ROBUST CONDITION MONITORING FOR MODERN POWER CONVERSION

by

Jason Maurice Anderson

A dissertation submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in
Electrical Engineering

Charlotte

2013

ABSTRACT

JASON MAURICE ANDERSON. Robust condition monitoring for modern power conversion. (Under the direction of DR. ROBERT W. COX)

The entire US electrical grid contains assets valued at approximately $800 billion, and many of these assets are nearing the end of their design lifetimes [1]. In addition, there is a growing dependence upon power electronics in mission-critical assets (i.e. for drives in power plants and naval ships, wind farms, and within the oil and natural-gas industries). These assets must be monitored. Diagnostic algorithms have been developed to use certain key performance indicators (KPIs) to detect incipient failures in electric machines and drives. This work was designed to be operated in real-time on operational machines and drives. For example the technique can detect impending failures in both mechanical and electrical components of a motor as well as semiconductor switches in power electronic drives. When monitoring power electronic drives, one is typically interested in the failure of power semiconductors and capacitors. To detect incipient faults in IGBTs, for instance, one must be able to track KPIs such as the on-state voltage and gate charge. This is particularly challenging in drives where one must measure voltages on the order of one or two volts in the presence of significant EMI. Sensing techniques have been developed to allow these signals to be reliably acquired and transmitted to the controller. This dissertation proposes a conservative approach for condition monitoring that uses communications and cloud-based analytics for condition monitoring of power conversion assets. Some of the potential benefits include lifetime extension of assets, improved efficiency and controllability, and reductions in operating costs.

DEDICATION

This thesis is dedicated to my wife, Whitney, my grandmother, Mabel Beatty, and my mother, Brenda Anderson.

ACKNOWLEDGMENTS

I would like to thank all four members of my dissertation committee: Robert Cox, Yogendra Kakad, Asis Nasipuri, and Martin Kane. I would like to thank Dr. Kakad for his encouragement and his words of wisedom throughout my time at UNC Charlotte. I would especially like to thank Dr. Robert Cox for all of the hard work that he has put into me for the past 7 years. I would not be as successful as I am today without your help and dedication.

I would like to thank my fellow collegues and friends: Dr. Douglas Isenberg, Mike McLain, John Troxler, Tony Harris, Paul O'Connor, Forrest Suter, Lane Foulks, Prayag Parikh, Dr. Phong Tran, and Rebecca Sawyer. Without each of your help I could not have made it through the last 7 years. You have helped make this journey a lot of fun!

I gratefully acknowledge the financial and technical support with regard to motor fault detection from Converteam Naval Systems (now GE Power Conversion), including Steve Mankevich, Joel Hough, and Adam Kabulski. I also gratefully acknowledge AREVA NP for their generous donation of the SpectraQuest Machinery Fault Simulator. Without these efforts, the work on motor fault detection would not have been possible.

Also I would like to acknowledge the support in the past few months of EPRI especially my manager Matt Olearczyk as I have began working while still finishing this up. Your support and encouragement are much appreciated.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

Since the introduction of steam power, the world has experienced several industrial revolutions. By the middle of the twentieth century, the cumulative effect was a large industrial network designed to extract energy from primary resources, convert it to electricity, and distribute it throughout society. More recently, the digital revolution produced technologies that vastly improved communication and computation. Today, there is a new movement in which these two tectonic shifts are converging. Digital technologies, namely networked sensors and actuators, are becoming embedded into our existing industrial infrastructure in a manner that could transform the ways in which we do business and live our lives. The energy-conversion process, all the way from primary energy extraction to end use, is a perfect candidate for such digital infusion. At present, energy conversion is highly inefficient, with about 60% of primary energy lost in the conversion to electricity [2]. The push for greater efficiency has driven technological innovation, including the proliferation of new digitally-enabled technologies at point-of-use. Examples include motor drives, switching power supplies, and interfaces for distributed generation sources. In addition to improved efficiency, there are also numerous operational benefits that can result from greater monitoring and control. Through the proper application of intelligent sensors, networked computing, and advanced analytics, it should become possible to detect impending failures, diagnose their root causes, and recommend the appropriate course of action with respect to risk mitigation and operational cost optimization. In the case of any individual component connected to or within the power system this means that we must move beyond data collection to advanced, risk-balanced decision-making.

To date, sufficient data sets typically do not exist to enable the necessary capabilities. This dissertation proposes a framework for obtaining such data, and it focuses specifically on electric motors and the power-electronic drives used to optimize their consumption and functionality.

1.1 Problem Overview

Condition-based maintenance (CBM), that is making maintenance decisions based on the true condition of a component, is a concept that has been discussed for several decades [3]. Large enterprises such as the United States military, power companies, and large manufacturers led the way in the early 1980s. Much of this early work was focused on demonstrating the capability of sensor based fault detection and diagnosis, as well as on the development of appropriate online sensors. As sensing technologies have matured and communications have improved, components and devices have become more intelligent. It is now possible to monitor many key assets from a centralized location.

Several factors are leading to the push for more intelligent asset management, including reduced sensor-deployment cost, increased computing power, and the growth of advanced analytics [2]. The decline in the cost of sensors has made it possible to monitor assets in a more economical manner than in the past. The continued advancement of microprocessors has also made it possible to place digital intelligence into and near physical assets. The advancements in big data software tools and analytic techniques provide the means to understand the massive quantities of data that are generated by intelligent assets.

Making sense of the large amount of data that can be generated by these new intelligent assets is one of the key components of the "Industrial Internet" concept that is currently being developed by General Electric (GE) and others [2]. As illustrated in Fig. 1.1, this idea can be expressed in terms of the flow and interaction of data, hardware, software, and intelligence. Data is harvested from intelligent devices and

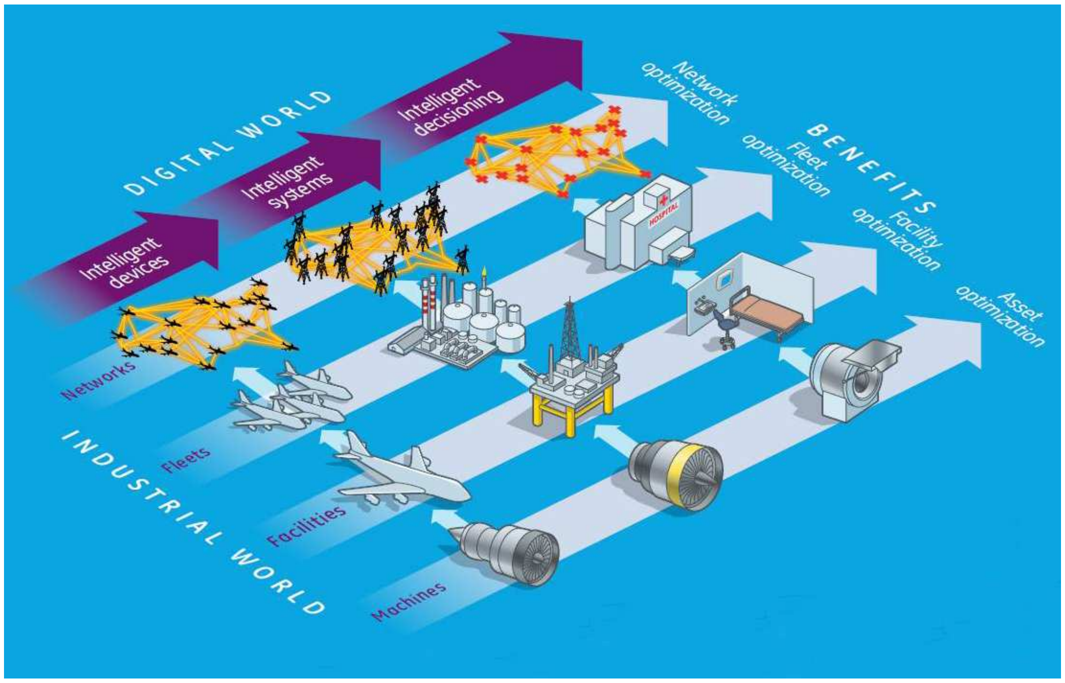


Figure 1.1: Applications of GE's "Industrial Internet" concept. Taken from [2].

networks. The data is stored, analyzed, and visualized using big data and analytics tools. The information gathered by these intelligent devices and components can be acted upon by decision makers. This concept has the potential to revolutionize the way numerous industries are operated [2].

In recent years various large enterprises have started to adopt the notion of networked asset monitoring. EPRI's Fleetwide Monitoring for Equipment Condition Assessment program [4] and the Navy's Integrated Condition Assessment System (ICAS) [5] are two examples of this approach. EPRI's vision for Fleetwide Monitoring, for instance, is to assist the power industry to evaluate and deploy advanced monitoring capabilities in a power plant. The scope is broad and focuses on monitoring individual instruments, components, and the complete system of components. To date, this work has focused heavily on fleetwide process monitoring. In this context, "fleetwide" refers to deployment across various power stations in a single companies fleet. Process monitoring, as its name implies, focuses on high-level process measurements such as flow rates, temperatures, and pressures. When a temperature exceeds a particular alarm value, for instance, it is likely that a given process is malfunctioning. One such example might be the temperature of a particular cooling fluid. Such process measurements are easy to obtain and are the crucial components of EPRI's fleetwide program and the Navy's ICAS program [6]. Process measurements such as coolant temperatures tend to change long after individual components begin to degrade. Instead of using only process variables, it is better to use individual component status parameters such as those obtainable from current sensors, vibration measurements, partial discharge monitoring [6].

Many excellent, early-stage fault detection technologies have been developed for power system components using status-parameter measurements. For example, motor current signature analysis (MCSA) is a technique that can be used to find incipient motor faults. Readings of motor current are recorded and analyzed in the frequency domain [7], [8]. The approach has been used to find early-stage faults such as those related to belts, couplers, and alignment. Vibration analysis is also an effective means for incipient fault detection. When applied to motors, for example, this approach exploits that fact that every motor has its own characteristic vibration spectrum. Changes to the individual frequency components of the vibration spectrum can give an early indication of a fault. Vibration analysis has been used to detect imbalance, misalignment, looseness [9], and bearing damage [10], [11].

Although many excellent individual technologies exist for monitoring the health of individual components, many integrated solutions that combine all of the individual technologies are not available. Major companies often make systems that focus on a single technology (i.e. partial discharge, MCSA, vibration, etc.). As shown later in Chapter 4, the reliance on an individual technology can lead to misdiagnoses. In order to achieve an integrated solution, EPRI and others are pushing for the development of smart components. A smart component is an augmented version of a standard component such as a motor or a pump. The smart component is understood to have embedded intelligence in the form of sensors, data transmission, and computation. Such devices should be able to continuously monitor their state of health predict their remaining useful life. Such components should be able to participate in an ad-hoc network of other similar components [12]. At present there is very little research on the integration of various sensing techniques for a single component. It is clearly an important area, but work is needed before such smart components can be deployed.

EPRI and other organizations recognize that vast data sets are needed in order to determine how to integrate individual technologies to make reliable risk-informed prognostic decisions. Examples of this include the diagnostic advisor project for power plants and a "big data" analytics modernization demonstration project in both the transmission and distribution programs. EPRI's diagnostic advisor system will facilitate the conversion of observed plant data to diagnoses (not just identified anomalies) [13]. This software will provide an extensible reasoning framework and database, which over time will be systematically filled with knowledge that is specific to plant assets. The framework and database will be flexible enough to integrate and process data from several sources, including observed signatures of actual faults that are contributed by field users, simulated asset fault signatures from plant simulators and/or physics-based models, and high-level fault-related status information such as those obtainable from operator rounds [13]. Other similar projects exist within EPRI's transmission [14] and distribution [15] programs. The common theme is that large data sets are needed in order to make any reliable decisions. At present, the collection of such data sets and the development of appropriate algorithms is crucial.

1.2 Proposed Idea

The approach proposed in this dissertation recognizes that changes in component data streams can be archived and this information stored in a database to begin to be able to develop diagnostic routines and risk-aware decision-making algorithms. The approach as shown in Fig. 1.2 leverages computing power now available at the

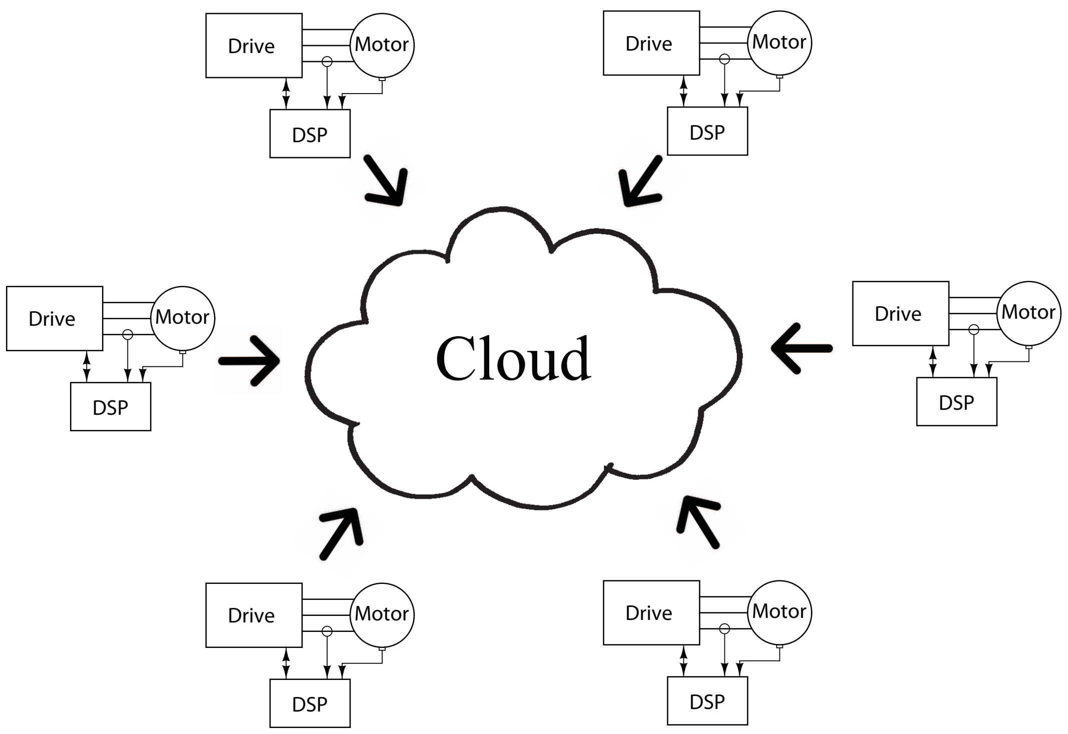


Figure 1.2: Proposed two step approach for fault detection and diagnosis.

point-of-load and separates fault detection from fault diagnosis. The basic idea is a two step approach:

- Step 1: Reliably detect faults at an early stage.
- Step 2: Beam the data into the cloud where it can be used to develop powerful diagnostic and prognostic routines.

During Step 1, features of a healthy system are learned during normal operating conditions, and these features are used to create a vector space. During operation, measured features are mapped onto this vector space. If a fault is developing, the corresponding mapping will change. This approach is based on a modification of a common facial-recognition algorithm [16], and it essentially detects that the feature space no longer appears as it should when healthy. Our scheme uses a single quantity known as the Health Indicator to determine that system behavior has changed. This approach is shown to be far more powerful than more simplistic, deterministic rules. Once a change has been observed, the underlying indicators can be captured and combined with maintenance records and process-variable measurements. The corresponding data sets can be used immediately for fault detection.

In Step 2, individual fault related-features are transmitted back to a centralized database from multiple locations. If a faulted condition occurs, an alarm would signal that investigation is needed. The prognosis could be performed by human experts and computerized algorithms. The data sets that would be recorded for similar assets across numerous locations could provide the information necessary to create powerful and automated expert systems.

The ultimate goal of this research is a suite of automated analytics capable of detecting faults, diagnosing their causes, and dispatching maintenance in a manner that optimizes economic impact. At present, such automation is not fully developed and remains a goal further down the technology road map of most organizations. This dissertation provides an approach that could be a major stepping stone on those road maps.

1.3 Primary Contributions

This dissertation's focus is on robust fault detection for modern power conversion, and it focuses specifically on power electronics and electric motors. The focus is specifically on the first step outlined in Sec. 1.2 - namely fault detection. With powerful detection schemes, large databases can be developed, and artificial intelligence and other tools can be used for diagnostics and prognostics.

In the case of motors, the proposed technique has been used to detect impending failures in both mechanical and electrical components. This dissertation integrates information from various techniques, including current-signature analysis and vibration analysis. As shown, the proposed scheme can detect minute changes store the data in appropriate databases.

In the case of power electronic drives, one is typically interested in the failure of switches and capacitors. To detect incipient faults in IGBTs, for instance, one must be able to track parameters such as the on-state voltage and the gate charge. This is particularly challenging in drives where one must measure voltages on the order of one or two volts in the presence of significant EMI. Sensing techniques have been developed to allow these signals to be reliably acquired and transmitted to the controller. The same fault detection algorithm has been used and it has been shown to be more sensitive than other schemes in the literature.

