the weighting of each signal may be changed during each iteration of the genetic algorithm until an optimal solution is reached (Haupt 2004).

Next, an iterative process is used to select optimal solutions to the problem. During each generation, or iteration, the population of candidates is evaluated using the fitness function selected. The candidates with the lowest fitness are often forced to "die out," and the candidates with the highest fitness are allowed to "reproduce," in an analogy to natural selection. Reproduction modifies the chromosomes of the new candidates by recombining the chromosomes of the previous candidates with the highest fitness. This process will generally tend to produce better solutions to the problem each generation, as the solutions with poor fitness are gradually discarded. However, reproduction alone does not tend to introduce new information into the chromosomes of all the candidates, since information from the candidates with the top fitness are merely recombined, not added. To prevent stagnation, mutations are allowed at a certain percentage each generation, which randomly changes the values of selected chromosomes (Haupt 2004).

The iterative genetic algorithm process will continue until stopping criteria have been met. Since overtraining might result if the genetic algorithm is trained for too many generations, a maximum number of iterations and a target fitness value are typically used as stopping criteria. Overfitting, also called overtraining, occurs when the algorithm achieves very high fitness on training data by learning the eccentricities and noise of the training data but does not learn the underlying functional behavior of the system. Hence, when the model is applied to data on which

it has not been trained, it performs poorly. This behavior is usually avoided by setting an appropriate target fitness value and maximum number of iterations.

For more information about genetic algorithms and their applicability to optimization problems, refer to (Haupt 2004) or (Mitchell 1998). Other optimization methods, such as gradient descent and machine learning, may also be used to determine ideal weightings for the signals in a prognostic parameter.

2.3 Auto-Associative Kernel Regression

In some cases, it is not ideal to use a raw signal to construct a prognostic parameter. For instance, when some of the monitored variables are correlated with each other and normal operation of the component involves a range of values for each signal, the signal itself should not be used as an input to a prognostic parameter, because an increase in a signal might simply be an expression of normal operation if other related variables are changing as well. A useful signal indicator for predicting the health of a component is the idea of a residual, or the deviation from normality calculated by subtracting the observed signal in a faulted condition from the predicted signal values at normal operation.

One robust method of determining the residual of a signal is an Auto-Associative Kernel Regression (AAKR) model, a nonparametric technique that employs historical data describing the component's behavior during normal operation. AAKRs have been used to determine when a sensor has drifted, but they can also be used to detect degradation within a component as the residual from the AAKR model increases over time (Hines 2006). Primarily, an AAKR consists of a memory matrix that contains values from monitored variables during normal operation. An AAKR memory matrix will look similar to Equation 2-7 below. For each row, there are n monitored signals, with m total observations of the monitored signals. Each column in the memory matrix is the value of a monitored signal, and each row of the memory matrix is a record of the measured values at a particular time step. However, the rows in the memory matrix do not necessarily refer to subsequent time steps; rather, a small number of representative

observations are selected out of the total observations of training components during normal operation and are termed memory vectors or prototype vectors. The total number of observations in an AAKR memory matrix is usually about 500 for steady-state operation.

$$MemoryMatrix = \begin{bmatrix} X_{11} & X_{12} & X_{13} & \dots & X_{1n} \\ X_{21} & X_{22} & X_{23} & \dots & X_{2n} \\ \dots & \dots & \dots & \dots & \dots \\ X_{m1} & X_{m2} & X_{m3} & \dots & X_{mn} \end{bmatrix}$$

Equation 2-7. AAKR Memory Matrix.

Data from an unfailed component that is being evaluated using residuals from the AAKR model will be represented by a query vector, shown in Equation 2-8. Each time step of the data from the unfailed component will be represented by a separate query vector, which contains the same number of monitored signals as an AAKR memory vector.

$$x = \begin{bmatrix} x_1 & x_2 & \dots & x_p \end{bmatrix}$$

Equation 2-8. AAKR Query Vector.

The similarity of the query vector to the memory vectors, which will eventually be used to determine the residuals of the unfailed component, will be calculated by the following process. First, the Euclidean distance of the query vector to each row of the memory matrix is found, using Equation 2-9. Depending on the distance of the query vector to the memory vector, each row will be given a similarity weighting. This weighting is calculated using a Gaussian similarity kernel, with a user-specified bandwidth. Using this process, an AAKR model may be thought of as an error-correction model. Equation 2-10 shows the process by which an AAKR model will make predictions, which is a weighted average of the observations in the memory matrix.

$$d_E = \sqrt{(X_{i1} - x_1)^2 + (X_{i2} - x_2)^2 + ... + (X_{ip} - x_p)^2}$$ **Equation 2-9. Euclidean Distance.**

$$x_{pred} = \frac{\sum_{i=1}^{p} w_i X_i}{\sum_{i=1}^{p} w_i}$$ **Equation 2-10. AAKR Prediction.**

Figure 2-8 shows sample results from signal #1 of an AAKR model constructed using 9 monitored signals. The original data are plotted using the blue points, and the AAKR model predictions are shown using the red lines. It can be seen that generally the AAKR predictions follow the blue data points exactly; this is an indication that the sensor or component is working correctly. As the AAKR predictions begin to drift away from the actual data points, it is more likely that sensor or component is experiencing degradation or a fault. Figure 2-9 shows a plot of the residuals from Figure 2-8. While there are several large spikes in the residuals, they do not appear to be increasing over time. If the sensor shown in the figure was drifting, or the component was experiencing a fault, it is likely that the residuals would either experience an abrupt increase and remain at that higher value, which would be a potential case for diagnostic analysis, or the residuals would gradually increase over time, which would be more applicable to prognostic analysis.

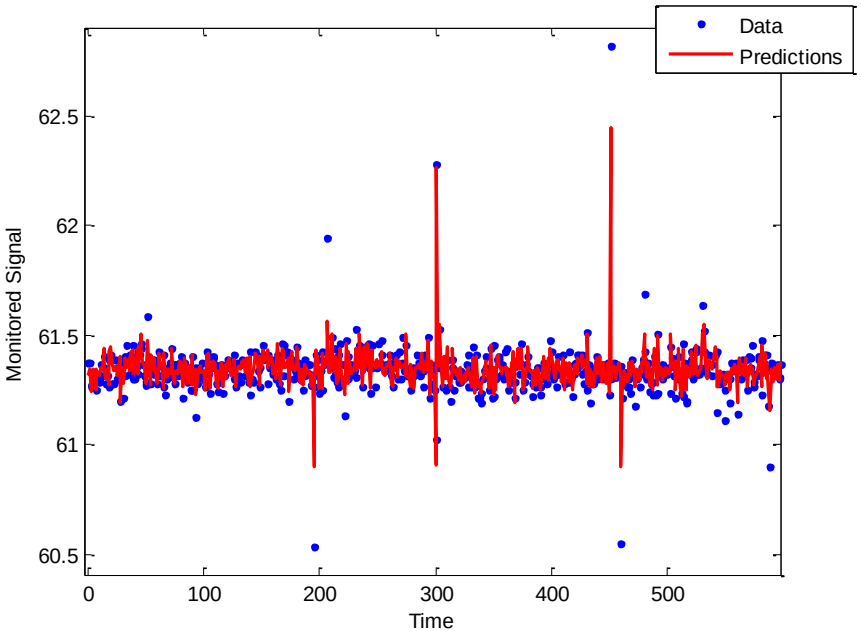

Figure 2-8. Sample AAKR Results.

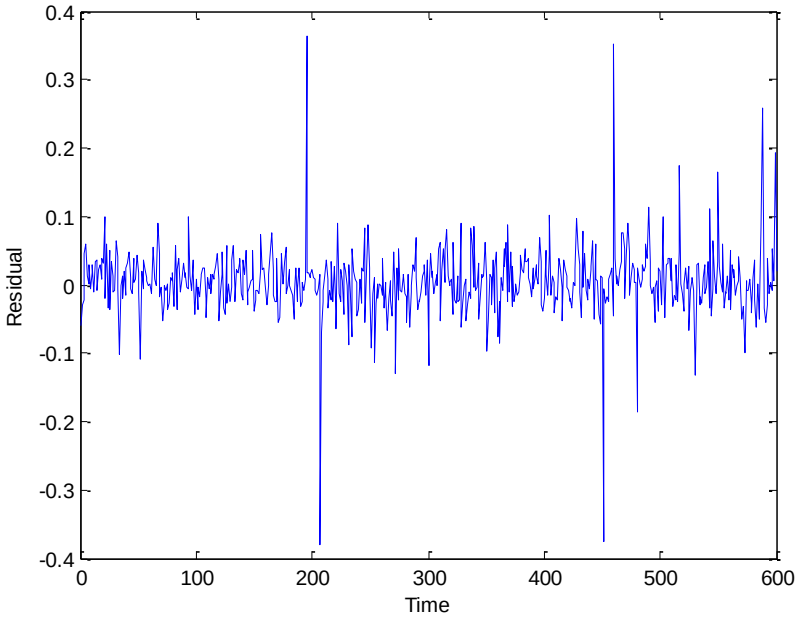

Figure 2-9. Sample AAKR Residuals.

AAKRs are commonly used to diagnose faults within sensors or components. Successful applications of condition monitoring using AAKR models include: wind turbine gearboxes (Guo 2011), induction motors (Diaz 2002), and ceramic capacitors (Sun 2012b). For more information about the mathematics behind AAKRs, refer to (Takeda 2011) or (Shawe-Taylor 2004). In Chapter 4, AAKR models will be used to generate residuals from monitored signals. These residuals will be used to create alarm codes, which will be treated as fault codes for analytical purposes. The method for creating the alarm codes is described in the next section.

2.4 Sequential Probability Ratio Test

While AAKR models are able to generate residuals from monitored signals in a component, it is sometimes difficult to determine whether an absolute increase in the residuals indicates degradation or if it is merely a result of normal data spread or noise within the data. In many cases, it is even more difficult to determine if an increase in residuals is related to an imminent fault. To some degree, an understanding of the residual level during historical faults in components is useful for these purposes, similar to the way that a failure threshold was established in section 2.1.3. However, a mathematical method called a Sequential Probability Ratio Test (SPRT) is able to determine whether a given residual likely belongs to one of two separate distributions, one describing a faulted condition or one describing normal operation.

The SPRT was developed by Wald (1945). Essentially, given a new sequential input with knowledge of historical inputs, the SPRT is designed to make one of three judgments: 1) the null hypothesis can be rejected; 2) the null hypothesis cannot be rejected; and 3) additional observations are needed (Wald 1945). For the purposes of this dissertation, the null hypothesis is that the current data point belongs to the normal operation distribution, which is assumed to be Gaussian. Rejecting the null hypothesis, then, would indicate that that the current data point is from a component which is experiencing a faulted condition. One of the primary advantages of the SPRT is that the number of data points required to make a judgment is not predetermined, as it is in most hypothesis testing algorithms.

Figure 2-12 illustrates how a decision might be made whether a residual is in a faulted condition or not. The blue distribution represents the typical range of residual values when the component is in normal operation. It can be seen that the mean value of the blue distribution is zero, which indicates that a zero residual is the most common value for a component in normal operation. The red distribution represents a shift from normal operation to a faulted condition, and the mean value from the faulted condition is a residual of about 40 units. The distributions overlap, representing unclear residual values which might belong to either distribution.

Figure 2-13 shows plotted residuals for a sample signal. It can be seen that the residuals begin increasing sharply near time step 400. This is an indication that either the sensor is experiencing a drift or the component is degrading. The SPRT can detect whether or not a fault has occurred by determining whether the signal belongs to a normal operation distribution or a faulted condition distribution. The false positive and false negative rate, also called a type I error (alpha) and type II error (beta) respectively, can be adjusted in the SPRT algorithm. Depending on the application involved, it might be more important to detect faults earlier but risk incorrect detection of faults, causing a high type I error rate, or it might be beneficial to be certain about the fault with a low type II error rate.

Sample results from the SPRT applied to the data from Figure 2-13 are shown in Figure 2-14. It can be seen that the SPRT begins detecting a fault near time step 415, which is slightly after the "true" fault began at time step 400. Earlier detection might be accomplished by adjustment of the alpha and beta parameters, but this might result in higher false positives.

For one application in this dissertation, SPRT fault detection will be used to generate alarm codes that determine when a component is not within the bounds of normal operation. These alarm codes will be treated similar to fault codes while performing the analysis described in Chapter 3. To accomplish the SPRT detection, functions within the Process and Equipment Monitoring (PEM) toolbox will be used (Garvey 2005).

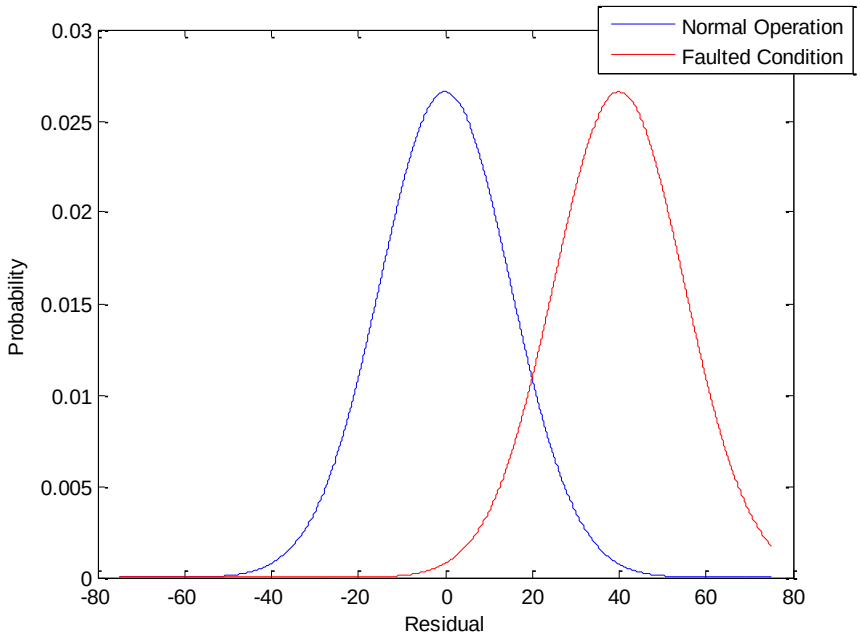

Figure 2-10. Sample SPRT Condition Distributions.

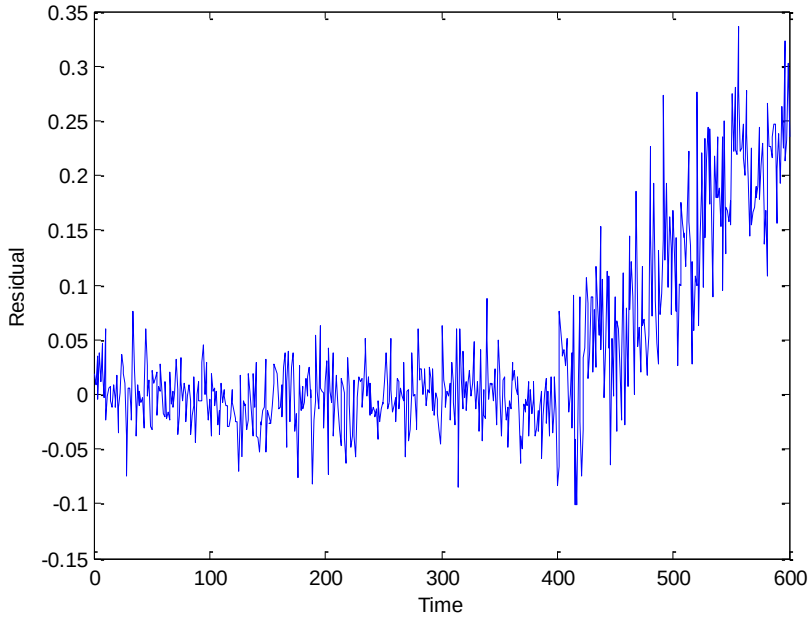

Figure 2-11. Sample Residuals for SPRT Testing.

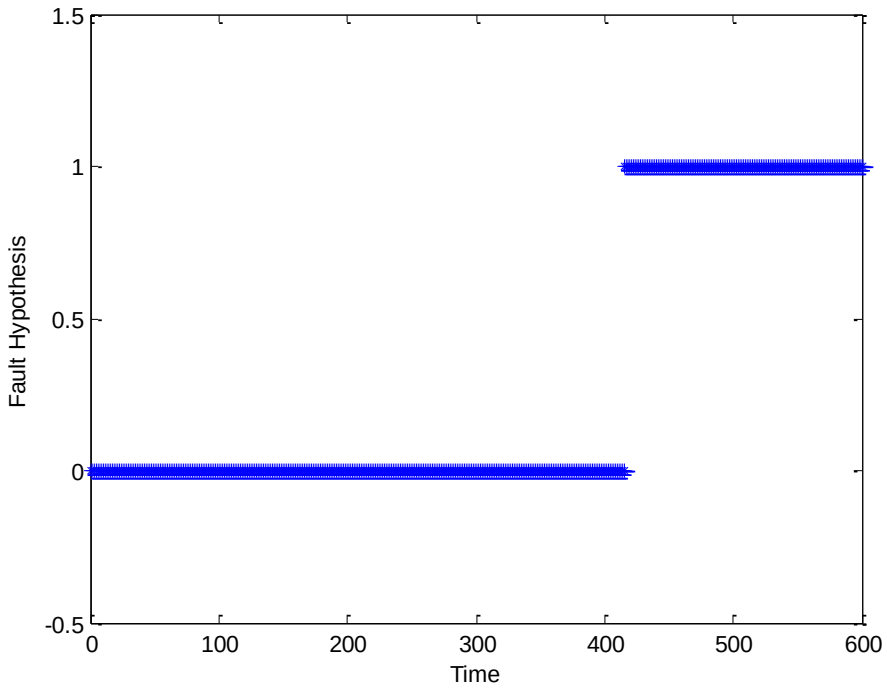

Figure 2-12. Sample SPRT Fault Detection.

2.5 Topic Models

Topic models are often used to analyze the occurrence of words or phrases in documents to make it easier to identify the commonalities between them. The number of occurrences of a particular word may be tracked throughout each document in a collection, and correlations between similar words may be used to investigate relationships between words. The probability of occurrence for groups of words can be characterized as a "topic," or a theme inherent to the grouping of words. Usually, topic models are created for easier sorting of documents or to find other documents that may be similar to a selected one.

However, there have been many successful uses of topic models in diverse applications of data analysis beyond simply documents. Computer images may be more easily categorized (Bart 2010), and internet search engines may be able to optimize their search results using a topic-based algorithm (Buntine 2004). Currently, it does not appear that topic models have been

applied to the field of reliability, but the idea of counting the number of times an event occurs to determine relationships within data may be very useful for failure analysis of components.

In this section, for ease of explanation, “words,” “documents,” and “topics” will be used to describe how topic models work. For the purposes of this dissertation, though, “words” are analogous to “fault codes,” “documents” to “time steps,” and “topics” to “fault modes” or “fault mechanisms.” A “corpus,” also known as a collection of documents, is analogous to the total number of components being analyzed. Figure 2-15 gives a graphical representation of how the terms used in topic models apply to reliability concepts.

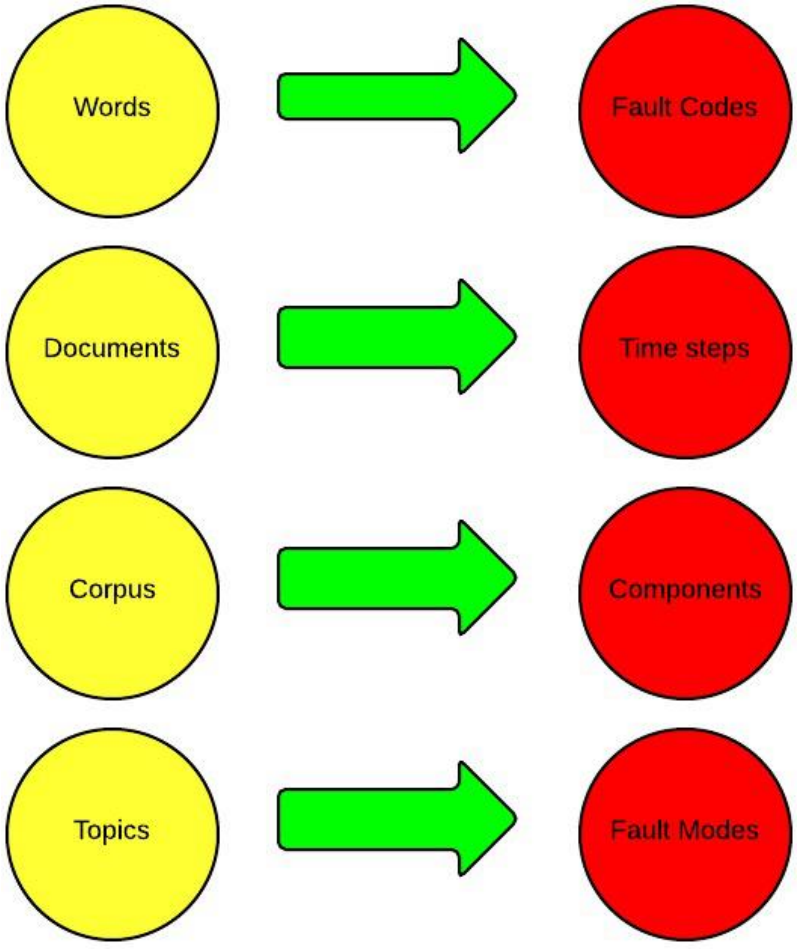

Figure 2-13. Topic Model Terms.

One of the most basic types of topic models, used in this dissertation, is a Latent Dirichlet Allocation (LDA) model (Blei 2003). Essentially, LDA is an iterative method of generating a collection of words using a topic distribution, and it is primarily a three-step process repeated over each document in the collection. The first step is to choose a topic distribution for each document. For example, if there are three topics, a distribution of 20% Topic 1, 40% Topic 2, and 40% Topic 3 might be chosen. This distribution can be also be randomly generated if no prior knowledge is assumed, or expert knowledge might be used to make an accurate initial guess. The second step, iterated for each word in the document, is to pick a topic using the distributions established in the first step. Finally, the third step is to generate a word based on the topic selected (Wang 2007).

To analyze a collection of documents in order to discover topics, Gibbs Sampling, which is an iterative Markov Chain Monte Carlo (MCMC) process, is a method that is typically used. As an initial guess, the Gibbs Sampling algorithm will assign each word in a document randomly to a topic, for each document in the collection. Next, these topics are updated iteratively to improve the accuracy of the word distributions in a topic. Two variables are calculated at each step: the proportion of words in a document that belong to a particular topic (p{topic *a* | document *b*}) and the probability that a particular word is assigned to topic *a* for all the documents in the collection (p{word *c* | topic *a*}). The algorithm assumes that all previous word assignments to topics are accurate except for this particular word, and so a new topic will be calculated for the word based on the equation:

$$topic = p(topic_a|document_b) * p(word_c|topic_a)$$ **Equation 2-11. Topic Assignment.**

This process is repeated iteratively until stagnation or until a predetermined number of iterations is reached. The end result is a selected number of topics with associated words and a distribution of topics for each document. Each document will have an associated probability that adds up to 100% across every topic. For example, a particular document with a two-topic model might have 60% association with Topic 1 and 40% association with Topic 2 (Casella 1992).

One potential application of topic models is that they can be used in the same way as Principal Component Analysis (PCA) to determine variables that are associated with one another. If "words" in topics models are replaced with "variables," grouping of variables can be accomplished much in the same way as PCA. For the purposes of this dissertation, topic models are more applicable than PCA because they may be used with a count matrix, which can be assembled from the fault codes of the component. As an example, a pre-operation test of a component with several potential fault codes can be characterized as a count matrix for each fault code that occurs. A group of these matrices from many training components may be analyzed using a topic model to determine which fault codes are associated with one another.

2.6 Summary of Literature Review

Prognostics, or the estimation of RUL, is a method which may be used to improve the reliability of components and optimize maintenance scheduling. Three main types of prognostics exist: Type I, Type II, and Type III. Type I prognostics uses a distribution of failure times from many components to calculate the expected failure time, which may be used to predict the RUL of an unfailed components. Type II prognostics, in general, takes into account the operating condition of the components as well. Components which are placed in a high-stress environment will likely fail sooner than those placed in a low-stress environment. Type III prognostics extrapolates monitored signals from an unfailed component to a failure threshold to predict their failure time.

Prognostic parameter construction is accomplished using a linear combination of monitored signals from a component. Ideally, each monitored signal will relate to the degradation of the component. Three performance metrics are generally used to determine how well a prognostic parameter will perform: monotonicity, trendability, and prognosability. A prognostic parameter should either increase or decrease regularly without changes in the direction of the trend, and the monotonicity metric will calculate how well the prognostic parameter performs in this regard. The trendability metric determines whether the constructed parameters have the same functional fit, which is optimal for a prognostic parameter. Finally, the

prognosability metric calculates whether the baseline values of the prognostic parameter are separable from the values of the prognostic parameter at failure. These three performance metrics may be used in an optimization algorithm, such as genetic algorithms, to determine the weights of each monitored signal in the linear combination that will be used to create the prognostic parameter.

An AAKR model is nonparametric and may be used to find the estimated value of a signal at a point in time, using previously known values as a reference point. When the prediction of the AAKR model differs from the observed value by a large amount, it is likely that the system is either in a new operating state or degradation is occurring in the system. An SPRT may be used to determine if the residuals between the expected and observed values are statistically significant.

Topic models are often used in document analysis, but they might also be applicable for fault code analysis. The input to a topic model is a frequency count matrix of the same form as a fault code matrix. Using topic models, the dimension of a large fault code database may be reduced by discarding topics that do not apply to the fault mode being investigated. Terms used in topic model analysis may be translated into reliability concepts, with “words” being analogous to “fault codes,” “documents” to “components,” and “topics” to “fault modes” or “fault mechanisms.”

CHAPTER 3
METHODOLOGY

Fault codes generally refer to a timestamp at which a component is not operating at normal condition. An example of when a fault code might be generated is when an actuator responds slower than a recommended guideline. While this slower speed might not be enough to warrant immediate removal and maintenance of the actuator, the fault code still gives useful information about the condition of the actuator. Over time, a large database may be created consisting of fault codes attached to timestamps for a particular component. This database might give useful information about the degradation of the component over time.

Fault codes are most commonly used for detection and diagnosis of a fault. Generation of a severe fault code will often require the component to be immediately serviced, and large numbers of less severe fault codes in a short period of time might be indicative of an imminent fault. The type of fault codes that are most useful for the kind of analysis described in this dissertation are those that do not require immediate removal of the component but rather represent a mild or moderate deviation of the component from normal behavior or an intermittent failure to meet design criteria.

3.1 Description of Simulated Data Set

A simulated data set was constructed to demonstrate the proposed analysis described in this dissertation. The simulated data were modeled after a butterfly valve, and appropriate fault codes were selected. These fault codes can be seen in Table 3-1. A total of 10 fault codes were simulated. The fault codes were assigned realistic descriptions, but the fault code descriptions did not have an effect on the behavior of the system. The simulated data is designed to behave similarly to the real-world data investigated in Chapter 4. Examples of sensors that might be used to determine the fault code are also given in the third column. For this hypothetical system, a series of fault

code were simulated for each time step, using MATLAB code found in Appendix A.1 of this dissertation.

A typical butterfly valve is shown in Figure 3-1. The central disc rotates along a vertical axis to inhibit or allow flow. An actuator on top of the valve controls the operation of the valve, opening or closing it. Butterfly valves are usually chosen because they may be constructed cheaply and require less maintenance support than other types of valves. One potential disadvantage of a butterfly valve is that it will always slightly impede flow, even when fully open, due to the width of the central disc.

Table 3-1. Simulated Valve Fault Codes.

#	**Fault Code Description**	**Sensor**
1	Valve response time slower than rated guidelines	Potentiometer
2	Stuck valve	Potentiometer
3	Valve does not fully close	Potentiometer
4	Pressure higher than rated guidelines	Pressuremeter
5	Valve does not seal	Pressuremeter
6	Temperature higher than rated guidelines	Thermocouple
7	Electrical signal does not provoke response	Current Clamp
8	Signal current too higher	Current Clamp
9	Baseline pressure drop too low	Pressuremeter
10	Flow higher than rated guidelines	Flowmeter

Figure 3-1. Butterfly Valve. (Emerson 2005)

The resulting fault code matrix will look similar to the one shown in Figure 3-2. In each index of the matrix, a "0" represents that a fault code was not registered, and a "1" represents that a fault code occurred. Each row of the fault code matrix is a time step at which a pre-operation test was conducted. Each column represents a fault code from Table 3-1; for instance, the first column is the first fault code, "valve response time slower than rated guidelines," and the fifth column is the fifth fault code, "valve does not seal." The entire fault code matrix is constructed for an individual component.

Data from a total of 10 valves was simulated using the MATLAB code in Appendix A.1, and these simulated valves are analyzed in the following sections. Figure 3-3 shows the simulated fault codes from the first valve. The x-axis refers to the increasing timestamp at which each measurement was taken. The y-axis gives the fault codes, numbered 1 to 10, which a valve might experience at that timestamp. These fault codes have been color-coded for ease of visualization. Looking at Figure 3-3, the fault codes that the valve experiences during the first operating cycle are "valve does not fully close," "valve does not seal," and "temperature higher than rated guidelines."

It can be seen that there are few numbers of fault codes near the beginning of life of the component, while the rate of fault code generation appears to increase near the end of life. This effect was purposefully simulated so that the simulated fault codes would approximate a monotonically increasing prognostic parameter. In a real-world situation, it is likely that some fault codes will be highly correlated with one another. For instance, the fault codes in this simulated data set "valve does not seal" and "valve does not fully close" would likely be highly related to each other. This effect was not generated in the simulated data, but it will be investigated further with the analysis in section 4.1.5.

The simulated data set only serves to demonstrate the proposed methods in this dissertation. While the fault codes in Table 3-1 are potentially those which might be recorded in a real system, it is unlikely that a typical valve would be monitored with enough sensors to determine these particular ten fault codes. Furthermore, a valve system does not necessarily represent the best application of fault code monitoring. A valve system was chosen because it is simply constructed and easy to visualize. Application of these methods to real-world data will be discussed in Chapter 4.

$$FC = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ . & . & . & . & . & . & . & . & . & . \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

Figure 3-2. Simulated Fault Code Matrix.

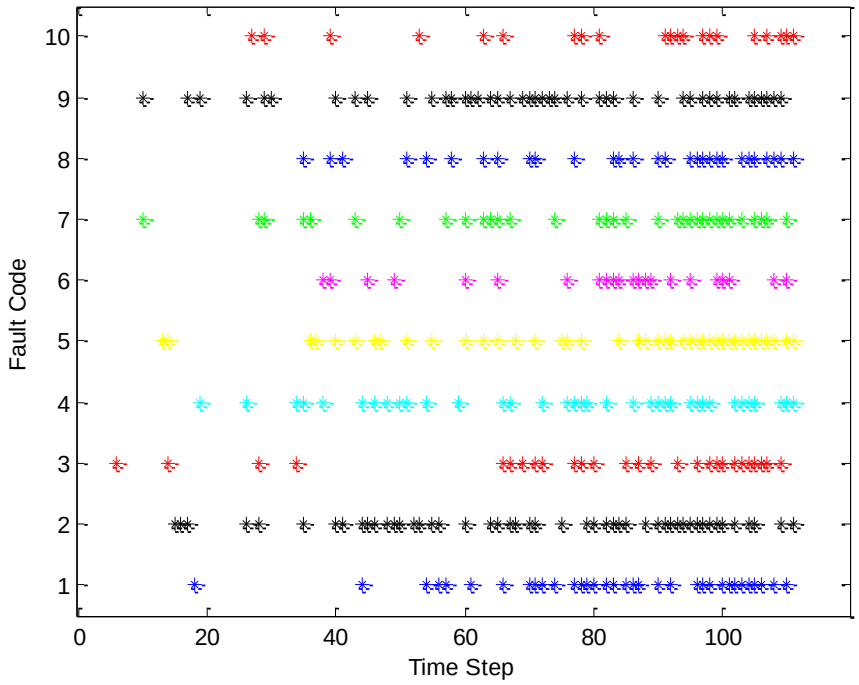

Figure 3-3. Fault Codes for Simulated Component 1.

3.2 Constructing Prognostic Parameters from Binary Fault Codes

Several methods were developed to construct a prognostic parameter from a matrix of binary fault codes. Essentially, the idea is to transform the data from a form that looks similar to Figure 3-2 to a prognostic parameter that looks like Figure 2-5. Three primary methods of accomplishing this purpose are investigated in the following sections: cumulative count, cumulative mean, and windowed mean. The MATLAB code use to perform these three methods is called *fc2pp*, given in Appendix A.2.

3.2.1 Cumulative Sum

The first method to transform the binary fault code data into a prognostic parameter uses the cumulative sum, also called cumulative count, of fault codes at each timestamp. For instance, the cumulative sum of fault codes at timestamp 3 will be the total number of all fault codes that occurred during timestamp 1, 2, and 3. Equation 3-1 below shows how a cumulative sum is obtained for a system with *n* fault codes at time step *k*.

$$CS(t=k) = \sum_{i=1}^{k} \sum_{j=1}^{n} FC(i,j)$$

Equation 3-1. Cumulative Sum.

Figure 3-4 shows the cumulative sum of fault codes obtained from the simulated data. It can be seen that the parameters are smooth and monotonically increasing, which are advantageous for a prognostic parameter. However, the parameters do not tend to fail in a fairly tight distribution, which is not optimal for a prognostic parameter.

A primary advantage of this method is that the resulting parameter will always be monotonically increasing. As described in section 2.2, a monotonic prognostic parameter is preferred because it is assumed that components do not self-heal. Since the cumulative sum of fault codes will always either stay the same or increase at each timestamp, the parameter must be monotonic.