1.4 Outline

The remaining chapters of this dissertation describe the components of the proposed model. Chapter 2 discusses the generalized health monitoring algorithm for fault detection. Chapter 3 describes the implementation condition monitoring algorithm for electrolytic capacitors and power semiconductors (IGBT and MOSFET). Chapter 4 shows the implementation of the condition monitoring algorithm for fault detection in electric machines. Chapter 5 presents some conclusions and future work.

CHAPTER 2: GENERALIZED HEALTH MONITORING ALGORITHM

The generalized health monitoring algorithm is described in this chapter. One of the great difficulties associated with condition monitoring is that the user must typically possess some degree of expertise in order to distinguish a normal operating condition from a potential failure mode. This is because monitoring spectral components can result from any number of sources, including those related to normal operating conditions. This in combination with the vast amount of available information, make it difficult to determine a set of rules for fault detection a priori. When using hard-and-fast rules, for instance, the developer is likely to omit certain situations out of ignorance or fail to implement rules that deal with the dynamic nature of certain conditions. Without knowing all possible fault conditions and symptoms, the developer finds himself in a difficult position.

In order to detect fault conditions without a set of rules, an unsupervised approach has been developed that learns key performance indicators (KPI) that are specific to the system and monitors for changes that may indicate a potential fault condition. The fault detection algorithm is given a hybrid spectrum consisting of KPIs extracted from data that is collected with various sensors. In the language of information theory, the relevant information is extracted from the feature vector, encoding it efficiently, and then comparing the encoded result to a database of healthy features encoded in a similar manner. Any differences indicate that the device may be degrading. The overall approach is shown in Figure 2.1. This approach is partially patterned after the facial recognition scheme presented in [16] and described in detail in this chapter.

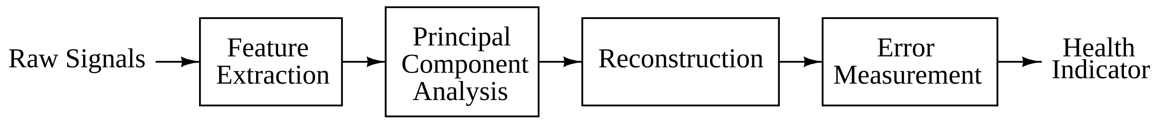


Figure 2.1: Block diagram of condition monitoring algorithm.

Figure 2.2: Training set of face images. Taken from [16].

2.1 Background on Facial Recognition Algorithm

The algorithm described in [16], [17] seeks to implement a facial recognition system that is efficient, simple, and accurate. The system does not depend on intuitive knowledge of the structure of the face like the mouth, nose, and eyes. Instead a linear combination of characteristic features called Eigenfaces are used for classification [16], [17]. Eigenfaces seeks to reduce the dimensionality of the training set by using principal component analysis (PCA) of the face images. PCA leaves only those features that are critical for facial recognition.

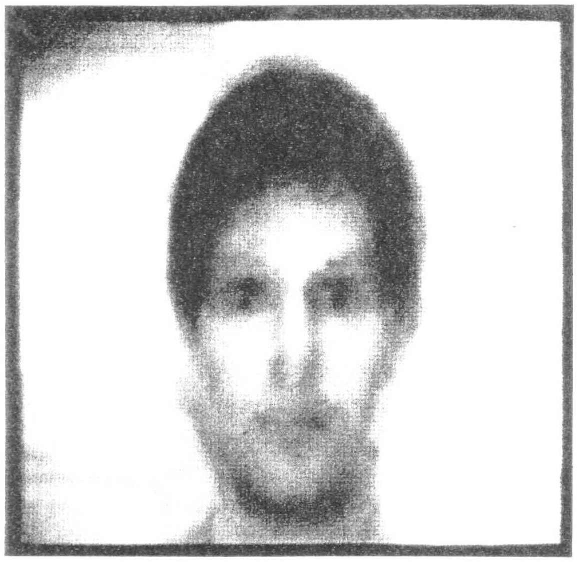


Figure 2.3: The average of all faces in the training set, Ψ. Taken from [16].

2.1.1 Learning Faces

The system is initialized by first acquiring a set of training images $(\mathbf{\Gamma}_1, \mathbf{\Gamma}_2, \mathbf{\Gamma}_3, \ldots \mathbf{\Gamma}_M)$ as shown in Fig 2.2. Eigenvectors and eigenvalues are computed on the covariance matrix of the training images and the M highest eigenvectors are kept. Finally, the known individuals are projected into the face space and their weights are stored. The images used for training are 8-bit intensity values of a 256 by 256 image. These images are converted from 256 by 256 images into a single dimension vector of size 65,536 because a two dimensional square matrix is needed to compute eigenvectors. The mean of the training images or the"average face" is defined as [16], [17]

$$\boldsymbol{\Psi} = \frac{1}{M}\sum_{n=1}^{M}\mathbf{\Gamma}_n. \tag{2.1}$$

and can be seen in Fig. 2.3. Each training image differs from the mean by $\boldsymbol{\Phi}$ which is [16], [17]

$$\boldsymbol{\Phi} = \mathbf{\Gamma}_k - \boldsymbol{\Psi}. \tag{2.2}$$

The vectors $\boldsymbol{u}_k$ is the eigenvectors and scalars λ_k is the eigenvalues of the covariance matrix of the face images, $\mathbf{\Phi}_i$. The covariance matrix is defined as [16], [17]

$$C = \frac{1}{M}\sum_{n=1}^{M} \boldsymbol{\phi}_n^T \boldsymbol{\phi}_n = \boldsymbol{A}\boldsymbol{A}^T \tag{2.3}$$

where $A = [\mathbf{\Phi}_1, \mathbf{\Phi}_2, \mathbf{\Phi}_3, \ldots, \mathbf{\Phi}_M]$. The eigenvalues are selected such that

$$\lambda_k = \frac{1}{M}\sum_{n=1}^{M} \left(\boldsymbol{u}_k^T \phi_n\right)^2 \tag{2.4}$$

is maximum, where

$$\boldsymbol{u}_l^T \boldsymbol{u}_k = \delta_{lk} = \begin{cases} 1, & \text{if } l = k \\ 0, & \text{otherwise} \end{cases} . \tag{2.5}$$

These formulas attempt to capture the source of the variance, which is later used for classification [16], [17]. The vectors $\boldsymbol{u}_k$ are referred to as the Eigenfaces, since they are eigenvectors and appear face-like in appearance as shown in Fig. 2.4. The major difficultly with this is that this computation will produce a large number of eigenvectors (N^2 by N^2). Since M is far less than N^2 at most there are $M-1$ non zero eigenvectors. A smaller matrix $L = \boldsymbol{A}^T\boldsymbol{A}$ will yield a smaller number of eigenvectors [16], [17]. The eigenfaces are defined as

$$\boldsymbol{u}_l = \sum_{k=1}^{M} \boldsymbol{v}_{lk} \boldsymbol{\phi}_k . \tag{2.6}$$

2.1.2 Recognizing Faces

Images can then be checked to determine if it is a known face or if it is a face at all. When a new image $\mathbf{\Gamma}$ is projected onto the face space using

$$\boldsymbol{\omega}_k = \boldsymbol{u}_k^T \left(\mathbf{\Gamma}_k - \mathbf{\Psi}\right). \tag{2.7}$$

The weights form a vector $\boldsymbol{\Omega}^T = [\omega_1, \omega_2, \omega_4, \ldots, \omega_M]$. The Euclidean distance measures the distance between the new image and a class of faces k. This distance is

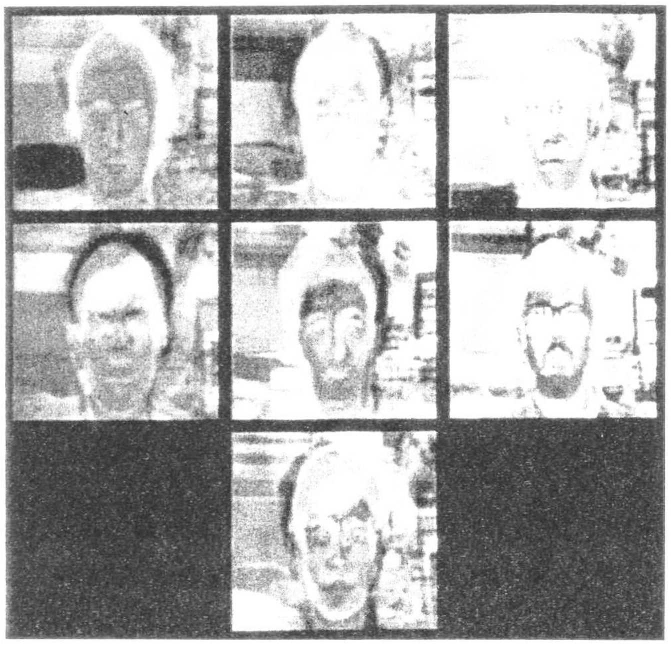


Figure 2.4: Eigenfaces that were calculated from the training images. Taken from [16].

defined as

$$\boldsymbol{\epsilon}_k^2 = \|\boldsymbol{\Omega} - \boldsymbol{\Omega}_k\|^2. \tag{2.8}$$

Note that if there are more than one examples of the face, these weights are averaged among all of the examples. If the distance measure, k is less than an empirically assigned threshold Θ_ϵ the face is recognized and assigned to class k [16], [17].

The distance function assumes that the new image,$\boldsymbol{\Gamma}$ is a face. To determine the validity of this assumption, the image is projected onto the face space, and the difference between the projected image and $\boldsymbol{\Gamma}$ is determined. The image is projected by computing

$$\boldsymbol{\Phi} = \boldsymbol{\Gamma} - \boldsymbol{\Psi}, \tag{2.9}$$

and projecting onto

$$\boldsymbol{\Phi}_f = \sum_{i=1}^{M'} \boldsymbol{\omega}_i \boldsymbol{u}_i. \tag{2.10}$$

The Euclidean distance determines the distance between the face space as [16], [17]

$$\boldsymbol{\epsilon}^2 = \|\boldsymbol{\Phi} - \boldsymbol{\Phi}_f\|^2. \tag{2.11}$$

If an image is presented to the algorithm and both $\boldsymbol{\epsilon}_k$ and $\boldsymbol{\epsilon}$ is greater than Θ_ϵ, then it is not a face.

2.2 Health Monitoring Algorithm

The following section describes in detail each block of the health monitoring algorithm as shown in Figure 2.1. The unsupervised fault detection algorithm is provided with a feature vector of KPIs. As in many pattern-recognition problems, this is a large number. When thinking in a rules-based context, one is likely to consider how each feature will be useful for the detection of one or more faults. In what is a potentially counterintuitive result, this is not necessarily the case. Generally speaking, features are not independent and thus classification accuracy does not improve as one includes more features. In fact, there is actually a diminishing point of returns that stems from the fact that as more features are included there is a possibility of over-fitting to the given data, a situation that can lead to errors. What is needed is an approach that combines features in such a way that it reduces the size of the data set, and reveals trends that best separate various data sets.

The heart of the fault detection algorithm is the principal component analysis (PCA) block. This unit provides information about meaningful trends within the monitored data. Generally speaking, PCA projects high-dimensional data onto a lower dimensional space, thus performing a transformation that best represents the information in the overall data set. In a sense, PCA is aggregating the set of KPIs and providing information about the relevant trends without having to develop a set of rules a priori. Experience has shown that that PCA leads to a robust detection mechanism.

PCA is an approach that reduces the dimensionality of the feature vector by projecting it onto a lower dimensional space. In general, PCA provides a linear transformation that best represents the data in a least-squares sense. Consider for a moment that n feature vectors are recorded. In this context, the feature vectors $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, ..., $\boldsymbol{x}_n$ each consist of n number of KPIs, and each vector might correspond to KPI values recorded at certain interval of time.

The feature vectors are sought to be represented by a single vector $\boldsymbol{x}_0$. More specifically, assume that that we want to find a vector $\boldsymbol{x}_0$ such that the sum of the squared distances between $\boldsymbol{x}_0$ and the various $\boldsymbol{x}_k$ is as small as possible. The squared-error criterion function J is defined as

$$J = \sum_{k=1}^{n} \|\boldsymbol{x}_0 - \boldsymbol{x}_k\|^2, \tag{2.12}$$

and seek the value of $\boldsymbol{x}_0$ that minimizes J. Perhaps somewhat intuitively, this problem is solved if $\boldsymbol{x}_0$ is equal to the sample mean $\boldsymbol{m}$ given by

$$m = \frac{1}{n} \sum_{k=1}^{n} \boldsymbol{x}_k. \tag{2.13}$$

This can be verified by expanding the equation for J, i.e.

$$J = \sum_{k=1}^{n} \|\boldsymbol{x}_0 - \boldsymbol{x}_k\|^2 = \sum_{k=1}^{n} \| \left(\boldsymbol{x}_0 - \boldsymbol{m}\right) - \left(\boldsymbol{x}_k - \boldsymbol{m}\right) \|^2, \tag{2.14}$$

Following some manipulation, this reduces to

$$J = \sum_{k=1}^{n} \| \left(\boldsymbol{x}_0 - \boldsymbol{m}\right) \|^2 + \sum_{k=1}^{n} \| \left(\boldsymbol{x}_k - \boldsymbol{m}\right) \|^2, \tag{2.15}$$

Since the second term on the right in Eq. (2.15) is independent of $\boldsymbol{x}_0$, the overall expression is minimized by the choice of $\boldsymbol{x}_0 = \boldsymbol{m}$. The sample mean $\boldsymbol{m}$ is a zero dimensional representation of the data and thus does not tell anything about the variability in the data set. To capture some of this, each feature vector is projected onto a line running through the sample mean. If $\boldsymbol{e}$ is a unit vector in the direction of the line, then each vector $\boldsymbol{x}_k$ can be represented as

$$\boldsymbol{x}_k = \boldsymbol{m} + a_k \boldsymbol{e}, \tag{2.16}$$

where a_k is a scalar corresponding to the distance of any vector $\boldsymbol{x}_k$ from the mean $\boldsymbol{m}$. An optimal set of coefficients a_k can be found by once again minimizing the squared-error criterion function,

$$J(a_1, a_2, \ldots, a_n, \boldsymbol{e}) = \sum_{k=1}^{n} \| (\boldsymbol{m} + a_k \boldsymbol{e}) - \boldsymbol{x}_k \|^2. \tag{2.17}$$

Recognizing that $\|\boldsymbol{e}\| = 1$, partially differentiating with respect to a_k and setting the derivative to zero [18], one can determine that Eq. (2.17) is minimized if

$$a_k = \boldsymbol{e}^T (\boldsymbol{x}_k - \boldsymbol{m}). \tag{2.18}$$

Geometrically, this result can be interpreted to mean that the least-squares solution can be obtained by projecting the vector $\boldsymbol{x}_k$ onto a line in the direction $\boldsymbol{e}$ that passes through the sample mean.

When performing PCA, the fundamental question regards the selection of the best direction $\boldsymbol{e}$ for the line. Without delving too deeply into the mathematics, it can be shown that the solution involves so-called scatter matrix [18] or sample covariance matrix, which is

$$\boldsymbol{S} = \sum_{k=1}^{n} (\boldsymbol{x}_k - \boldsymbol{m}) (\boldsymbol{x}_k - \boldsymbol{m})^T. \tag{2.19}$$

To find the eigenvalues, set the determinant of the following matrix is equal to zero [19],

$$|\boldsymbol{S} - \lambda I| = 0. \tag{2.20}$$

The roots of the characteristic equation is the eigenvalues and the eigenvectors that correspond to those eigenvalues are found by substituting the eigenvalues back into (2.20) and solving.

Ultimately, it is found that the best one-dimensional projection of the data is obtained when data is projected onto a line through the sample mean in a direction of the eigenvector of the scatter matrix having the largest eigenvalue [18]. This result can be readily extended from a one-dimensional projection to a d-dimensional projection. In this case, each vector is written like so

$$\boldsymbol{x}_k = \boldsymbol{m} + \sum_{i=1}^{d'} a_{k,i}\boldsymbol{e}_i. \tag{2.21}$$

where $d' \leq d$. In this case, the squared-error criterion function

$$J_{d'} = \sum_{k=1}^{n} \| \left(\boldsymbol{m} + \sum_{i=1}^{d'} a_{k,i}\boldsymbol{e}_i \right) - \boldsymbol{x}_k \|^2, \tag{2.22}$$

is minimized when the vectors $\boldsymbol{e}_i$ are the d' eigenvectors of the scatter matrix having the largest eigenvalues [18]. The eigenvectors of the scatter matrix are orthogonal and they form a a set of natural basis vector to represent any feature vector $\boldsymbol{x}_k$. Ultimately, the coefficients a_i for each vector in Eq. (2.21) are known as its principal components.

During reconstruction $\boldsymbol{x}_k$ is mapped onto the training space and an approximation of $\boldsymbol{x}_k$ is computed as

$$\hat{\boldsymbol{x}}_k = \boldsymbol{m} + \sum_{i=1}^{d'} a_{k,i}\boldsymbol{e}_i. \tag{2.23}$$

Following reconstruction, the algorithm calculates the error between the projection and the original data $\boldsymbol{x}_k$ This is done by taking the two norm of the residual vector $\boldsymbol{r}$, where

$$\boldsymbol{r} = \hat{\boldsymbol{x}}_k - \boldsymbol{x}_k. \tag{2.24}$$

The Health Indicator, denoted as (HI_k), represents the error between the measured features and their expected values or projection onto the healthy features. The Health Indicator is calculated by finding the distance between the measurements and the approximation at each time t_k. This is defined as

$$HI_k = \boldsymbol{r}^T\boldsymbol{r} = \left(\hat{\boldsymbol{x}}_k - \boldsymbol{x}_k\right)^T \left(\hat{\boldsymbol{x}}_k - \boldsymbol{x}_k\right). \tag{2.25}$$

In a very basic sense if the error is small, the system is operating under normal conditions or is “healthy”. If the error grows, a problem may be developing.

CHAPTER 3: FAULT DETECTION IN POWER ELECTRONIC DRIVES

A method for the online detection of incipient faults in power electronic drives is presented in this chapter. The early detection of incipient faults is desirable in mission-critical applications such as shipboard propulsion drives, drives in nuclear power plants, etc. Techniques for monitoring the health of the two most sensitive components in power electronic systems, namely electrolytic filtering capacitors and controllable semiconductor switches (i.e. IGBTs and MOSFETs) are described [20]. The chapter begins with a brief discussion of the primary failure mechanisms for these components. It then presents a online technique designed to measure capacitor ESR, which is a key indicator of capacitor health. Section 3.3 uses the health monitoring algorithm described in Chap. 2 to extract important device features (i.e $V_{CE,ON}$ and R_{ON}) and compare them to healthy values recorded over a range of operating conditions. An experimental implementation in an IGBT-based drive is described and experimental results are shown. An on-line feature extraction scheme for MOSFETs is presented in Sec. 3.4. This scheme exploits the nature of carrier-based PWM in order simplify the measurement process of the key indicator, on-state resistance of the power MOSFET, $R_{DS,ON}$.

3.1 Background

Power electronic drives are becoming increasingly common in mission-critical applications. High-power, medium-voltage examples include propulsion motors aboard all-electric ships [21], large industrial motors such as those driving recirculation pumps in nuclear power plants [22], and large multi-megawatt, grid-tied inverters for wind turbines and photovoltaics [23]. Lower voltage drives are also becoming common in

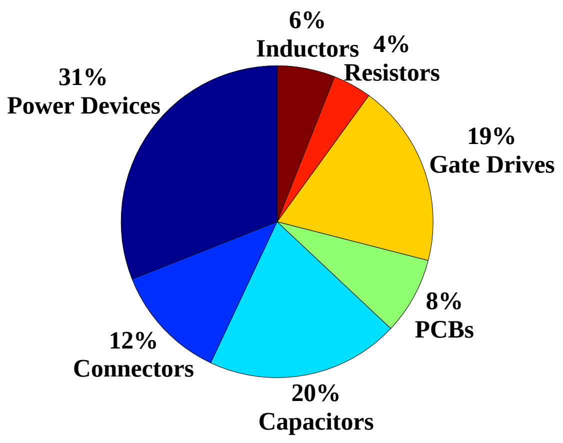


Figure 3.1: Pie chart of the distribution of component failure in a power electronic drive.

vehicles and aircraft [24]. This increasing dependence on power electronics in mission-critical applications has created a need for real-time techniques that can detect early stage faults.

Figure 3.1 shows the results of a survey that shows the distribution of component failures in power electronic drives. The two components most prone to failure in switch-mode drives are electrolytic filtering capacitors and power semiconductors [20]. One approach for on-line monitoring of capacitor health is presented based on monitoring the capacitor ESR, which is a key indicator of capacitor health. Techniques for monitoring capacitor health have been described in the literature, and are relatively easy to implement in digitally controlled drives [20, 25]. In the case of MOSFET and IGBT switches, however, the focus has been to develop fault-detection schemes that allow one to detect complete device failures such as shorted or open transistors [26], [27], [28], [29], [30]. In the event that catastrophic failure does not occur, various schemes allow operation in a degraded mode until service can be performed [27], [28], [31].

Although post-fault detection and fault tolerance will always be a necessary safety feature in any mission-critical drive, there is good reason to consider incipient fault detection and online condition monitoring. Consider, for instance, an IGBT that slowly wears over its lifetime because of thermal cycling. Ultimately, such a device will short circuit, which is desirable in a drive with $N + 1$ components. Eventually, however, the bond wires burn away, leading to an indeterminate failure state that may cause an arc flash and subsequent collateral damage to the rest of the circuit or to nearby humans [32].

The literature includes a number of articles describing the physics of failure in MOSFETs and IGBTs [33], [34], [35], [36]. Authors have described and verified the relationships between device health and device parameters (i.e. on-state resistance, threshold voltage, etc.). Until recently, however, very few works have described the use of such features for monitoring switch health in real-time in an operational circuit [37], [26].

A major limiting factor in the development of condition monitoring for switches has been the difficulty in obtaining the required signals, which are often very small and thus highly susceptible to corruption from switching noise [30]. Several useful health-related parameters are most easily accessed during switching when noise can be particularly problematic. Consider gate charge, for example, which can help to identify naturally occurring gate-oxide degradation [38], [39]. Figure 3.2 shows the noise corrupting the gate-current measurements in a 208 V motor drive. Clearly, switching noise makes it difficult to consistently extract the injected gate charge. Another issue apparent in Fig. 3.2 is that the signals of interest change rapidly and thus must be sampled at very high rates and/or carefully conditioned using well-designed analog circuits. Given the noise issues, such sampling and conditioning must be performed near the switch. The rest of this section will give background on the failure mechanisms that are associated with electrolytic bus capacitors and power semiconductors.

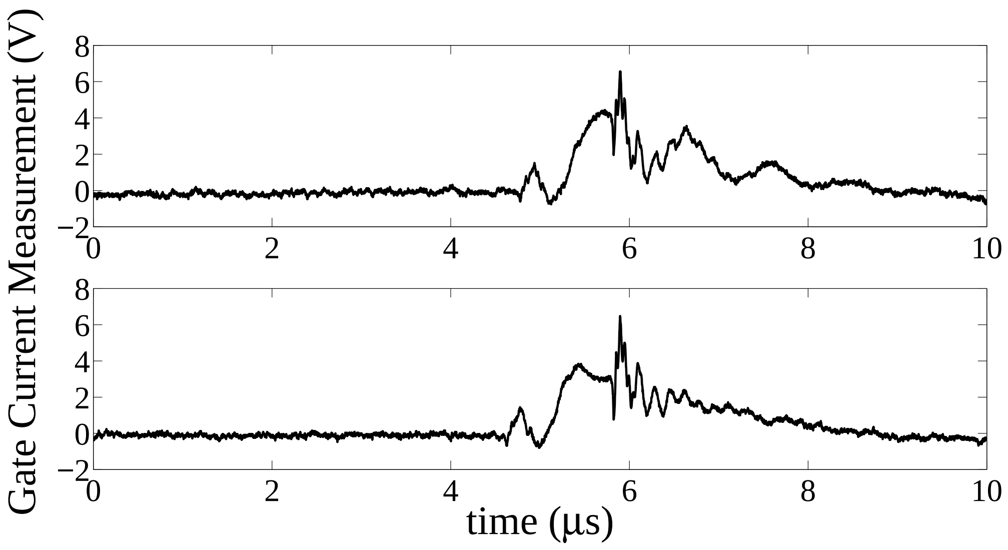


Figure 3.2: Output of a differential amplifier in a prototype drive measuring the voltage across a 10 Ω gate resistor during two different turn-on instances. Note that the gate current has a slightly different shape during the two instances as a result of switching noise.

3.1.1 Electrolytic Capacitor Failures

Electrolytic bus capacitors are one of the weakest links in motor drives [40]. Degradation occurs for various reasons, including thermal stresses, transients, reverse bias, and strong vibrations [41]. Thermal stress caused by high ambient temperatures and self-heating from ripple currents is the leading cause of premature failure. This thermal degradation can be neatly summarized. As a capacitor ages, heat from the environment and internal resistance causes the electrolyte to vaporize and escape through the end seal. This loss of electrolyte causes a corresponding increase in ESR. The relationship between these two quantities can be expressed empirically as [42]

$$\frac{ESR}{ESR_0} = \left(\frac{V_0}{V}\right)^2, \tag{3.1}$$

where ESR and ESR_0 are the corresponding resistance values at time t and time zero; similarly, V and V_0 represent the volume of electrolyte at the same two instants. As V decreases over time, Eq. 3.1 shows that there is a corresponding increase in

ESR. This higher resistance ultimately increases the rate of heat generation, which in turn increases the rate of vaporization. The lower volume of electrolyte then further increases ESR, which in turn increases heating. Ultimately, this positive feedback mechanism accelerates capacitor failure [42]. Because of its central role in the failure process, ESR is a reliable indicator of capacitor health [41, 42].

3.1.2 Power Semiconductor Failures

There are two different categories of failure mechanisms in power transistors. The first group includes intrinsic mechanisms related to the physics of the actual semiconductors. Some of the most prevalent examples are dielectric breakdown and electromigration [38, 43, 44]. The other group includes factors related to transistor packaging, such as contact migration, bond-wire lift, and die-solder degradation [45, 46, 47, 48].

3.1.2.1 Example Intrinsic Failure Mechanisms

Dielectric breakdown occurs when a strong electric field creates a current channel in an insulating medium [49]. During conduction, breakdown can occur between the gate and the drain/collector terminal or between the gate and the source/emitter terminal. Two different forms of breakdown are noted in the literature [38]. Catastrophic breakdown of the gate oxide typically results from severe thermal or electrical over-stress (i.e. electrostatic discharge, junction overvoltage, etc.). Time-dependent dielectric breakdown (TDDB), which occurs more gradually over time, refers to the natural breakdown of the gate oxide. TDDB is caused by chronic defect accumulation in the SiO_2 insulator during standard operation [39]. At least three defect-generation mechanisms have been identified [38]. These include impact ionization, hot carrier injection, and so-called trap creation attributed to the redistribution of hydrogen within the device. Before causing a complete failure, these naturally-occurring phenomena affect various device parameters [38]. For instance, they can change the gate leakage current. Similarly, any charges that become trapped in the gate oxide affect important device parameters, such as the threshold voltage V_T and the transconductance g_m [38]. Note that breakdown can also occur between the drain and the source or between the collector and emitter when the device is in a blocking state. Electromigration is another intrinsic failure phenomenon [43]. This mechanism results when high current densities within the silicon cause adjacent metal connections to migrate. If any voids form in the interconnects as a result of this process, then the connection may open circuit or the overall device resistance may increase [43, 50].

3.1.2.2 Example Extrinsic Failure Mechanisms

Various extrinsic failure phenomena have also been observed. Bond-wire lift is one of the most commonly occurring examples [46]. This phenomenon is a failure in the bond between the package wire and the silicon die. Thermal expansion mismatch between the bond solder and the attachment point is the primary cause. Bond-wire lift leads to higher junction temperatures (T_j), and thus it impacts parameters such as $V_{CE,ON}$ and on-state resistance [51]. Changes in these parameters can increase power dissipation, thus cause further increases in T_j. The resulting positive feedback mechanism ultimately leads to a complete device failure [46]. Die-solder degradation is another extrinsic issue [47]. Solder attaching the silicon die to the package heat sink can develop cracks and voids due to dissimilar thermal expansion in the two materials [48]. The junction-to-case thermal impedance, Ω_{jc}, thus increases, which leads to a higher T_j. A positive feedback mechanism is thus created once again. As in the case of bond-wire failures, this mechanism affects parameters such as the on-state resistance and $V_{CE,ON}$ [46]. A third extrinsic failure mechanism is contact migration. This phenomenon, which is related to electromigration, occurs when voids between external metal contacts and silicon cause metal to diffuse into the semiconductor. Ultimately, this diffused metal can short-circuit internal pn junctions. Before causing a complete failure, this mechanism impacts parameters such as $V_{CE,ON}$ and on-state resistance [46].

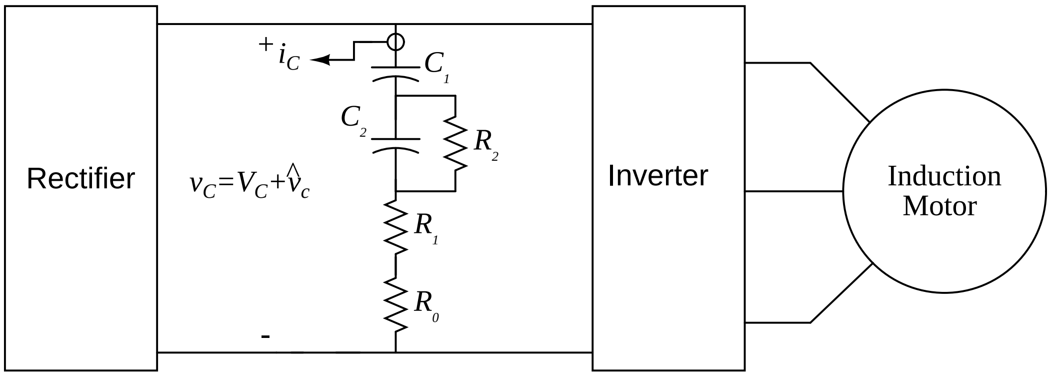


Figure 3.3: Block diagram of a typical VSD showing a comprehensive bus capacitor model. The capacitor current, i_C, and ac capacitor voltage, $\hat{v}_c$, are both measured.

3.2 Real-Time Condition Monitoring of Electrolytic Capacitors

The following section describes a method for the condition monitoring of electrolytic bus capacitors in real time.

3.2.1 Method

An on-line impedance spectroscopy technique was developed to monitor the value of capacitor ESR in a variable-speed ac drive of the form shown in Fig. 3.3. The block diagram includes a comprehensive low-frequency capacitor model developed from first principles [42, 52, 53]. C_1 is the rated terminal capacitance, R_1 is the resistance of the electrolyte, C_2 is the dielectric capacitance, R_2 is the dielectric loss resistance, and R_0 is the resistance of the foil, tabs, and terminals. This model has been developed specifically for investigation of capacitor performance over the frequency range relevant for dc bus capacitors (i.e. from dc to tens of kiloHertz) [42, 52]. The capacitor current i_C and the ac component $\hat{v}_c$ of the capacitor voltage are monitored.

Figure 3.4 outlines the proposed online ESR measurement process. Note that the procedure begins with measurements of i_c and $\hat{v}_c$ sampled at several kiloHertz. Figure 3.5 shows example waveforms. The frequency spectra of these quantities are then computed using the Fast Fourier Transform (FFT). Note that the signals are first

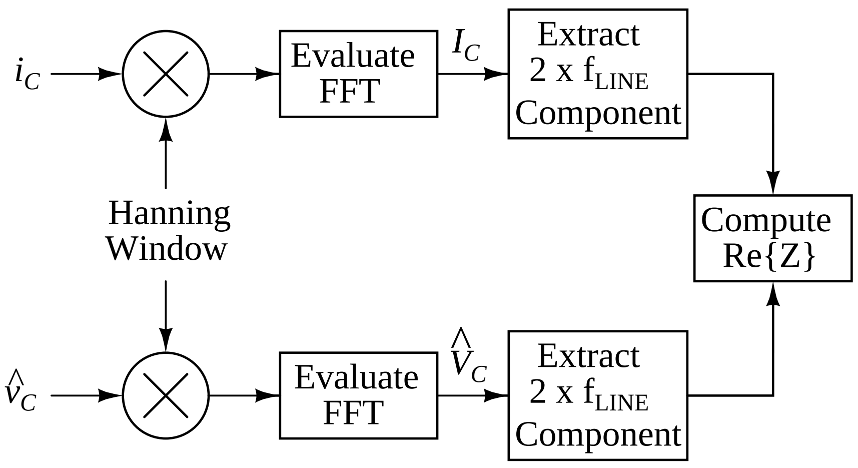


Figure 3.4: Block diagram showing the computation of the dc bus capacitor ESR.