However, the main disadvantage of this method is that the prognostic parameter does not return to normal after maintenance has been performed, which is the downside of a highly monotonic parameter. In some cases this behavior may be preferred, but in some situations, especially when repair rates are higher than replacement rates, monotonicity may actually be a negative aspect of a prognostic parameter. For instance, valve 7 actually experienced a simulated maintenance action at timestamp 50. The degradation of valve 7 was reset to zero at this timestamp, but according to Figure 3-4,

the prognostic parameter of valve 7 did not return to zero. Rather, the prognostic parameter appeared to flatline between timestamps 50 and 70. This disadvantage could be mitigated by keeping careful maintenance records of all components. When the component experiences a maintenance action, the prognostic parameter could be reset to zero. However, this might not entirely be correct either, as a maintenance action will likely not fully reset the actual degradation to zero, and it is difficult to gauge exactly how well the maintenance action fixed the problem. In some situations where maintenance can be verified to repair the component to as good as new or as good as used, this method might be more appropriate.

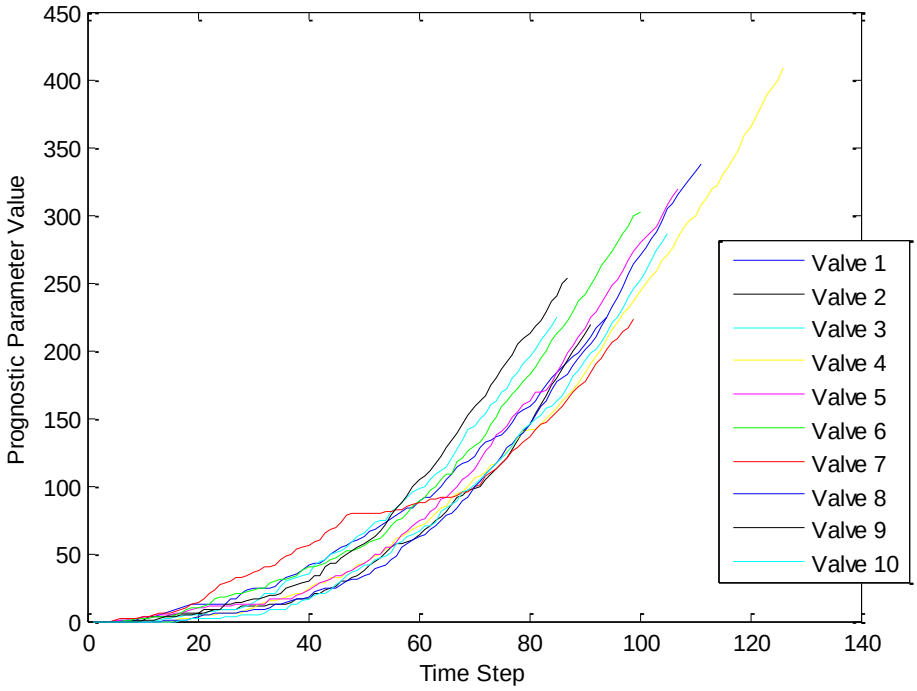

Figure 3-4. Cumulative Sum of Fault Codes from Simulated Data.

3.2.2 Cumulative Mean

Another method which was investigated to transform the binary fault code data into a prognostic parameter was the cumulative mean. In other words, the cumulative mean of the fault codes at time step 5 will be the mean number of fault codes from time steps 1 through 5. Equation 3-2 below shows how a cumulative mean is obtained for a system with *n* fault codes at time step *k.*

$$CM(t=k) = \sum_{i=1}^{k} \sum_{j=1}^{n} \frac{FC(i,j)}{k}$$

Equation 3-2. Cumulative Mean.

Figure 3-5 shows the cumulative mean per timestamp for the simulated valves. These constructed parameters are not necessarily as smooth as those created using the cumulative count method described in the previous section, and there will be much more variance in the parameter during early life because less measurements have been taken. Hence, this construction method is more useful for data that have been collected over many operating cycles. Furthermore, the cumulative mean parameters are not necessarily monotonic.

The cumulative mean method is a slight improvement to the cumulative count method in that it provides a mechanism by which the parameter may return to "normal" values after an unrecorded maintenance action. If the number of fault codes begin decreasing after a maintenance action, the mean number of fault codes will also begin decreasing, albeit slowly. This can be seen in Figure 3-5 for valve 7. Near timestamp 50, the parameter appears to decrease until timestamp 70, when it begins increasing again. This decrease reflects the unrecorded maintenance action near timestamp 50, and it is an improvement over the cumulative count method, which only resulted in a flat line. However, while the prognostic parameter decreased near the maintenance action, it did not return to normal as an ideal prognostic parameter should have done.

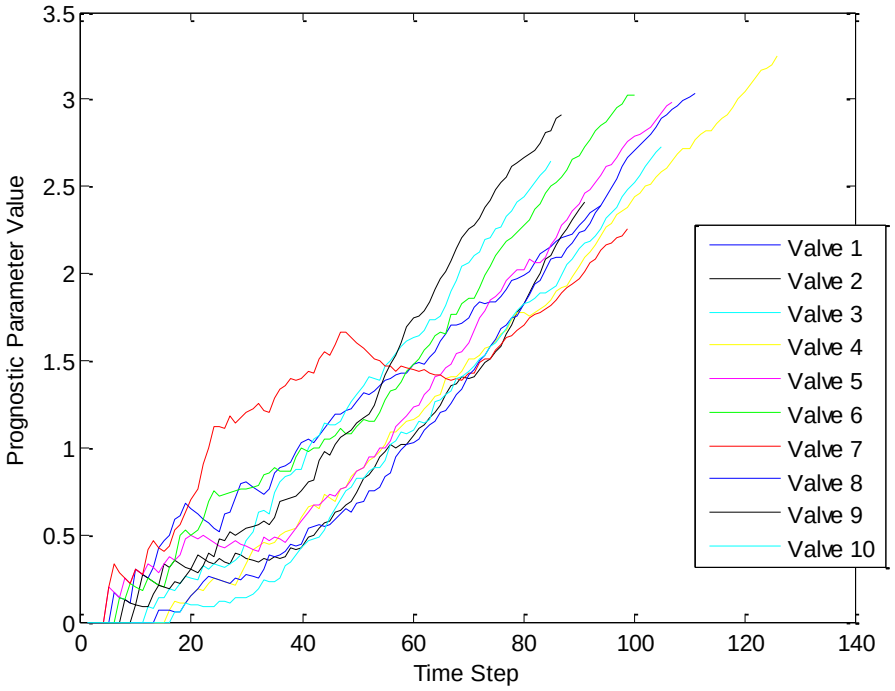

Figure 3-5. Cumulative Mean of Fault Codes from Simulated Data.

3.2.3 Windowed Mean

The final method used to transform the binary fault code data into a prognostic parameter is a windowed mean. A windowed mean is calculated by finding the average of a predetermined number of previous observations. For instance, the windowed mean with a window size of 5 at timestamp 6 would be the mean of the values from timestamps 2 through 6. Equation 3-3 shows how a windowed mean would be obtained for a system with *n* fault codes at time step *k* for a window size of *m*.

$$CM(t=k) = \sum_{i=(k-m)}^{k} \sum_{j=1}^{n} \frac{FC(i,j)}{m}$$

Equation 3-3. Windowed Mean.

Figure 3-6 shows the windowed mean of the simulated valves with a window size of 5, and Figure 3-7 shows the same data with a window size of 10. It can be seen from the figures that a window size of 10 tends to produce smoother data and results in a more monotonically increasing parameter. It is likely that for this data set, a window size near 10 will be ideal to capture normal degradation and unrecorded maintenance activities.

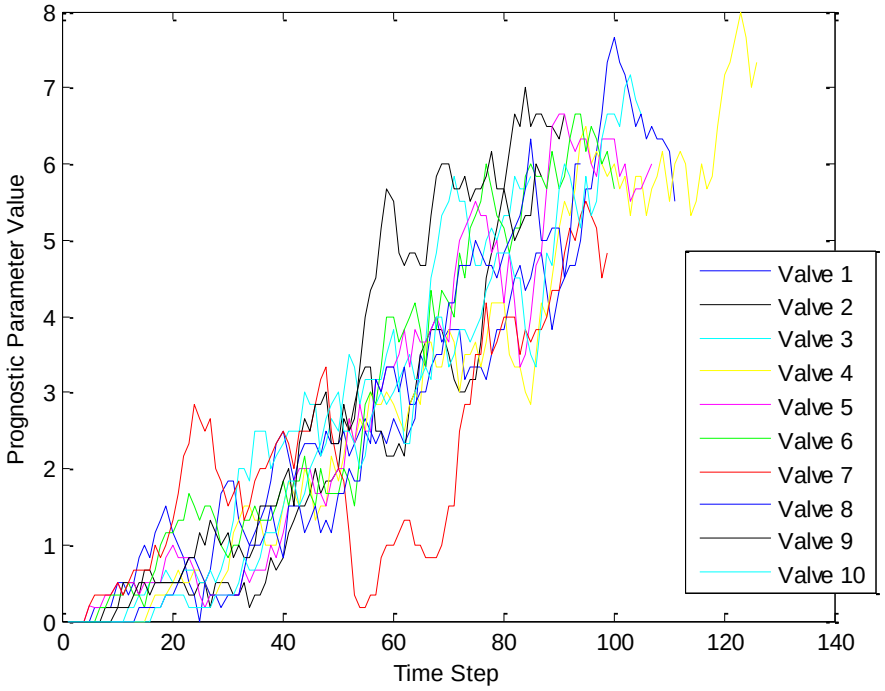

Figure 3-6. Windowed Mean (Window Size 5) of Fault Codes from Simulated Data.

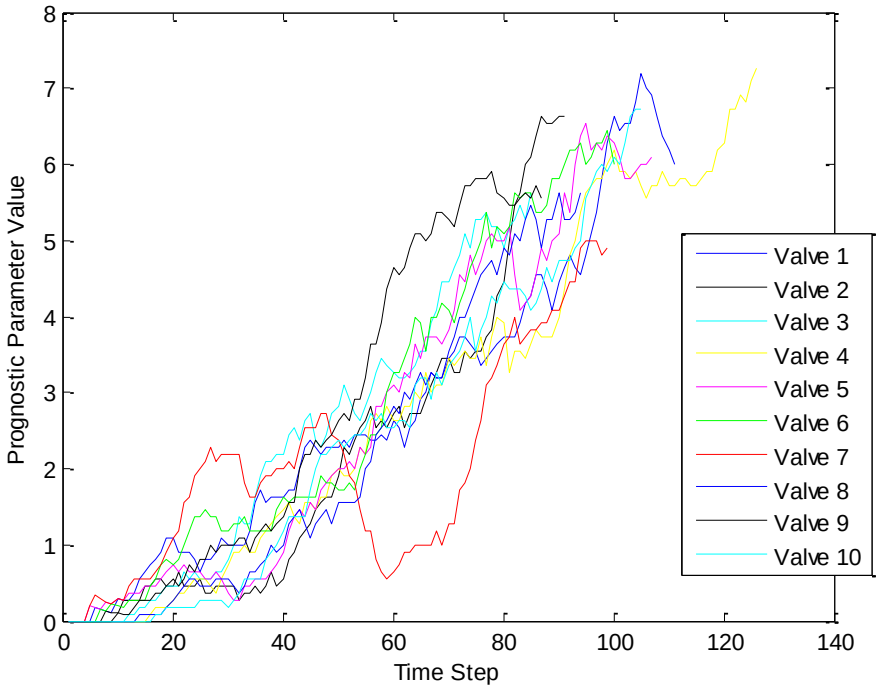

Figure 3-7. Windowed Mean (Window Size 10) of Fault Codes from Simulated Data.

The windowed mean provides the best solution to the problem of an unrecorded or unreliable maintenance action. It can be seen that valve 7 shows a sharp decrease near timestamp 50, returning to almost normal values before increasing again near timestamp 70. This parameter reflects more accurately the actual degradation of the simulated valve. Depending on which window size is selected, the parameter may return more quickly or more slowly to normal values after maintenance is performed. For example, a window size of 5 might require about 5 timesteps before the parameter returns to a normal value. This is an important consideration which must be taken into account when a window size is selected.

3.2.4 Performance Metric Results

The prognostic performance metrics described in section 2.2 were applied to the simulated data from this section, and results are shown in Table 3-2. For reference, values about 0.7 for all three performance metrics are generally considered to be good, while values less than 0.3 are considered to be unacceptable. However, in this case, due to the non-monotonicity of the system introduced by the unrecorded maintenance action, monotonicity will be much less important than the other two performance metrics.

Table 3-2. Performance Metric Results from Simulated Data.

	Monotonicity	Trendability	Prognosability	Total
Cumulative Count	1.00	0.97	0.80	2.77
Cumulative Mean	1.00	0.89	0.82	2.71
Windowed Mean (5)	0.90	0.89	0.66	2.48
Windowed Mean (10)	0.95	0.89	0.76	2.60

All three methods performed very well in regards to monotonicity and trendability, with values near 0.9. The prognosability, likely the most important of the three metric, also achieves good results. The prognosability metric for a window of 10 is higher than for the window of 5, implying that a window of 10 is better suited for this type of data than a window of 5. While the three metrics were in general higher for the cumulative count and cumulative mean than for the windowed mean, they do have the disadvantage of not reacting well to unrecorded maintenance actions. Each of the three methods may be the best selection for a particular application due to their unique strengths and weaknesses.

3.2.5 Prognostic Parameter Optimization Using Ideal Weightings

It is likely that some fault codes may be more related to the degradation of the component than others. When a large number of fault codes may be present in a database, it is important to determine how well correlated each fault code is with degradation, and constructing an ideal prognostic parameter may involve weighting the fault codes based on how relevant to degradation they might be. One method to accomplish this might be to use an optimization algorithm, such as gradient descent, which were described in section 2.2.

A MATLAB program, found in Appendix A.2, was developed which uses the function "fminimax" to find ideal weightings for each of the fault codes which maximize the three performance metrics described in (Coble 2009). The "fminimax" function in MATLAB uses gradient descent to find a maximum or minimum; in this case, the maximum being searched for is the linear combination of monotonicity, prognosability, and trendability which was described in Equation 2-6. In this dissertation, the prognostic parameters constructed using ideal weightings with the "fminimax" function will be referred to as the optimized prognostic parameters.

Figure 3-8 shows the prognostic parameter resulting from optimized fault code weightings for the cumulative count method, and Figure 3-9 likewise shows the updated prognostic parameter for a windowed mean with a window size of 10. These figures may be compared to Figure 3-4 and Figure 3-7 respectively. Overall, the optimized prognostic parameters appear to be noisier than the regular prognostic parameters. However, since the prognostic parameters were optimized using the three performance metrics, they will likely outperform the unoptimized prognostic parameters in regards to the three performance metrics.

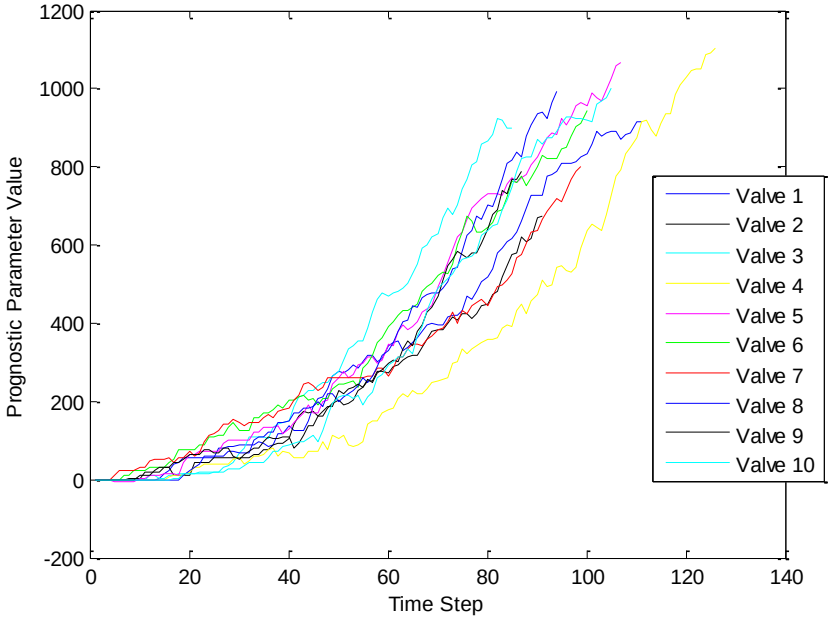

Figure 3-8. Cumulative Sum Parameter Using Optimized Weightings.

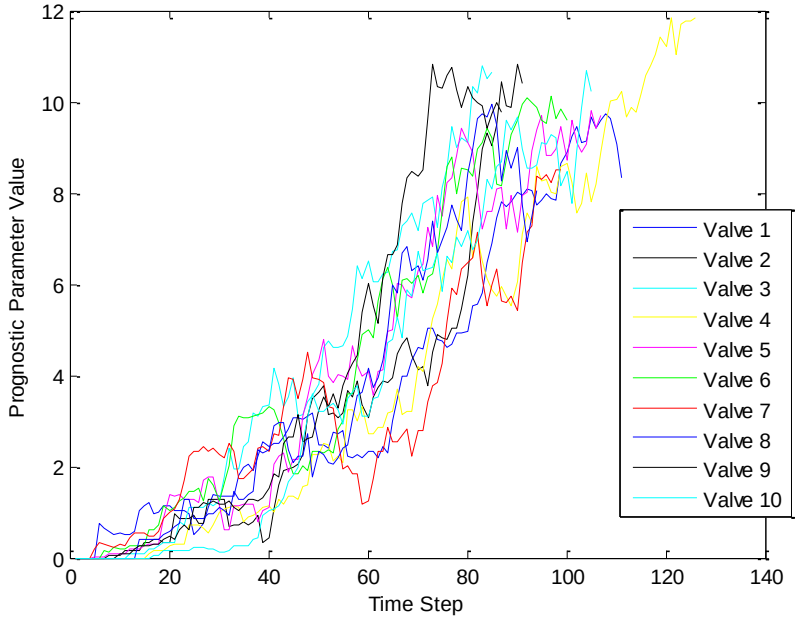

Figure 3-9. Windowed Mean Parameter (Window Size 10) Using Optimized Weightings.

Table 3-3 shows the resulting three performance metrics using the prognostic parameters constructed using ideal weightings. These results may be compared to Table 3-2, which shows the results from the un-optimized prognostic parameters. Overall, the results from the optimized prognostic parameter are an improvement over the results from Table 3-2. In particular, the windowed mean with a window size of 10 was improved, with the prognosability increasing from 0.763 to 0.869.

Table 3-3. Performance Metric Results from Optimized Prognostic Parameter.

	Monotonicity	Trendability	Prognosability	Total
Cumulative Count	1.00	0.98	0.87	2.84
Cumulative Mean	1.00	0.92	0.84	2.77
Windowed Mean (5)	0.93	0.93	0.78	2.65
Windowed Mean (10)	1.00	0.89	0.87	2.76

Since the optimization process in this section was performed on simulated data, there is not a large amount of difference between the optimized and the non-optimized results, shown in Table 3-2 and Table 3-3. However, when real-world data is involved, it is likely that some fault codes will be much more applicable to degradation than others, and optimization may be a more useful process. The application of the optimization on real-world data will be investigated in Chapter 4.

The method described in this section for converting fault codes into a prognostic parameter will be primarily useful for situations in which fault codes exist but monitored signals cannot be used to construct an accurate prognostic parameter. This situation is unlikely, because fault codes must almost always be obtained from some type of condition monitoring, and it will generally be better to construct a prognostic parameter with the monitored signals rather than potentially losing information by downsizing the monitored signals into fault codes. However, this method may be applicable in a case in which the collected data is too large to be stored, and alarm codes are stored instead.

3.3 Incorporating Fault Codes into Existing Prognostic Parameters

When an existing prognostic parameter has already been created, usually by incorporating information from monitored signals, fault code data may potentially be used to supplement the prognostic parameter. The methodology described in this section will be especially useful for components with both collected fault codes and multiple monitored signals which are related to degradation. Since in an ideal case, a good prognostic parameter may be generated from signal monitoring alone, the fault code database will instead be used to supplement the existing prognostic parameter and improve RUL estimates, rather than being used to construct the prognostic parameter alone, as was done in section 3.2. To illustrate the improvement in results when fault code data are supplemented, Figure 3-10 shows the measured degradation from the same simulated data as described in section 3.1.

Looking at Figure 3-10, Valve 7 appears to have anomalous behavior near time stamp 50. As was explained in the previous section, valve 7 experienced a maintenance action near timestamp 50, and its degradation was reset to zero. While the actual degradation from components will likely not be measureable in a real-life situation, this figure can represent a degradation estimate based on the behavior of monitored signals.

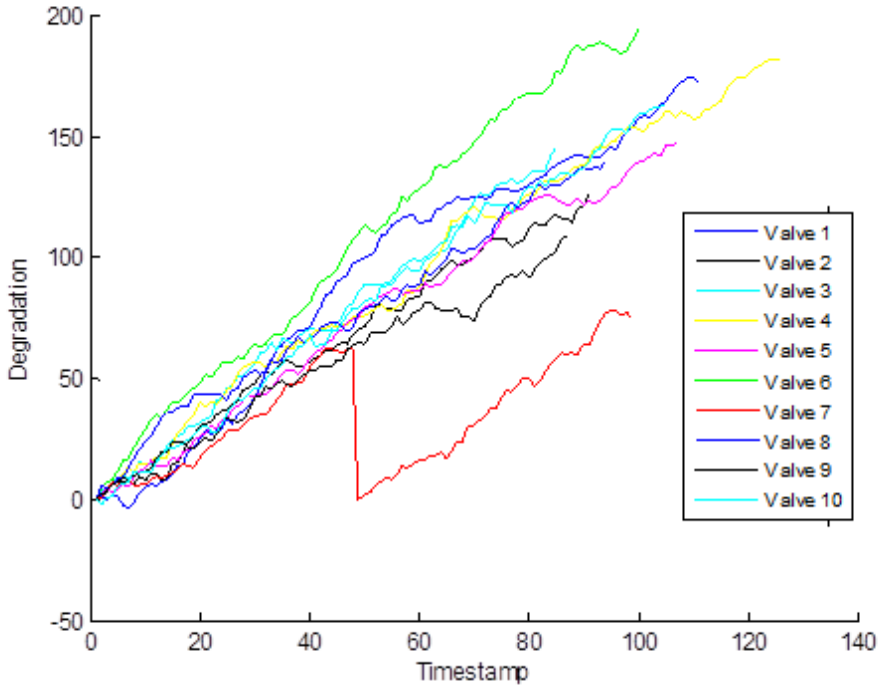

Figure 3-10. Measured Degradation from Simulated Data.

3.3.1 Weighted Average Merging

One method to incorporate fault code data into an existing prognostic parameter is to merge the parameters using a weighted average. This method is shown in Equation 3-4. MP stands for the final merged parameter, FCP is the fault code parameter constructed using fault codes and the methods described in section 3.3, and DP is the prognostic parameter which may be created using monitored signals and the methods from section 2.2. However, this method will work only if the prognostic parameter and the fault code parameter are matrices of the same size.

$$MP = w_{FCP} * FCP + w_{DP} * DP$$ **Equation 3-4. Weighted Average Merging.**

For analysis of the simulated data in this section, a weighting of 1 was used for both w_{FCP} and w_{DP}. Figure 3-11 shows the results when the measured degradation from Figure 3-10 is merged in this fashion with the optimized windowed fault code data with a window size of 10 from Figure 3-9. While the merged parameter looks noisier than the parameter from Figure 3-10, the performance metrics have improved with a monotonicity of 0.933, a trendability of 0.884, and a prognosability of 0.918. These numbers may be compared with the windowed mean parameter alone from Table 3-3.

Although a merging procedure using weighted averaging is simplistic, incorporating two sources of information into the prognostic parameter may not only improve accuracy but may also provide more robust degradation tracking, since there is the potential that degradation will be easier to detect using a fault code matrix than monitored signal values.

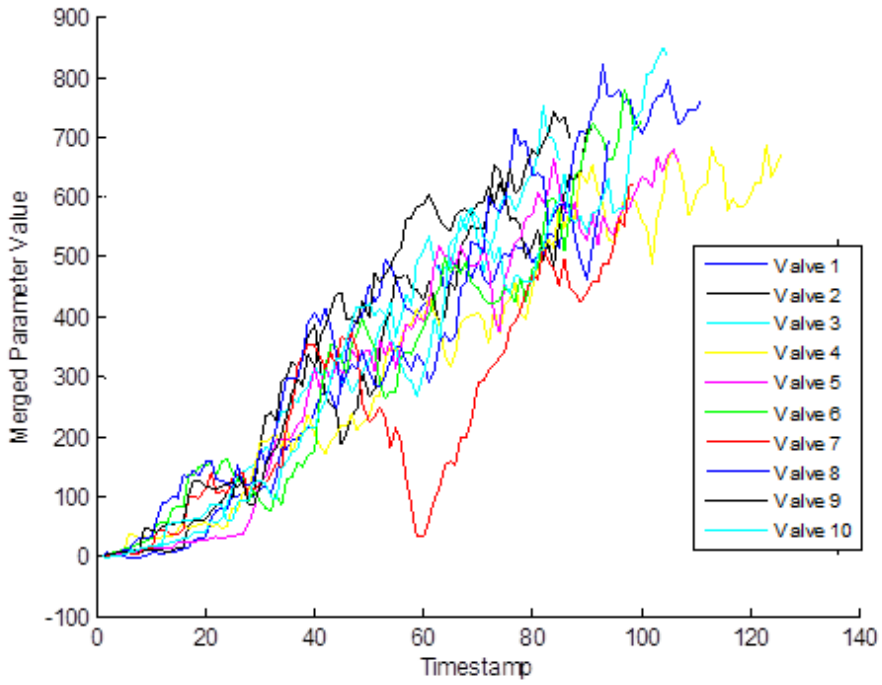

Figure 3-11. Merged Parameter from Simulated Data.

3.3.2 Interpolation Procedure

However, merging a parameter using this method is only feasible if the prognostic parameter from monitored signals and the fault code parameter have observations at the same time steps. In most situations, this will not be the case, because fault codes are typically registered only when aberrant behavior has occurred, and monitored signals are usually updated constantly. For this case, interpolation may the best way to appropriately size the parameters.

An example of how this might be accomplished is shown in Figure 3-12. The monitored signal parameter and the fault code parameter are different sizes. The red stars show the measured value of the monitored signal parameter, and the blue stars show the measured value of the fault code parameter.

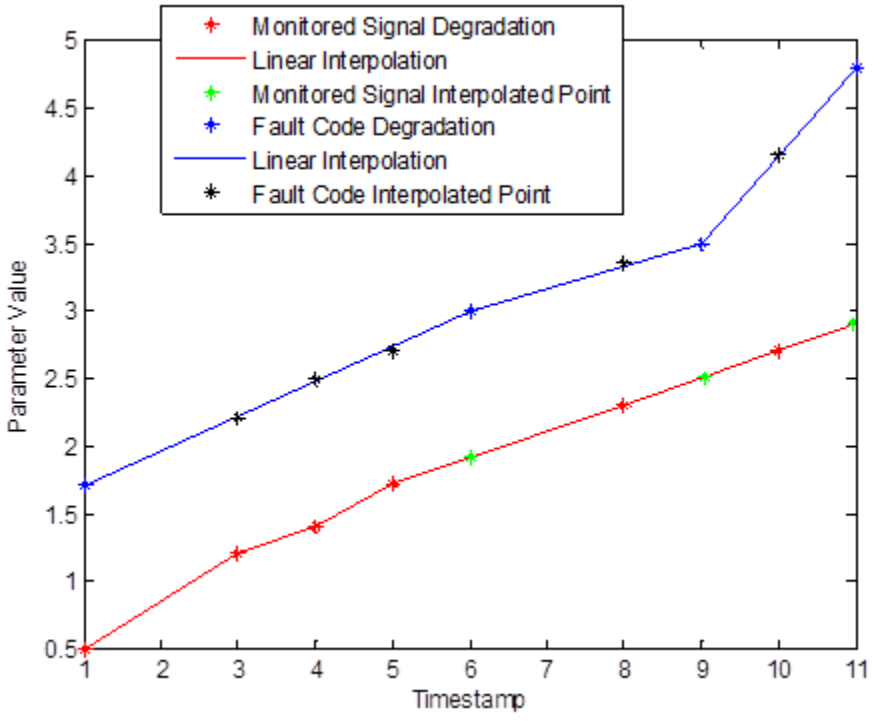

Figure 3-12. Data Interpolation for Merging Procedure.

Looking at the figure, there are several timestamps for which there is a measurement for one parameter but not the other. The solid lines show the linear interpolation of the data points for each of the two parameters. For timestamps at which there is a data point for only one of the parameters but not both, a green star shows an interpolated data point for the monitored signal parameter, and a black star shows an interpolated data point for the fault code parameter. By including the measured data points and the interpolated data points, both signals will be the same size and Equation 3-1 is now applicable.

Care must be taken that interpolation is not too heavily used, since less confidence can be placed in interpolated data points than measured data points. If the total number of time steps of the fault code matrix is much less than the monitored signal matrix, it is recommended that the procedure described in section 3.3.3 be used instead.

3.3.3 Optimized Merging Procedure

The merging procedure described in section 3.3.1 may be applicable for a simplistic system, but in general, a weighting of each fault code based on relevance to degradation can be accomplished more organically within the merging process, without the necessity of the initial parameter construction step described in section 3.2. Furthermore, with proper construction of the code used to merge the fault code information into a prognostic parameter, the interpolation procedure described in 3.3.2 will no longer be necessary as well, because base degradation may be found using the monitored signals and additional "shocks" of degradation may be added for each fault code.

Perhaps the most straightforward way to merge the two parameters is to solve it as an optimization problem using gradient descent or genetic algorithms, as was described in section 3.2.5. However, a major problem was encountered when this method alone was used; specifically, the optimization method alone could not adequately determine an appropriate scaling between the fault code matrix and the prognostic parameter. Since the fault code data is binary and the prognostic parameter is typically continuous, it was

necessary to first make a good initial guess about a scaling factor between the two types of data, especially when the values of the prognostic parameter are large compared to the frequency of fault codes. Essentially, gradient descent optimization is still being employed to solve the problem, but a module in the function will allow a more optimal initial guess before the gradient descent is run. Furthermore, a function with an optimization routine encapsulated inside would allow more user-customizable options, which may be necessary for investigation of unfamiliar data sets.

To solve these issues, a multi-module iterative function was designed to make a "best-guess" of both the scaling factor and initial weights for the fault code columns. This function, named *intfc*, is given in Appendix A.3. Three inputs are supplied to the function: a prognostic parameter, a fault code matrix, and a weighting matrix for the importance of each of the three performance metrics. The prognostic parameter and fault code matrix must be based on the same time scale. If the time scales are different between the two inputs, the original data may be interpolated as demonstrated in section 3.3.2 so that the time steps in both the prognostic parameter and fault code matrix are equivalent. Overall, the function works in three primary steps: appropriately scaling the data with respect to amplitude of the parameter, determining which fault code columns are relevant to degradation, and calculating weights for each column based on performance metric results. Detailed explanations of these three steps will be explored in the next few paragraphs.

First, to optimize the initial guess of the weights, it will scale the fault code data depending on the values of the prognostic parameter. The scaling parameter can potentially be user-selectable, but the default value depends on the average increase in the degradation each time step. This default scaling factor will give the initial guesses roughly equal importance for both the prognostic parameter and the fault code data, assuming there is a significant number of faults in the final row of the fault code matrix. The selected scaling factor largely depends on the typical values of the prognostic parameter, which might be very large compared to the binary values of the fault code matrix.

The second step determines which fault code columns are relevant to degradation. If the results of the performance metrics are below a user-selectable threshold for a particular fault code, it is unlikely that the fault code is related to the fault mode that is being investigated. In most cases, this step would be unnecessary, because weighting of each fault code using gradient descent would simply result in very low weights for faults that are unrelated to degradation. However, this step was included for fault code analysis because it drastically reduced the amount of computation time for large fault code databases. By quickly determining if there is any useful information at all in a particular fault code, the dimensionality of the fault code database could be reduced, and computation time decreased.

Finally, the last step of the program will weight each fault code column using a gradient descent optimization routine with a fitness function that is based on the performance metrics described in section 2.2. This step is the essential function of the program, although the previous two steps will help initialize the gradient descent optimization routine to produce more accurate results in a timely fashion.

Equation 3-5 shows how the optimized merged parameter (OMP) is generated using the three steps mentioned above. In the first step, the scaling factor, shown in red, is found. This scaling factor is multiplied by the importance vector described in the second step, shown in blue, which will determine if a fault code column is used in the analysis at all. Next, the resulting vector will be multiplied by the weights found in the third step, shown in green, which calculates how closely the fault code column is related to degradation. Finally, the resulting vector will be multiplied by the original fault code matrix, shown in black.

$$OMP = SF * \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ \dots \\ 1 \end{bmatrix}' * \begin{bmatrix} 0.78 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0.55 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & \dots & 0.92 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 & 1 & \dots & 0 \\ 0 & 0 & 0 & 1 & \dots & 0 \\ 1 & 1 & 0 & 0 & \dots & 1 \\ 1 & 1 & 1 & 0 & \dots & 1 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 1 & 1 & 0 & 1 & \dots & 1 \end{bmatrix}'$$

Equation 3-5. Optimized Merging Procedure.

The function *intfc* was applied to the simulated data described in section 3.1. Figure 3-13 shows the resulting values when the prognostic parameter and fault code matrix data are merged using the *intfc* function. This figure may be compared with Figure 3-11, which is the simple weighted average merging procedure. The data in Figure 3-13 is less noisy due to the incorporation of the additional information from the fault code matrix. In general, the procedure used for the function *intfc* is expected to give better results than the weighted average method described in section 3.3.1.