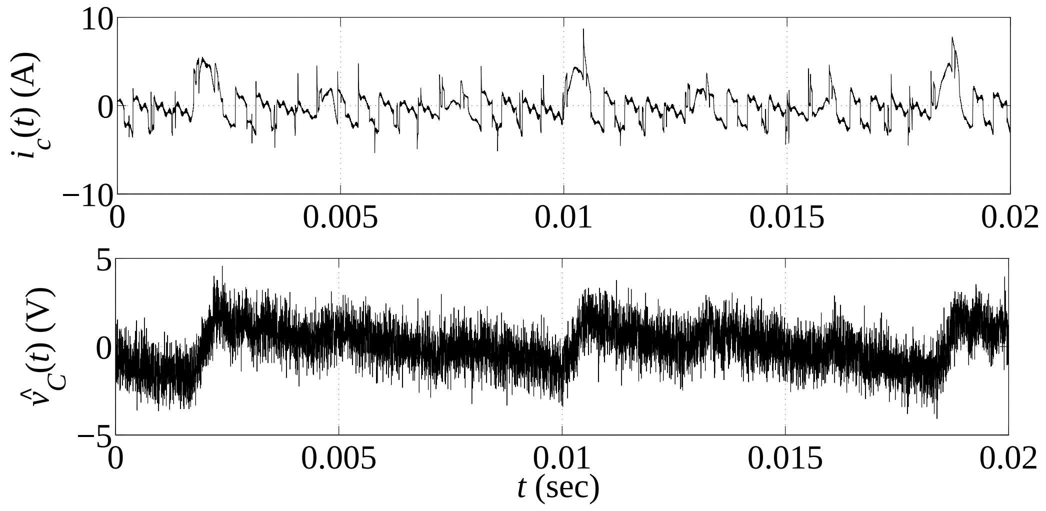


Figure 3.5: Measured capacitor current, i_C, and ac capacitor voltage, $\hat{v}_c$.

modulated with a Hanning window in order to minimize spectral leakage. Figure 3.6 presents the spectra corresponding to the waveforms of Fig. 3.5. ESR is ultimately computed by extracting the spectral lines closest to twice the ac line frequency (i.e. 120 Hz). In terms of the FFTs defined in Fig. 3.4, ESR is thus

$$ESR = \Re\left\{\frac{\hat{V}_C\left(2f_{line}\right)}{I_C(2f_{line})}\right\}. \tag{3.2}$$

Measurements are performed at twice the line frequency because physical arguments justify the existence of appreciable signals at that frequency [54]. Content at

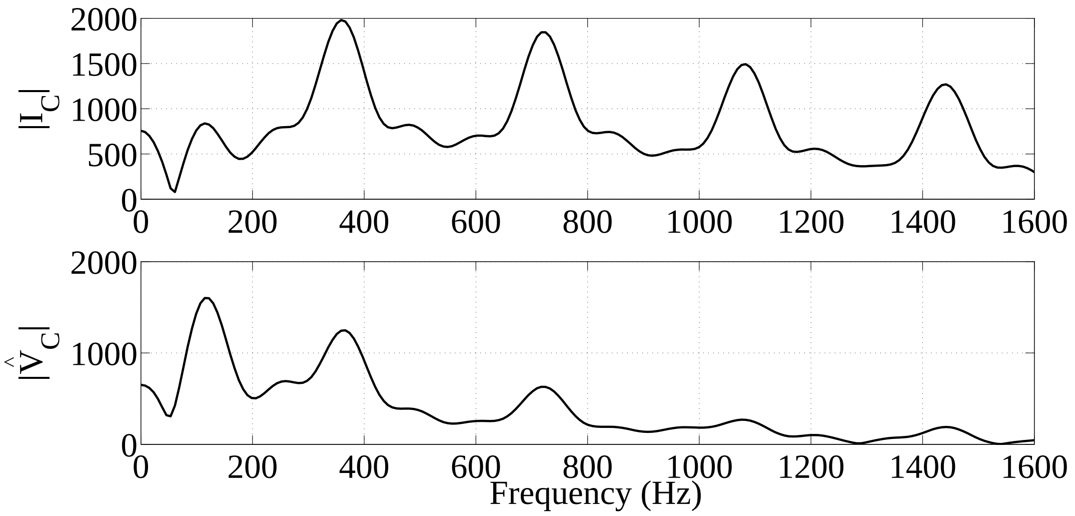


Figure 3.6: Magnitudes of the FFTs of the measured capacitor current (top) and the measured ac capacitor voltage (bottom).

twice the line frequency results from imbalances between the voltages applied at the input of the six-pulse rectifier. Note that measurements could be performed at any number of other frequencies at which appreciable signal content is expected, such as $6f_{line}$ [54]. It is desirable, however, to avoid frequency content that depends directly on motor speed. In terms of the comprehensive capacitor model, Eq. 3.2 yields

$$ESR = \Re\{Z_{cap}\} = R_1 + R_0 + \frac{R_2}{(1+\omega^2 R_2^2 C_2^2)}. \tag{3.3}$$

3.2.2 Experimental Demonstration

The approach shown in Fig. 3.4 was used to determine the ESR of a dc bus capacitor in a prototype drive of the form shown in Fig. 3.7. Accelerated age testing was performed on the prototype using a controlled temperature chamber. The capacitor ESR was found to increase over time as expected [41, 42, 52]. Table 3.1 shows the impedance of the dc bus capacitor before and after the accelerated age testing. The capacitor was aged by holding it at its rated temperature for 30 hours. The change in the real part of the capacitor impedance is quite noticeable. Note the results were validated using a BK Precision 889B LCR/ESR Meter.

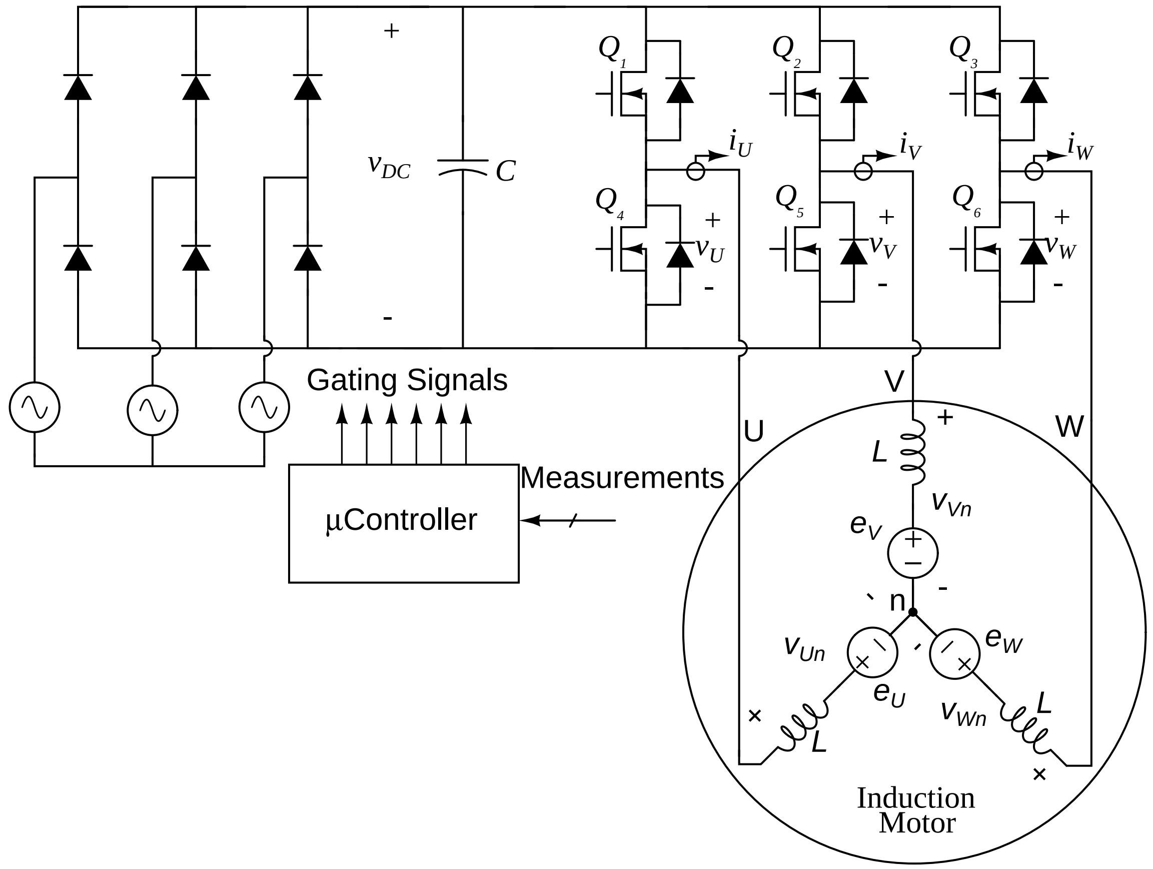


Figure 3.7: Partial schematic of the test circuit. A high common-mode amplifier measures the dc bus voltage v_C and a Hall-effect transducer measures i_C.

Table 3.1: Measured impedance of dc bus capacitor before and after accelerated aging.

	Before Aging		After Aging	
	Measured Using Proposed Method	Measured Using LCR/ESR Meter	Measured Using Proposed Method	Measured Using LCR/ESR Meter
$\|Z\|$ (Ω)	1.91	1.9	1.82	1.83
$\angle Z$ ($^\circ$)	-87.29	-87.33	-82.83	-82.08
$\Re\{Z\}$ (Ω)	0.09	0.089	0.226	0.251
$\Im\{Z\}$ (Ω)	-1.91	-1.9	-1.8	-1.81

3.3 Real-Time Switch Condition Monitoring of IGBTs

Real-time switch condition monitoring introduces new requirements into the design of the overall drive. First, one must obtain high-rate samples of switch terminal variables such as v_{CE}, i_G and i_C. Additionally, these signals must be processed and analyzed over time. This section shows a drive architecture that addresses these issues, and it presents an algorithm that can be included to track switch health. Note that the algorithm can be used in any drive capable of measuring the required quantities.

3.3.1 Drive Architecture and Sensing

Figure 3.8 shows a three-phase full-bridge motor drive that has been constructed such that it is possible to sample all of the required terminal variables. The key feature is the advanced gate drive concept from [27]. It is anticipated that fully developed versions of these devices could sample terminal quantities locally at a very high rate, and then transmit appropriately down-sampled versions over a fiber-optic link to a digital signal processor (DSP) performing overall control. The raw signals that are measured include the phase current (i_C), the collector-to-emitter on-state voltage ($v_{CE,on}$), the gate-to-emitter voltage (v_{GE}), gate charge indicator (Q_i), case temperature (T_C), and ambient temperature (T_A).

From a health monitoring perspective, it is critical to sample raw signals such as $v_{CE,ON}$ and i_C fast enough to be able to extract features such as V_J and R_{ON}. Sampling rate is one issue that must be addressed regardless of the actual circuit architecture. In general, sampling must be performed at a rate above the switching frequency. This may limit application in lower power drives with switching frequencies in excess of 10 kHz. In high power, medium voltage (MV) drives, which are more likely to require condition monitoring, power-dissipation limits tend to cap switching frequencies at values on the order of 1kHz [55].

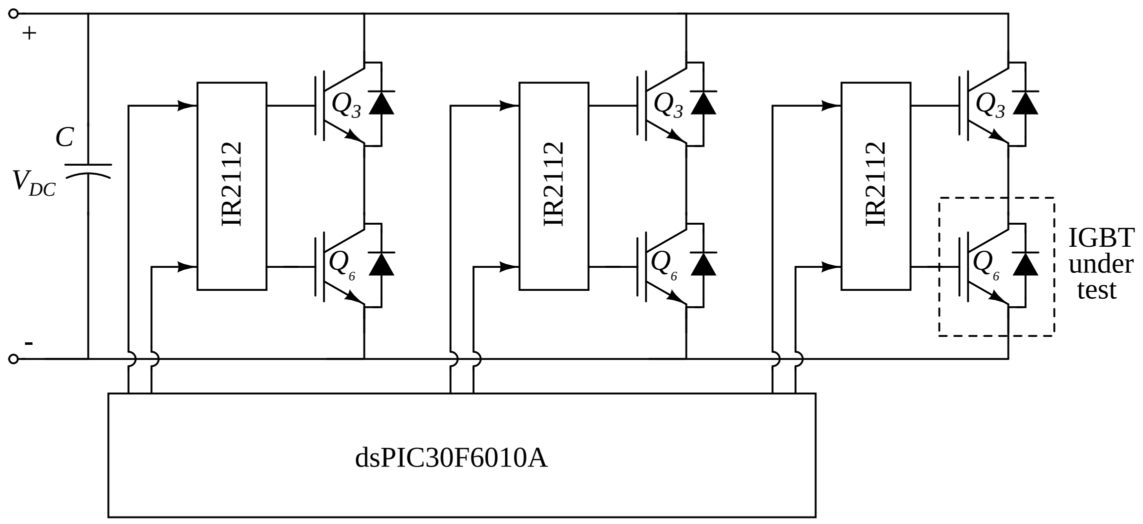


Figure 3.8: Schematic of custom-built IGBT based motor drive used for driving a $\frac{1}{2}$hp induction motor. The control signals are supplied using a dsPIC30F6010A.

At these switching frequencies, it likely that device variables could be sampled fast enough to extract the relevant features. The raw signals are measured using a DT9816. DT9816 is a low-cost 16-bit simultaneous sampling USB data acquisition board which is used to sample 6 analog signals at 50 kHz each [56].

3.3.1.1 Sensing Circuits for Switch Condition Monitoring

Measurement circuits and sensors are used to measure the raw signals that are used to extract the relevant features. The phase current, i_C, is easily measured using the LTS 6-NP which is a Hall-effect transducer. The collector-to-emitter voltage, on the other hand, is difficult to measure directly because the required amplifier would be exposed to common-mode swings on the order of several hundred volts and the subsequent analog-to-digital converter would have difficulty measuring the very low on-state voltage with adequate resolution. To overcome these issues, v_{CE} is measured in the on-state using the desaturation detection circuit included in gate drives to detect if an IGBT is no longer in the saturation region of operation.

The onstate collector-to-emitter voltage, $V_{CE,ON}$ is measured using the circuit shown in Fig. 3.9. The measured voltage, v_X, which is referred to the high volt-

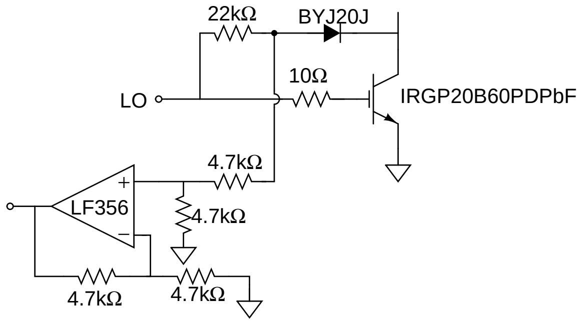


Figure 3.9: Circuit used for the measurement of on-state collector-to-emitter voltage, $V_{CE,ON}$ in experimental drive.

age side of the drive, is connected to a simple operational amplifier subtracter circuit. When the IGBT is conducting, v_X has the form

$$v_X = V_D + v_{CE}(i_c), \tag{3.4}$$

where V_D is the forward voltage drop across the diode in the desaturation circuit. Since the voltage at the output of the differential amplifier in Fig. 3.9 is referenced to the high voltage bus, the circuit shown in Fig. 3.11 is used to transmit its output to the data-acquisition system. Similar isolation would be required in an FPGA-based design, but its exact location in the signal path would depend upon the specifics of the design.

The optical transmission circuit from [27] maintains appropriate isolation when measuring $V_{CE,ON}$. The main component of the optical transmission circuit is the HCNR201 which is a high-linearity wide-bandwidth analog optocoupler consisting of a high-performance AlGaAs Light Emitting Diode, LED that illuminates two closely matched photodiodes, PD1 and PD2. The input photodiode, PD1, can be used to monitor, and therefore stabilize, the light output of the LED. The non-linearity and drift characteristics normally associated with LED can be virtually eliminated. The output photodiode, PD2, produces a photocurrent that is linearly related to the light output of the LED. This allows for a highly stable linear gain characteristic of the

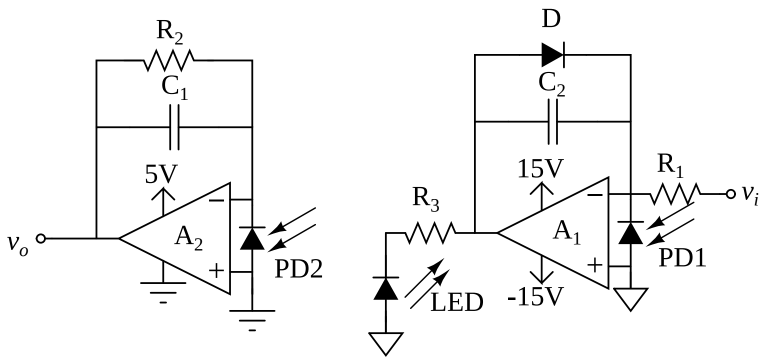


Figure 3.10: Analog circuit used to transmit signals from across isolation barrier.

overall optical transmission circuit [57].

The operation of the basic circuit may not be immediately obvious just from looking at Fig 3.11, particularly the input part of the circuit. Stated briefly, the operational amplifier A_1 adjusts the LED current, and therefore the current in PD1, I_{PD1}, to maintain its + input terminal at 0 V. Since the + input of A_1 is at 0 V, the current through R_1, and therefore I_{PD1} is [57]

$$I_{PD1} = \frac{v_i}{R_1}. \tag{3.5}$$

Note that I_{PD1} depends only on the input voltage and the value of R_1 and is independent of the light output of the LED.

There is a very linear relationship between the input optical power and the output current of a photodiode. Therefore, by stabilizing and linearizing I_{PD1}, the light output of the LED is also stabilized and linearized. Since light from the LED also falls on both of the photodiodes, I_{PD2} is stabilized as well. The physical package construction determines the relative amounts of light that fall on the two photodiodes and, therefore, the ratio of the photodiode currents. This results in very stable operation over time and temperature. The photodiode current ratio is expressed as a constant, K, where [57]

$$K = \frac{I_{PD2}}{I_{PD1}}. \tag{3.6}$$

The operational amplifier, A_2 and resistor R_2 form a trans-resistance amplifier that converts I_{PD2} back into a voltage, v_o, which is [57]

$$v_o = I_{PD2}R_2. \tag{3.7}$$

Combining Eq. (3.5)-(3.7)yields a transfer function relating the output voltage to the input voltage,

$$v_o = K\left(\frac{R_2}{R_1}\right)v_i. \tag{3.8}$$

Therefore the relationship between v_i and v_o is constant, linear, and independent of the light output characteristics of the LED. The optical transmission circuit's gain can be adjusted by simply changing the ratio of R_2 to R_1. The parameter K can be thought of as the gain of the optocoupler and for this HCNR201 it is approximately 1 [57].

Figure 3.11 shows the unity gain optical transmission circuit that was designed to measure $V_{CE,ON}$. Each of the measurement circuits that are referenced to the dc bus of the motor drive requires an optical transmission circuit to transmit the signal back to be acquired using the data acquisition board.

Figure 3.12 shows the circuit used to directly extract the gate charge parameter Q_G. Note that a high-bandwidth differential amplifier measures the voltage across the gate resistor to obtain i_G. The subsequent signal is amplified and passed to an operational transconductance amplifier (OPA660) configured as a nano-second pulse integrator. The output voltage of the pulse integrator is the time integral of its input voltage which is calculated as

$$V_O = \frac{g_m}{C}\int_0^T V_{BE}dt. \tag{3.9}$$

whereV_O is the output voltage, V_{BE} is the base-to-emitter voltage, g_m is the transconductance of the diamond transistor, T is the integration time, and C is the integration capacitance. A sample-and-hold (AD781) triggered by the data-acquisition system

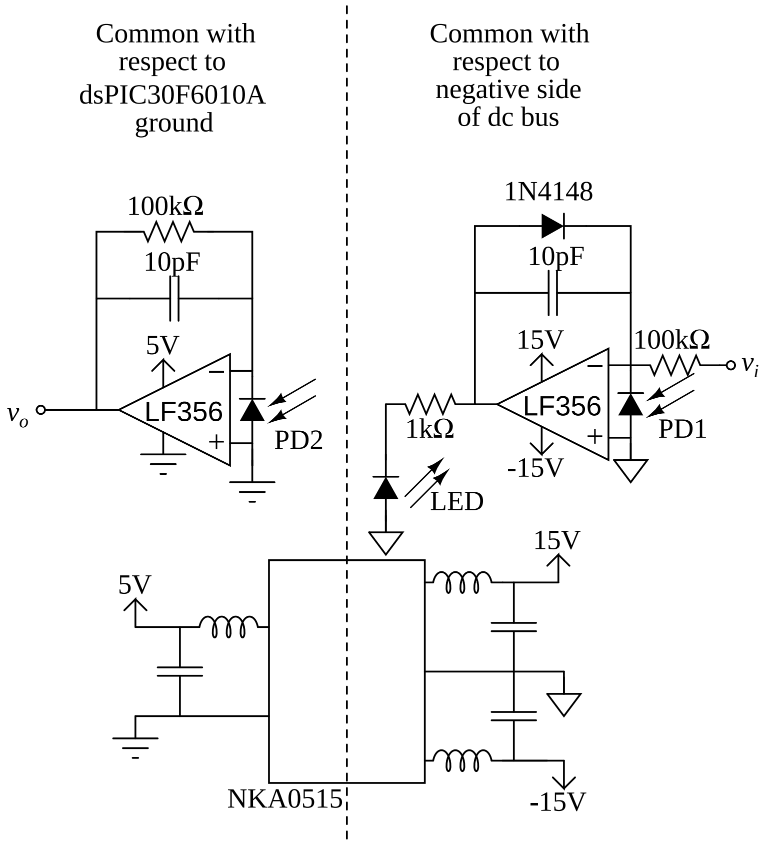

Common with respect to dsPIC30F6010A ground
Common with respect to negative side of dc bus
100kΩ
10pF
5V
LF356
v_o
PD2
1N4148
10pF
15V
100kΩ
v_i
LF356
PD1
1kΩ
-15V
LED
15V
5V
NKA0515
-15V

Figure 3.11: Optical transmission circuit used to provide isolation between the low voltage data-acquisition system and measurements recorded with respect to the high voltage bus. The NKA0515 provides isolated dc supplies for all of the measurement circuits included here.

acquires the output voltage of this circuit.

Figure 3.13 shows the output of the integrator as well as the trigger signal and the corresponding sample-and-hold output. Note that the integrator output rises during the device turn on, i.e. while v_{GE} is rising. Once i_G has fallen to zero and v_{GE} has become steady, the integrator capacitor begins to slowly discharge through the output resistance of the OPA660. Note that a sample is acquired during this discharge in order to avoid the switching noise at device turn-on. Figure 3.14, which shows the integrator output signal during four different device turn-ons, provides the rationale. Note that each signal is clearly impacted differently by the common-mode switching noise. Once this has dissipated, however, each output is quite consistent. The output at the point labeled 'Sample' is taken as the feature Q_G. Further discussion follows in Sec. 3.3.3.4, which shows that the output at the sampling point varies significantly as the gate oxide exhibits signs of degradation.

The next useful feature that must be measured is is V_{GE}. Figure 3.15 shows the circuit measuring the gate-to-emitter voltage, which is simply a differential amplifier. A simple voltage divider was used to reduce the size of gate voltage so that the output of the operational amplifier would be in a range so that a circuit similar to the one in Fig. 3.11 is able to transmit the output back to the data-acquisition system. Samples are averaged over each conduction interval to compute $V_{GE,ON}$ for that interval.

A solid-state sensor measures the ambient temperature, and a thermocouple measures the case temperature of the transistor as shown in Fig. 3.16. These measurements are also averaged over an interval of 1 second.

3.3.1.2 Feature Extraction

Once the raw signals have been collected by the data acquisition board, the relevant features must be extracted from them. The features that are extracted include $I_{C,rms}$, V_J, R_{ON}, and Q_i. Several key features are obtained by processing the measurements of i_C and v_{CE}. Section 3.3.1.1 shows the approach for measuring V_{CE}. Although the

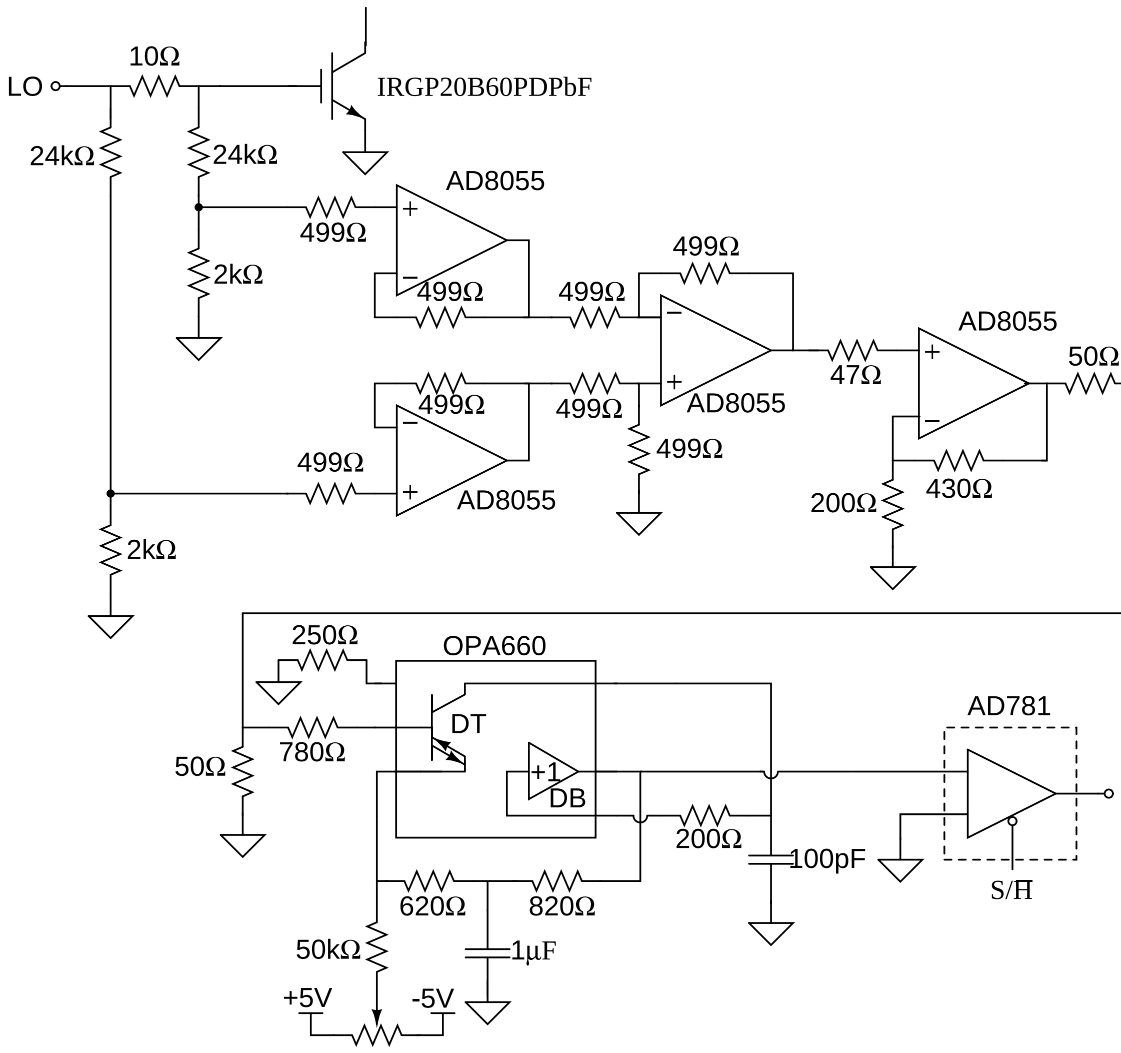


Figure 3.12: Circuit used to extract the gate charge injected during turn-on. The OPA660 is an operational transconductance amplifier that integrates using the 100pF capacitor. The trigger signal for the AD781 sample-and-hold is provided from the microcontroller using an optocoupler.

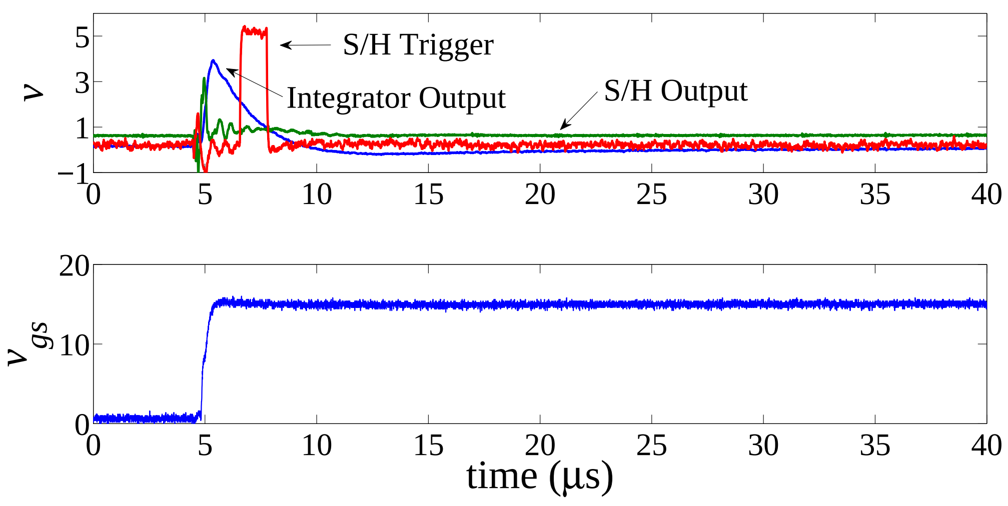


Figure 3.13: Top trace: Integrator output, sample-and-hold output, and trigger signal during a device turn-on. Note the effect of switching noise. Bottom trace: The corresponding gate-to-emitter voltage.

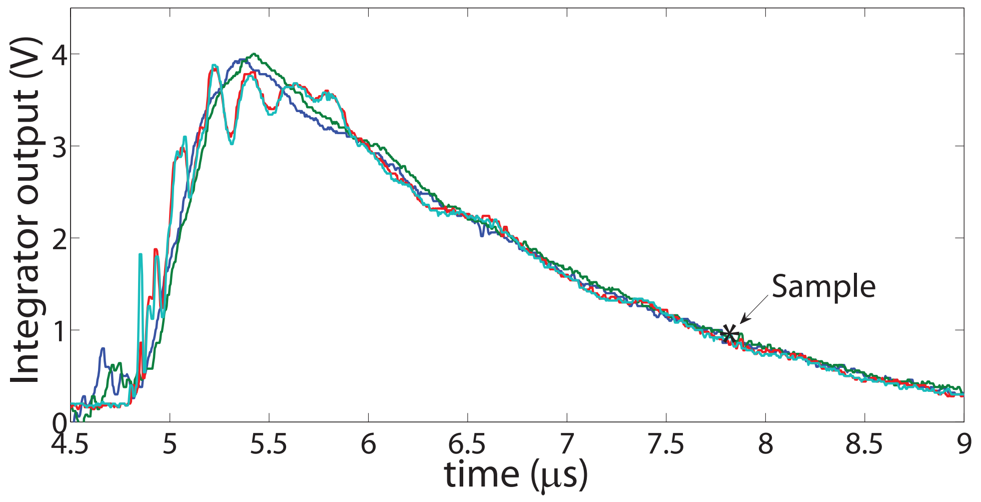


Figure 3.14: Integrator output during four different device turn-ons. Note that when the signals are rising, i_G is being integrated. The sampling point is shown.

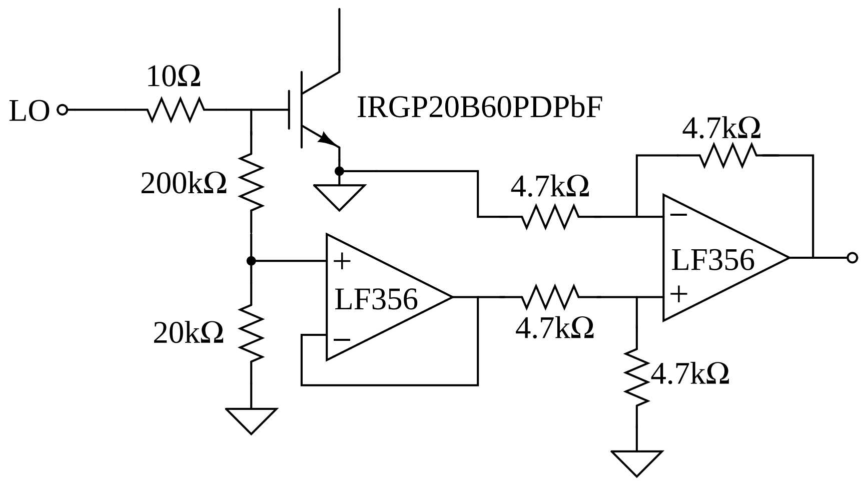


Figure 3.15: Circuit used for the measurement of gate-to-emitter voltage, V_{GE} in experimental drive.