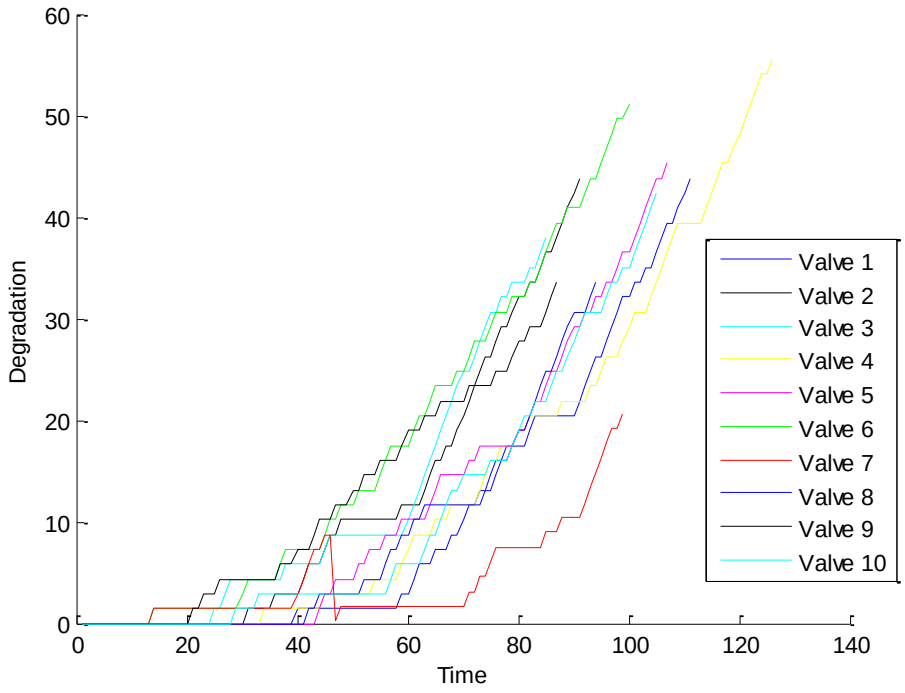

Figure 3-13. Merged Parameters for Simulated Valves.

Table 3-4 shows the assigned weights calculated by *intfc* compared to the sum of the three performance metrics, called the "prognostic performance," which is directly used in step three of the optimization process described above. These results were achieved when the fault codes are used alone (without a previously-constructed prognostic parameter generated from monitored signals) to create a prognostic parameter with the windowed mean method with a size of 10, as demonstrated in section 3.2.3. As can be seen in the table, fault codes which were assigned high weights using the *intfc* function also had high values for the sum of the performance metrics. This effect occurs because both of these values attempt to quantify the relevance of the fault code in regards to degradation, and the *intfc* function explicitly uses performance metric values in its fitness function.

Table 3-4. Performance Metrics vs. Assigned Weights.

Fault Code	Prognostic Performance	Assigned Weights
1	2.12	0
2	2.87	0.085
3	2.33	0.0031
4	2.50	0.024
5	2.86	0.10
6	2.91	0.097
7	2.19	0
8	2.23	0
9	2.76	0.10
10	2.06	0

3.3.4 Summary of Results

Table 3.5 shows the performance metric results from sections 3.2 and 3.3. The best two results from the fault code parameter construction without merging are shown under the heading "fault code parameters," and the two primary methods of merging the fault code information with a previously constructed prognostic parameter are given under the heading "merged parameters."

Table 3-5. Summary of Performance Metric Results from Sections 3.2 and 3.3.

	Monotonicity	Trendability	Prognosability	Total
Fault Code Parameters				
Windowed Mean (5)	0.93	0.93	0.78	2.65
Windowed Mean (10)	1	0.89	0.87	2.76
Merged Parameters				
Weighted Average	0.93	0.88	0.92	2.74
Intfc	1	0.89	0.93	2.82

Both merging procedures appear to work better than a prognostic parameter constructed from fault codes alone. Furthermore, the best results come from the *intfc* function. However, depending on the particular data set being analyzed, any of these methods might be applicable. If monitored signals are unavailable or unreliable, construction of a prognostic parameter using fault codes alone might yield better results. When both monitored signals and a fault code matrix is available, a merging procedure should likely be used. If computation time must be reduced, a simple weighted average merging procedure might be the best option, but if computation time is not an issue, optimization of weights for the fault codes using a function such as *intfc* will likely improve the accuracy of the resulting prognostic parameter. Since computation of the weights needs to be done only once, it may be a lesser consideration if time is available before implementation of a system.

3.4 Operational Fault Code Matrix Generation

The methods described in this dissertation may still be applicable even when a pre-operation test for the component is not regularly performed. Although collection of raw signals will usually be preferred, fault codes may be generated during normal operation for a monitored component, especially when data storage is a concern or raw signal storage is unfeasible for some other reason. These fault codes may be produced each time aberrant behavior is detected from the component, and the resulting fault code matrix may be used directly in the analysis described in this dissertation.

One potential issue may be how to address multiple occurrences of aberrant behavior in the same data collection time period. If each row in the fault code matrix refers to a minute of operation, it is important to determine how many entries should be placed in the fault code matrix if aberrant behavior is observed multiple times during this time period. A solution may be to choose time steps that are short enough such that this behavior is unlikely to occur, but this is not always feasible because it might involve too

much time-consuming optimization of the length of the time step. An easier way to address the problem is to allow multiple entries to be present in the fault code matrix. Instead of ones and zeros, a fault code matrix may thus also contain twos, threes, and fours as well, referring to the number of faults which occurred during a set time period. An upper threshold should likely be chosen so that an arbitrarily large amount of fault codes are not inserted into the matrix, but this behavior is unlikely to occur except directly at component failure, in which case prognostics is not necessary anyway.

A potential method for online fault code generation may be to determine when a raw signal from a component crosses a threshold value which is established using prior unfaulted cases. Once the signal crosses the threshold, a fault code is generated for that time step in the fault code matrix. The resulting fault code matrix may be directly analyzed using the methods described earlier in sections 3.1 through 3.3.

A simulated data set was generated to illustrate how this process might be accomplished. The code used in this simulation is given in Appendix A.4. The simulated machine is a 60 Hz motor with a faulty bearing with bearing fault frequency at 350 Hz. After each accelerated degradation cycle, degradation is progressively induced in the bearing, which will increase the 350 Hz peak in the frequency spectrum of the signal. Random noise is added to each test to more accurately approximate real-life operation of a motor. Essentially, this simulated data set is designed to be similar to the data that will be investigated in section 4.2. Figure 3-14 shows the raw signal and FFT of the first simulated test, and Figure 3-15 shows the final simulated test and FFT.

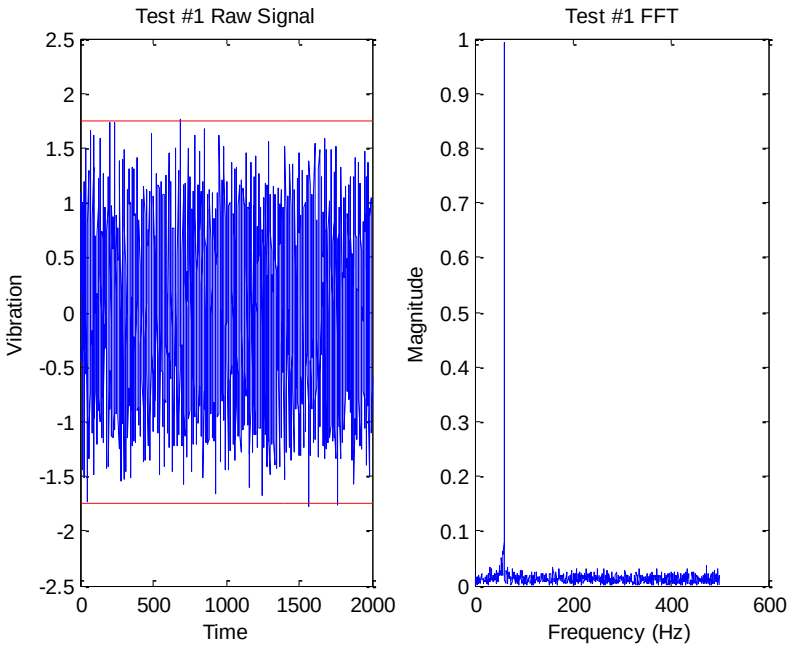

Figure 3-14. Simulated Vibration Signal Test #1.

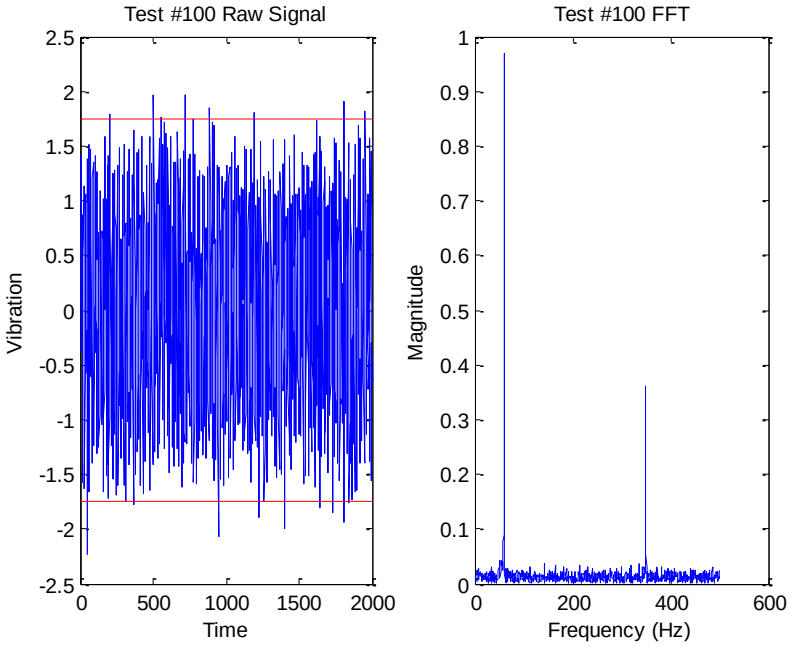

Figure 3-15. Simulated Vibration Signal Test #100.

It can be seen from the raw signal comparison between Test #1 and Test #100 in Figures 3-14 and 3-15 that there is an increase in the number of times the vibration signal crosses the red line threshold, which was set at 1.75 units as an illustration of a possible threshold. In a real-life situation, this threshold might be optimized based on guidelines for operation. For example, motors might be rated for vibration magnitudes below a particular value, and this value might be appropriate to use for a threshold. The difference in the FFT of the first and last test can also be seen in these two figures. As time passes and degradation increases, the magnitude of the frequency spectrum near the 350 Hz value increases.

To create the fault code matrix, the number of times that the raw signal crosses the red threshold line is recorded for each test. This method will result in a long matrix with 1 column and 100 rows, equal to the number of tests. If multiple signals are present, additional columns may be added as well. The resulting matrix can be directly analyzed using the methods described in sections 3.1 through 3.3.

Typically, this type of data would be analyzed by extracting features from the frequency spectrum of the monitored vibration signals. However, there may be some cases where this methodology would not be feasible, such as limited data storage availability or inadequacy of online software to perform the FFT in real-time. The method described here is much less computationally intensive, and for this example the data storage is 99.95% less than storing the raw signals.

Figure 3-16 shows the number of fault codes registered per test. As expected due to the simulation parameters, the number of registered faults is increasing over time, since the degradation of the simulated bearing is increasing. As mentioned earlier, it is important to decide whether or not the total number of fault codes per test is placed directly into the fault code matrix. If the amount of time per row of the fault code matrix is reduced, the number of fault codes registered for that row will also be reduced, and the fault code matrix will tend towards ones and zeroes. However, a better option might be to use the number of fault codes per test as an additional weighting factor in the method described in section 3.3. For instance, if 33 fault codes are registered during a test, an

additional weight of 33 may be multiplied by the weights found using the gradient descent during optimization.

The example of the bearing vibration signal shows how a single signal might be transformed into a fault code during operation of a component. However, there are likely other signals that are being monitored, and these signals can likewise be transformed into fault codes to create more columns in the fault code matrix. For instance, vibration might be monitored in more than one direction, allowing additional columns to be present in the fault code matrix. Other signals that might be relevant are bearing temperature, the frequency spectrum of motor power or current, or ultrasonic or acoustic testing. Furthermore, other columns in the fault code matrix which are not based on monitored signals can be created using maintenance records, observations of aberrant noise coming from the component, and so on.

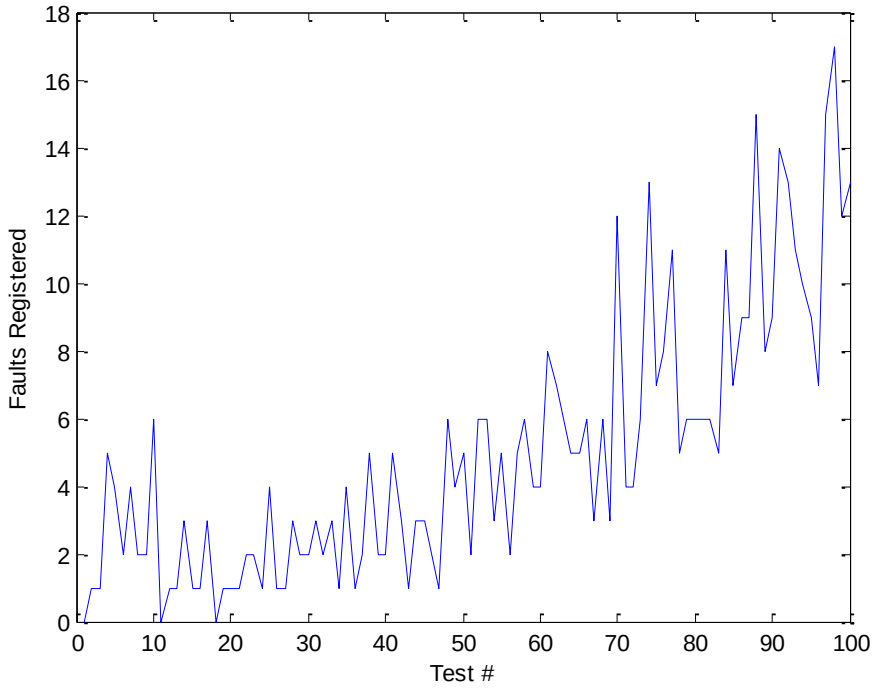

Figure 3-16. Fault Codes Per Test.

Another method of obtaining fault code data from operating components is explored later in Chapter 4 using the accelerated degradation testing motor experiment. This method uses an AAKR model and SPRT detection to find a fault, which may be registered in a fault code matrix in the same way as the threshold method described in this section. Again, multiple fault codes may be generated per time step, and it is important to decide on an appropriate time step and a maximum number of fault codes that can be generated per time step.

3.5 Dimension Reduction of Large Fault Code Databases Using Topic Models

When databases contain large numbers of potential fault codes, it may be difficult or time consuming to perform the analysis described in section 3.3. Ideally, expert knowledge may be able to limit the number of fault codes which are relevant for a particular fault mode. However, in many cases, expert knowledge is unavailable or the system is complex enough that it is difficult to narrow down which fault codes are applicable to a fault mode. In this case, topic models can be used to reduce the dimensionality of the fault code matrix, diminishing the time it might take for a gradient descent optimization routine to optimize the weights for each fault code. Analysis using topic models may also reveal relationships between fault codes that were not found using expert knowledge alone.

A simulated data set was generated to illustrate how topic models might be useful during the analysis of fault codes. A total of ten valves were simulated until failure, with 1000 potential fault codes each. Although 1000 fault codes seems large, there are many real-world applications where thousands of potential fault codes may be registered, and topic models may be useful in this type of situation by reducing the total amount of fault codes being analyzed.

Degradation was simulated for each fault mode using a linear accumulation of damage with random noise. The probability of occurrence of a fault code was determined

by the amount of degradation at each time step of the simulation. Since in real life the occurrence of a fault code is likely related to the amount of the damage the component has accrued, this simulation should give reasonable values for each of the simulated valves. The valves were allowed to fail according to two different fault modes, and a hard threshold was established so that when the components reached a certain level of degradation for the specific fault mode, failure would occur. The MATLAB code used to generate the data for the simulation is given in Appendix A.5.

Figure 3-17 shows the resulting prognostic parameter when the simulated fault code database is run through the function *fc2pp*, using the windowed mean method with a window size of 10. The resulting parameters look fairly monotonic, which is an effect of the simulation process, since very large amounts of fault codes were simulated, and the resulting averages are statistically very predictable.

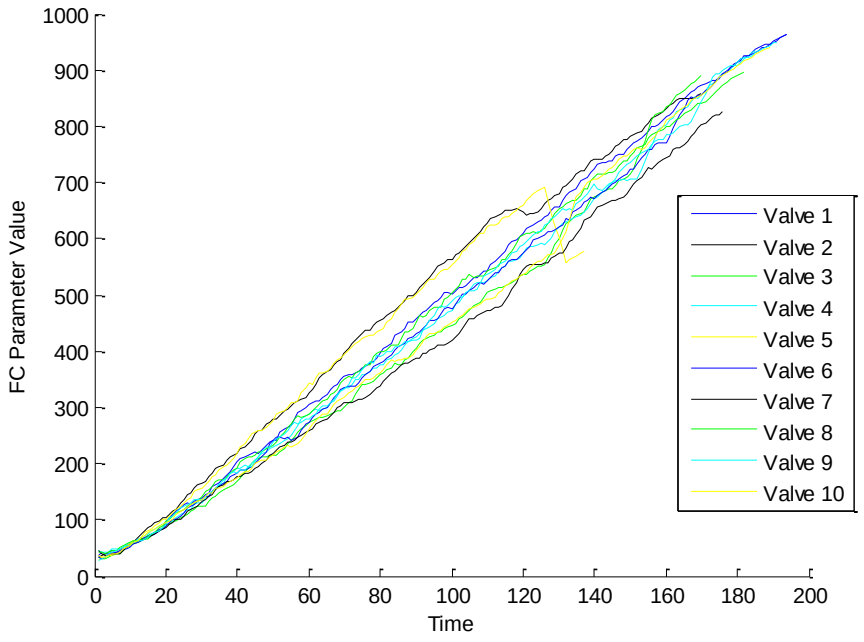

Figure 3-17. Prognostic Parameter Values Using fc2pp.

A histogram of the final values of the prognostic parameter constructed using *fc2pp* is shown in Figure 3-18. It can be seen that the distribution of values is fairly wide. As was explained in section 2.2, it is important that this distribution of prognostic parameter values at failure be tight rather than wide, so these results are not necessarily ideal. However, using topic models to separate out irrelevant fault codes can improve prognostic parameter construction, especially when more than one fault mode is present, as in this case.

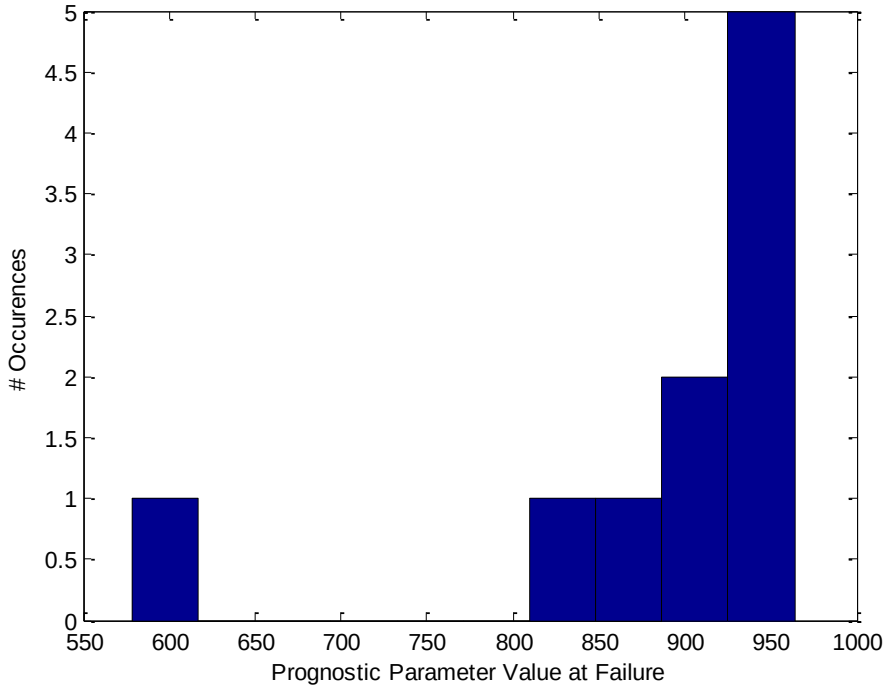

Figure 3-18. Histogram of Prognostic Parameter Values at Failure Using fc2pp.

Next, the simulated data were analyzed using a topic model to determine if improvements could be made to this prognostic parameter. The MATLAB Topic Modeling Toolbox 1.4 was used for the data analysis in this section (Steyvers 2013). Two topics were chosen, since it was known through simulation parameters that two primary fault modes were present. Typically, a topic model requires the number of topics to be pre-specified. In many cases, the number of fault modes is already known, but different numbers of topics may be selected during analysis to determine if accuracy is improved.

As was mentioned in section 2.5, the terminology typically used in topic models may be translated into terms suitable for reliability analysis. "Words" are analogous to "fault codes," "documents" are "components," and "topics" are most closely associated with "fault modes" or "fault mechanisms." The major input to the MATLAB Topic Modeling Toolbox is a frequency count matrix of the occurrence of each word per document. In other words, the frequency count matrix will consist of the number of each fault code per component, which may be found by summing each column in a fault code matrix (Griffiths 2007).

Next, Latent Dirichlet Analysis (LDA) is performed on the fault code matrix. The central assumption of LDA is the exchangeability of both words and documents, which means that they are both conditionally independent and identically distributed with respect to an underlying probability distribution (Blei 2003). In other words, it is assumed that the probability of an occurrence of a word in a particular document is dependent on an inherent probability distribution. In terms of reliability, the assumption is that fault codes occur with a random probability which increases based on the accrued degradation of the component, which seems like a reasonable assumption.

LDA is accomplished by the following process. The first step assumes that the total number of words for a particular document has been drawn from a Poisson distribution, shown below in Equation 3-3. This assumption is made to simplify the derivation used to create LDA, and it is similar to the assumption that the distribution of component lifetimes may be approximated by a normal distribution. In Equation 3-6, X is

a discrete random variable and λ is the expected value of the Poisson distribution. For all k greater than or equal to zero, the Poisson distribution may be applicable (Blei 2003).

$$Pr(X = k) = \frac{\lambda^k e^{-k}}{k!}$$ **Equation 3-6. Poisson Distribution.**

The next step is to draw a random topic θ from the Dirichlet distribution, shown in Equation 3-7. In this equation, p_i is the probability density for the parameter u_i, and Z(u) is a normalization constant. The parameters u_i in this equation are the topics, and the probabilities p_i for each u_i may be specified beforehand or may be drawn from the multinomial distribution, shown in Equation 3-8. For example, in a 2-topic model, Topic 1 might be "dogs" with a pre-specified probability p_1 of 2/3, and Topic 2 might be "cats" with a probability p_2 of 1/3. The sum of all p_i should equal 1 (Blei 2003).

$$Dirichlet(p, u) = \frac{1}{Z(u)} \prod_{i=1}^{n} p_i^{u_i - 1}$$ **Equation 3-7. Dirichlet Distribution.**

$$Pr(X_1 = x_1 \;\&\; ... \;\&\; X_k = x_k) = (\frac{n!}{x_1! * ... * x_k!} * p_1^{x_1} * ... * p) \; iff \; \sum_{i=1}^{k} x_i = n$$

Equation 3-8. Multinomial Distribution.

For each word in a document, or in reliability terms, for each fault code in a component, a topic is randomly assigned to the topic using the Dirichlet distribution given in Equation 3-7. For instance, in the above example with "dogs" and "cats," a phrase "wagging tail" might be assigned to topic "dogs" with probability 2/3 and assigned to topic "cats" with probability 1/3.

The performance of this assignment is evaluated in the next step. First, the proportion of words for each document *d* that have been assigned to topic *t* is calculated, and this proportion is multiplied by the probability that word *w* has been assigned to topic *t* throughout all documents that contain the word *w*. The resulting probability for choosing a topic *t* is shown in Equation 3-9. Thus, a new topic will be chosen for each word given *P(assignment),* or the probability of assignment to topic *t*. (Blei 2003).

$$P(assignment) = P(topic\ t|document\ d) * P(word\ w|topic\ t)$$

Equation 3-9. Topic Assignment Process. (Blei 2003)

This process will be iteratively accomplished for a user-selected number of iterations. Eventually, topic assignment will be relatively steady-state, and the word assignment to each topic will be determined. The output of the toolbox will be a KxN matrix, where K is the number of topics selected, and N is the number of words. Again, "topics" most closely refer to fault modes, and "words" refer to fault codes. For this particular case, there will be 2 topics and 1000 words. The KxN matrix can be multiplied by the initial fault code matrix to create parameter values for each of the two potential fault modes.

Figure 3-19 shows the resulting parameter for Topic 1, and Figure 3-20 shows the values of the parameter for Topic 2. It can be seen that Topic 1 has 9 failure points (shown as the red stars) clustered together with high parameter values, and 1 failure point with a much lower value. Conversely, Topic 2 has 9 failure points clustered together with low parameter values, and 1 failure point with a higher parameter value.

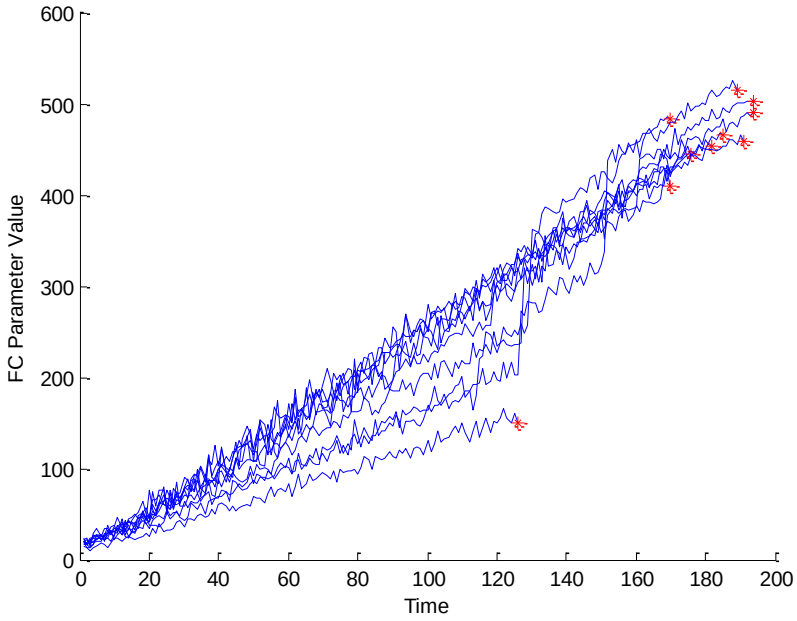

Figure 3-19. Topic 1 Parameter Results.

The interpretation of these results is that 9 out of the 10 simulated components failed according to the failure mode associated with Topic 1, and 1 out of the 10 components failed according to the failure mode associated with Topic 2. These conclusions can be reached by measuring the parameter values for each of the two topics at failure; if the parameter value is higher for Topic 1 than Topic 2, the failure mode is likely the one associated with Topic 1, and if the parameter value is higher for Topic 2 than for Topic 1, the failure mode is likely the one associated with Topic 2. Since the data was simulated, it is known which components failed due to which fault mode, and the data analysis in this section achieved correct results 100% of the time. However, this data is simulated, and it is likely that real-world data will present extra challenges to this methodology. Unfortunately, it is difficult to obtain a real-world data set that contains fault codes, multiple known fault modes, and direct measurements of degradation.

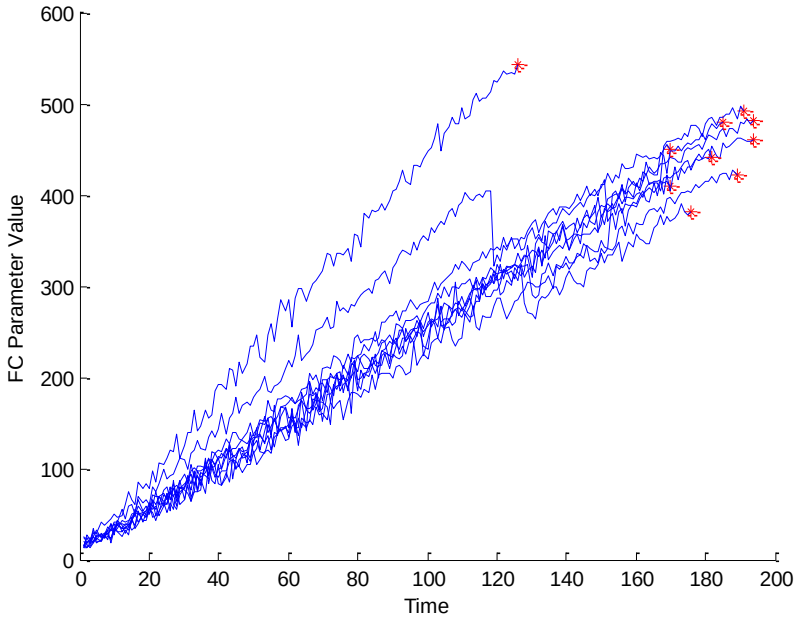

Figure 3-20. Topic 2 Parameter Results.

To accurately estimate the degradation of a component using the above analysis, the following method was used. After the data was normalized so that each topic was given equal weight in the analysis, the prognostic parameter value was considered to be the maximum value of the two topics. Of course, these weights could also be adjusted when it is known that one failure mode is more likely to occur than another failure mode.

For the simulated data mentioned in the sections above, Figure 3-21 shows the maximum value of the two topics. It can be seen that the parameters are smoother than those in Figures 3-19 and 3-20, and the distribution of values at failure (shown in Figure 3-22) are much narrower and look like they may be approximated by a Weibull distribution, which may be ideal for a prognostic parameter, as was discussed in Section 2.2. Figure 3-22 may be compared to Figure 3-18, whose distribution does not appear to be normally distributed or narrow.

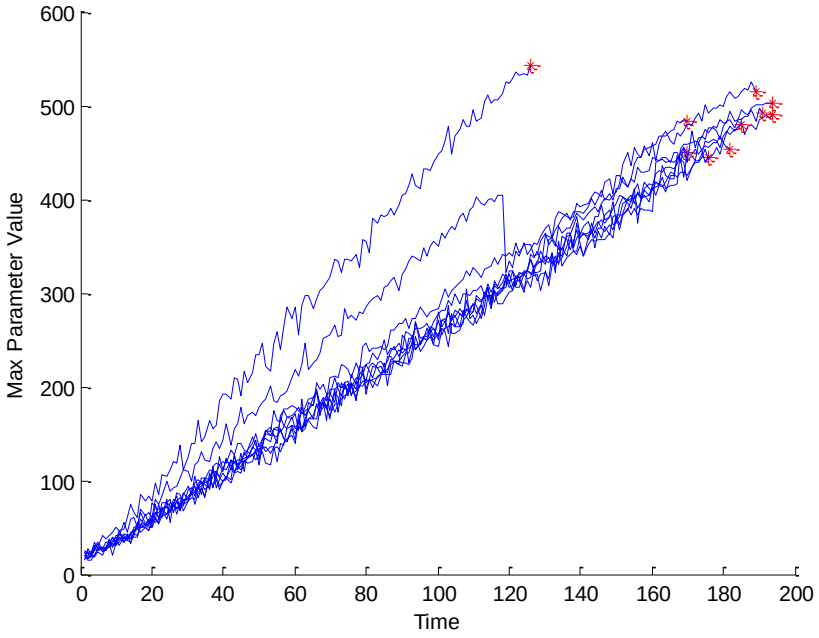

Figure 3-21. Maximum Values of Topics 1 and 2.

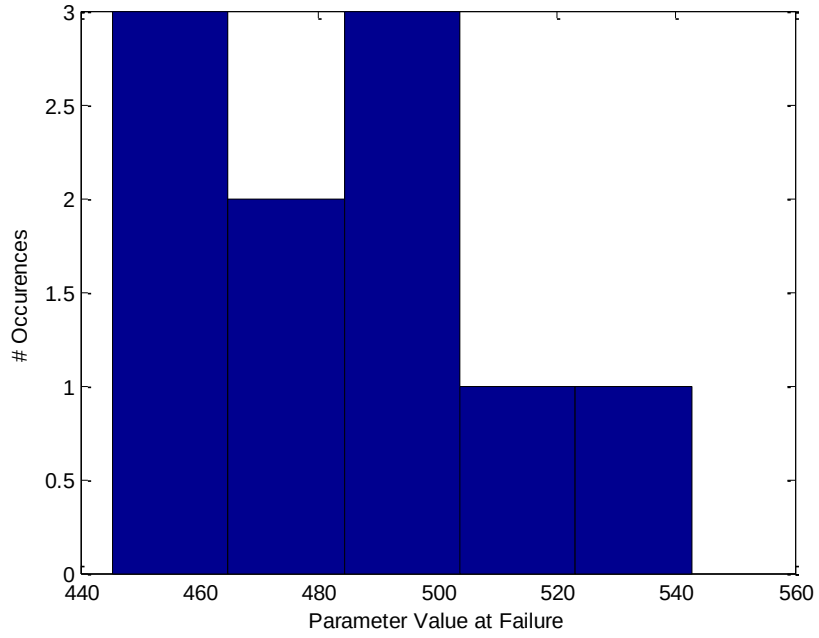

Figure 3-22. Distribution of Parameter Values at Failure.

Table 3.6 shows the performance metric results from this section. Due to the fact that the data was simulated, there is not a large difference between the performance metric results of the three models constructed using topic models. However, it can be seen that the windowed mean method with a window size of 5, constructed using the function *fc2pp*, produces worse results than any of the three topic models. Since two fault modes were present in this case, the *fc2pp* method will produce less accurate results than fault mode isolation accomplished using topic models.

Table 3-6. Prognostic Performance Metric Results from Topic Models.

	Monotonicity	Trendability	Prognosability	Total
Windowed Mean, 5	1.00	0.82	0.87	2.69
Topic 1	1.00	0.87	0.92	2.79
Topic 2	1.00	0.86	0.92	2.78
Max Value of T1 & T2	1.00	0.87	0.93	2.80

As demonstrated in this section, topic models may be useful for dimension reduction of large fault code databases, discovery of which fault codes may be associated with a particular fault mode, and increased accuracy of a constructed prognostic parameter when multiple fault modes are present. Unfortunately, due to the difficulty in obtaining data sets with preferred qualities, these three improvements could not be explored in great detail. However, the positive results from the simulated data set in this section indicates that topic models may be a fertile area for ongoing research involving reliability estimations using fault codes.

3.6 Summary of Methodology

In this chapter, the proposed methodology to use fault code information to improve the RUL estimation of components was discussed. First, three methods used to transform a binary fault code matrix directly into a prognostic parameter were discussed. The cumulative sum method found the total number of fault codes which had occurred during the entire lifetime of the component up until the current time, the cumulative mean method calculated the mean number of fault codes at the current time, and the windowed mean found the mean calculated during a window of time before the current test, such as the mean over 5 operational cycles. Each of these methods had benefits and detriments. Generally, the cumulative sum and cumulative mean methods will be more monotonic, but they do not necessarily return to normal values when a component is serviced. The windowed mean method parameter values will quickly return to normal after maintenance but are usually less monotonic and more prone to noise.

A method of incorporating fault code information into an existing prognostic parameter was also discussed. To accomplish this purpose, a function called *intfc* was developed in MATLAB. The function had three primary modules: appropriately scaling the data, determining which fault code columns are relevant to degradation, and calculating weights for each fault code based on performance metric results. The output of this function is an integrated prognostic parameter, and if useful information about the health of the component is contained within the fault code data, the accuracy of RUL estimation should be increased.

For data sets that do not have a pre-operational fault test, a method for transforming normal operational data into fault codes was investigated. Usually, it would be disadvantageous to discard operational data which might give information about the health of a component. However, in industries where data storage is a significant problem, or previously obtained data has already been reduced to fault codes, these methods might be useful.

Finally, topic models were considered as a dimension reduction and fault isolation method. When large fault code databases are examined, it is likely that a large amount of spurious codes is present in the data, and many of the fault codes may not pertain to the fault mode being investigated. Topic models may be used to discard unnecessary fault codes in the analysis, reducing noise and increasing the accuracy of the model. Furthermore, topic models can allow isolation of fault modes when more than one fault mode is present within the fault code database. Without expert knowledge, it may be difficult to understand the interactions between fault codes and which fault codes are applicable to a particular fault mode. Topic models may be useful to determine relationships between fault codes even when little information is known about them.

CHAPTER 4
APPLICATIONS

In this chapter, the methods described in Chapter 3 will be applied to real-life data. Favorable results in this section will indicate that the theoretical methods developed in this dissertation have practical applications for data sets that contain fault code information, especially for those systems which must undergo a test to ensure they are operating correctly before they are placed into operation.

4.1 Actuator Degradation Data Set

The data set that was used for the analysis in this section is export-controlled, and specific information about the data may not be discussed. However, a general description of the type of data involved is given here. The data were taken from an actuator system which was monitored using two given signals. Data from a total of twenty components were received that contained both the two monitored signals and associated fault codes. In previous work on the export-controlled research project that inspired this research, these monitored signals were used to generate a prognostic parameter, as they were related to the health of the component. Results from that analysis were generally favorable.

4.1.1 Data Set Description

Fault codes for the actuator system are generated through a mandatory pre-operation check to determine that everything is operating correctly. Approximately 8000 fault codes might possibly be encountered during this pre-operation check, about one percent of which were identified by expert knowledge as related to the actuator failure being investigated. Each fault code is also associated with a timestamp. For this analysis, the given fault codes have been renumbered and the descriptions of the fault codes have been removed for purposes of export control.

The timesteps given in this data set were based on the operating cycles of the actuator system. For example, time step 3 refers to the third time that the actuator was placed into operation since the beginning of the given data. As such, the time steps may not be directly translated into time units such as days; however, since the actuator system primarily accrued damage through operation alone, the degradation over time should increase fairly regularly per timestep. Another potential difficulty with the given data is that baseline values for the actuators were not available. The data for each component began somewhere in the middle of its lifetime. This made Type I prognostic analysis impossible, and differences in the lifetime of the components should not be necessarily treated as actual high variability in the lifetime, as data may have been received from different portions of each component's operational life.

Finally, data from forensics were not available for the actuator systems. It is difficult to determine which fault modes the components were experiencing or if they were accruing degradation at all, apart from the information found using the monitored signals and fault codes. Since prognostics is best accomplished when data from different fault modes can be isolated into separate models, this made analysis of the data difficult. Furthermore, it is known that a potentially large number of "no fault found" (NFF) cases are present within the data. In other words, the actuator was removed because of aberrant behavior but when examined no fault could be found. It is likely, then, that many of the actuators from the twenty which were received do not actually have a fault but were instead incorrectly classified.

Throughout the analysis in section 4.1, results were only favorable for six out of the twenty components, and results from the other components will not be shown, as it is assumed they are a NFF, and this assumption is supported by other data that cannot be presented here. In the future, it is possible that forensics may be performed on these twenty given actuators, and more information will be known about the failure mode present and whether the actuator was a NFF. In section 4.1.7, a method of determining which actuators are NFF is discussed, although verification of the accuracy of this method could not be established.

4.1.2 Constructing Prognostic Parameters from Fault Codes

Using the methods described in section 3.2, the fault codes from the export-controlled data were assembled into prognostic parameters. In this section, only data from the 66 fault codes which were identified by expert knowledge as related to the actuator fault will be considered. A later section will analyze the entire fault code database using topic models.

Figure 4-1 shows the results from the cumulative count method. It can be seen from the figure that the parameter is monotonically increasing, as is expected from the cumulative count method. It also appears that the components have widely varying lifetimes, but this is likely a result of the data being obtained starting at a random point in the lifetime of the component, rather than starting at the beginning of the component's lifetime, as would be preferable. There is a large variability in the values of the prognostic parameter at the failure of a component. This is not preferable for a prognostic parameter and may indicate that the cumulative count method is unsuitable for this type of data. One possibility is that some of the fault codes which are included in the cumulative count are not related to degradation; optimization using ideal weights, which will be investigated in the next section, can be used to determine if this is the case. One of the components, Actuator 4, appears to have a more steady increase in degradation than the other components, which have sharp increases in the parameter near failure.

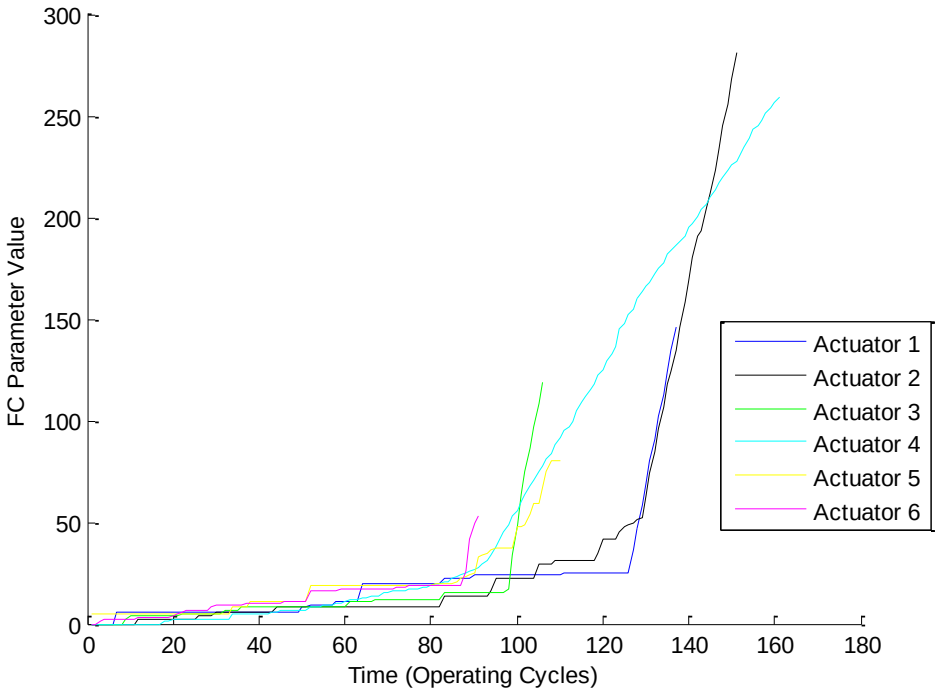

Figure 4-1. Cumulative Count Method for Export-Controlled Data.

The results from the cumulative mean method can be seen in Figure 4-2. Once again, the explanation for the differing lifetimes can likely be attributed to not having the full data for any of the components. Also, since the constructed parameters might have relatively low number of data points, the variance is much higher early on in the component life, which results in lower reliability in the early lifetime.

Typically, results from the cumulative mean method will look very smooth compared to the windowed mean method, since it is inherently smoothing. All of the actuators show a fairly strong upward trend, which likely does mean that the component is experiencing degradation since the mean number of fault codes are regularly increasing. Interestingly, the parameter values for a few of the actuators appear to increase heavily near the beginning of the data, and then the values gradually return to

normal. This behavior likely indicates that there was a maintenance action near this data period which resolved a problem that was causing a fault. Actuator 4, once again, appears to have a smoother upward trend than the other components.

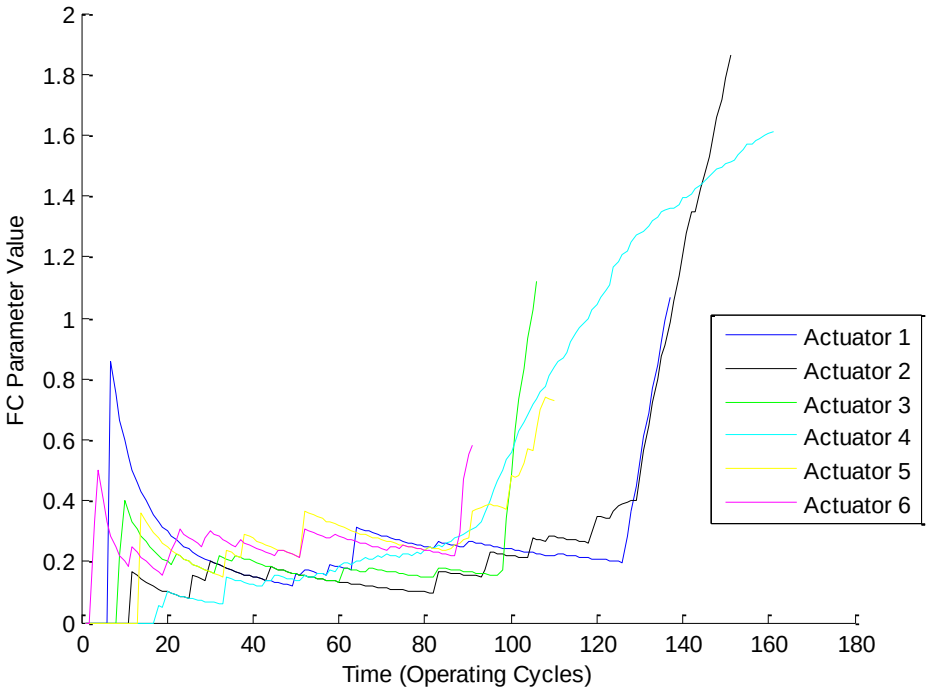

Figure 4-2. Cumulative Mean Method for Export-Controlled Data.

Figure 4-3 shows the windowed mean method with a window size of 5. Overall, the results seem to be much noisier than the other two methods, but this is a result of the inherent smoothing of the other two methods. Looking at the distribution of values at failure, the windowed mean method appears to have a more narrow distribution than the other two methods. Figure 4-4 shows the windowed mean method with a window size of 10. The parameter using this method appears to be slightly smoother, and it will likely

achieve better results from the performance metric. Other window sizes were selected, but the results usually had more smoothing than was acceptable, resulting in almost constant values, especially when the window size was greater than 20.

The windowed mean methodology reveals that there is often a sharp increase in the parameter right before the failure of the actuator. This effect likely indicates that the parameter will be more useful for diagnostics than prognostics, since the slope of accumulated degradation changes throughout the lifetime of the component. However, it is possible that integration of this parameter with monitored signals, investigated in section 4.1.4, will yield better results.

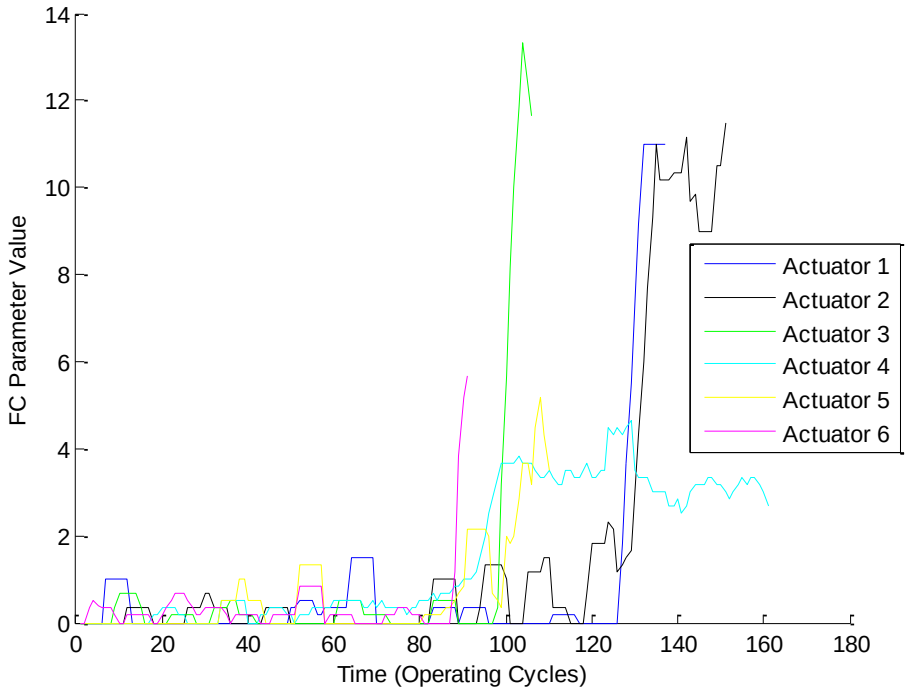

Figure 4-3. Windowed Mean Method (Window Size 5) for Export-Controlled Data.

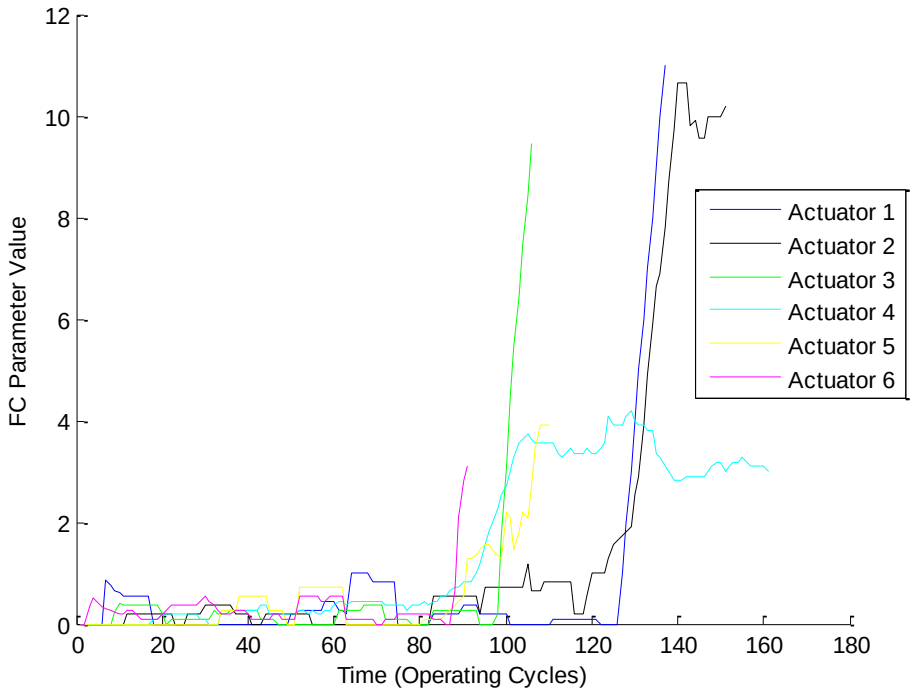

Figure 4-4. Windowed Mean Method (Window Size 10) for Export-Controlled Data.

The performance metric results from each of the four methods are shown in Table 4-1. As expected, the cumulative count and the cumulative mean methods perform better in the monotonicity and trendability metrics; however, they do not perform as well in the prognosability metric. Furthermore, the cumulative count and cumulative mean methods do not react well to unrecorded maintenance action, which is likely present in this data set. The best method to use for this data set is likely a windowed mean with a window size of 10, as it gives slightly better results than the windowed mean with window size of 5, and the windowed mean method will react well to unplanned maintenance actions. The prognosability of the windowed mean methods are fairly good, and it is likely that these two parameters would be useful in determining the remaining useful life of the actuators.

Table 4-1. Performance Metric Results from Export-Controlled Data.

	Monotonicity	Trendability	Prognosability	Total
Cumulative Count	1.00	0.79	0.55	2.34
Cumulative Mean	0.72	0.50	0.65	1.87
Windowed Mean (5)	0.46	0.31	0.68	1.45
Windowed Mean (10)	0.66	0.48	0.67	1.81

4.1.3 Parameter Optimization Using Ideal Weighting

Next, the parameter optimization process was accomplished for the export-controlled data. This parameter optimization process is described in section 3.3 and uses gradient descent to optimize the weights assigned to each fault code column. Once again, the three parameter construction methods (cumulative sum, cumulative mean, and windowed mean) from section 3.2 were employed, using two different window sizes for the windowed mean method. Figure 4-5 shows the results from the parameter optimization for the windowed mean with a window size of 5. Overall, the results from the optimization process appear improved over the results from the previous section. For instance, the parameter values at failure for the components shown in Figure 4-5 appear to be in a tighter distribution than the comparable values in Figure 4-4. This will likely be reflected in an increased prognosability metric in the performance metric results. Results from the other three methods are shown in Appendix B.1.

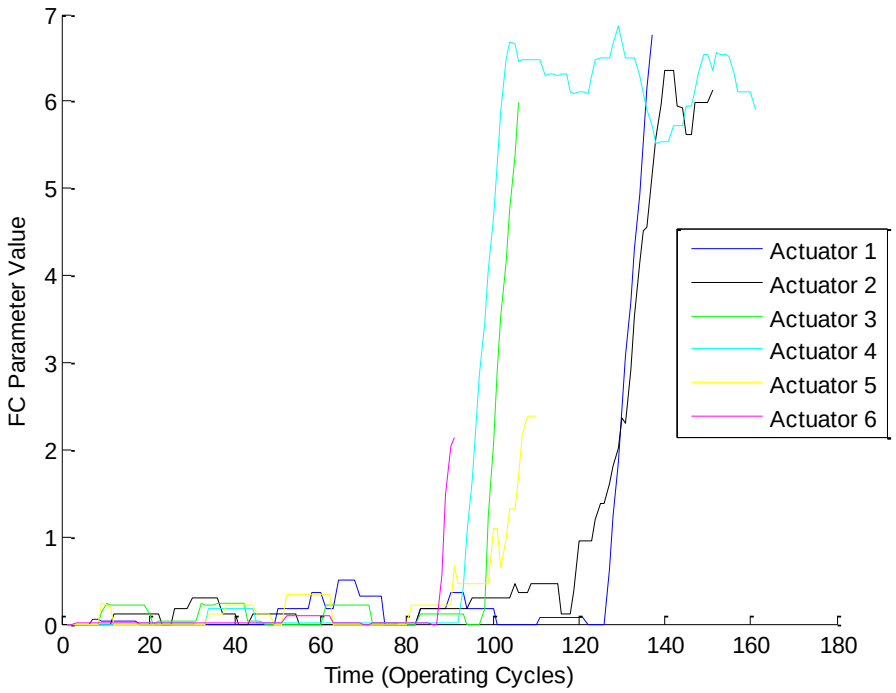

Figure 4-5. Optimized Windowed Mean (Window Size 5) for Export-Controlled Data.

The performance metric results from the optimized prognostic parameters are shown in Table 4-2. In general, the performance metric values from the optimized prognostic parameters are increased from the original parameters from the previous section. The values from Table 4-2 may be compared to the values from Table 4-1 previously. Despite the total of the performance metrics being highest for the cumulative sum method, the best results in Table 4-2 are from the two windowed mean methods, since the prognosability metrics are higher for these two methods, and the prognosability may be the most important of the three metrics in this case. Both of the windowed methods have a prognosability value above 0.7, which is considered to be good.

Table 4-2. Performance Metric Results from Optimized Parameters.

	Monotonicity	Trendability	Prognosability	Total
Cumulative Sum	1.00	0.82	0.58	2.40
Cumulative Mean	0.75	0.54	0.66	1.95
Windowed Mean, 5	0.59	0.42	0.78	1.79
Windowed Mean, 10	0.70	0.48	0.77	1.95

A weakness in the methodology of this section is that the optimization procedure assumes that the degradation of each component is linearly increasing over time, which is not necessarily true. Validating the weights calculated by the optimization procedure would usually require an external measurement of the degradation of the actuator. Since this information was not available, it was assumed that prognostic parameters with higher metric values are preferable, which is why optimization resulted in higher performance metrics. However, if unrecorded maintenance actions are present in the data, it is possible that the "humps" in the data seen in Figure 4-3 and 4-4 represent actual changes in degradation and should not be smoothed as was inherently accomplished in the optimization process.

Overall, the results from this section were slightly better than expected, likely because of the optimization procedure used in section 4.1.3, which employed gradient descent to optimize the weighting of the fault code columns. It appears that fault codes alone in this case can establish a viable prognostic parameter, without any input from monitored signals. The effectiveness of these parameters in predicting RUL will be investigated in section 4.1.6. In most situations, particularly those with a smaller number of fault codes, it is likely that fault codes alone will not be able to establish a prognostic parameter, and some information from existing monitored signals must be used.

4.1.4 Integration with Existing Prognostic Parameter

As mentioned in section 4.1, two monitored signals which related to degradation were collected from the actuator systems. Of these monitored signals, a total of 12 features were extracted, which cannot be discussed because of export-control, and these features were used to construct a prognostic parameter according to the methodology described in section 2.2. The resulting prognostic parameters are shown in Figure 4-6. It can be seen that the prognostic parameters are very noisy, and it is unlikely that these parameters alone will perform well in a GPM. By integrating information from the fault codes, the usefulness of the prognostic parameters shown in Figure 4-6 may be improved.

The optimized fault code prognostic parameters from the previous section were integrated with the prognostic parameter formed from the two monitored signals according to the method described in section 3.4, in which the function *intfc* was used to find optimal weights for the fault code columns. Since the timesteps between the two prognostic parameters did not directly match up, an interpolation method as described in section 3.4 was employed.

Figure 4-7 shows the results from the integrated parameter. This figure may be directly compared to Figure 4-5, which showed the optimized fault code parameter alone. Overall, the results from the integrated parameter appear to have improved over both the monitored signal parameter alone and the fault code parameter alone. All of the actuators have a similar trend and appear to be monotonically increasing at a regular rate. Strangely, actuator 4 had much larger values than any of the other actuators, especially at failure. From Figures 4-1 through 4-4 it can be seen that actuator 4 behaved differently than the other actuators as well, which might indicate that this actuator is experiencing a different fault mode from the other actuators.

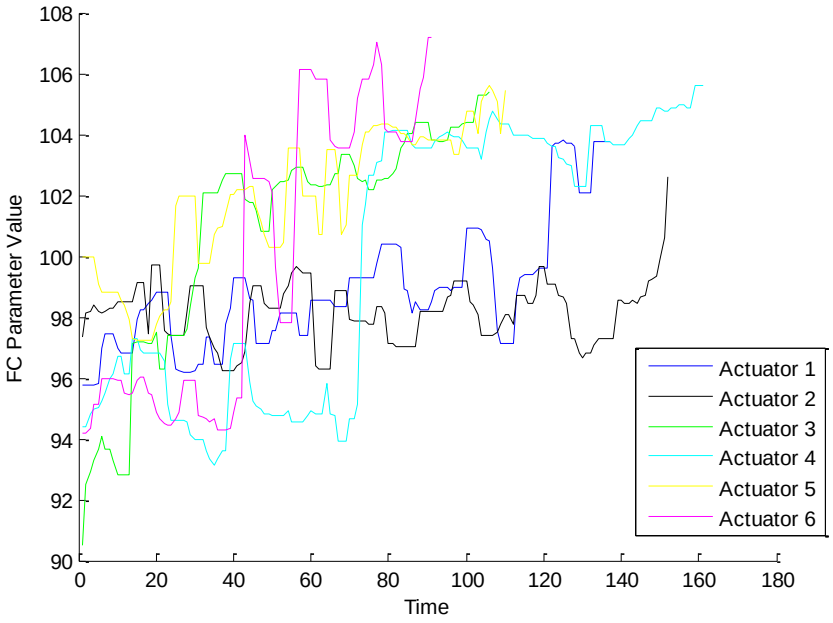

Figure 4-6. Constructed Prognostic Parameters.

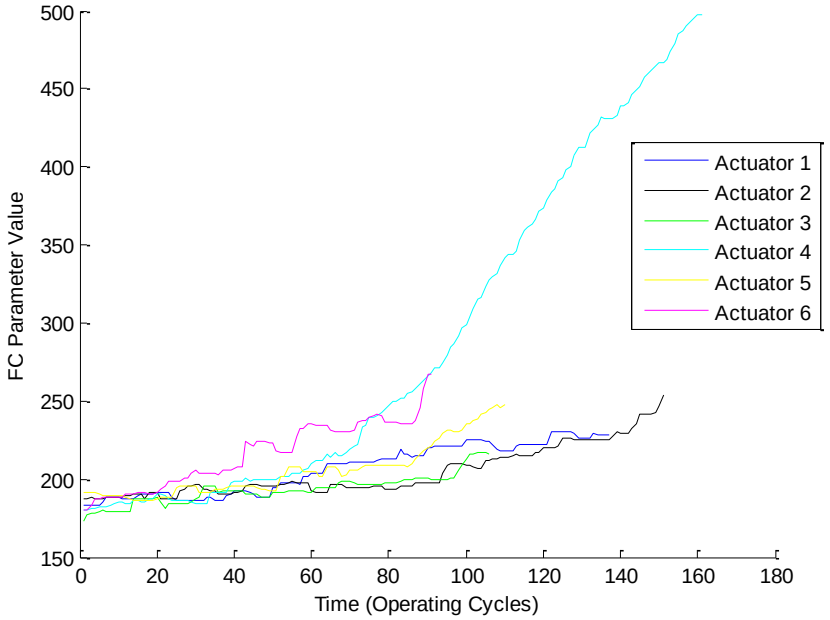

Figure 4-7. Incorporated Prognostic Parameter for Windowed Mean Method with Window Size 10.

Overall, the performance metric results from the integration procedure in this section were better than those in the previous sections, with a monotonicity of 0.92, a trendability of 0.36, and a prognosability of 0.84. These results seem to indicate that the integration process is an improvement over the previous methods described in sections 4.1.1 through 4.1.3. However, it is possible that the degradation accumulation of the components is not regular for each timestep, which would certainly affect the monotonicity and trendability metrics. In other words, some operating cycles of the component are more stressful than others, which affect the amount of degradation accumulated per cycle. This effect could be investigated with an estimation of the amount of stress, but this information was not available. A better dataset with time measured in hours or days would likely give more regular accumulations of degradation than this particular dataset, in which the data was measured in operating cycles.

4.1.5 Dimension Reduction Using Topic Models

In this section, the data set which contained the entire set of approximately 8000 fault codes rather than the about 80 identified by expert knowledge as relating to the actuator fault was analyzed. Limited information was available about these fault codes, and it is likely that they are related to systems other than the actuator system of interest in this analysis. However, it is possible that they might still contain useful information related to degradation, as components may be interrelated and degradation in one component may affect another.

First, the entire fault code data set was analyzed using the methods in section 3.2, transforming fault codes into a prognostic parameter. Figure 4-8 shows the results when a windowed mean with a size of 10 is used. Additional results from other methods are given in Appendix B.2. From the figures, it can be seen that the resulting parameter is very noisy and does not appear to be useful in determining the degradation of the actuator. It is likely that the parameter is so noisy because many of the fault codes which were used are not actually relevant to the degradation of the actuator. To solve this issue,

topic models will be used to reduce the dimensionality of the 7774 fault codes and indicate which groups of fault codes are relevant for the degradation of the actuator.

First, a topic model was constructed from the 7774 fault codes with a total number of 12 topics. 12 topics were chosen because there were roughly 10 total systems which were represented in the fault code database, and slightly extra topics will potentially allow noise reduction as well. Furthermore, there may be multiple fault modes per system. Ideally, the fault codes relating to the actuator degradation will be clustered within the same topic and may be used as an input to the *fc2pp* function, which will create a prognostic parameter. The MATLAB Topic Modeling Toolbox 1.4 was used for the topic model analysis in this section (Steyvers 2013).

Figure 4-9 shows the total number of fault codes for each topic. Each fault code was assigned to a particular topic based on the maximum likelihood that it belonged to the topic, which was calculated by the MATLAB Topic Modeling Toolbox 1.4. Fault codes which did not occur frequently enough (less than 3 occurrences throughout all actuators) to be included in the analysis were removed during this step, resulting in a total of 503 fault codes. Topics 6 and 11 consisted entirely of fault codes which did not occur frequently enough and were removed entirely. Topic 1 was also excluded from analysis because it contained only one fault code.

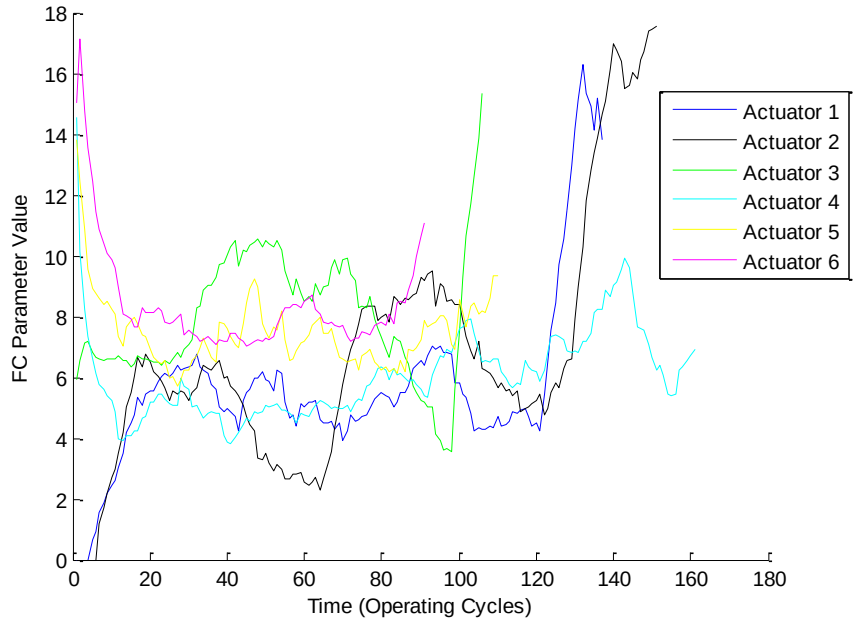

Figure 4-8. All Fault Codes, Windowed Mean with a Size of 10.

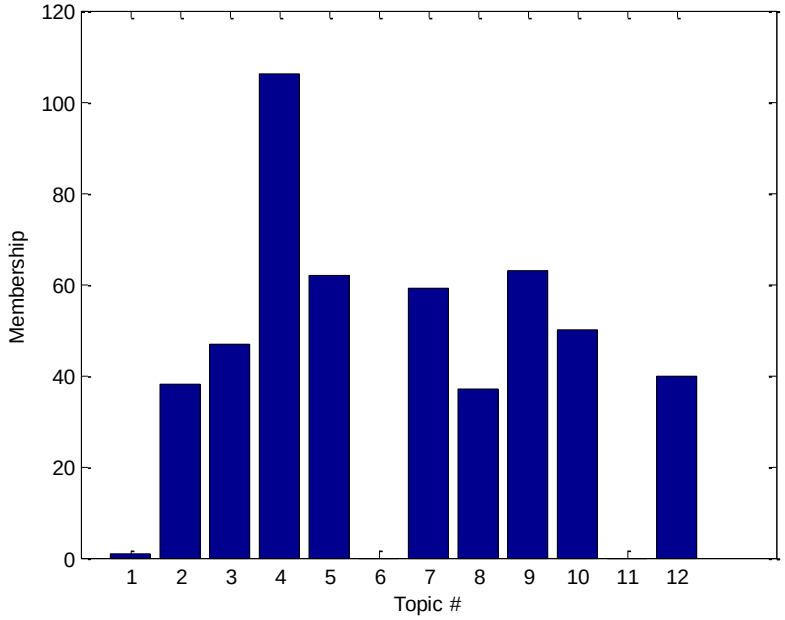

Figure 4-9. Total Topic Membership Count.

The analysis described in section 3.2 was performed on the remaining 9 topics. Figures 4-10 and 4-11 show the resulting parameters for Topic 3 and Topic 7 respectively. These two topics contain the fault codes which are associated with the fault mode of interest for the actuators, identified by expert knowledge. However, through the topic model analysis about 30 other fault codes were also identified as similar to the ones identified by expert knowledge, and these additional fault codes may provide extra information about the degradation of the actuators.

Additional figures for the other topics can be found in Appendix B.2. It can be seen that several of these topics experience large spikes in the data at various points in their lifetime. Although forensic information was not available for any of the systems that these actuators were attached to, it can be inferred that these spikes indicate a fault within the system that does not apply to the actuator. If expert knowledge was not available and it was not known that Topic 3 and Topic 7 correspond to the fault codes which are relevant to degradation, the other topics could be eliminated by correlating them with faults that did not occur in the system in question. In other words, a sharp increase in a topic parameter near a fault that did not originate in the actuator is a sign that the topic does not apply to the actuator.