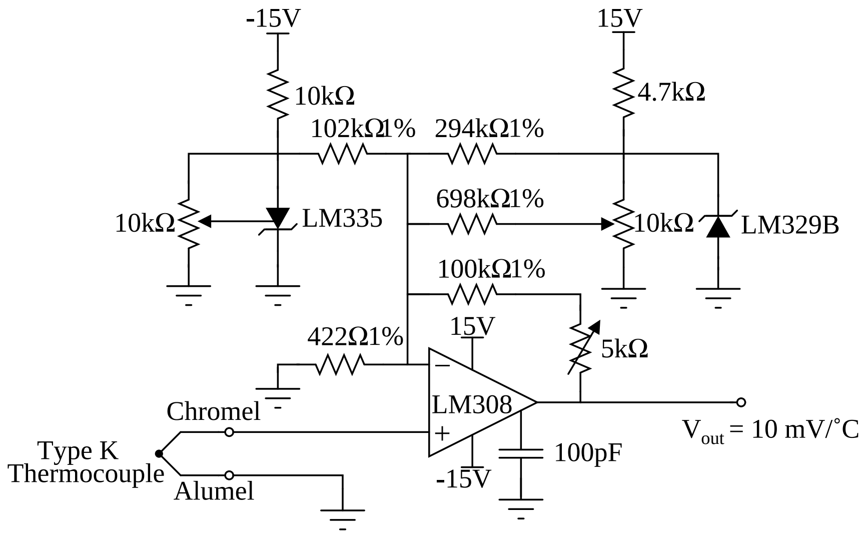


Figure 3.16: Circuit used for the measurement of IGBT case temperature in experimental drive.

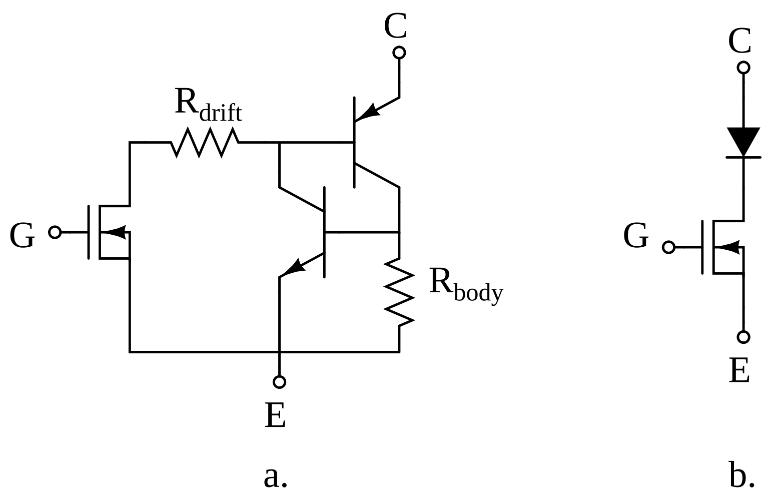


Figure 3.17: Measurement of phase current I_C and v_{CE} for the low side IGBT under test. When v_{CE} is negative, the anti-parallel diode is conducting.

literature suggests that V_{CE} can be directly applied in fault detection, this is only true under carefully controlled conditions that do not necessarily apply in the field. In general, the circuit model as shown in in Fig 3.17b for a conducting IGBT is a diode in series with a power MOSFET, meaning that

$$V_{CE,ON} = V_J + i_C R_{ON}, \tag{3.10}$$

where R_{ON} is the resistance of the MOSFET channel and V_J represents the combination of the voltage drops across the pn junction and the drift regions shown in in Fig 3.17a [54].

Given that most incipient faults have only a small impact on $V_{CE,ON}$, this result suggests that it may difficult to distinguish between changes in i_C and true fault conditions. This problem is further compounded by the fact that both V_J and R_{ON} are affected by temperature [54]. Feature extraction is particularly problematic in variable-speed ac drives in which the current and $V_{CE,ON}$ are subject to continuous variation as seen in Fig. 3.18. Figure 3.18 illustrates the effect of Eq. 3.10 in an operational drive. Note that $V_{CE,ON}$ essentially follows the current when the IGBT is conducting. To isolate the effect of the current, the two parameters are estimated separately in Eq. 3.10 using a least squares approach.

Samples of the current i_c and the voltage v_{CE} during conduction as shown in Fig. 3.18 are taken for 1 second and a linear fit on the data is performed. Figure 3.19 the data collected with the measurement circuits showing V_{CE} and I_C for 4 cycles of the current. Figure 3.19 shows one second of relevant measured points and the best-fit line corresponding to the data shown in Fig. 3.19. The estimated y-intercept (V_J) and the slope (R_{ON}) are ultimately extracted as features. The value R_{ON} for this particular second of data is 170.6 mΩ and V_J is 1.053 V.

These parameters and the corresponding rms current are extracted. The root mean square of the phase current is calcualted over the each individual period of a 60 Hz sinusoid and is defined as

$$I_{C,rms} = \sqrt{\frac{1}{N}\sum_{1}^{N} \|i_c\|^2}, \tag{3.11}$$

where N is the number of points in a 60 Hz period which for a sampling frequency of 50 kHz is 833. The mean of the rms current is then computed over one second intervals. Ultimately, estimates of T_C, T_A, Q_G, I_C, V_J, and R_{ON} over a one second interval are used to assemble a feature vector at time t_k. Threshold voltage and possibly other features could be included in the future.

3.3.2 Health-Monitoring Algorithm for Power Switches

Figure 3.20 shows a health-monitoring algorithm for power switches. This algorithm is designed to distinguish between the effects of true faults and other naturally occurring phenomena such as changes in temperature and operating conditions. Various algorithms can be used. In general, multiple samples of the input data are needed to calculate any feature. As a result, indicators are extracted at a rate well below the sampling frequency and in this case at a interval of one second. As shown in Fig.3.20, various feature estimates are ultimately combined with measurements of the case temperature and passed to the remainder of the algorithm.

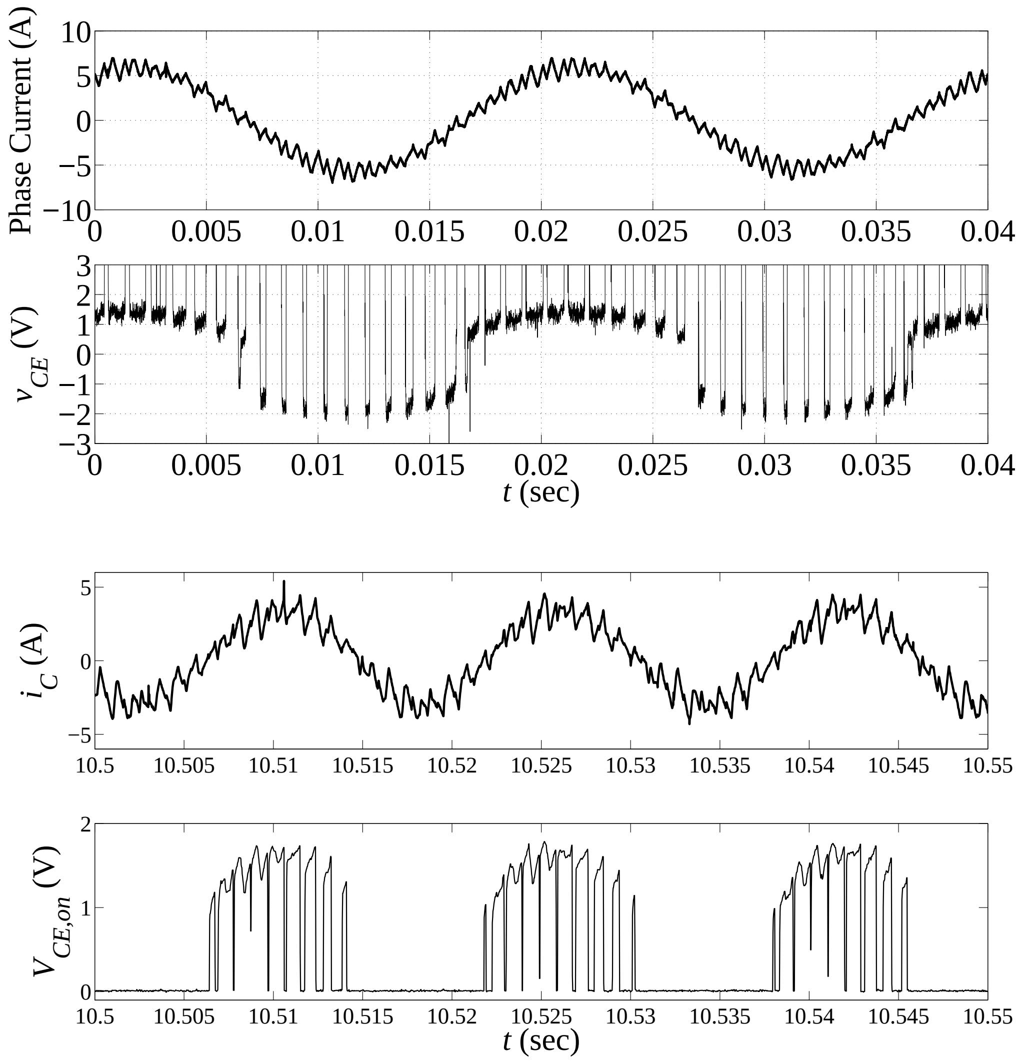


Figure 3.18: Oscilloscope measurement of phase current I_C and v_{CE} and measurements of the phase current I_C and $v_{CE,on}$ with the sensing circuits for the low side IGBT under test. When v_{CE} is negative, the anti-parallel diode is conducting.

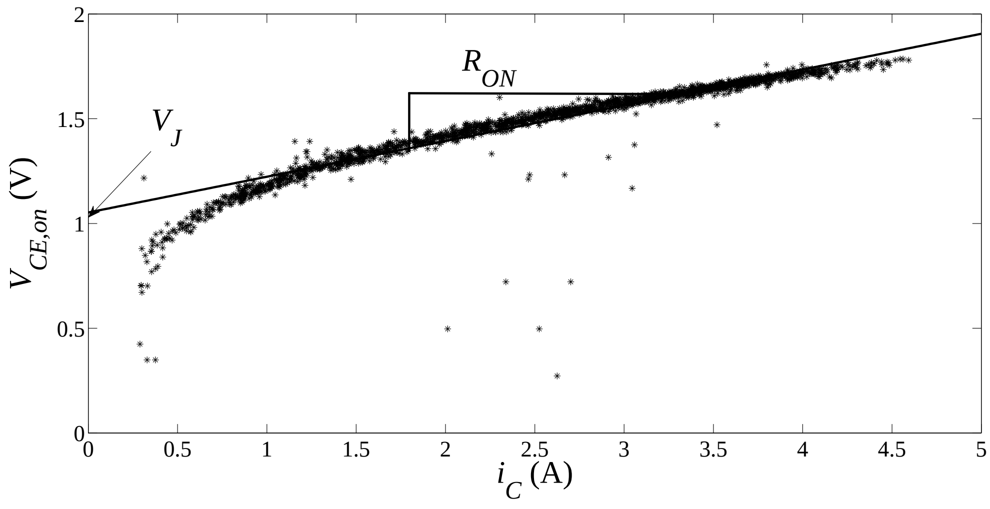
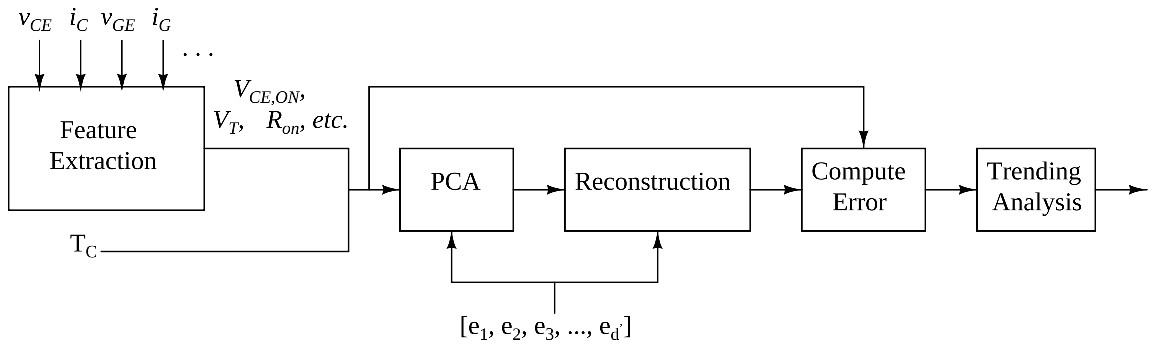


Figure 3.19: Plot showing v_{CE} versus i_C over one second. Note that a fit is performed to extract V_J and R_{ON}.



v_{CE} i_C v_{GE} i_G . . .
Feature Extraction
$V_{CE,ON}$, V_T, R_{on}, etc.
T_C
PCA
Reconstruction
Compute Error
Trending Analysis
[e_1, e_2, e_3, ..., $e_{d'}$]

Figure 3.20: Generalized condition-monitoring algorithm for an operational IGBT based on principal-components analysis (PCA). The $\boldsymbol{e}_i$ are based on healthy conditions. New input signals and parameters can be added as needed.

Feature vectors computed at the time t_k are grouped into a column vector $\boldsymbol{x}_k$. In our current implementation in the IGBT based drive, this vector includes the following features, each measured once per line cycle:

- Voltage drops across the pn junction and drift regions, V_J
- On-state resistance of the MOSFET channel, R_{ON}
- Injected gate charge at turn-on, Q_G
- Root-mean-squared value of the collector current, I_C
- Average on-state gate-to-emitter voltage, $V_{GE,ON}$
- Ambient temperature, T_A
- Case temperature, T_C

Assuming that other parameters will be added, the general assumption is made that $\boldsymbol{x}_k$ has a length d. The basic approach of the principal-component-based algorithm is to compare each measurement $\boldsymbol{x}_k$ to an expectation. This expected vector is computed by projecting $\boldsymbol{x}_k$ onto a vector space created using healthy features. These healthy values are learned during a training phase in which the drive is new and presumably in good condition. During training, the healthy vectors are decomposed into a small set of characteristic vectors that best describes the distribution of the healthy parameters. During operation, each measured vector is projected onto this space.

The training space includes features recorded over a range of expected operating conditions. In all, there are M such vectors and they are denoted as $\Gamma_1, \Gamma_2, \Gamma_3, , \Gamma_M$. These training vectors are subject to a PCA in which one seeks a set of orthonormal vectors $\boldsymbol{e}_i$ that best describe the distribution of the data. The j-th training vector can thus be expressed as

$$\boldsymbol{\Gamma}_j = \boldsymbol{m} + \sum_{i=1}^{d'} a_{j,i}\boldsymbol{e}_i, \tag{3.12}$$

where $\boldsymbol{m}$ is the sample mean, i.e.

$$\boldsymbol{m} = \frac{1}{M}\sum_{j=1}^{M} \boldsymbol{\Gamma}_j. \tag{3.13}$$

Note that the distribution of the data is best described using $d' \leq d$ orthonormal vectors [18, 16]. During the training phase, one calculates these vectors by minimizing the squared-error criterion function

$$J = \sum_{j=1}^{M} \left\| \left(\boldsymbol{m} + \sum_{i=1}^{d'} a_{j,i} \boldsymbol{e}_i \right) - \boldsymbol{\Gamma}_j \right\|^2 . \tag{3.14}$$

[18] and [16] show that the $\boldsymbol{e}_i$ correspond to the eigenvectors of the sample covariance matrix which is

$$\boldsymbol{S} = \sum_{j=1}^{M} \left(\boldsymbol{\Gamma}_j - \boldsymbol{m} \right) \left(\boldsymbol{\Gamma}_j - \boldsymbol{m} \right)^T . \tag{3.15}$$

The actual $\boldsymbol{e}_i$ are the eigenvectors corresponding to the d' largest eigenvalues of $\boldsymbol{S}$ [18, 16]. Ultimately, this process yields a compact basis that efficiently encodes the relevant features of a healthy switch over a range of expected operating conditions.

During normal operation of the drive, the prognostic algorithm in Fig. 3.20 projects the features measured at time t_k onto the space spanned by the $\boldsymbol{e}_i$. This projection is performed by the block labeled PCA, which computes the coefficients

$$a_{k,i} = \boldsymbol{e}_i^T \left(\boldsymbol{x}_k - \boldsymbol{m} \right) . \tag{3.16}$$

The next block uses these coefficients to reconstruct an approximation of $\boldsymbol{x}_k$. The resulting estimate is thus denoted as

$$\hat{\boldsymbol{x}}_k = \boldsymbol{m} + \sum_{i=1}^{d'} a_{k,i} \boldsymbol{e}_i . \tag{3.17}$$

Following reconstruction, the algorithm calculates the two-norm of the residual vector $\boldsymbol{r} = \hat{\boldsymbol{x}}_k - \boldsymbol{x}_k$. This quantity represents the error between the measured features and their projection onto the healthy features. If the error is small, the switch is healthy; if the error grows, a problem may be developing. The final block monitors for such variations.

3.3.3 Incipient Fault Detection in IGBTs

The condition monitoring algorithm described in Section 3.3.2 has been tested using an IGBT-based drive of the form shown in Fig. 3.8. The switching frequency is set to approximately 1 kHz in order to mimic operation in a medium voltage drive. This section begins with a description of the methods used to degrade the IGBTs and also shows testing results.