Several interesting features may be noted about these figures. First, actuator 4 again behaves differently than the other actuators. For topic 3, the values of actuator 4 are much lower than the other actuators, and for topic 7, the values are much higher than the other actuators. Although unfortunately forensic information was not available to confirm how each actuator failed, it is possible that topic 3 and topic 7 related to different failure modes of the actuator, and actuator 4 is experiencing a different fault mode than the other actuators. For Topic 3, actuators 1, 2, and 3 have large spikes in the parameter near the middle of their lifetimes, and then quickly return to normal values. It is likely that there was an unrecorded maintenance action during this time period which allowed the actuators to return to normal operation. Topic 3 appears to correspond to a fault mode for which the maintenance action is replacement, since the values so quickly return to normal. In contrast, Topic 7 appears to correspond to a fault mode in which degradation

is more gradual. For Topic 7, none of the actuators have significant spikes in the data. There is an upward trend in actuator 4 for Topic 7, but no other actuator for this topic has clear upward trends.

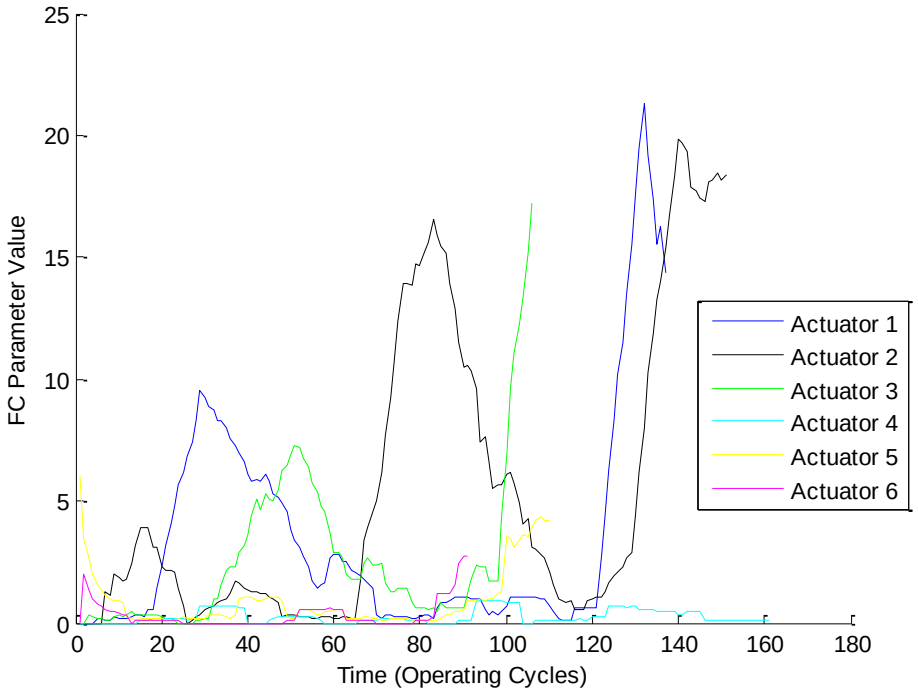

Figure 4-10. Topic 3.

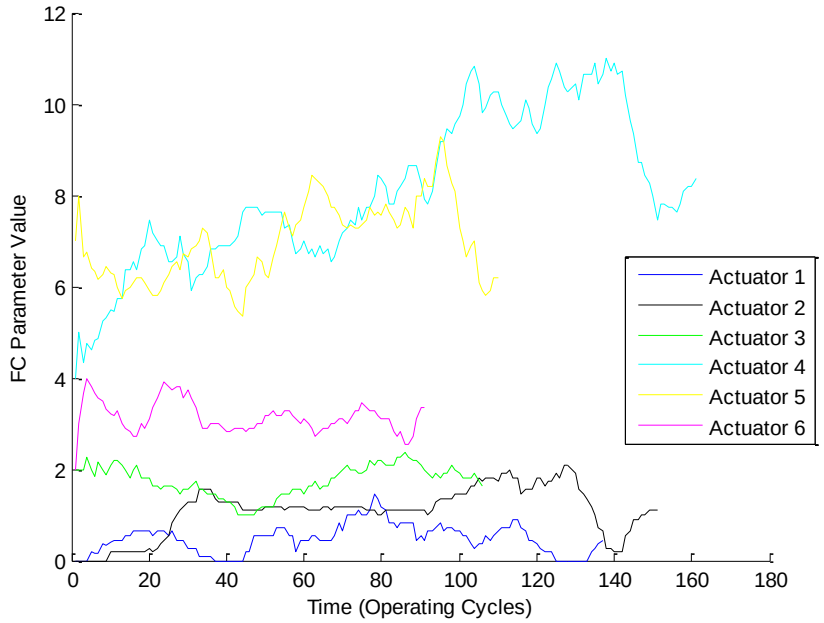

Figure 4-11. Topic 7.

Next, the integration process described in section 4.1.4 was employed for topics 3 and 7. The monitored signals from the data were used as the input prognostic parameter to *intfc*, and the fault codes for each topic was used as the other input. Performance metric weights were [1 1 1], giving all three equal importance. Both of these topics were analyzed separately as they appear to be related to different fault modes. Figures 4-12 and 4-13 show the integrated parameter for topics 3 and 7 respectively.

When integrated with the monitored signal values, both of these parameters show strong upward trends, despite the fault code parameters shown in Figure 4-10 and Figure 4-11 being less trendable. This effect occurs because of several assumptions of the integration process: first, that the parameter should be optimized using the three performance metrics, and second, that degradation in the equipment is occurring at regular rates per test. It is likely that this second assumption may not be true, but fault code information on a time scale such as hours or days was not available, and operating cycles were all that was given.

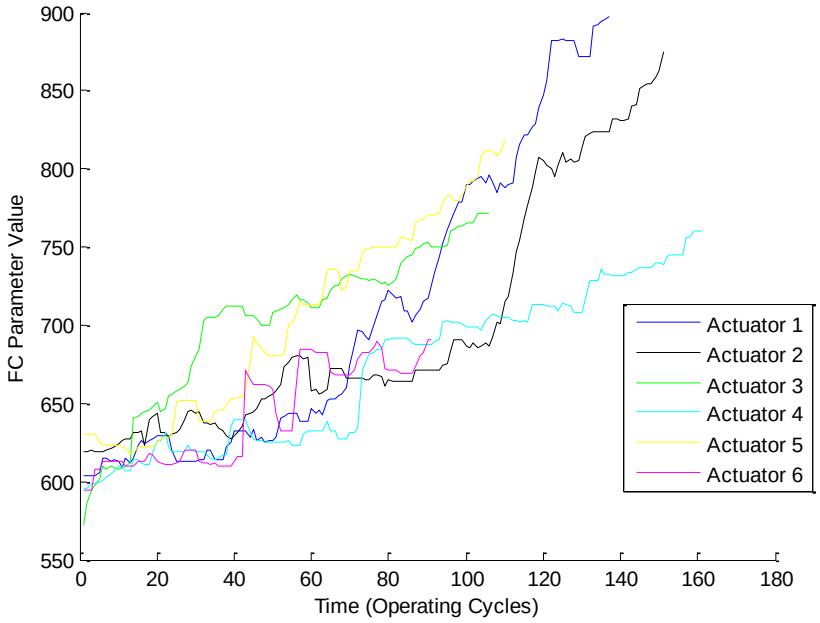

Figure 4-12. Topic 3 Integrated Parameter.

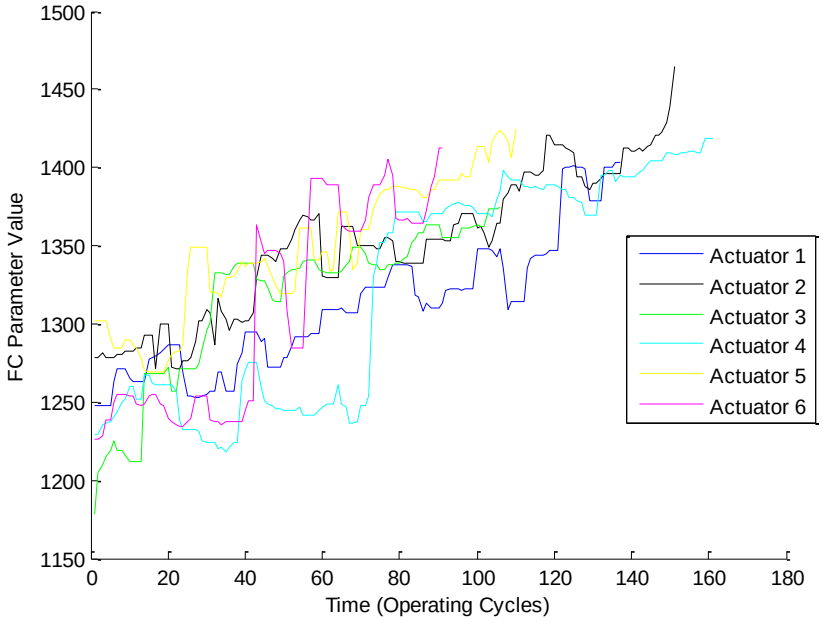

Figure 4-13. Topic 7 Integrated Parameter.

The performance metric results from the integrated parameters are shown below in Table 4-3. These results may be compared to the performance metric results from section 4.1.4, the integrated parameter using fault codes identified by expert knowledge, which were a monotonicity of 0.92, a trendability of 0.36, and a prognosability of 0.84. The results from Topic 7 are very comparable to the expert knowledge identified fault code parameter, which indicates that topic modeling may be used to identify applicable fault codes even without expert knowledge. Results from topic 7 were generally better than for topic 3, which likely occurred because topic 7 had a more gradual upward trend than topic 3.

Table 4-3. Performance Metric Results from Integrated Topics 3 and 7.

	Monotonicity	**Trendability**	**Prognosability**
Topic 3	1.00	0.70	0.68
Topic 7	0.85	0.66	0.84

Overall, the results in this section demonstrate that separation of fault modes using topic modeling of fault codes may be possible. Corroboration of these results could be accomplished using forensics of the failed actuators to determine the fault mode that was experienced by each actuator, but since this information was not available, this methodology may be validated in future work with fault codes. Since dimension reduction was performed using this data set, the improvement in the performance metrics using topic models is at least partially due to noise reduction. The favorable results in this section likely indicate that dimension reduction using topic models is feasible.

4.1.6 Summary of Performance Metric Results

A summary of the results from Tables 4-1 through 4-3, along with additional performance metric calculations from parameters which were not directly mentioned, can be found in Table 4-4. Two types of datasets were used, either the fault codes identified by expert knowledge as related to the actuator failure, or the entire fault code data set. Methods which either used or did not use optimization were executed in order to discover how the optimization process changes the results of the performance metrics.

The best results appear to come from the Topic 7 integrated model and the expert knowledge integrated model. The cumulative sum models, especially for the methodology where all fault codes are used, also achieved very high values, but this is likely mainly due to the way that the cumulative sum methodology is constructed, as it will inherently yield very large montonicity and trendability values.

4.1.7 Remaining Useful Life Estimation

In this section, the remaining useful life of the actuators was estimated using a General Path Model, which was described in section 2.1.3. Each of the fault code parameter methods described in the previous sections of 4.1 were employed, including: fault codes alone using the three construction methods, parameter optimization using the three construction methods, integration with the monitored signals, and dimension reduction and fault mode isolation using topic models. A leave-one-out method was used, where each model was built using 5 training cases and tested on the sixth actuator. The data from the sixth actuator was clipped at 30% of its total lifetime. For each parameter set, 6 estimations of remaining useful life will be made. Table 4-5 shows the error as a percentage of the average number of operating cycles for the data received when the estimated RUL is compared to the actual RUL of the actuator for all parameter construction methods. Since the data received was not the entire lifetime of the component, the error in the table will not be in terms of a percent of lifetime. The error as percent of lifetime is likely much lower than the results indicate in this table.

Table 4-4. Summary of Performance Metric Results.

Fault Codes	Method	Optimized	M	T	P	Total
Expert Knowledge	Cumulative Sum	no	1.00	0.79	0.55	2.34
Expert Knowledge	Cumulative Mean	no	0.72	0.50	0.65	1.87
Expert Knowledge	Windowed Mean, 5	no	0.46	0.31	0.68	1.45
Expert Knowledge	Windowed Mean, 10	no	0.66	0.48	0.67	1.81
Expert Knowledge	Cumulative Sum	yes	1.00	0.82	0.58	2.40
Expert Knowledge	Cumulative Mean	yes	0.75	0.54	0.66	1.95
Expert Knowledge	Windowed Mean, 5	yes	0.59	0.42	0.78	1.79
Expert Knowledge	Windowed Mean, 10	yes	0.70	0.48	0.77	1.95
All	Cumulative Sum	no	1.00	0.91	0.82	2.73
All	Cumulative Mean	no	0.35	0.78	0.42	1.55
All	Windowed Mean, 5	no	0.28	0.10	0.60	0.98
All	Windowed Mean, 10	no	0.20	0.29	0.67	1.16
All	Cumulative Sum	yes	1.00	0.91	0.83	2.74
All	Cumulative Mean	yes	0.41	0.79	0.43	1.63
All	Windowed Mean, 5	yes	0.19	0.62	0.64	1.45
All	Windowed Mean, 10	yes	0.20	0.68	0.66	1.54
Expert Knowledge	Integrated	yes	0.92	0.36	0.84	2.12
All	Integrated	yes	0.81	0.45	0.81	2.07
Topic 3	Windowed Mean, 10	yes	0.28	0.26	0.41	0.95
Topic 7	Windowed Mean, 10	yes	0.78	0.57	0.83	2.18
Topic 3	Integrated	yes	1.00	0.70	0.68	2.38
Topic 7	Integrated	yes	0.85	0.66	0.84	2.35

The number of over-predictions, shown in the last column, is important because RUL estimations should be conservative enough to under-predict the actual failure time of a component. Ideally, maintenance should be performed right before the component actually fails, and if the failure time is over-predicted, the component will fail before maintenance can be performed. Most of these results do not over-predict a majority of the time, and the two best results in the table do not over-predict at all.

Overall, lack of information hindered RUL estimation. Baseline values for the parameters could not be obtained, which impeded optimization in regards to the prognosability metric. Furthermore, the actual lifetime of these actuators was not known, since the data received began in the middle of the actuators' lifetimes. It is likely that some of these actuators experienced unrecorded maintenance actions, which might have greatly affected the parameter estimation of the degradation of the actuators. Also, without forensic information from the actuators, it is possible that some of them should actually have been classified as NFF, which would hinder GPM model construction. Finally, it is unclear the time period of the operating cycles by which the actuators were measured, and hence it is impossible to tell how these results might translate into time units such as days.

According to Table 4-5, the best results come from the Topic 7 fault codes integrated with the monitored signal, which was a mean absolute error of 14.8 operating cycles. Since Topic 7 had the best results from the performance metrics and had the most gradual slope of all the topics, it is likely the best prognostic parameter to use in this case. Although the results from this section cannot be converted into a time unit such as days, the relative improvements of one method over another can be seen from Table 4-5. It appears that topic modeling is a useful method for analyzing fault code data, especially when integrated with monitored signals that relate to the degradation of the component.

Table 4-5. Summary of GPM Results.

Fault Codes	Method	Optimized	MAE (% Data)	# Overpredict
Expert Knowledge	Cumulative Sum	no	27.96	1
Expert Knowledge	Cumulative Mean	no	27.49	0
Expert Knowledge	Windowed Mean, 5	no	21.09	1
Expert Knowledge	Windowed Mean, 10	no	17.75	0
Expert Knowledge	Cumulative Sum	yes	27.84	1
Expert Knowledge	Cumulative Mean	yes	27.45	0
Expert Knowledge	Windowed Mean, 5	yes	18.05	2
Expert Knowledge	Windowed Mean, 10	yes	14.24	2
All	Cumulative Sum	no	24.85	1
All	Cumulative Mean	no	22.90	1
All	Windowed Mean, 5	no	22.43	2
All	Windowed Mean, 10	no	21.56	1
All	Cumulative Sum	yes	24.03	2
All	Cumulative Mean	yes	22.38	3
All	Windowed Mean, 5	yes	19.74	1
All	Windowed Mean, 10	yes	18.23	1
Expert Knowledge	Integrated	yes	13.72	1
All	Integrated	yes	16.58	2
Topic 3	Windowed Mean, 10	yes	18.35	1
Topic 7	Windowed Mean, 10	yes	7.45	0
Topic 3	Integrated	yes	11.12	0
Topic 7	Integrated	yes	6.41	0

4.1.8 No Fault Found Cases

The analysis in the previous sections has assumed that out of the 20 actuators whose data were received, 14 were NFF and 6 were actually experiencing degradation. Unfortunately, verification of which actuators were NFF could not be performed due to data restrictions. However, the statistics of the fault codes could be used to distinguish which actuators were likely to be experiencing a failure. Several average values for all of the actuators are shown in Table 4-6. "Avg # FC/Test" refers to the average number of all fault codes per test, "Avg # EK FC/Test" refers to the average number of the fault codes identified by expert knowledge as related to the actuator failure per test, "Avg # T3 FC/Test" refers to the average number of Topic 3 (identified in section 4.1.5) fault codes per test, and "Avg # T7 FC/Test" refers to the average number of Topic 7 (identified in section 4.1.5) fault codes per test. All of the values in this table have been multiplied by the same random integer for export control purposes, so the numbers in this table do not necessarily represent actual fault code rates.

In this section, a different numbering scheme has been used than in the previous sections of 4.1, so actuator numbers may not necessarily match up to those which were referred to earlier. However, the 6 actuators which were assumed to have a fault have been bolded so that comparisons can be made between the hypothesized NFF and faulty actuators.

According to Table 4-6, there does not appear to be a difference in the average number of total fault codes between the actuators assumed to be NFF and those assumed to be faulty. This effect likely occurs because much of the entire fault code database applies to systems other than actuator failure, and the failure data received was for actuator replacement only. Hence, the total number of fault codes is likely to be relatively constant across all actuators.

Overall, the 6 actuators which were included in the analysis tend to have higher values of average Expert Knowledge fault codes per test. Since the Expert Knowledge fault codes relate directly to the actuator failure mode that is being investigated, this is a

strong indication that these 6 actuators have experienced more degradation than the other 14 and explains why better results were obtained from these 14. Values for Topic 3 and Topic 7 are also typically higher for the 6 investigated actuators than for the other actuators.

Table 4-6. Average Statistics for Actuators 1-20.

Actuator	Avg # FC/Test	Avg # EK FC/Test	Avg # T3 FC/Test	Avg # T7 FC/Test
1	**14.3**	**1.96**	**2**	**0.33**
2	**16.7**	**3.94**	**0.82**	**1.42**
3	14.0	0.88	0.86	0.58
4	**11.2**	**1.67**	**0.18**	**0.14**
5	13.3	0.56	0.35	0.46
6	9.2	0.19	0.27	0.33
7	14.9	0.23	0.22	0.34
8	10.2	0.14	0.26	0.29
9	11.8	0.28	0.29	0.17
10	10.8	0.18	0.42	0.28
11	11.6	1.65	0.44	0.91
12	12.5	0.25	0.62	0.50
13	9.8	0.50	0.08	0.30
14	**11.2**	**2.96**	**0.78**	**0.99**
15	**11.1**	**3.30**	**0.62**	**1.16**
16	10.3	0.00	0.32	0.61
17	8.8	0.12	0.15	0.31
18	13.9	0.21	0.19	0.46
19	12.7	0.71	0.38	0.81
20	**10.4**	**1.06**	**0.49**	**1.33**

Figure 4-14 shows a parameter constructed with fault codes alone using a windowed mean of 10 for one of the NFF-designated actuators. It can be seen that the parameter values are much lower than those in Figure 4-4, which used the same methodology to construct parameters for the faulty actuators. It is possible that the spikes in the data correspond to unrecorded maintenance actions, but since the values are much lower than those in Figure 4-4, it is more likely that they are simply spurious fault codes in the data. Other cases looked very similar to Figure 4-14, with typical values much lower than those of the 6 faulty actuators.

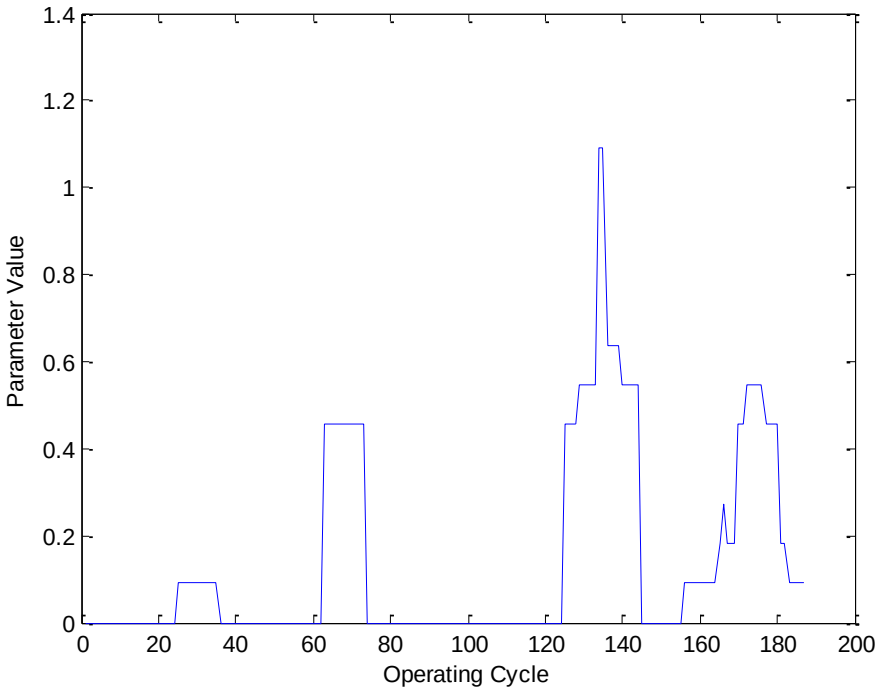

Figure 4-14. NFF Case, Windowed Mean with a Size of 10.

4.1.9 Summary of Results

The analysis performed in this section demonstrated that the methodology developed in Chapter 3 will yield favorable results when applied to a real-world data set. Prognostic parameters were constructed from fault codes alone without any additional monitored signals, although the resulting parameters often had a large spike right before failure and very little trending in the beginning of life. However, since the spikes occurred between 5 and 20 operating cycles before failure, these parameters may still be useful in predicting the RUL of a component. For this dataset, the windowed mean method was determined to be the best of the three fault code parameter construction methods which were described in section 3.2. Furthermore, optimization using gradient descent was able to improve these results.

Integration of the fault codes with a monitored signal related to degradation using the function *intfc* yielded very favorable results, with improved performance metrics over both the monitored signal alone and the fault code parameters alone. The results in section 4.1.4 demonstrated the efficacy of using two sources of information about the degradation of a component in order to construct a prognostic parameter.

Dimension reduction was accomplished for the large fault code database with approximately 8000 fault codes which may or may not be related to degradation of the fault mode being investigated. From the results in section 4.1.5, topic models appear to be a feasible method for fault code analysis. Using topic models, fault codes could be discarded which were not related to the actuator degradation being investigated. Integration of the monitored signals with the topic model parameters was also performed, and the performance metric results showed improvement.

The results in this section were validated using a GPM which estimated the RUL of each of the actuators using a "leave one out" method. Since accurate RUL estimation is what determines the effectiveness of a prognostic model, this section was particularly important in validating the methodology described in this dissertation. The most accurate RUL estimation was from the Topic 7 integrated parameter, which indicates that topic

models may be useful not only for dimension reduction but also for fault mode isolation. It can be theorized, although lack of information does not allow a conclusion, that Topic 7 and Topic 3 refer to different fault modes for the actuators, and Topic 7 was the more prevalent fault among the actuators.

Several difficulties were introduced in this analysis due to the lack of information about the data set. First, no forensic investigation was performed on the actuators which were studied, so fault mode was not able to be determined. With forensic information, different prognostic models may be made for each fault mode, which would improve accuracy. Second, the data was received in the middle of the lifetime of the actuators, which did not allow baseline measurements to be taken. Finally, the number of actuators that was useful in model construction was low due to the potentially high number of NFF actuators present. Additional actuators that were known to be faulty would likely improve results significantly.

4.2 Motor Degradation Data Set

Another data set was also investigated with the methods described in Chapter 3. Since the fault codes from section 4.1 were obtained from a pre-operation test of the actuators, this section will demonstrate the other potential use of fault code analysis, which is fault code generation during operation. The methodology described in section 3.4, which detailed the creation of fault codes using operational data, will be used to convert monitored signals into a fault code matrix which can be analyzed using the fault code analysis methods described in sections 3.2 and 3.3.

4.2.1 Data Set Description

An accelerated degradation experiment was developed for another project, and the data has been repurposed for this research. Data from ten 5-HP motors were collected over a period of about one year. The accelerated degradation procedure was accomplished according to IEEE Standard 117, which recommended temperatures and

methods to degrade the insulation of motors (IEEE 1974). According to the IEEE standard, the degradation was accomplished by two primary methods. First, high temperatures reduce the effectiveness of the insulation inside the motors, and the electrical properties of the motors may degrade enough that electrical failure occurs. For 5-horsepower motors to fail in the estimated time period of the experiment, heating at 180 degrees C was selected. The second form of degradation was corrosion induced by a moisture chamber. The procedure described in the IEEE standard was modified slightly and is given in the paragraph below. The equipment and instrumentation used in the experiment is summarized in Table 4-7.

The ten motors were divided into two testing groups, high stress and low stress, so that comparisons could be made between the two stressing conditions. Each week, the motors were subject to 72 hours of heating in a laboratory-grade oven at either 140 degrees C or 160 degrees C depending on which testing group they were in. After heating, the motors were removed and allowed to air cool for about 6 hours. Next, the motors were quenched in an enclosed shallow water pool for about 15 minutes, which induced moisture degradation, according to the IEEE Standard 117 mentioned above. The motors were then heated again in the ovens for 72 hours at either 140 degrees C or 160 degrees C. Finally, the motors were allowed to air cool for approximately 18 hours before they were placed on the testing bed. This procedure is summarized in Figure 4-15.

Once per week each motor was placed on a testing bed for one hour, and signals that are potentially relevant to the degradation of the motor were monitored. On the testing bed, the motor was coupled with a generator, which was attached to a load bank to dissipate the electricity generated. Data from four transient startups were taken each week, and two seconds of data every fifteen minutes were taken from the steady-state operation of the motors, also resulting in four steady-state measurements per week. Figure 4-16 shows a picture of the test bed for reference. A wire cage was placed over the rotor and coupling for safety considerations.

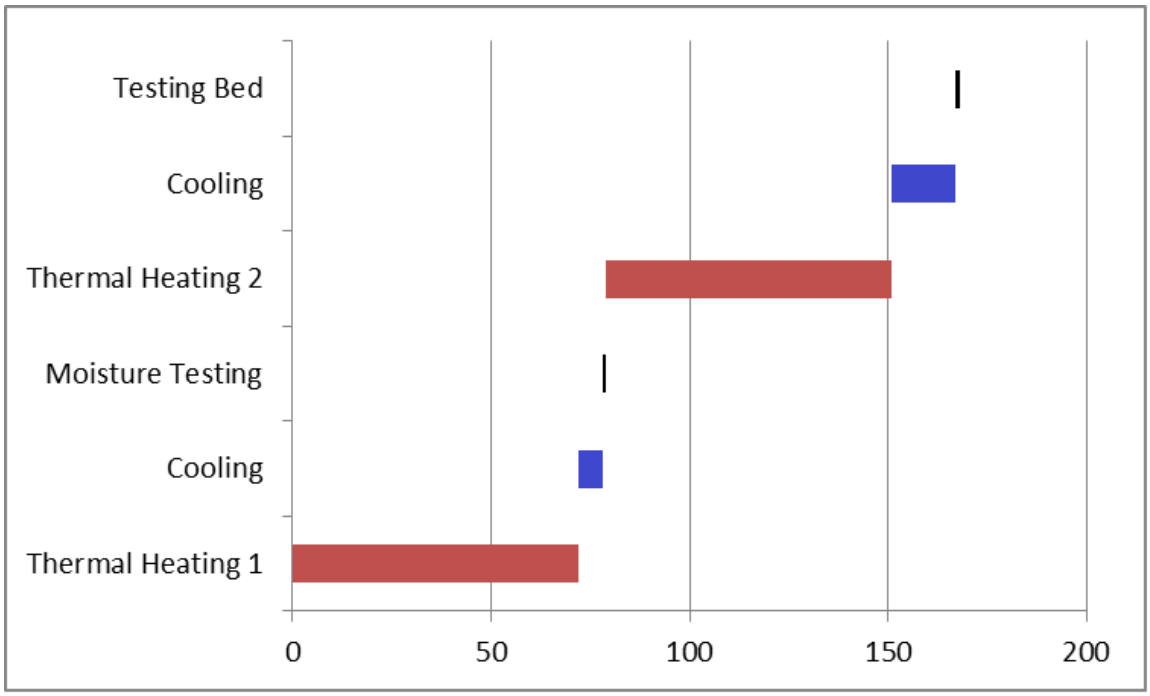

Figure 4-15. Motor Testing Procedure.

Table 4-7. Motor Experiment Equipment.

Equipment	**Manufacturer**	**Model #**
Motor	US Motors	S5P1A
Generator	Winco	TB6000C
Load Bank	Avtron	K490
Oven	Lab Companion	EW-52402-91
Current Clamp	Fluke	i200s
Accelerometer	PCB	ICP 352C18
Acoustic Sensor	PCB	130D20
Tachometer	PCB	LaserTach
Thermocouple	Omega	CO1K

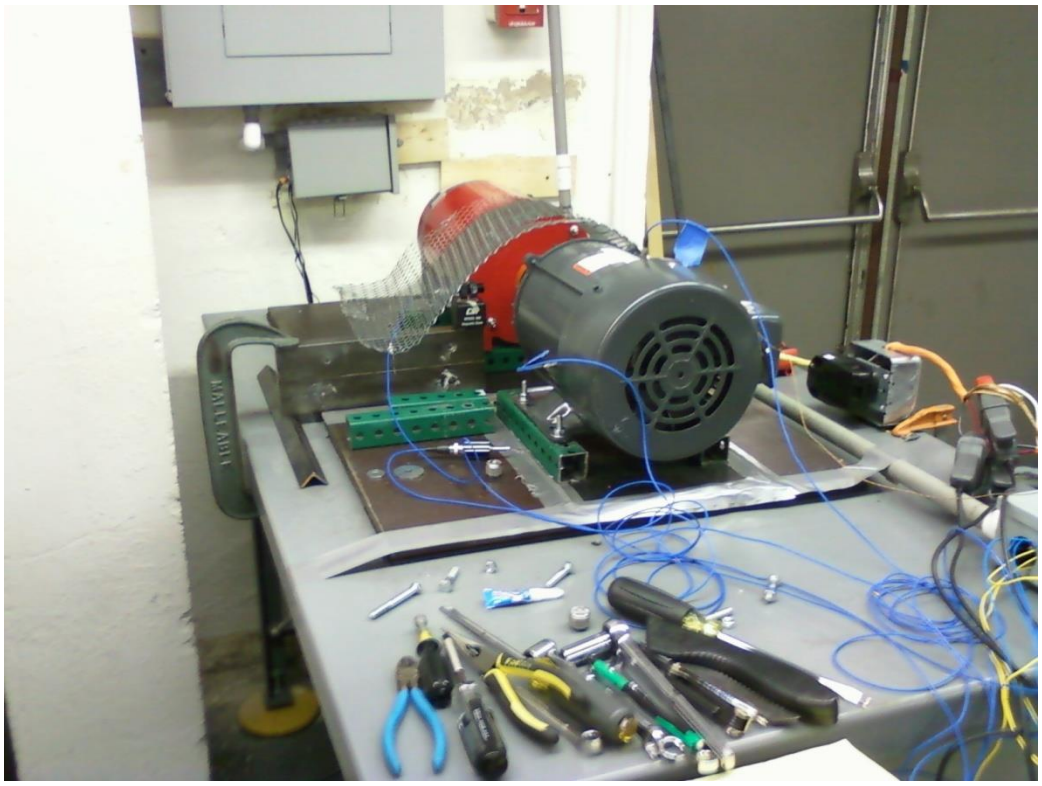

Figure 4-16. Motor Test Bed.

A total of 13 signals was collected during test bed operation: motor current for all three phases, motor voltage for all three phases, horizontal vibration of the motor, vertical vibration of the motor, acoustics from a microphone near the motor, a tachometer signal from the spinning rotor of the motor, a temperature sensor placed inside the motor, and the resulting load bank current and voltage. All of the signals were collected at 10240 Hz so that high-frequency analysis could be performed.

4.2.2 Generation of Fault Codes from Monitored Signals

In this section, the methodology described in section 3.4, which involved the construction of fault codes from operational data, was used to transform monitored signals collected on the test bed into a fault code matrix that may be analyzed using the methods described in this dissertation.

First, an AAKR model was constructed using the motor data. The first test of the motors was considered to be the training, or baseline, values, since no heating had yet been performed in the ovens, and hence no degradation had occurred. Therefore, the first test for each motor should contain no degradation. The testing values for each motor were the subsequent tests after the first one. Degradation should increase over time due to the accelerated degradation testing, so the residuals from the AAKR model will be increasing as well. Nine features were extracted from the motor data to be used in the AAKR model: the three current signals, the two vibration signals, the three voltage signals, and the acoustic signal. These nine features were used to construct a fault code database with nine total fault codes, based on the residuals from the AAKR model.

Figure 4-17 shows the residuals from the AAKR model of the horizontal and vertical vibration of Motor 2. It can be seen from the figure that near the end of the life of the motor the residuals have greater variance. The mean absolute residuals with a window size of 4 tests are shown in Figure 4-18. As expected, the horizontal and vertical vibration are very similar to each other. Overall, except for the progressive decrease in the parameter near the middle of the data, there appears to be an upward trend, but the data is noisy, and it does not appear that this parameter alone would be very useful for prognostics. However, the fault code generation method which will be executed in this section may be able to improve the prognostic parameter somewhat.

A SPRT detection method, described in section 2.4 and involving the estimation of a faulted condition based on deviation from a baseline mean and variance, was used to determine if a fault had occurred in the system during operation. Fault logic was used such that a fault would only be registered if 3 out of the last 5 observations were

considered to have been faulted. Since each test had four runs, the mean number of fault codes generated per week was calculated. Figures 4-19 and 4-20 show the mean number of fault codes per week for the second phase of current and the horizontal vibration, respectively. Each figure has been median filtered with a window size of 3. Additional results from the other seven fault codes may be found in Appendix B.3.

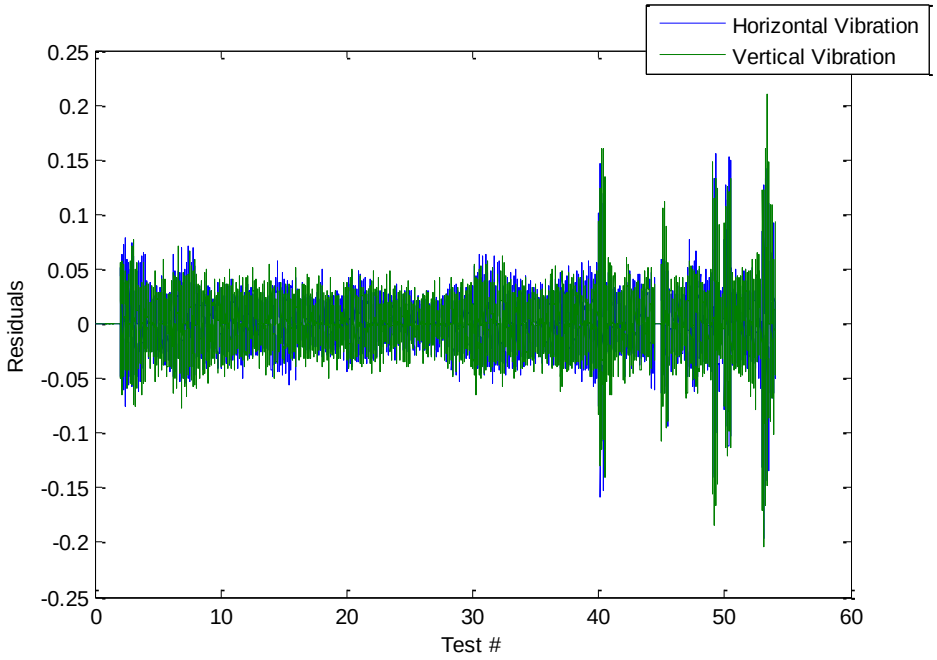

Figure 4-17. Residuals from Motor 2 Vibration.

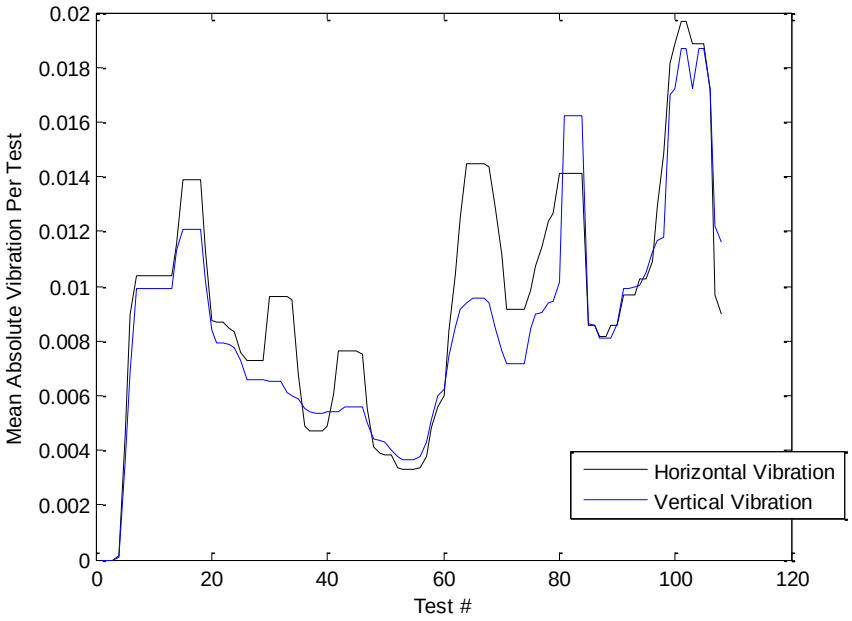

Figure 4-18. Mean Absolute Vibration Per Test, Motor 2.

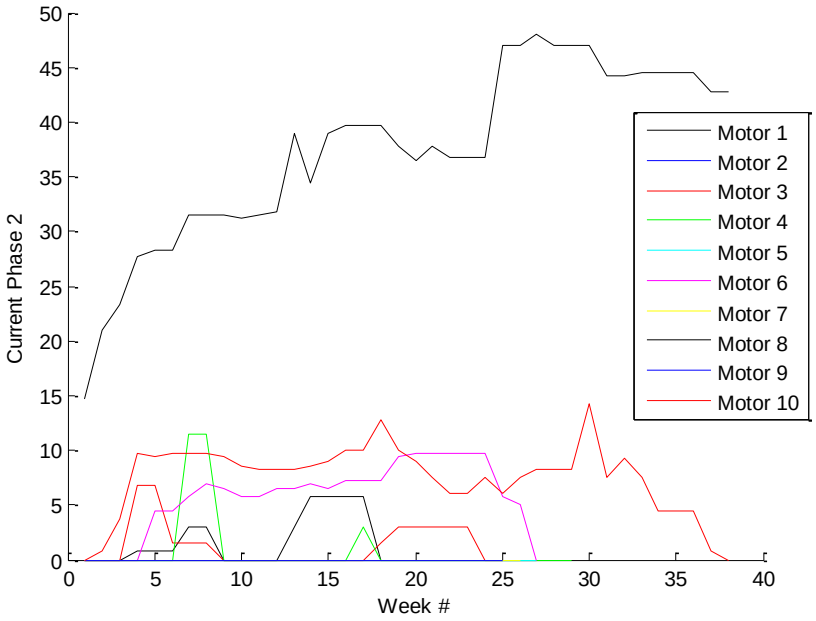

Figure 4-19. Second Phase Current Fault Codes Per Week.

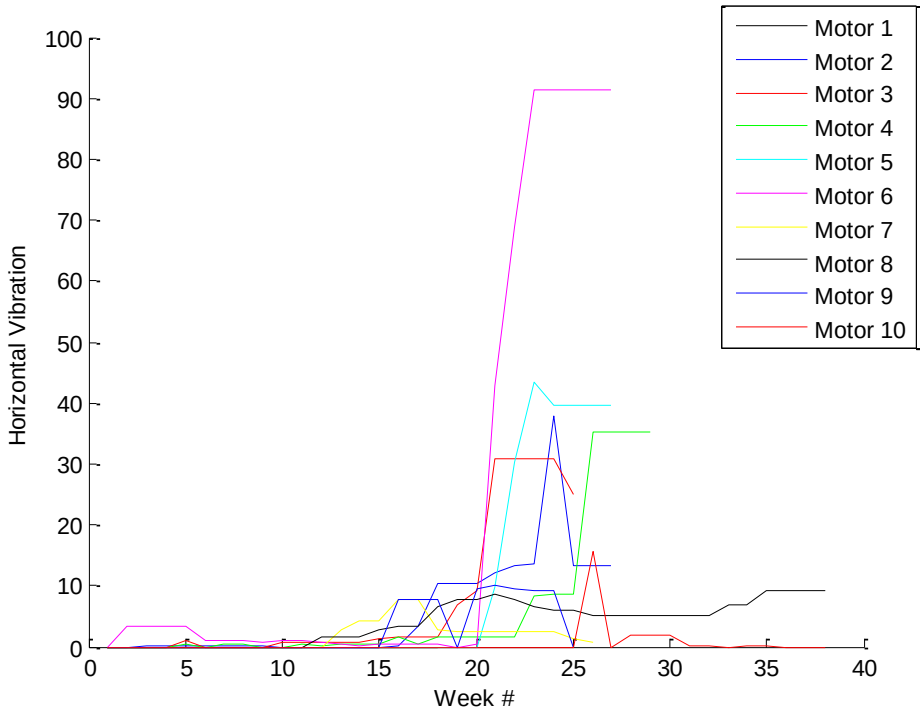

Figure 4-20. Horizontal Vibration Fault Codes Per Week.

These figures show that plotting the fault codes per week alone may yield a viable prognostic parameter, although the data used to construct the parameters in this case was fairly noisy. However, in later sections these fault codes will be optimized to increase the accuracy of the results.

4.2.3 Constructing and Optimizing Prognostic Parameters from Fault Codes

The methodology described in section 3.2 was used to construct a prognostic parameter from the fault codes generated in the previous section. All three methods, including the cumulative sum, cumulative mean, and windowed mean with a window size of both 5 and 10, were applied. Optimization was also used to determine how the resulting parameters might be improved. Figure 4-21 shows the constructed parameter using optimization and a windowed mean with a size of 10. Although the distribution of

values at failure is wider than ideal, overall the parameters appear to be useful in regards to prognostics. Results from the other methods are shown in Appendix B.4. The accuracy of these parameters for RUL estimation will be investigated in the next section.

The performance metric results are shown in Table 4-8. This table shows that the optimized cumulative sum method appears to give the highest results. However, as was seen earlier in this dissertation, the cumulative sum method has the disadvantage of reacting poorly to unrecorded maintenance actions. In this case, it is known that no unrecorded maintenance took place, so the cumulative sum method is valid, but when this information is not known, the cumulative sum and cumulative mean methods should not be used.

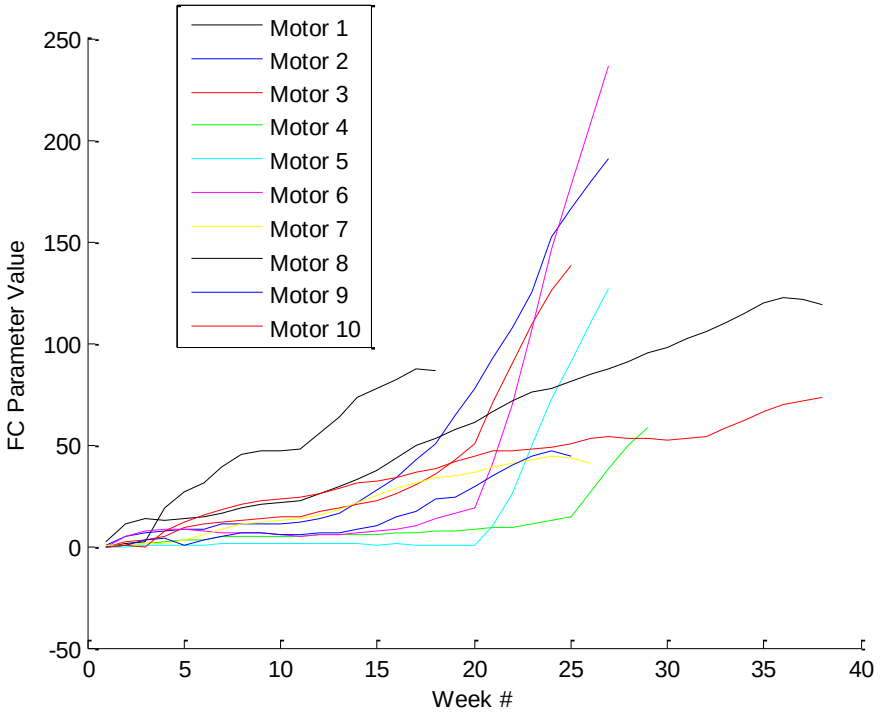

Figure 4-21. Motor Fault Codes, Optimized Windowed Mean with a Window Size of 10.

Table 4-8. Performance Metric Results for Motor Fault Code Parameters.

Method	Optimized	Monotonicity	Trendability	Prognosability	Total
Cumulative Sum	No	1.00	0.39	0.67	2.06
Cumulative Mean	No	0.78	0.35	0.46	1.59
Window Mean, 5	No	0.68	0.32	0.57	1.57
Window Mean, 10	No	0.73	0.30	0.52	1.55
Cumulative Sum	Yes	1.00	0.72	0.68	2.40
Cumulative Mean	Yes	0.79	0.37	0.49	1.65
Window Mean, 5	Yes	0.87	0.28	0.60	1.75
Window Mean, 10	Yes	0.90	0.56	0.59	2.05

Overall, the results from Table 4-8 indicate that fault code generation from operational data may be a viable method when data storage is a concern or raw signals may not be stored for another reason. In the next section, the accuracy of these methods in determining RUL will be investigated.

4.2.4 Remaining Useful Life Estimation

Using the parameters constructed in the previous section, the RUL of each of the motors was calculated. The procedure used in this section was the same as in section 4.1.7. Again, a "leave one out" method was employed where the model was trained using nine of the motors and tested on the tenth motor. The test data was clipped at about 30% of the total life of the motor so that accurate extrapolation of degradation could be accomplished. The average RUL from the ten resulting predictions is shown in Table 4-9. The number of overpredictions is also shown.

Table 4-9. Remaining Useful Life Results for Motor Data.

Method	Optimized	MAE (% Life)	# Overpredicted
Cumulative Sum	no	31.1	5
Cumulative Mean	no	32.3	3
Window Mean, 5	no	25.3	3
Window Mean, 10	no	19.2	1
Cumulative Sum	yes	5.7	0
Cumulative Mean	yes	19.9	2
Window Mean, 5	yes	14.9	2
Window Mean, 10	yes	8.4	3

Looking at the table, the best results are from the cumulative sum model, with a mean absolute error as a percent of life of 5.7% and 0 overpredictions. Ideally, models should not be constructed with the cumulative sum or cumulative mean methods because maintenance will reset the degradation on the component, but in this case, since it was known that the motors did not have any unrecorded maintenance, both of the cumulative models were valid.

4.2.5 Summary of Results

The results in this section indicate that a fault code matrix may be generated from operational data. Typically, it is not advisable to construct fault codes from monitored signals using the methods described in 4.2.2, since raw signals alone will usually give better results than constructed fault codes. Rather, these fault codes are likely already present in operational data and may take the form of maintenance records, operator notes about aberrant signals or strange operational behavior, or operating limits being exceeded causing an alarm during operation.

Using fault codes alone, parameters were constructed using the three methods described in section 3.2: cumulative sum, cumulative mean, and windowed mean. The

best results generally came from the cumulative sum method, since it was known that the motors did not experience any unrecorded maintenance actions. The integration method was not investigated in this section, since all relevant monitored signals were used to construct the fault code database. Since the fault code database was small, topic models were not used for dimension reduction or fault mode isolation.

Despite the data being used for a purpose other than which it was originally intended, results from the GPM were favorable, with a best MAE of 5.7% as a percent of average lifetime and zero overpredictions using the optimized cumulative sum method. The lack of fine resolution in data due to the accelerated degradation may have been a detriment to RUL estimation accuracy.

CHAPTER 5
CONCLUSIONS AND RECOMMENDATION FOR FUTURE WORK

5.1 Conclusions

Condition-based monitoring using prognostics has been proposed as a way to improve safety and reduce maintenance costs by minimizing the unplanned failures of power plant components. Since nuclear power plants generally have high levels of thermal and radiative stress, and during operation many parts of nuclear power plant are difficult to reach for maintenance, prognostics may be an ideal method to ensure that components will not fail during an operating cycle. Furthermore, many components in nuclear power plants are high-value and safety-critical, and correct estimation of their health will allow more efficient maintenance scheduling, which may reduce and optimize operating costs.

Fault codes are sources of information about the condition of a component that are currently being underutilized. Typically, fault codes are used for diagnostic purposes to determine when a fault has occurred in a component and must be immediately serviced. However, less severe fault codes which indicate minor aberrant behavior in a component may be used to determine the condition of a component. In this case, fault codes may indicate a progressive fault within a component as the number of fault codes increase over time. This fault code information may be incorporated into prognostic models such as General Path Models (GPM), and the Remaining Useful Life (RUL) of a component may be estimated. The usefulness of fault code information for RUL estimation may extend to fields as varied as aviation, transportation, and high-value military assets.

To increase the accuracy of RUL estimations for plant components, the following contributions were made:

1. Development of a set of techniques that transforms binary fault code data into a prognostics parameter which is optimized to find ideal weightings for each fault code.

2. Incorporation of fault code information into already-existing prognostic parameters. Gradient descent is used to optimize the weighting for each fault code.
3. Application of topic models as a dimension-reduction and fault mode isolation technique for large fault code databases.
4. Development of MATLAB functions that can be used to apply these techniques to real-world or simulated data.

Several methods for transforming fault codes into prognostic parameters were developed in this dissertation, including: cumulative sum, cumulative mean, and windowed mean. A MATLAB function called *fc2pp* was developed to automate these methods for a generalized data set. Typically, the cumulative sum and cumulative mean methods are to be avoided except in special cases where no unrecorded maintenance actions occur and the amount of degradation reduction per maintenance action is known. In most cases, a windowed mean with an appropriately chosen window size will give the best results.

Optimization was used to improve the results of the fault code parameter construction methods. By weighting each fault code based on performance metric results and how relevant the fault code was to degradation, the RUL estimation accuracy of each construction method may be improved. The fitness function of the optimization method was a linear combination of the three performance metrics, which were monotonicity, trendability, and prognosability (Coble 2009).

A method of incorporating fault codes into an existing prognostic parameter was also developed in this dissertation. A function called *intfc* was created which took both monitored signals and fault codes as inputs and merged them. The merging process was accomplished in three primary steps. First, the function scaled the data appropriately so that fault codes and monitored signals were given roughly equal important. Second, the fault codes which were relevant to degradation were determined and all other fault codes were discarded. This step decreased the average running time of the function. Finally,

weights for each fault code column were calculated using an optimization procedure with a fitness function that was a linear combination of the three performance metrics.

The applicability of topic modeling to fault code analysis was explored. Topic models may be used to reduce the dimensionality of a large fault code database, especially when it is suspected that some of the fault codes are not applicable to the fault mode that is being investigated. A methodology to use topic models in order to isolate fault modes was also discussed, although lack of information about fault modes from real-world data sets used in the analysis did not allow verification of the proposed methods.

The fault code analysis proposed in this dissertation was validated with two real-world data sets. First, a data set from actuator failures was analyzed, which contained two fault code matrices which were used as inputs, one with about 8000 fault codes that were not necessarily applicable to actuator failure, and one with the roughly one percent of this fault code database which were identified by expert knowledge as related to actuator failure. Using the construction methods of cumulative sum, cumulative mean, and windowed mean, prognostic parameters were created using the fault code data alone. Results from this section indicated that the data would be useful for diagnostic purposes, as the parameter spiked sharply before failure but did not trend well near the beginning and middle of life.

The integration method described in section 3.3, which incorporated information from both fault codes and a monitored signal, was tested on the actuator data set, since monitored signals from the actuators that related to degradation were available. Overall, the prognostic parameter resulting from the integration process had improved performance metric results over both the fault code parameter and the monitored signals alone. Topic modeling was used to analyze the large fault code database from the actuators as well, and results indicate that topic models may be able to distinguish between fault modes, although verification of this was not possible due to lack of information about the fault modes of the failed actuators.

In order to validate whether or not these methods could be used to accurately predict the RUL of a component, a GPM was created for each of the methods described above. The best results were obtained from a model built using the data from Topic 7 with an optimized windowed mean model. The mean absolute error from these results was 6.41% as a percent of the average lifetime of the data received, which is considered to be a good prediction due to the original noise within the data. Results from the other models were generally favorable as well.

Another data set from an accelerated degradation experiment using 5 horsepower motors was also used to validate the methods described in this dissertation. Monitored signals from the experiment were transformed into fault codes by generating an Auto-Associative Kernel Regression model and Sequential Probability Ratio Test monitoring which resulted in fault alarms. These fault codes were then used to construct a prognostic parameter using the construction methods of cumulative sum, cumulative mean, and windowed mean. Finally, RUL estimations were made for the degraded motors, and the best results were from the optimized cumulative sum method with a mean absolute error of 5.7 percent as a percent of average lifetime of the motors.

The analysis in this dissertation clearly indicated for these two real-world data sets that fault codes analysis may result in higher performance metrics and greater accuracy in RUL prediction. Although future work may be able to determine if topic modeling can perform fault mode isolation, the applicability of topic modeling for fault code analysis was established from these results, as the best results from the actuator data set came from an optimized topic model parameter. Overall, the results from the two real-world data sets indicate that fault code analysis is a viable method for the generation of prognostic parameters and the prediction of RUL.

5.2 Recommendations for Future Work

One of the greatest challenges in data analysis is finding good data. In this case, especially for the actuator data set, a lack of information about the failure modes of the components was a great hindrance. If forensics was performed for all of the components for each of the two real-world data sets, the fault mode isolation effects of topic modeling could be verified, Remaining Useful Life (RUL) estimates for all fault code analysis methods could be improved by creating separate models for each fault mode, and verification could be received that failure actually occurred when hypothesized. The lack of baseline values for the actuator data set hindered the estimation of RUL. In the future, the methodology established in this dissertation could be investigated further with a dataset that has separable and distinctive failure modes and contains a full data from entry into service to failure. With a distinct number of failure modes, the effectiveness of the fault mode isolation from topic models may be investigated more thoroughly. Furthermore, a dataset from nuclear power plant operation would be very beneficial to establish the usefulness of this method to nuclear engineering.

The optimization method developed for this dissertation, contained in the function *intfc*, was ideal for the data sets investigated. The MATLAB code was designed to be as general as possible so that favorable results may be obtained even from data sets that it was not designed for. However, the number of data sets that were available to test the *intfc* code was small, and further development of the *intfc* function may be necessary. Additional construction methods other than cumulative sum, cumulative mean, and windowed mean may also be necessary to include in an updated *fc2pp* function.

Another dimension reduction method which may be applicable to the methods described in this dissertation is Principal Component Analysis (PCA). Although PCA is typically used for continuous data, there have been modifications of the method so that it may be used for binary data as well, such as Binary Principal Component Analysis (Leeuw 2004) and Sparse Logistic Principal Components Analysis for Binary Data (Lee 2010). PCA may be able to reduce the noise in a large fault code database and isolate

fault modes in the same way that topic models were employed. The effectiveness of PCA vs. Topic Models may be investigated in future work.

Finally, one paper found during the literature survey employed three values for a fault: normal, warning, and faulted, instead of the typical binary faulted/unfaulted state (Yoo 2008). With proper data collected which employs an indicator of these three states, the methodology described in this dissertation may be extended. For instance, a potential way of interpreting these three states may be that a “normal” label has a value of 0, a “warning” label has a value of 0.5, and a “faulted” label has a value of 1.

LIST OF REFERENCES

Bardell, P.H., Jr.; Mcanney, W.H., "Built-in test for RAMs," *Design & Test of Computers, IEEE* , vol.5, no.4, pp.29,36, Aug. 1988

Bart, E., Welling, M., Perona, P.; "Unsupervised organization of image collections: taxonomies and beyond." *Trans. Pattern Recognit. Mach. Intell.* (2010)

Blei, D. M., Ng, A. Y., & Jordan, M. I. "Latent Dirichlet Allocation". *Journal of Machine Learning Research*, 993-1022. (2003)

Bond, L.J.; Ramuhalli, P.; Tawfik, M.S.; Lybeck, N.J., "Prognostics and life beyond 60 years for nuclear power plants," *Prognostics and Health Management (PHM), 2011 IEEE Conference*, pp.1,7, 20-23 June 2011

Boskoski, P.; Gasperin, M.; Petelin, D., "Bearing fault prognostics based on signal complexity and Gaussian process models," *Prognostics and Health Management (PHM), 2012 IEEE Conference on*, pp.1,8, 18-21 June 2012

Buntine, W., Löfström, J. "A Scalable Topic-Based Open Source Search Engine." *Proceedings of the IEEE/WIC/ACM Conference on Web Intelligence*, 228-234. (2004)

Casella, George; George, Edward I. "Explaining the Gibbs sampler". *The American Statistician*. 167–174. (1992)

Chen, He-tao; Hong-jie Yuan, "Reliability assessment based on proportional degradation hazards model," *Industrial Engineering and Engineering Management (IE&EM), 2010 IEEE 17Th International Conference,* pp.958,962, 29-31 Oct. 2010

Coble, Jamie. "Process and Equipment Prognostics [PEP] Toolbox." University of Tennessee. 2010

Coble, J.B.; Hines, J.W., "Prognostic algorithm categorization with PHM Challenge application," *Prognostics and Health Management, 2008. PHM 2008. International Conference on* , vol., no., pp.1,11, 6-9 Oct. 2008

Coble, J.B., and J.W. Hines, "Identifying Optimal Prognostic Parameters from Data: A Genetic Algorithms Approach," *Annual Conference of the Prognostics and Health Management Society 2009*

Coble, Jamie. *Merging Data Sources to Predict Remaining Useful Life- An Automated Method to Identify Prognostic Parameters*. Diss. University of Tennessee, 2010. Print.

Cox, David, "Regression Models and Life-Tables". *Journal of the Royal Statistical Society*, pp. 187–220, 1972

Diaz, I.; Hollmen, J., "Residual generation and visualization for understanding novel process conditions," *Neural Networks, 2002. IJCNN '02. Proceedings of the 2002 International Joint Conference*, vol.3, pp.2070,2075, 2002

Doctor, Stephen R. "Measurement Challenges Associated With Irradiated Reactor Components." *Materials Research Society Proceedings* 503 (1997)

Ebeling, C. E., An Introduction to Reliability and Maintainability Engineering, Waveland Press, Illinois USA (2010

Emerson Process Management. *Control Valve Handbook*. http://www.documentation.emersonprocess.com/groups/public/documents/book/cvh99.pdf. 2005

Franklin, M.; Saluja, K.K.; Kinoshita, K., "Row/column pattern sensitive fault detection in RAMs via built-in self-test," *Fault-Tolerant Computing, 1989. FTCS-19. Digest of Papers., Nineteenth International Symposium*, pp.36,43, 21-23 June 1989

Garvey, D.R.; John, M.; Baumann, J., "Nonparametric life consumption modeling of high end drilling tools," *Reliability and Maintainability Symposium (RAMS), 2010 Proceedings – Annual*, pp.1,6, 25-28 Jan. 2010

Garvey, Dustin. "Process and Equipment Monitoring [PEM] Toolbox." University of Tennessee. 2005

Glickman, J.F., "Common organizational level tester [military aircraft testing]," *AUTOTESTCON 2003. IEEE Systems Readiness Technology Conference. Proceedings*, pp.40,42, 22-25 Sept. 2003

Griffiths, T.L., Steyvers, M., & Tenenbaum, J.B.T. (2007). Topics in Semantic Representation. *Psychological Review*, 114(2), 211-244.

Guo, Peng, and Nan Bai. "Wind Turbine Gearbox Condition Monitoring with AAKR and Moving Window Statistic Methods." *Energies* (2011)

Haupt, Randy. *Practical Genetic Algorithms*. 2nd ed. N.p.: n.p., 2004. Print.

Hines, J. W., and D. Garvey. "Development and Application of Fault Detectability Performance Metrics for Instrument Calibration Verification and Anomaly Detection."*Journal of Pattern Recognition Research* (2006): 2-15.

Hines, J.W., and Usynin, Alexander. “Current Computational Trends in Equipment Prognostics.” *International Journal of Computational Intelligence Systems*: vol. 1, no. 1. 2008.

Ho, M.; Kecheng Shen, "Using a proportional hazards model to determine the effect of covariates on pooled data — A case study," *Reliability, Maintainability and Safety (ICRMS), 2011 9th International Conference*, pp.1370,1375, 12-15 June 2011

Hofmeister, J.P.; Vohnout, S.; Mitchell, C.; Heimes, F.O.; Saha, S., "HALT evaluation of SJ BIST technology for electronic prognostics," *AUTOTESTCON, 2010 IEEE* , pp.1,7, 13-16 Sept. 2010

Huang, Tingting; Tongmin Jiang, "An extended proportional hazards-proportional odds model in accelerated life testing," *Reliability, Maintainability and Safety, 2009. ICRMS 2009. 8th International Conference*, pp.1173,1176, 20-24 July 2009

"IEEE Standard Test Procedure for Evaluation of Systems of Insulating Materials for Random-Wound AC Electric Machinery," ANSI C50.32-1976 and IEEE Std 117-1974 (Reaffirmed 1984) (Revision of IEEE Std 117-1956), pp.i-22, 1974

Kirkland, L.V.; Pombo, Tony; Nelson, K.; Berghout, F., "Avionics health management: searching for the prognostics grail," *Aerospace Conference, 2004. Proceedings. 2004 IEEE*, vol.5, pp.3448,3454 Vol.5, 6-13 March 2004

Kumar, D.; Westberg, U., "Proportional hazards modeling of time-dependent covariates using linear regression: a case study [mine power cable reliability]," *Reliability, IEEE Transactions*, vol.45, no.3, pp.386,392, Sep 1996

Lee, Seokho, “Sparse Logistic Principal Components Analysis for Binary Data,” Ann Appl Stat, 2010

Leeuw, Jan de, “Principal Component Analysis of Binary Data by Iterated Singular Value Decomposition.” Elsevier Science. June 2004.

Lu, C.J., and W. Meeker, “Using Degradation Measures to Estimate a Time-to-Failure Distribution,” *Technometrics* 35 (2) 1993: 161 – 174.

Lybeck, N.; Marble, S.; Morton, B.; , "Validating Prognostic Algorithms: A Case Study Using Comprehensive Bearing Fault Data," Aerospace Conference, 2007 IEEE , pp.1-9, 3-10 March 2007

Ma, XiangFu; Wenhu Huang; JiaZhong Zhang, "Built-In Test System for Submarine-Launched Missile Model," *Pervasive Computing Signal Processing and Applications (PCSPA), 2010 First International Conference on* , pp.1268,1272, 17-19 Sept. 2010

Meeker, W.Q., L.A. Escobar, and C.J. Lu, “Accelerated degradation tests: modeling and analysis,” *Technometrics*, vol. 40, no. 2, pp. 89-99, 1998.

Mitchell, M.E., "Lessons learned using boundary scan and built-in test for integration and diagnostic test of the U.S. Navy joint standoff weapon," *AUTOTESTCON '95. Systems Readiness: Test Technology for the 21st Century. Conference Record*, pp.151,159, 8-10 Aug. 1995

Mitchell, Melanie. An Introduction to Genetic Algorithms (Complex Adaptive Systems). 3rd ed. N.p.: n.p., 1998. Print.

Mukherjee, N.; Chakraborty, T.J.; Karri, R, "Built in self test: a complete test solution for telecommunication systems," *Communications Magazine, IEEE,* vol.37, no.6, pp.72,78, Jun 1999

Prasad, P. V N; Rao, K. R M, "Failure analysis and replacement strategies of distribution transformers using proportional hazard modeling," *Reliability and Maintainability Symposium, 2003,* pp.523,527, 2003

Pritchard, J. K., Stephens, M., & Donnelly, P. “Inference of population structure using multilocus genotype data”. *Genetics*, 155, 945-955. (2000)

Ramuhalli, P.; Coble, J.; Meyer, R.M.; Bond, L.J., "Prognostics health management and life beyond 60 for nuclear power plants," *Future of Instrumentation International Workshop (FIIW),* 2012, pp.1,4, 8-9 Oct. 2012

Randall, Robert B. *Vibration-based Condition Monitoring*. N.p.: Wiley, 2011. Print.

Rastogi, U.; Jain, V.K.; Srinivas, G.; Guptan, R., "Databases and their analysis for applications in quantitative risk analysis of NPP," *Reliability, Safety and Hazard (ICRESH), 2010 2nd International Conference,* pp.176,180, 14-16 Dec. 2010

Shah, M.D., "Fault detection and diagnosis in nuclear power plant — A brief introduction," *Engineering (NUiCONE), 2011 Nirma University International Conference,* pp.1,5, 8-10 Dec. 2011

Shawe-Taylor, John. *Kernel Methods for Pattern Analysis*. N.p.: n.p., 2004. Print.

Smith, H.R.; Wiedenbrug, E.; Lind, M.; "Rotating Element Bearing Diagnostics in a Nuclear Power Plant: Comparing Vibration and Torque Techniques," *Diagnostics for*

Electric Machines, Power Electronics and Drives, 2007 IEEE International Symposium, pp.17-22, 6-8 Sept. 2007

Steinmetz, M.J., "Built-in-test instrumentation and 21 rules of thumb," *Instrumentation & Measurement Magazine, IEEE* , vol.5, no.3, pp.30,38, Sep 2002

Steyvers, Mark; Griffiths, Tom, "MATLAB Topic Modeling Toolbox 1.4." http://psiexp.ss.uci.edu/research/programs_data/toolbox.htm#References. December 22, 2013.

Sun, Mingming; Fangyi Wan; Zeng Qin, "Health monitoring for propagating crack faults," *Prognostics and System Health Management (PHM), 2012 IEEE Conference*, pp.1,6, 23-25 May 2012

Sun, Jianzhong; Shunfeng Cheng; Pecht, M., "Prognostics of Multilayer Ceramic Capacitors Via the Parameter Residuals," *Device and Materials Reliability, IEEE Transactions on* , vol.12, no.1, pp.49,57, March 2012

Takeda, Hiroyuki. *Locally Adaptive Kernel Regression Methods for Multi-dimensional Signal Processing*. N.p.: n.p., n.d. 2011. Print.