3.3.3.3 IGBT Degradation Methods

Several methods have been presented in the literature to degrade or artificially age power semiconductors [35, 34]. Electrical and thermal stress are the two most common mechanisms by which power semiconductors are degraded.

Several accelerated thermal ageing schemes have been shown in the literature [34, 45, 58, 59]. Chronic temperature overstress and thermal cycling are the most prevalent thermal stress methods. Thermal overstress subjects the transistor to high temperatures for extended periods of time and can be artificially created by switching the transistor without proper heat sinking. The transistor is operated in extreme temperatures well beyond the safe operating area of the device. The transistor is aged by placing it into a Class-A amplifier and the case temperature is monitored with a temperature sensor. A controller is used to keep the temperature of the case at a constant value for an extended period of time.

Figure 3.21 shows a schematic of the testing setup that is used to age the IGBTs that are to be used in the motor drive for fault detection. The gate signal in this circuit is a 1 kHz square waveform with a 50% duty ratio, D and a 15 V amplitude. The collector current, I_C, is set using the load resistor, R_{Load}, such that current flowing through the IGBT creates the desired power dissipation in the device. The power dissipated by the IGBT is defined as

$$P_{diss} = V_{CE,on} I_C D + P_{sw}, \tag{3.18}$$

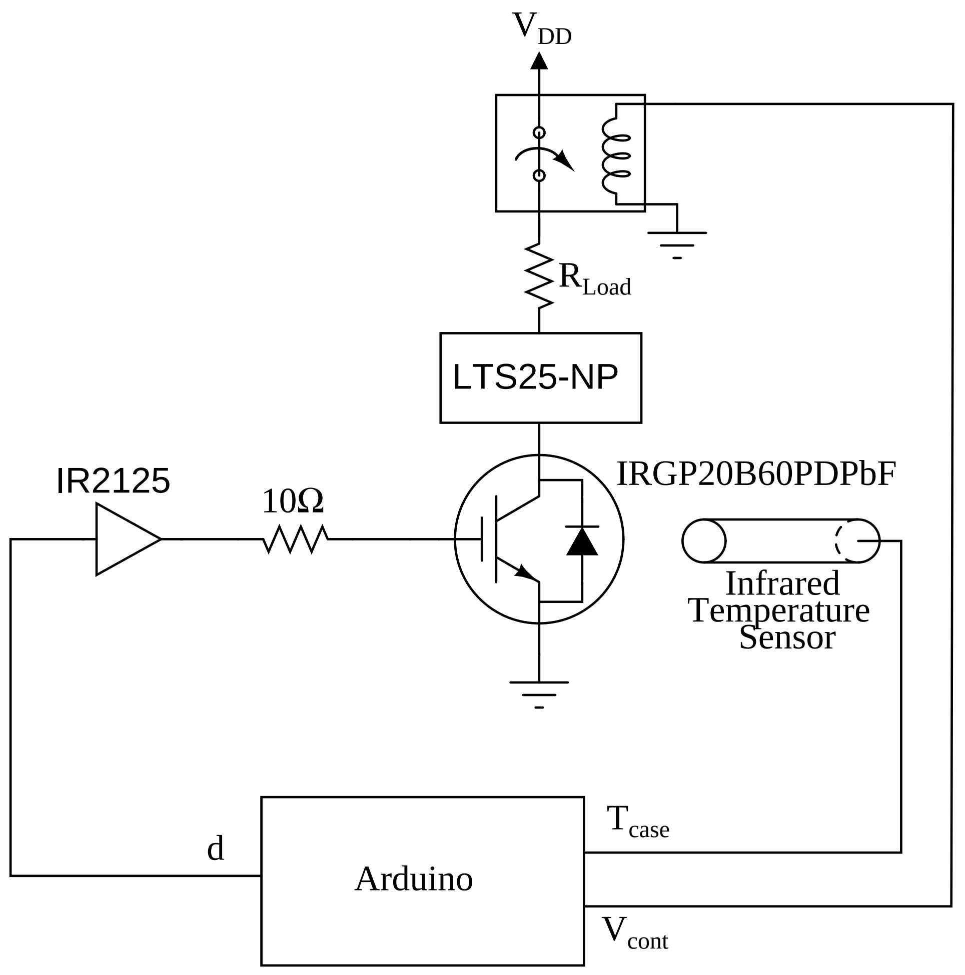


Figure 3.21: Circuit used to perform accelerated aging of the IGBT.

where $V_{CE,on}$ is the on-state collector-to-emitter voltage and $P_{sw}$is the power loss due to switching which is minimal due to the low switching frequency.

The thermal circuit in Fig. 3.22 models the thermal conduction of the IGBT in the degradation circuit. Note that without the power dissipated in the IGBT, P_{diss} due to thermal conduction is

$$P_{diss} = \frac{T_j - T_c}{\theta_{jc}}, \tag{3.19}$$

where T_j is the junction temperature, T_c is the case temperature, and θ_{jc} is the junction-to-case thermal impedance in °C$/W$ specified by the manufacturer.

$T_{j,set}$ is the desired junction temperature during the first degradation test and it is set to 125% of the maximum operational junction temperature defined by the manu-

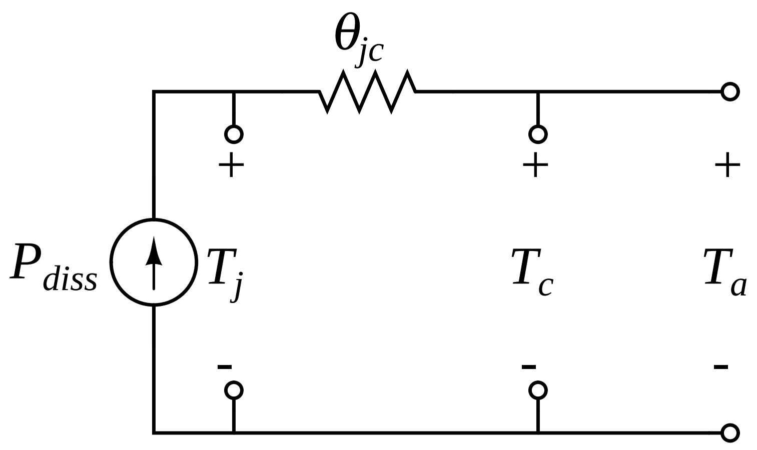


Figure 3.22: Thermal equivalent circuit for IGBT when placed in thermal degradation circuit.

facturer. For the IRGP20B60PDPbF the maximum operational junction temperature is 150°C so therefore $T_{j,set}$ is 187.5°C. Since the junction temperature is not easy to measure the case temperature, $T_{c,set}$ is actually set to desired value using Eq.(3.18) and Eq.(3.18) and is defined as

$$T_{c,set} = T_{j,set} - P_{diss}\theta_{jc}. \tag{3.20}$$

During degradation a microcontoller, the Arduino, controls the case temperature at $T_{c,set}$ and also monitors I_C to insure the IGBT does not latch up. If the IGBT latches up during degradation V_{DD} is disconnected using the relay which is also controlled by the Arduino and the junction temperature is allowed to fall. The degradation is restarted again after the junction temperature has been reduced but with $T_{j,set}$ reduced by 10°C. The failure mechanisms that can be caused by a thermal overstress include die-attach failures [59], gate latch-up [58], and time-dependent dielectric breakdown.

Thermal cycling is also a prevalent accelerated aging methodology. This method subjects power transistors to rapid changes in temperature differences causing thermal expansion and contraction. This method employs the same circuit as shown in Fig. 3.21 but the device is operated only in the safe operating area. The temperature of the transistor is cycled using the degradation circuit in the following steps:

- Hold case temperature at 50°C for 5 minutes.
- Increase case temperature to 80°C for 5 minutes.
- Increase case temperature to 110°C for 10 minutes.
- Increase case temperature to 140°C for 20 minutes.
- Decrease case temperature to 110°C for 10 minutes.
- Decrease case temperature to 80°C for 5 minutes.
- Decrease case temperature to 50°C for 5 minutes.

The failure mode associated with thermal cycling is the failure of the transistor to switch as a result of the increased resistance of the device. As a result of the increased resistance, the drain current dropped and the transistor fails to switch. Die solder degradation and wire lift are associated with thermal cycling.

An electrical failure was developed by using a method to mimic the degradation of the gate dielectric. Figure 3.23 shows the test setup for the gate dielectric wearout detection. When a gate dielectric breakdown occurs a substantial leakage current is seen during the on and off state of an IGBT. This condition is created artificially by adding in parallel a resistor R_{leak} between the gate and emitter during operation. A 47k Ω, 4.7k Ω, 470 Ω, and 47 Ω resistor was placed between the gate and emitter to simulate and increase in the a leakage gate current. An increase in the gate leakage current which occurs after gate dielectric wear out can be seen as increase in the gate charge indicator described in Sec. 3.3.1.1.

3.3.3.4 Experimental Results

To illustrate the effectiveness of the proposed algorithm, two different early stage faults were introduced into a single IGBT switch. First, the monitored switch was removed and subjected to an accelerated aging test to produce an early stage die-attach fault.

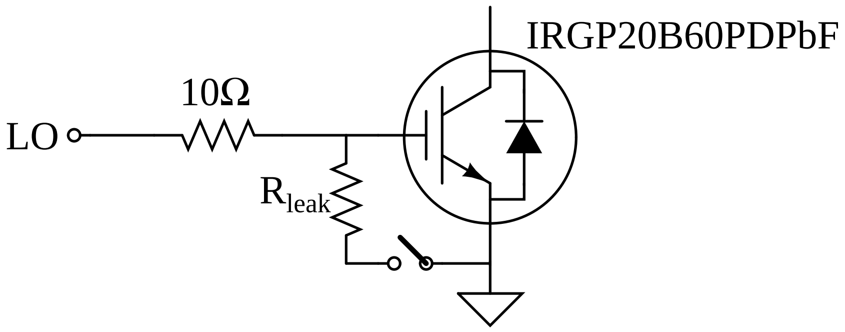


Figure 3.23: Test setup for a simulated gate dielectric wearout.

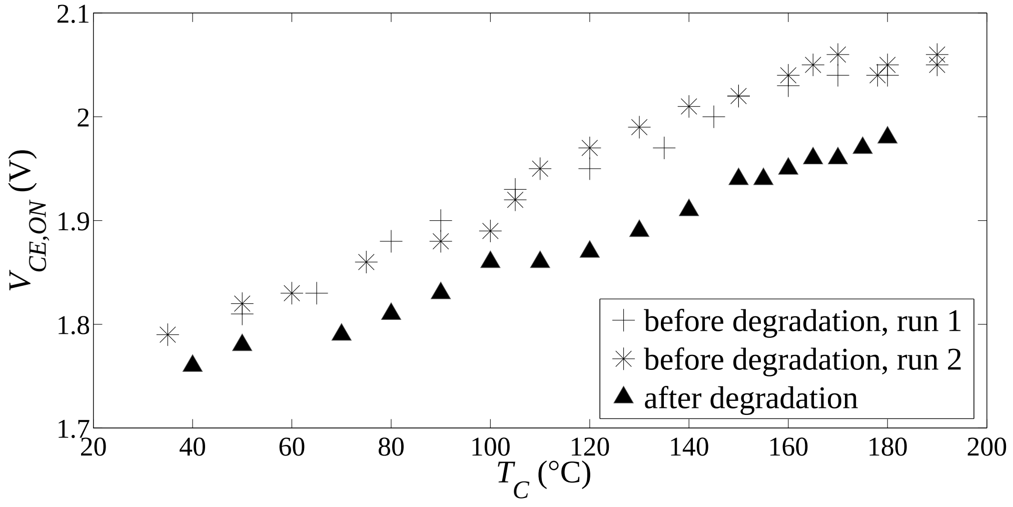


Figure 3.24: $V_{CE,ON}$ versus temperature for the IGBT, both before and after aging.

Figure 3.24 shows results recorded during the aging process. Note the positive temperature coefficient; this is expected for IGBTs [59]. Additionally, note that the component has a lower average on-state v_{CE} at all temperatures once it has degraded. This result was also obtained in [59]. It is believed that the drop occurs because the degraded die attach increases θ_{jc} and thus creates a higher internal temperature at any given T_C. Since the pn junction has a negative temperature coefficient of resistance [60], an increase in the temperature of this junction lowers average on-state v_{CE}.

Figure 3.25 shows results obtained using the proposed condition-monitoring algorithm once the degraded part was returned to the drive. Note that the part was

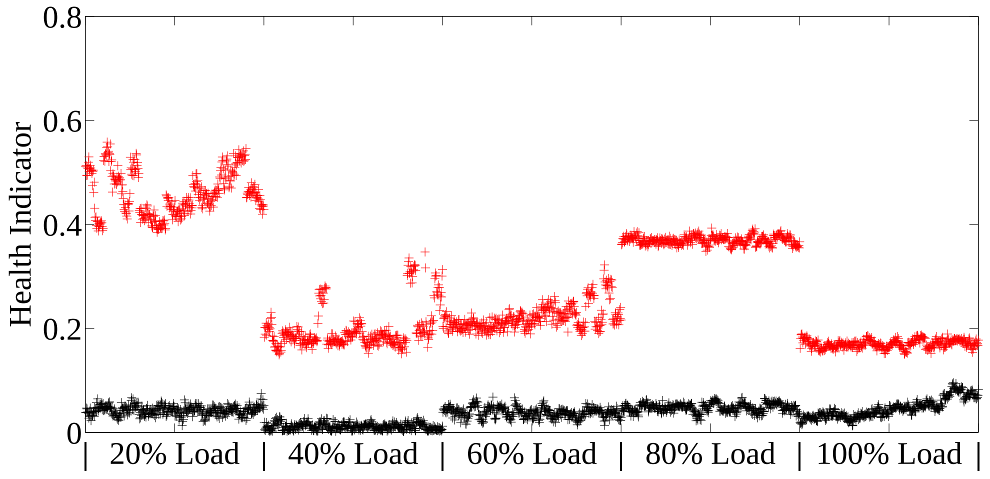


Figure 3.25: The reconstruction error recorded by the health-monitoring algorithm before and after the die-attach fault was induced. Note that there are a number of samples recorded at each loading condition and that the higher error at each load step corresponds to the degraded transistor.

tested in both the healthy and degraded state, and that the load on the machine was varied in order to simulate changes in operating conditions. The figure shows the reconstruction errors in both the healthy and degraded conditions, using data extracted at 20% load steps. Note that the degraded part caused a significantly higher reconstruction error at each load step and that changes in the loading condition had minimal impact. Note that drive operation appeared normal despite the fact that the device was nearing failure.

As another example, the effect of a breakdown in the gate oxide was simulated, which results in an increased leakage current [38]. As in [26], this fault was simulated by placing resistance between the gate and the emitter of the monitored switch. Different resistor values were used to demonstrate the ability to detect early stage faults. Figure 3.26 shows the reconstruction error versus time for both healthy and faulted operation. The gate charge indicator changes with an increase in leakage current from hundreds of microamps to tens of milliamps. Note that time series are shown for the healthy part in 10% increments from no load to full load. To simulate

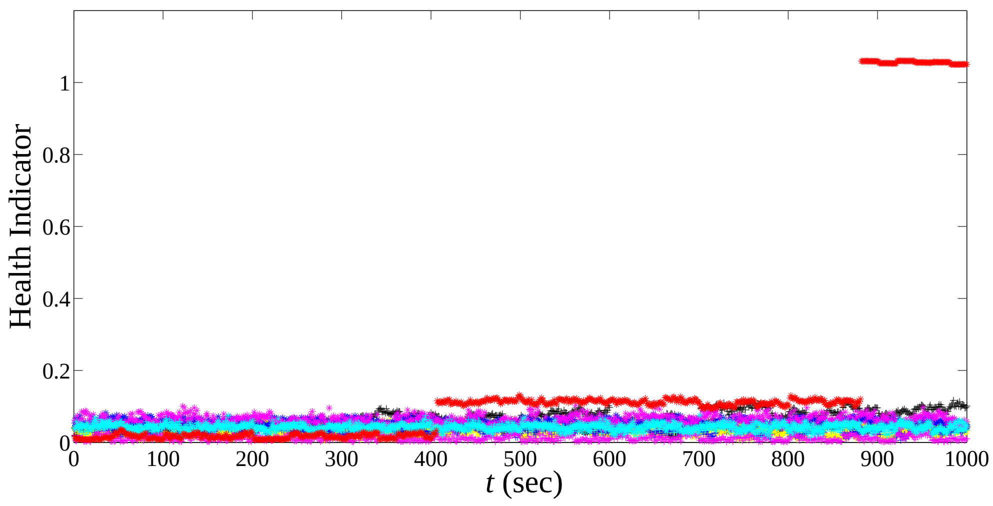


Figure 3.26: The reconstruction error recorded by the health-monitoring algorithm in 10% load steps for a healthy part. Note that they are all reasonably similar. The high errors beginning at ≈400s and ≈900s correspond to increased gate leakage current, i.e. ~ 3.2 mA and ~ 32 mA, respectively, at 100% load.

a faulted part, a 4.7k Ω resistor was added at $t \approx 400$s and then 470 Ω at $t \approx 900$s with the drive running at full load. Note that the corresponding leakage currents are small (~ 3.2 mA and ~ 32 mA, respectively), and that both conditions correspond to scenarios in which the drive maintains complete functionality despite the presence of the soon-to-fail switch.