Upadhyaya, B.R., M. Naghedolfeizi, and B. Raychaudhuri, "Residual Life Estimation of Plant Components," *P/PM Technology*, June, 1994: 22 – 29

Vessey, J. P.; Williams, T. H., "Built-in test and the oceanographic sensor," *OCEANS '94. 'Oceans Engineering for Today's Technology and Tomorrow's Preservation.' Proceedings*, vol.2, no., pp.II/368,II/371 vol.2, 13-16 Sep 1994

Voyiatzis, I.; Paschalis, A.; Gizopoulos, D.; Kranitis, N.; Halatsis, C., "A concurrent built-in self-test architecture based on a self-testing RAM," *Reliability, IEEE Transactions on* , vol.54, no.1, pp.69,78, March 2005

Wagner, L. Y. "Electric Motor Failure Modes." *Philips Electric*. N.p., n.d. Web. 20 May 2013. <http://www.phillipselectric.com/pdf/techtalk1.pdf>

Wald, Abraham. "Sequential Tests of Statistical Hypotheses". *Annals of Mathematical Statistics*, pp. 117–186, June 1945

Wang, Xiaogang; Grimson, Eric. "Spatial Latent Dirichlet Allocation" . Proceedings of Neural Information Processing Systems Conference (NIPS). (2007).

Wheeler, Kevin R., Tolga Kurtoglu, and Scott D. Poll. "A Survey of Health Management User Objectives in Aerospace Systems Related to Diagnostic and Prognostic Metrics." *International Journal of Prognostics and Health Management* 2010.01 (2010)

Xiao, Shan Qing; Pan Meng Chun; Weng Fei Bing, "Application of BIT design for electric vehicle (EV)," *Power Electronics and Drive Systems, 1999. PEDS '99. Proceedings of the IEEE 1999 International Conference*, vol.1, pp.516,518, 1999

Xu, Jiuping; Xu, Lei, "Health management based on fusion prognostics for avionics systems," *Systems Engineering and Electronics, Journal of*, vol.22, no.3, pp.428,436, June 2011

Yoo, Changsun; Youngshin Kang; Bumjin Park, "Hardware-in-the-loop test for fault diagnosis system of tilt rotor UAV," *Control, Automation and Systems, 2008. ICCAS 2008. International Conference*, pp.320,323, 14-17 Oct. 2008

Zhang, Xiaodong; Xu, R.; Chiman Kwan; Liang, S.Y.; Qiulin Xie; Haynes, L., "An integrated approach to bearing fault diagnostics and prognostics," *American Control Conference, 2005,* pp.2750,2755 vol. 4, 8-10 June 2005

Zhenhua, Wen; Liu YuanPeng, "Applications of Prognostics and Health Management in aviation industry," *Prognostics and System Health Management Conference (PHM-Shenzhen), 2011*, pp.1,5, 24-25 May 2011

APPENDICES

APPENDIX A- MATLAB CODE

A.1 Fault Code Simulation

```
rel=3.5*[0.1,0.01,0.15,0.05,0.06,0.1,0.09,0.12,0.04,0.2];
ft=[111,91,85,126,107,100,99,94,87,105];

for ii=1:10
    dp{ii}(1)=0;
    for jj=2:ft(ii)
        if ii==7&&jj==50;dp{ii}(jj)=0;dp{ii}(jj-1)=0;end
        dp{ii}(jj)=dp{ii}(jj-1)+1.5+randn*2;end;end

for ii=1:10
    fd(ii)=dp{ii}(end);end

for ii=1:10
    for jj=1:ft(ii)
        for kk=1:10
            faultcodes{ii}(jj,kk)=(rand>0.2)*(rand<(dp{ii}(jj)/fd(ii)-
rand*rel(kk)));
        end;end;end

clear fd ft ii jj kk rel

colorstring=['bkrcymgbkr'];

close all

for ii=1:10
    eval(['plot(1:111,' num2str(ii) '*faultcodes{1}(1:111,' num2str(ii)
'),''' '*' colorstring(ii) ''')'])
    hold on
end

xlabel('Time Step')
ylabel('Fault Code')
```

A.2 fc2pp- Fault Code to Prognostic Parameter Function

```
function [PP model] = fc2pp(freqmat,varargin)

%fc2pp - Fault Code To Prognostic Parameter
%
%Written by:   Eric Strong
%Last Updated: 01-10-14
%
%This code will transform a fault code frequency count matrix
%into a prognostic parameter.
%
%DESCRIPTION OF VARIABLES-
%freqmat = a frequency of fault code count matrix of size m x n
%          where m is the time step (such as days) and n is the
%          fault code ID with number of fault codes occurring.
%          ex. matrix- [0 0 1;
%                       1 1 1;
%                       0 0 3]
%          In the first day, there is one fault in the
%          component, fault code # 3. On the next day, there is
%          1 fault for each fault code. On the final day, there
%          are 3 faults from fault code #3.
%
%OPTIONAL ARGUMENTS:
%'cumsum' = this is a user-passed argument that changes the
%          construction of the prognostic parameter to a cumulative
%          sum of all fault codes. All of the columns of freqmat
%          are summed together, and the cumulative sum of the
%          resulting vector is found.
%'cummean' = this is a user-passed argument that uses the cumulative
%          mean to generate the prognostic parameter. Once again, the
%          columns of freqmat are summed together, and the cumulative
%          mean (as in, the value for day 3 will be the mean of values
%          from days 1-3, and the value for day 10 will the mean of
%          values from 1-10, and so on) will be found.
%'winmean' = this is a user-passed argument that will use a windowed
%          mean to generate the prognostic parameter. After summing
%          the columns of freqmat, the windowed mean will be found.
%          This argument must be followed by the window size, so that
%          the function call will look like fc2pp(freqmat,'winmean',5)
%'reduce'  = this argument will reduce the maximum number of fault
%          codes per time step to 1. For instance, if 5 faults from
%          fault code # 3 occurred during week 2, this argument will
%          reduce 5 faults to 1 fault in the freqmat. This is useful
%          if extra fault codes are generated from multiple forensic
%          runs, and it is unlikely that these additional fault codes
%          are relevant for degradation.
%'optweight' = this argument will use gradient descent to optimize
%          weightings for each fault code based on maximizing the
%          metrics prognosability, trendability, and monotonicity. See
%          the reference for more information
```

```
%%The fc2pp function also uses the function "ppmetrics" from the PEP
%%toolbox, developed by Dr. Jamie Coble for the University of Tennesse

if iscell(freqmat)
    if size(freqmat,1)>size(freqmat,2);freqmat=freqmat';end
%     for ii=1:size(freqmat,2)
%         freqmat{ii}=freqmat{ii}(:,(sum(freqmat{ii},1)~=0));end
else
    freqmat{1}=freqmat(:,(sum(freqmat,1)~=0));end

input=freqmat;

model                                                                =
struct('method','normal','type','winmean','window',5,'reduce',...
    'no','weights',[]);

flags=varargin;
if size(flags)>0
    for ii=1:length(flags)
        if strcmp(flags{ii},'cumsum')
            model.type='cumsum';end
        if strcmp(flags{ii},'cummean')
            model.type='cummean';end
        if strcmp(flags{ii},'winmean')
            model.type='winmean';model.window=flags{ii+1};end
        if strcmp(flags{ii},'optweight')
            model.method='optweight';end;
        if strcmp(flags{ii},'reduce')
            model.reduce='yes';end;end;end

win2=model.window;

if strcmp(model.reduce,'yes')
    for ii=1:size(input,2)
        input{ii}(input{ii}~=0)=1;end;end

if strcmp(model.method,'optweight');
    if strcmp(model.type,'cumsum')
        for ii=1:size(input,2)
            inputs{ii}=cumsum(input{ii});end;end
    if strcmp(model.type,'cummean')
        for ii=1:size(input,2)
            for jj=1:size(input{ii},1)
                inputs{ii}=mean(input{ii}(1:jj,:));end;end;end
    if strcmp(model.type,'winmean')
        for ii=1:size(input,2)
            kk=0;
            for jj=1:win2
                inputs{ii}(jj,:)=mean(input{ii}((jj-kk):jj,:),1);
                kk=kk+1;end
            for jj=(win2+1):size(input{ii},1)
```

```
                inputs{ii}(jj,:)=mean(input{ii}((jj-
win2):jj,:),1);end;end;end

    fitfcn = @(x) paramfit(x',inputs,[1 1 1],false);
    for yy=1:size(inputs{1},2)
        for xx=1:size(inputs,2)
            temp{xx}=inputs{xx}(:,yy);end
        [m,p,t]=ppmetrics(temp);
        x0(yy)=m+p+t;end
    x0(isnan(x0))=0;
    warning off
    options=optimset('Display','off','MaxFunEvals',500);
    [model.weights]=fminimax(fitfcn,x0',[],[],[],[],[],[],[],options);
    for ii=1:size(input,2)
        input{ii}=input{ii}*model.weights;end
end

if strcmp(model.type,'cumsum')
    for ii=1:size(input,2)
        PP{ii}=cumsum(sum(input{ii},2));end;end

if strcmp(model.type,'cummean')
    for ii=1:size(input,2)
        for jj=1:size(input{ii},1)
            PP{ii}(jj,1)=mean(sum(input{ii}(1:jj,:),2));end;end;end

if strcmp(model.type,'winmean')
    for ii=1:size(input,2)
        kk=0;
        for jj=1:win2
            PP{ii}(jj,1)=mean(sum(input{ii}((jj-kk):jj,:),2));
            kk=kk+1;end
        for jj=(win2+1):size(input{ii},1)
            PP{ii}(jj,1)=mean(sum(input{ii}((jj-
win2):jj,:),2));end;end;end

end
```

A.3 intfc- Integrating Fault Codes into Existing Prognostic Parameters Function

```
function [PP weights] = intfc(progparam,freqmat,optweight)

%%If this error occurs, try using the interpolation function
%                                                                    if
(iscell(freqmat)&&iscell(progparam))&&size(progparam)~=size(freqmat)
%         error('Size of parameter is not equal to size of count
matrix.');end

%%Data input: columns in freqmat should be fault codes, rows should be
%%time. Progparam should be a 1 x m where m is the same number of rows
as
%%in the freqmat.

%%If not enough arguments are supplied, use default values
if nargin<3
    optweight=[1 1 1];end

%%Ensure that the dimensions on the two input matrixes are standardized
to
%%be 1 x number of components
if size(freqmat,1)>size(freqmat,2)
    freqmat=freqmat';end
if size(progparam,1)>size(progparam,2)
    progparam=progparam';end

%%Intialization and Matrix Sizes
input=freqmat;
numcomps=size(progparam,2);
numfc=size(freqmat{1},2);

%%%Module 1- Scaling
%%Find a scaling factor for the fault code data that is appropriate
based
%%on the values of the non-integrated prognostic parameter
nn=1;
for ii=1:numcomps
    for jj=2:size(progparam{ii},1)
        temp(nn)=progparam{ii}(jj)-progparam{ii}(jj-1);
        nn=nn+1;end;
    meantemp(ii)=mean(progparam{ii});end
mscale=mean(temp);meanall=mean(meantemp);

%%%Module 2- Parameter Construction and Selection
%%Select the fault codes that are relevant to degradation, and provide
%%a good initial guess for weights.
```

```
%%This module uses the "ppmetrics" function from the PEP toolbox,
%%developed by Dr. Jamie Coble
for ii=1:numcomps

nfm{ii}(1,:)=meanall.*mscale.*freqmat{ii}(1,:)+ones(1,numfc).*progparam
{ii}(1,1);
    for jj=2:size(progparam{ii},1)
        nfm{ii}(jj,:)=nfm{ii}(jj-
1,:)+ones(1,numfc).*(progparam{ii}(jj)-progparam{ii}(jj-1))+...
            meanall.*mscale.*freqmat{ii}(jj,:);end;end
for jj=1:numfc
    [basem basep baset]=ppmetrics(progparam);
    for kk=1:numcomps
        np{kk}=nfm{kk}(:,jj);end
    [newm newp newt]=ppmetrics(np);
    if (optweight*[basem basep baset]')<(optweight*[newm newp newt]');
        iw(jj)=(optweight*[newm  newp  newt]')-(optweight*[basem  basep
baset]');
    else
        for mm=1:numcomps
            nfm{mm}(:,jj)=0;end
    end;end
if size(iw,2)<numfc
    iw((size(iw,2)+1):numfc)=0;end

%%%Module 3- Optimization
%%This module uses the "paramfit" function from the PEP toolbox,
%%developed by Dr. Jamie Coble
fitfcn = @(x) paramfit(x,nfm,optweight,false);
warning off;options=optimset('Display','off','MaxFunEvals',400);
[optweights]=fminimax(fitfcn,iw,[],[],[],[],[],[],[],options);

%%%Final Parameter Construction
for ii=1:numcomps
    PP{ii}=nfm{ii}*optweights';end
```

A.4 Bearing Vibration Simulation

```
deg=0;
for ii=1:100
    Fs = 1000;T = 1/Fs;L = 2000;t = (0:L-1)*T;
    deg=deg+(0.008).*rand;
    y{ii} = sin(2*pi*60*t) + deg*sin(2*pi*350*t) + 0.3*randn(size(t));
end

for jj=1:100
    a=find(abs(y{jj})>1.8);
    fc(jj)=size(a,2);
end

NFFT = 2^nextpow2(L);
f = Fs/2*linspace(0,1,NFFT/2+1);
Y = fft(y{1},NFFT)/L;
subplot(1,2,1);plot(y{1});title('Test #1 Raw Signal');
xlabel('Time');ylabel('Vibration');axis([0 2000 -2.5 2.5])
hold on;plot(1:2000,ones(1,2000).*1.75,'r')
plot(1:2000,ones(1,2000).*-1.75,'r');hold off;
subplot(1,2,2);plot(f,2*abs(Y(1:NFFT/2+1)));title('Test #1 FFT')
xlabel('Frequency (Hz)');ylabel('Magnitude')

figure
Y = fft(y{100},NFFT)/L;
subplot(1,2,1);plot(y{100});title('Test #100 Raw Signal');
xlabel('Time');ylabel('Vibration');axis([0 2000 -2.5 2.5])
hold on;plot(1:2000,ones(1,2000).*1.75,'r')
plot(1:2000,ones(1,2000).*-1.75,'r');hold off;
subplot(1,2,2);plot(f,2*abs(Y(1:NFFT/2+1)));title('Test #100 FFT')
xlabel('Frequency (Hz)');ylabel('Magnitude')
```

A.5 Large Fault Code Database Simulation

```
for ii=1:10
    deg1=0;deg2=0;
    bal(ii)=rand;n=0;

    while deg1<1 && deg2<1
        deg1=deg1+0.01*bal(ii)+0.0003*rand;
        deg2=deg2+0.01*(1-bal(ii))+0.0003*rand;
        n=n+1;

        for kk=1:400
            fc{ii}(n,kk)=rand<(deg1-randn*0.08);end
        for kk=401:1000
            fc{ii}(n,kk)=rand<(deg2-randn*0.08);end
    end;end
```

APPENDIX B- ADDITIONAL FIGURES

B.1 Additional Figures for Section 4.1.3

These figures show additional parameters constructed using the fc2pp function on the actuator degradation data set from Section 4.1.3.

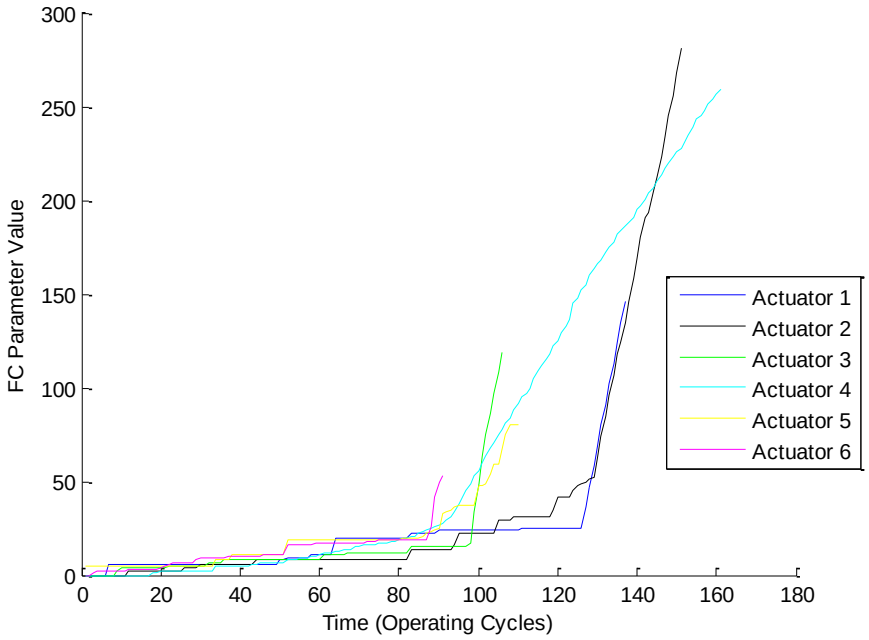

Figure B.1. Cumulative Sum.

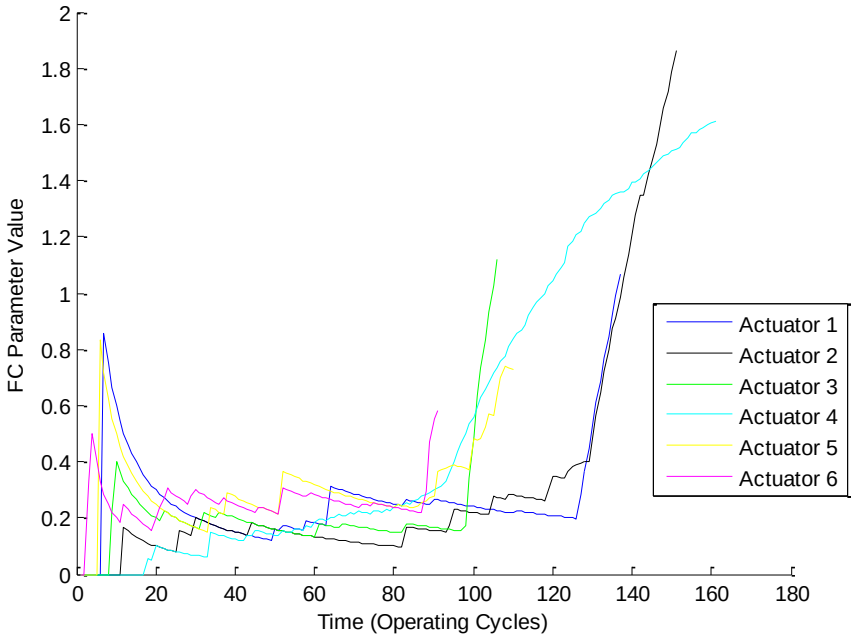

Figure B.2. Cumulative Mean.

B.2 Additional Figures for Section 4.1.5

These figures show additional parameters constructed using topic model dimension reduction from Section 4.1.5.

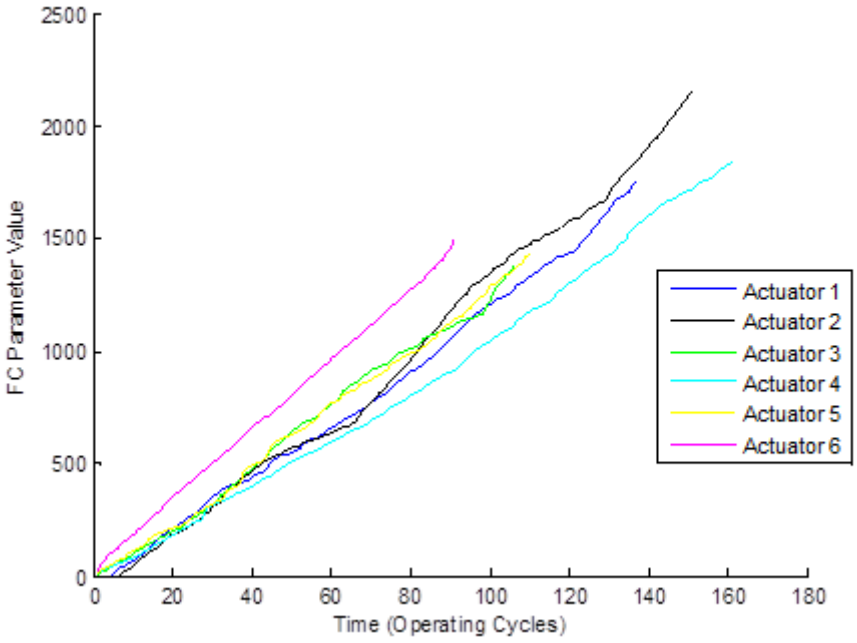

Figure B.3. Non-optimized, Cumulative Sum.

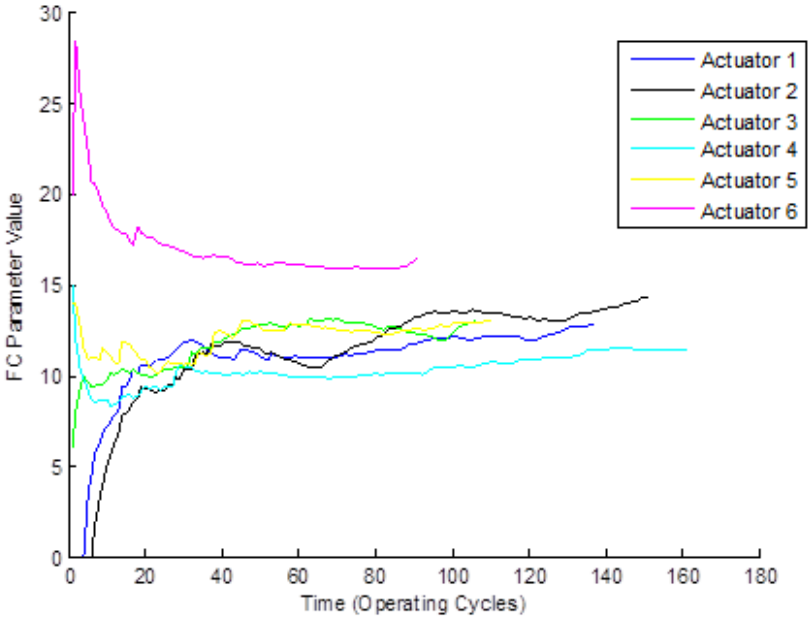

Figure B.4. Non-optimized, Cumulative Mean.

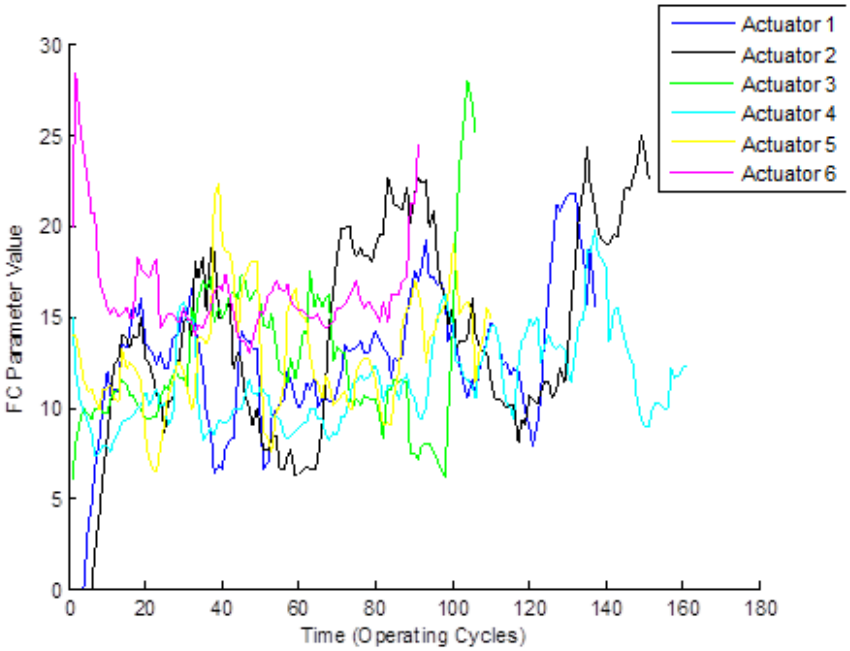

Figure B.5. Non-optimized, Windowed Mean Size 5.

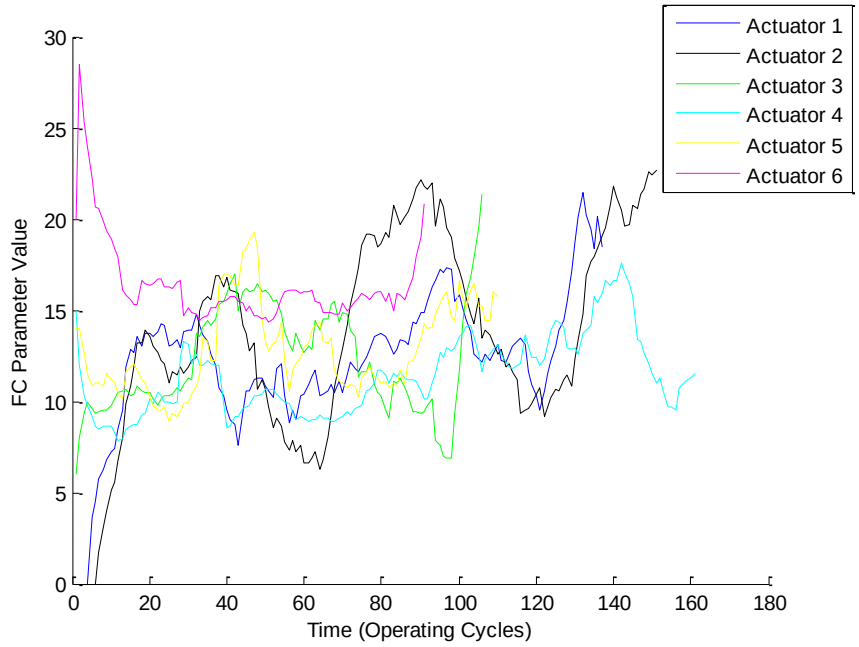

Figure B.6. Non-optimized, Windowed Mean Size 10.

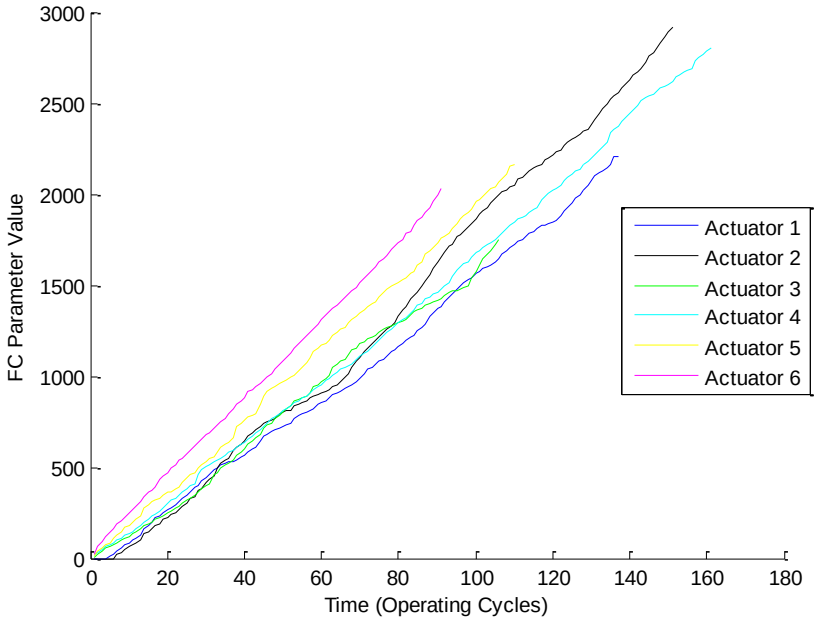

Figure B.7. Optimized, Cumulative Sum.

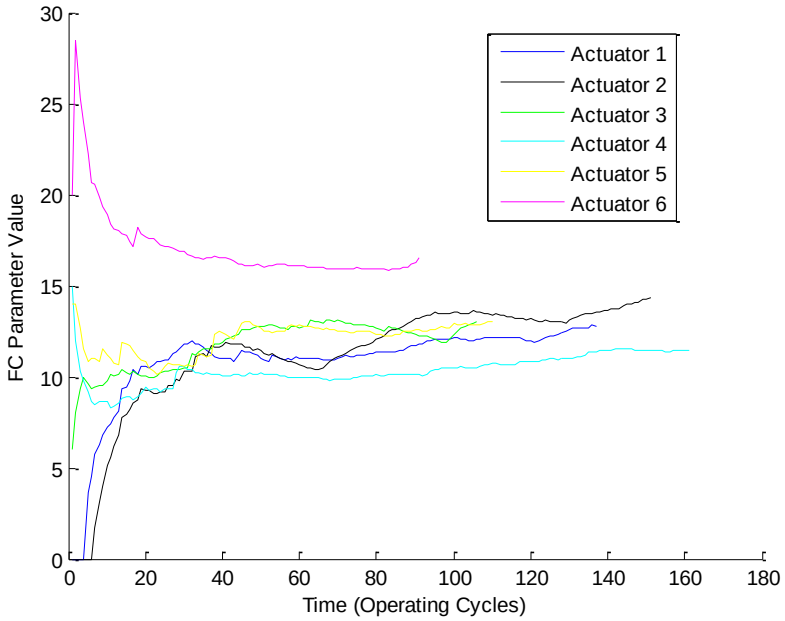

Figure B.8. Optimized, Cumulative Mean.

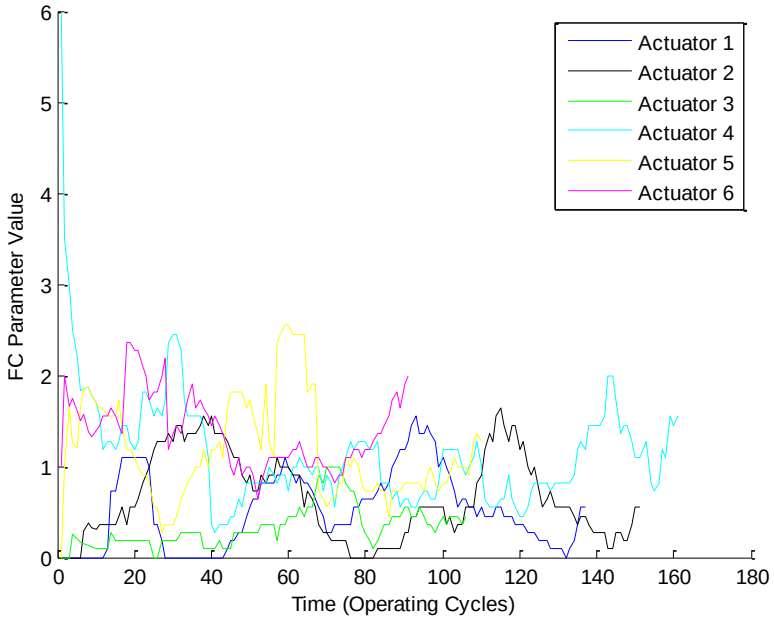

Figure B.9. Topic 1.

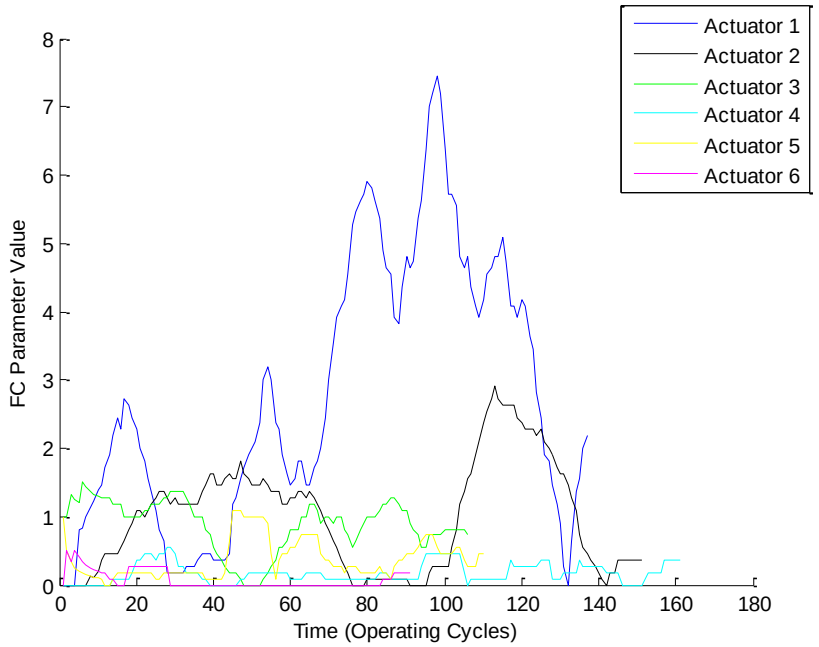

Figure B.10. Topic 2.

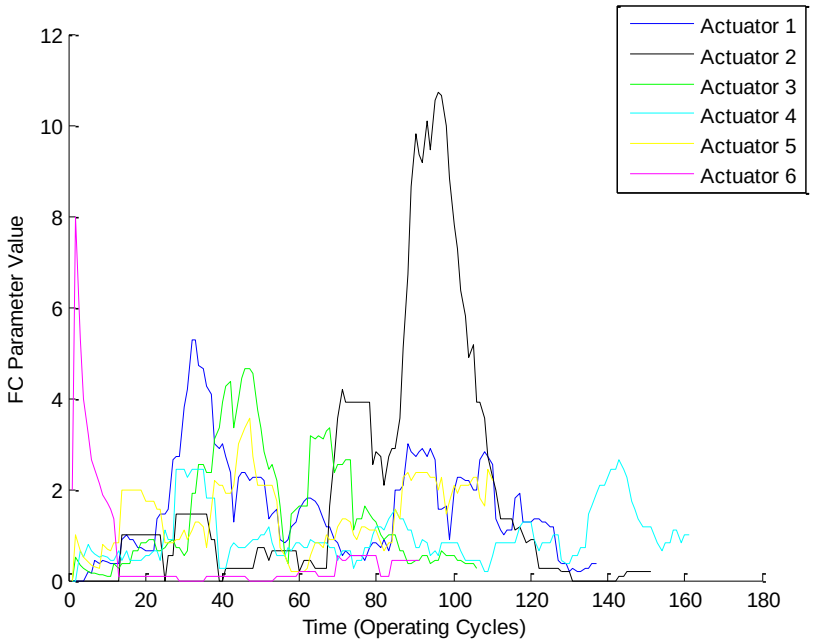

Figure B.11.Topic 4.

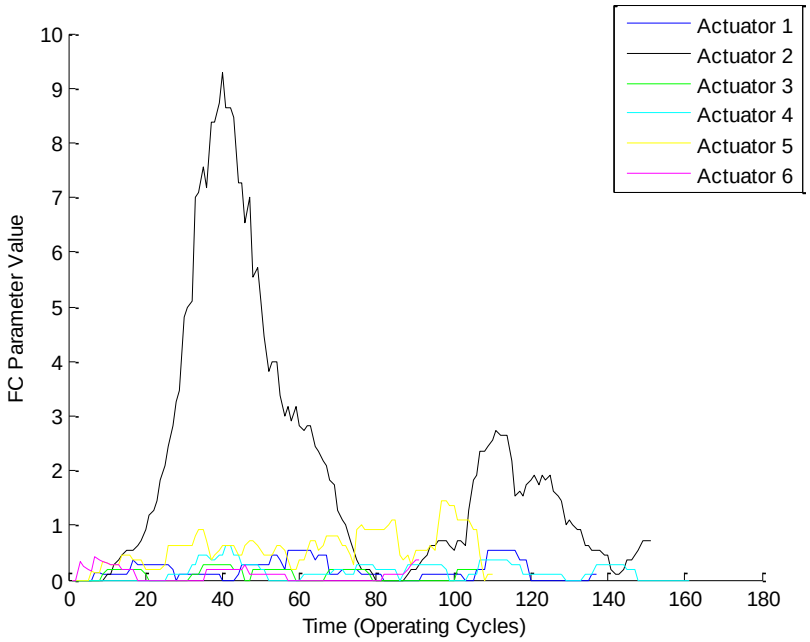

Figure B.12. Topic 5.

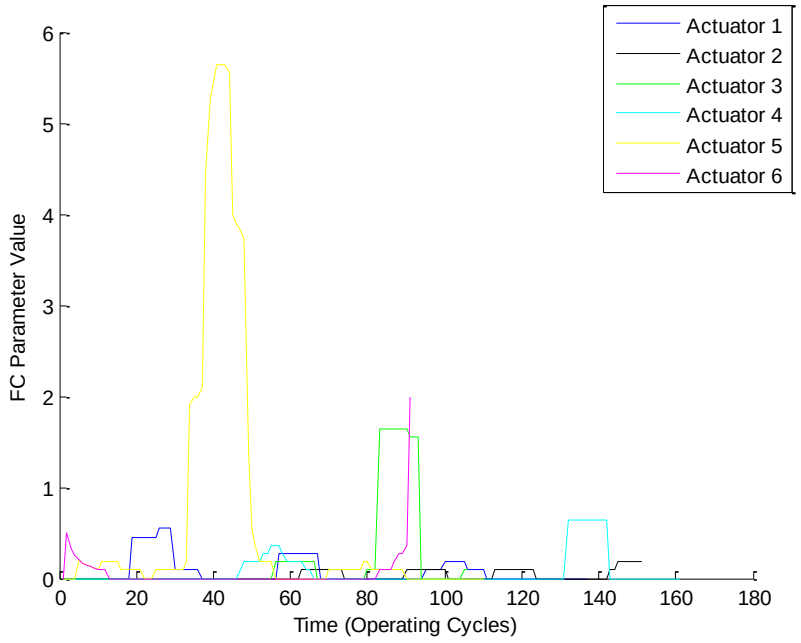

Figure B.13. Topic 6.

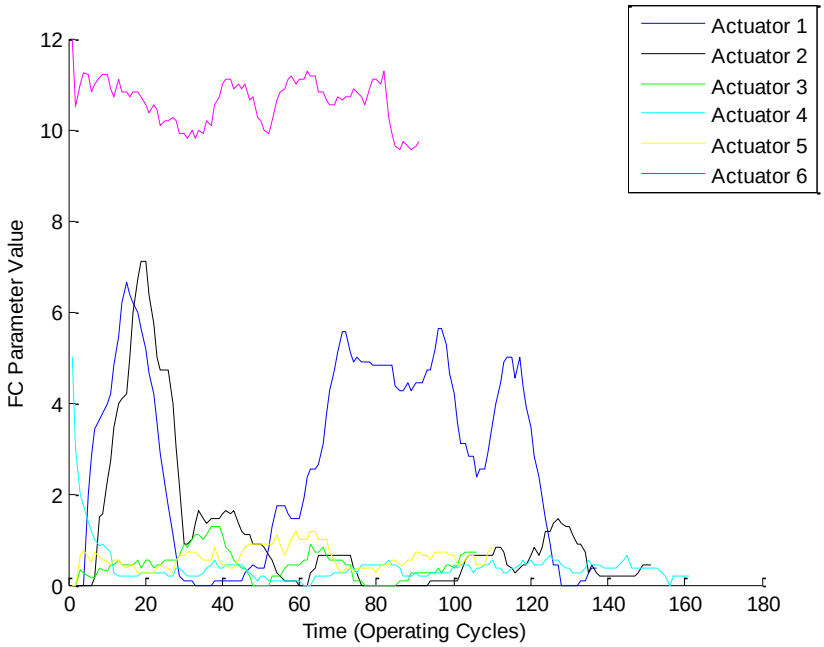

Figure B.14. Topic 8.

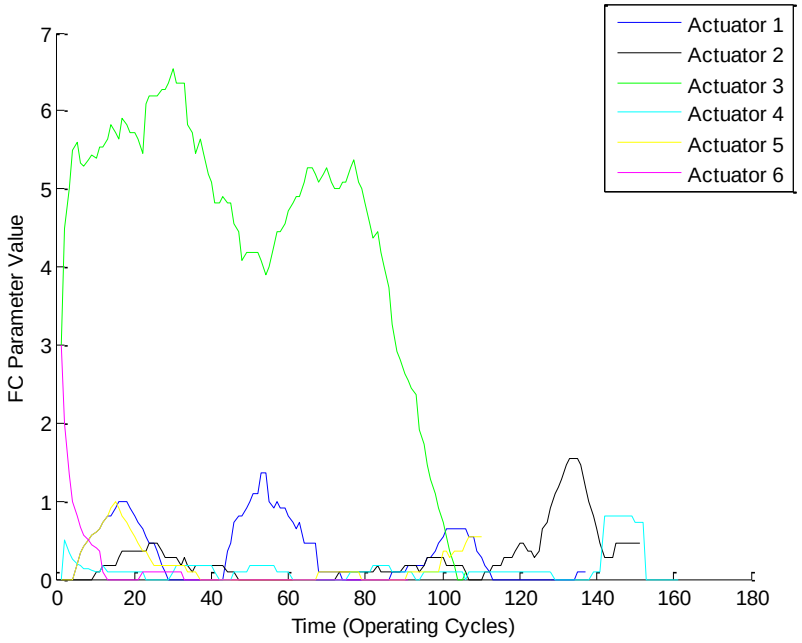

Figure B.15. Topic 9.

B.3 Additional Figures for Section 4.2.2

These figures show additional parameters constructed for fault code generation from the motor degradation data set shown in Section 4.2.2.

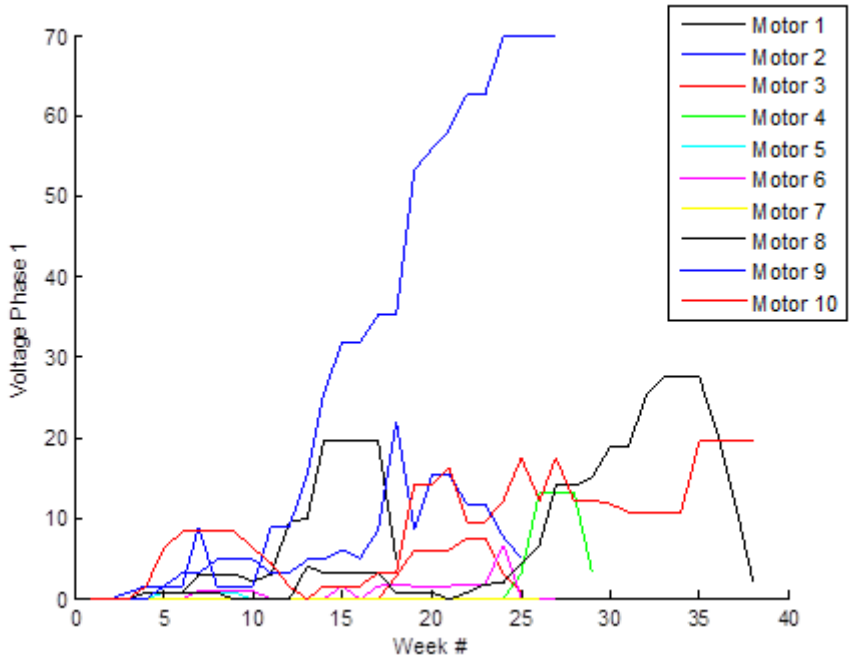

Figure B. 16. Voltage Phase 1.

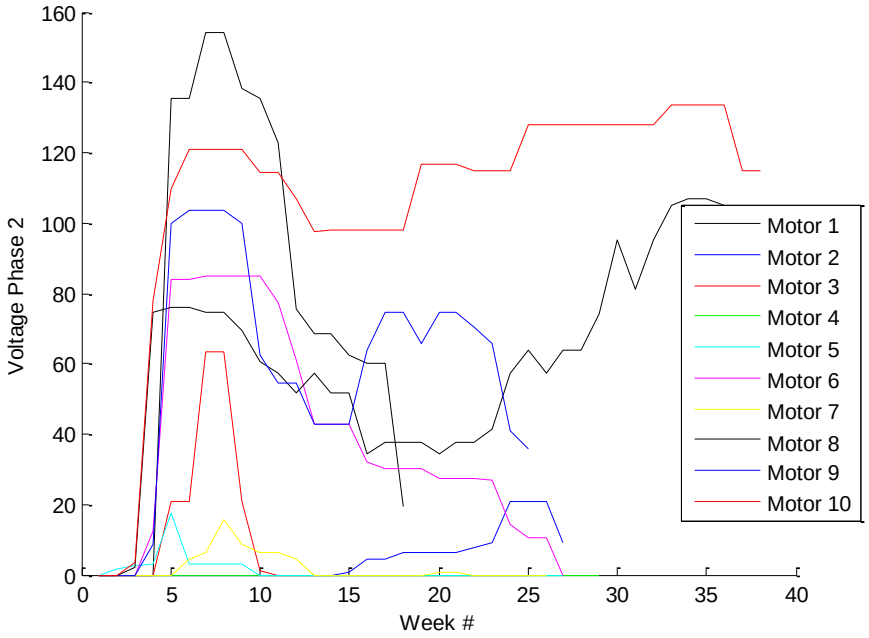

Figure B.17. Voltage Phase 2.

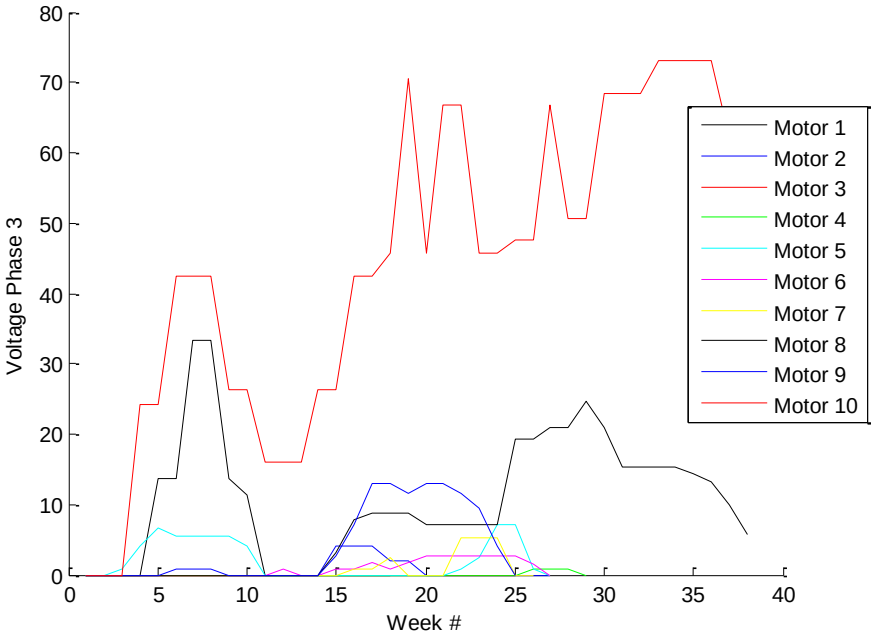

Figure B. 18. Voltage Phase 3.

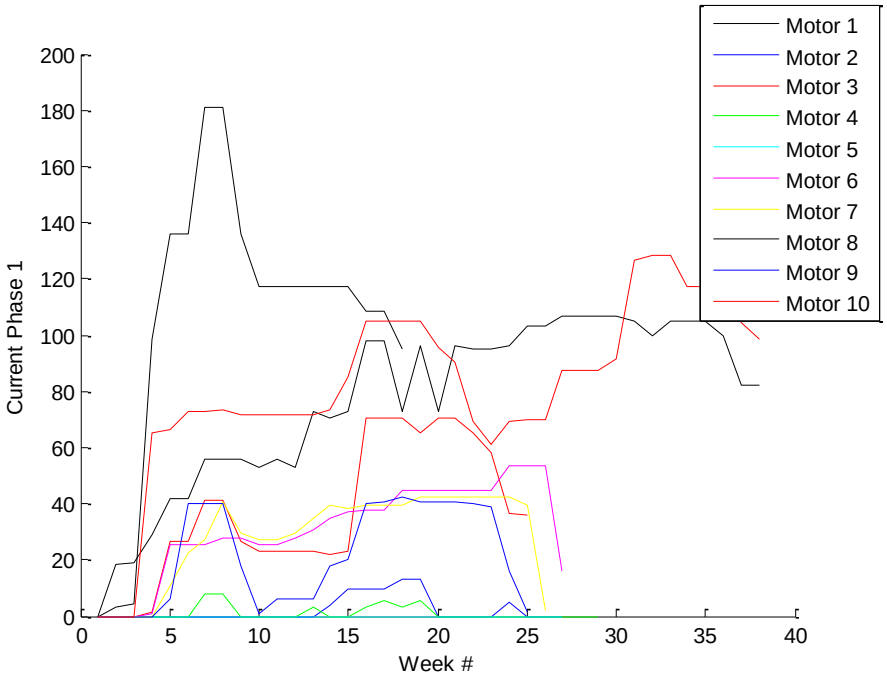

Figure B.19. Current Phase 1.

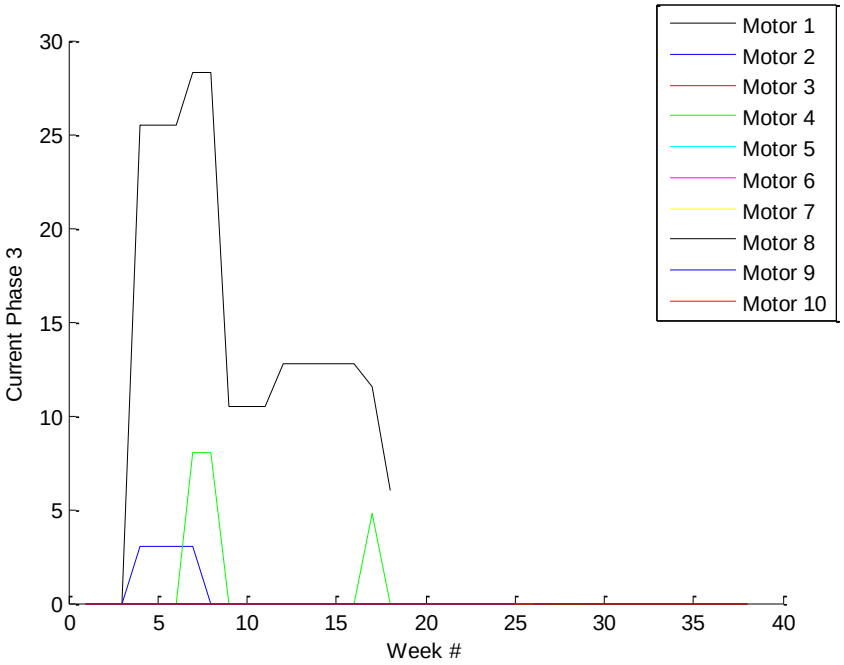

Figure B.20. Current Phase 3.

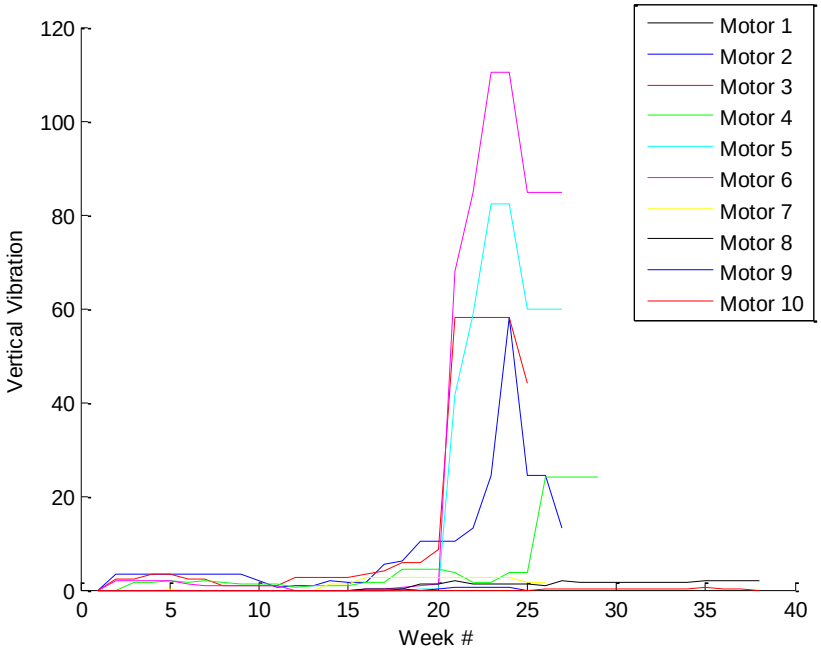

Figure B.21. Vertical Vibration.

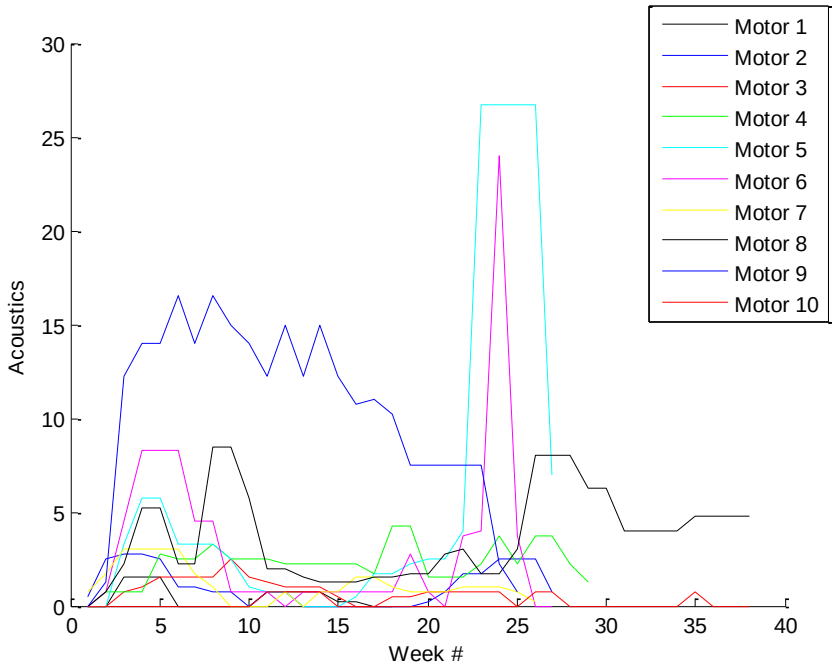

Figure B.22. Acoustics.

B.4 Additional Figures for Section 4.2.3

These figures show additional parameters constructed from fault codes from the motor degradation data set shown in Section 4.2.2.

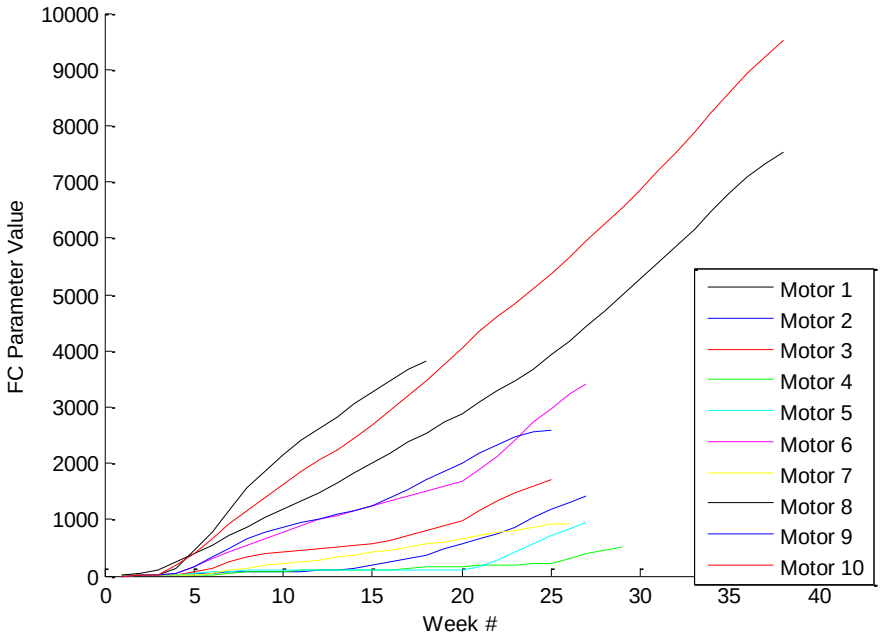

Figure B.23. Cumulative Sum, Non-optimized.

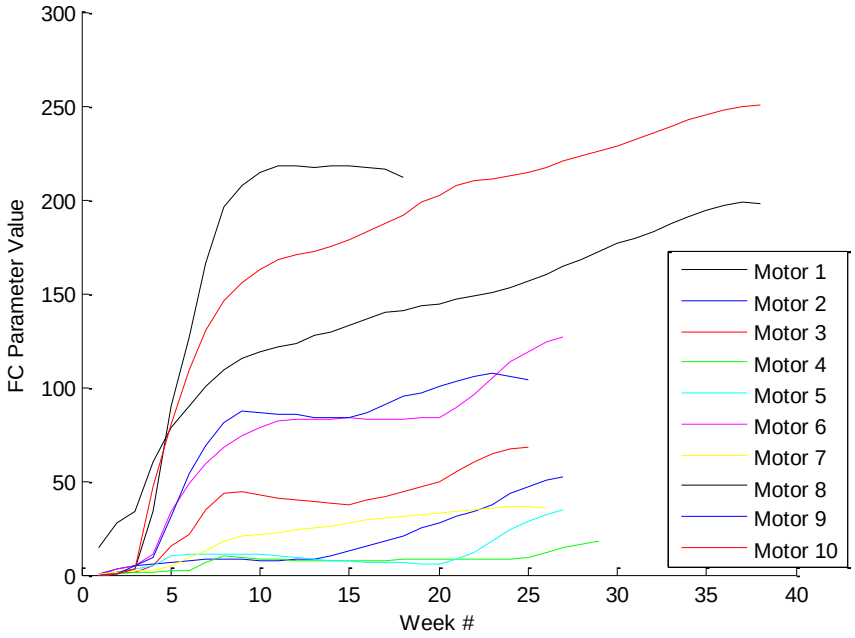

Figure B.24. Cumulative Mean, Non-optimized.

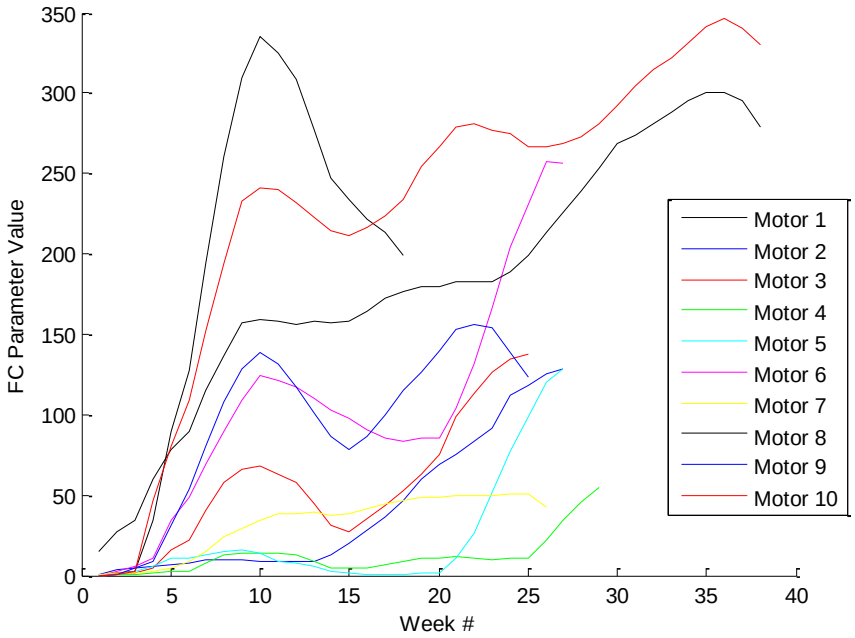

Figure B.25. Windowed Mean with a Window of 5, Non-optimized.

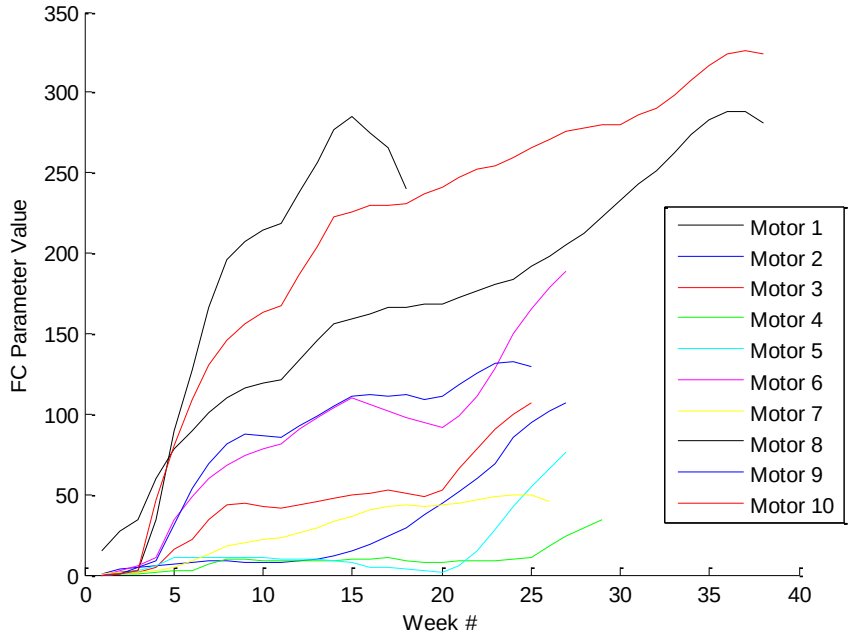

Figure B.26. Windowed Mean with a Window of 10, Non-optimized.

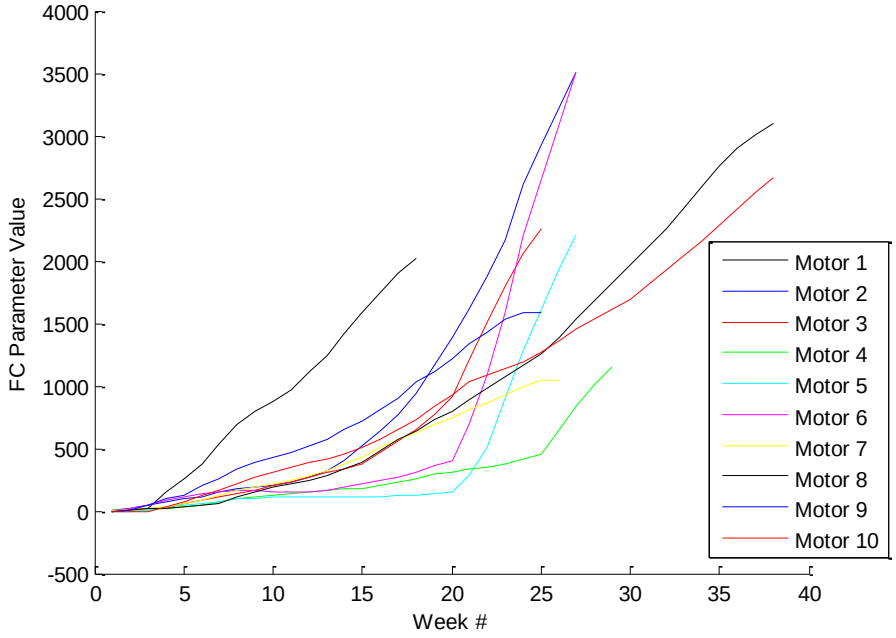

Figure B.27. Cumulative Sum, Optimized.

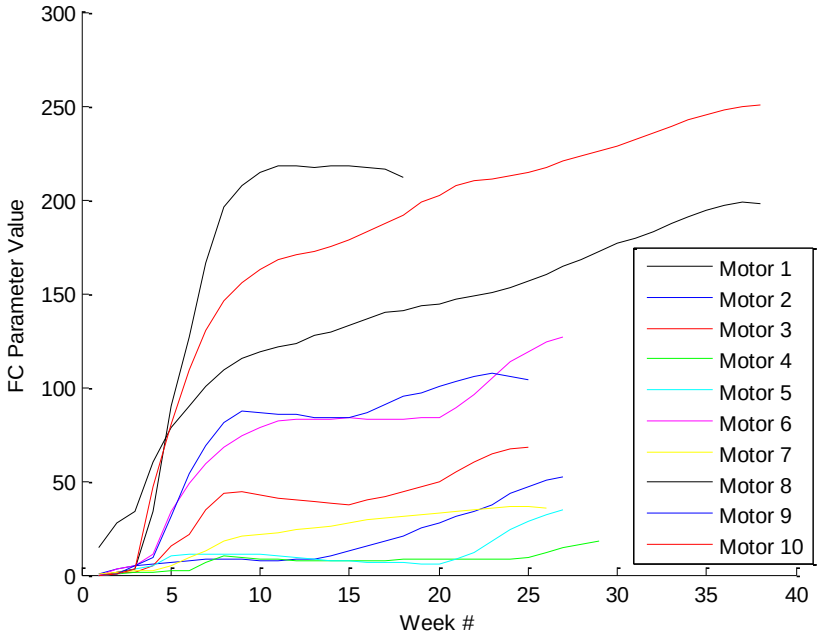

Figure B.28. Cumulative Mean, Optimized.

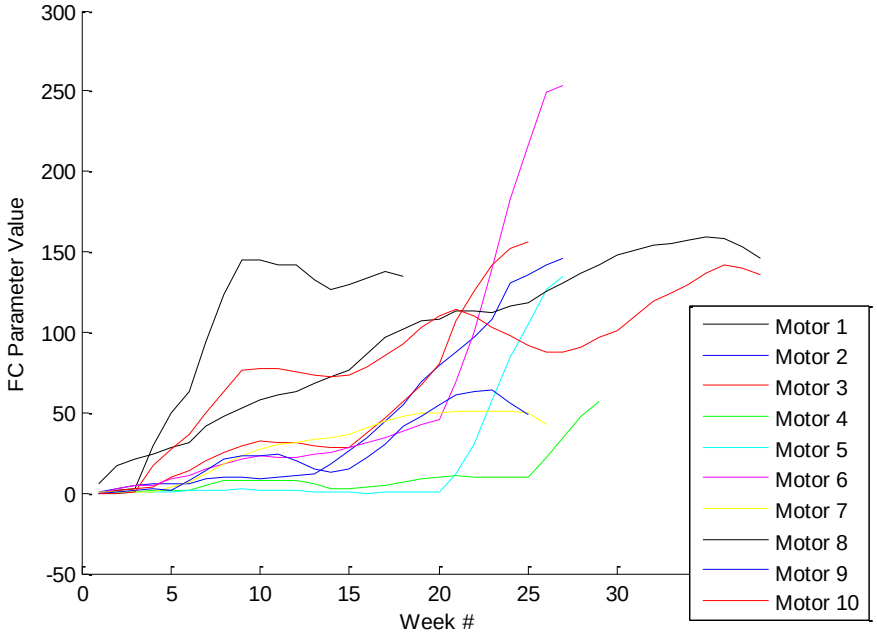

Figure B.29. Windowed Mean with a Window of 5, Optimized.

VITA

Eric Strong was born on May 27, 1988, in Chattanooga, TN. He graduated from Ringgold High School in Ringgold, GA in 2006. During high school, he received several honors, including receiving the AP Scholar with Distinction award and attending the Governor's School in the summer of 2009 for physics. Eric also became an Eagle Scout during this time. He then attended Vanderbilt University in Nashville, TN, receiving a bachelor's degree in Physics in May 2010. While an undergraduate, he was a member of Vanderbilt Student Volunteers for Science, a mentoring program for local middle schools that taught science to kids using laboratory experiments and emphasized available opportunities in science. He was also a member of Wilskills, a conservation group that handled recycling and biofuel manufacturing on campus, along with camping, canoeing, and caving weekend trips.

In May 2010, Eric began attending the University of Tennessee in Knoxville, TN. He received his Masters in Nuclear Engineering in May 2012, and he received a second Masters in Reliability Engineering with a specialization in prognostics in May 2013. In November 2012, Eric was elected as the student executive committee member for the Human Factors Instrumentation and Control Division (HFICD) of the American Nuclear Society (ANS). In October 2013, Eric presented preliminary research related to this dissertation at the Doctoral Consortium of the Prognostics and Health Management (PHM) Society conference in New Orleans, LA and received Honorable Mention for Best Presenter.