3.4 Method for Incipient Fault Detection in Power MOSFETs

The on-line monitoring approach shown in Fig. 3.8 can also be applied to MOSFETs. Figure 3.27 shows the schematic of the MOSFET-based drive constructed for testing purposes. Note that this system uses the same control board and motor as the IGBT-based drive.

3.4.1 Feature Extraction

Although various health indicators also exist for MOSFETs, the use of $R_{DS,ON}$ is focused on here. Onstate resistance can be measured on-line in motor drives using naturally occurring ripple waveforms generated by the action of carrier-based PWM.

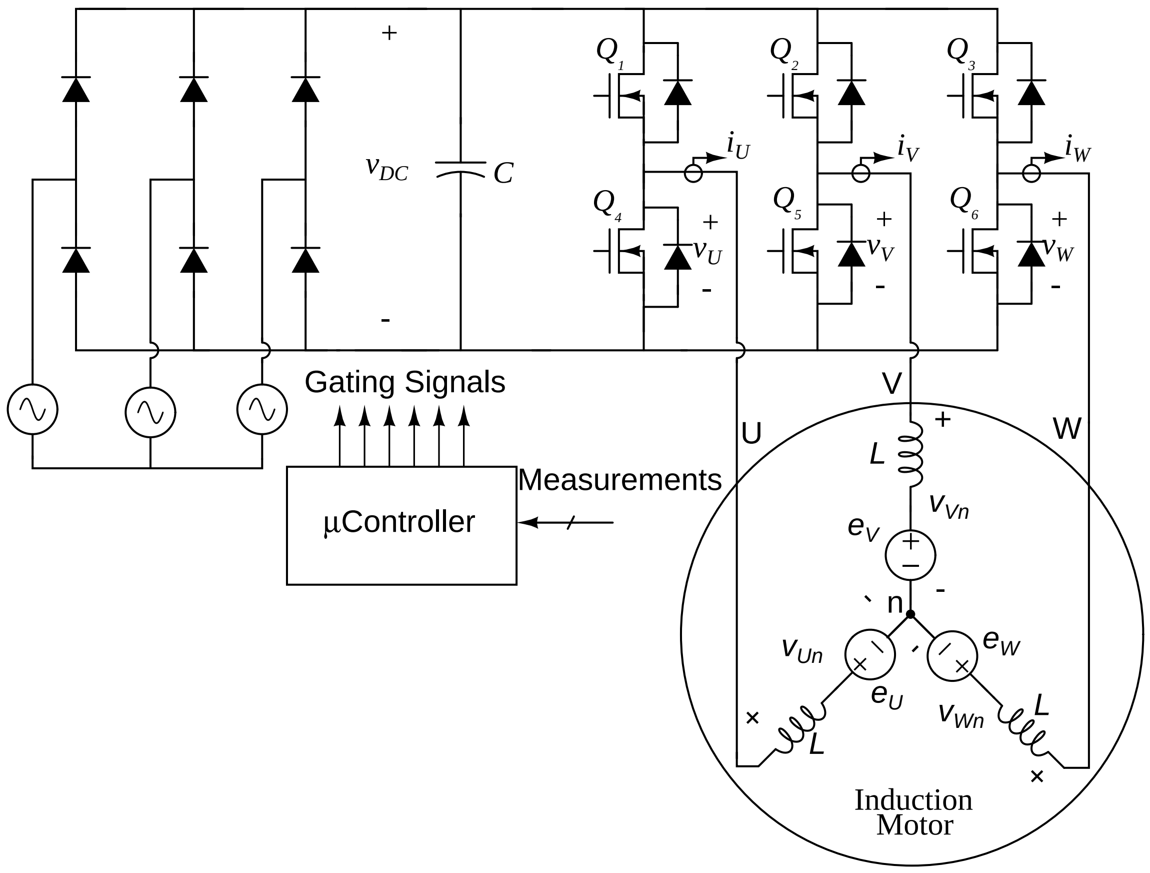


Figure 3.27: Partial schematic of the experimental FET-based drive. The drain-to-source voltage, v_{DS}, is measured for each transistor and the motor phase currents i_U, i_V, and i_W are also measured.

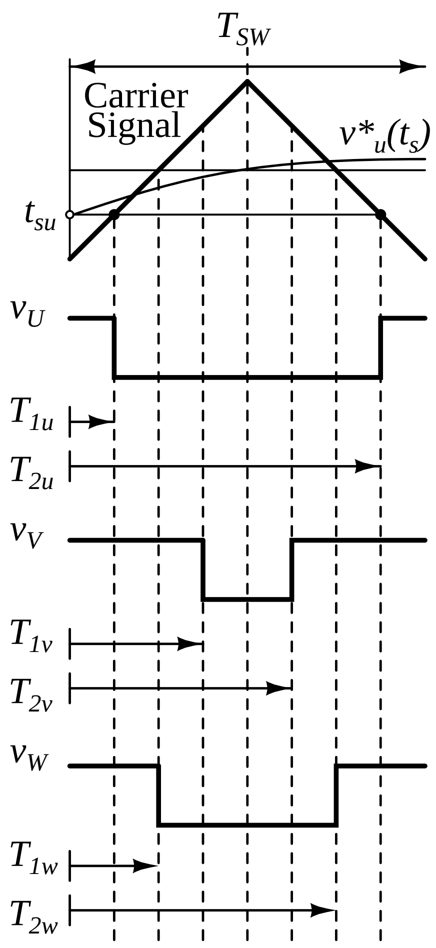


Figure 3.28: Notional example of the voltages v_U, v_V, and v_W during a single switching period. Also shown is the fictitious triangular carrier signal and the reference waveform for phase U. The voltages are defined in Fig. 3.27. The switching time instants, T_1 and T_2, are also shown for each of the phases.

The exact details depend upon the manner in which the drive is controlled.

Consider the motor drive shown in Fig. 3.27, and assume that it is driven using a digitally-implemented carrier-based PWM scheme. Specific instances of carrier-based PWM include space-vector modulation and regular sampling [61]. Typically, these schemes are implemented using capture-compare modules in microcontrollers. The most common technique is to use timers to align the switching instants with a fictitious carrier waveform as shown in Fig. 3.28. As a result, each phase voltage pulses once per switching period. The exact timing of switching depends upon the specifics of the microcontroller, but switching is often selected to be symmetrically aligned with the peaks of the carrier as shown in Fig. 3.28 [61]. The measurement approach described below does not depend on this symmetry, however. Symmetric regular sampling is used only for illustrative purposes, and other techniques could be considered.

The proposed measurement approach is based on the current ripple generated by the phase voltages shown in Fig. 3.28. These voltage waveforms can be written as the superposition of a fundamental frequency component and a ripple term. Using the simplified induction machine equivalent circuit in Fig. 3.27, note that all of the voltage ripple for a given phase appears across the inductance L. The resulting current is thus of the form [54]

$$i_{ripple} = \frac{1}{L} \int_{0}^{t} v_{ripple}\left(\tau\right) \mathrm{d}\tau. \tag{3.21}$$

The term v_{ripple} can be written for any one of the three phases. For phase U, for instance, it is

$$v_{U,ripple} = v_{U_n} - v_{U_{n,1}} \tag{3.22}$$

where v_{U_n} is the voltage across phase U with respect to the motor neutral and $v_{U_{n,1}}$ is the corresponding fundamental component of v_{U_n}. In terms of the phase voltages shown in Fig. 3.28 [54],

$$v_{U_n} = \frac{2}{3} v_U - \frac{1}{3}(v_V + v_W). \tag{3.23}$$

A careful analysis of Eqs. (3.21)-(3.23) shows that the pulses in v_V and v_W affect the ripple current flowing in phase U. If one measures the voltage across Q_4 during an appropriate interval aligned with one of the pulses on the other two phase legs, then one can easily estimate the on-state resistance of Q_4 using the equation

$$R_{DS,4,ON} = \frac{\Delta v_U}{\Delta i_U}, \tag{3.24}$$

where Δi_U is the ripple current during the measurement interval and Δv_U is the corresponding voltage ripple.

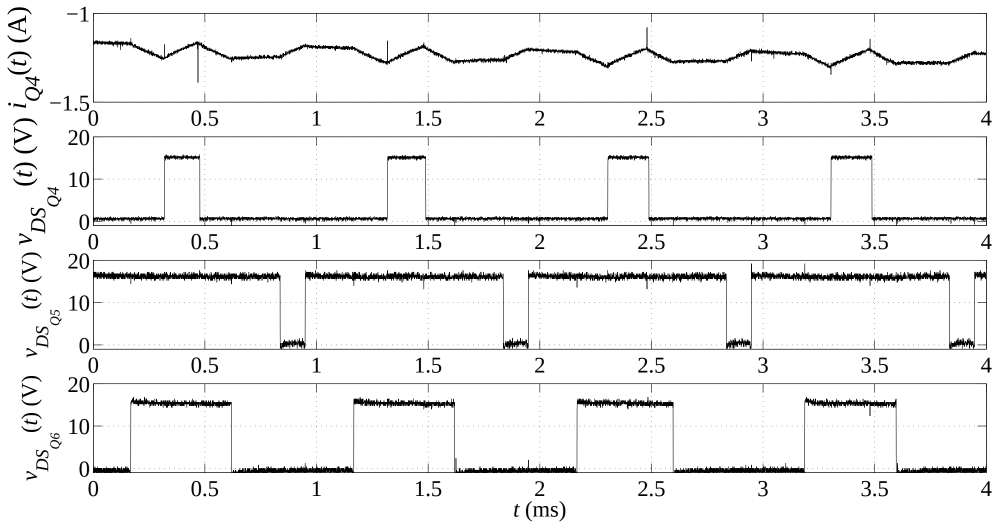


Figure 3.29: The nature of i_U, v_U, v_V, and v_W (from top to bottom) in the experimental drive.

Figure 3.29 presents an experimental example showing how the ripple current changes as the MOSFETs change state. $R_{DS,4,ON}$ would be measured when v_V is low. The on-state resistance of the other FETs would be measured during similar intervals. Timing is critical to the successful implementation of the measurement process described above. Figure 3.28, for instance, shows times during which Q_4, Q_5, and Q_6 are each conducting. Note that Q_4 is on for the longest time, and that its ripple current should exhibit 5 distinct states based on the action of the other phase legs. By comparison, the ripple current flowing through Q_5 only exhibits one distinct state. In this case, one can trigger high-speed sampling of the voltage and current for Q_4 using the same interrupts that control the switching of Q_5. Thus, one can exploit the nature of the PWM waveforms to obtain the best possible measurements of Δi and Δv. Measurements should be performed over a sufficiently long interval during which the current is approximately linear. The timing shown in Fig. 3.28, for instance, is ideal for the measurement of $R_{DS,4,ON}$ because the pulses on the two other phases are of a sufficient length and do not overlap. One can predetermine the appropriate measurement times by considering the nature of the symmetric, center-aligned PWM signals shown in Fig. 3.28. For any one phase, the switching time instants T_1 and T_2 can be computed in real time using the geometrical relationships [61].

$$T_1 = \frac{1}{4}T_{SW} \cdot (1 + v^*(t_s)) \tag{3.25}$$

$$T_2 = \frac{1}{2}T_{SW} + \frac{1}{4}T_{SW} \cdot (1 - v^*(t_s)) \tag{3.26}$$

where $v^*(t_s)$ is the most recent sample of the reference waveform for the given phase. When using regular sampling, these references are of the form

$$v_u^*(t_s) = M \sin(\omega_m t_s) \tag{3.27}$$

$$v_v^*(t_s) = M \sin\left(\omega_m t_s - \frac{2\pi}{3}\right) \tag{3.28}$$

$$v_w^*(t_s) = M \sin\left(\omega_m t_s + \frac{2\pi}{3}\right) \tag{3.29}$$

where M is the modulation depth and ω_m is the angular frequency of the modulating signal. The pulse widths for each phase are thus computed by taking the difference

$$\Delta T = T_2 - T_1 = \frac{1}{2}T_{SW} - \frac{1}{2}T_{SW} \cdot v^*(t_s). \tag{3.30}$$

These times are independent of ω_m and can thus be rewritten by substituting $\omega_m t_s = \theta$ into (3.27)-(3.29) to yield

$$\Delta T_u = \frac{1}{2}T_{SW} - \frac{1}{2}T_{SW} \cdot M \sin(\theta) \tag{3.31}$$

$$\Delta T_v = \frac{1}{2}T_{SW} - \frac{1}{2}T_{SW} \cdot M \sin\left(\theta - \frac{2\pi}{3}\right) \tag{3.32}$$

$$\Delta T_w = \frac{1}{2}T_{SW} - \frac{1}{2}T_{SW} \cdot M \sin\left(\theta + \frac{2\pi}{3}\right). \tag{3.33}$$

Equations (3.31)-(3.33) can be plotted to determine appropriate measurement times. Figure 3.28 corresponds to $M = 1$ and $\theta = \frac{-2\pi}{3}$. One can perform analysis to determine if the current will be approximately linear during the chosen measurement period. The measurement timing can be easily coordinated with switching.

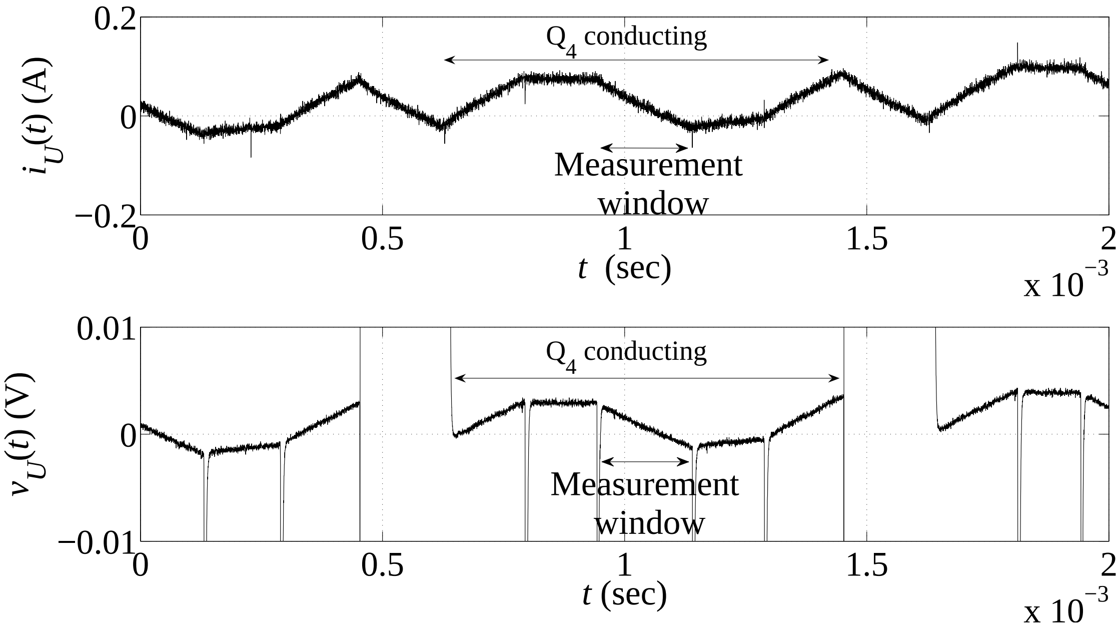


Figure 3.30: Measured $v_U(t)$ and $v_U(t)$. The measurement window is indicated.

3.4.2 Experimental Demonstration

The proposed measurement scheme has been demonstrated using a prototype drive of the form shown in Fig. 3.27. The microcontroller generates center-aligned PWM waveforms. Figure 3.30 shows the drain-to-source voltage across Q_4 and the phase U current. Δi_U and Δv_U are measured over the indicated window. As described above, the measurement is relatively straightforward to implement because the appropriate sampling window can be predetermined. Measurements are performed once every 100 periods of the modulating waveform. When applying the proposed approach with the waveforms shown in Fig. 3.30, the value of $R_{DS,ON}$ was found to be 40.2 mΩ. For the given conditions (i.e. ambient temperature and current), the manufacturer datasheet predicts a value of 44 mΩ [62]. Estimates of $R_{DS,ON}$ over a range of temperatures could be used to create the vector space needed for the condition-monitoring algorithm. The key implementation details are the same as for the IGBTs, with the need to perform a training step at the outset. To test the detection mechanism, the drive was placed

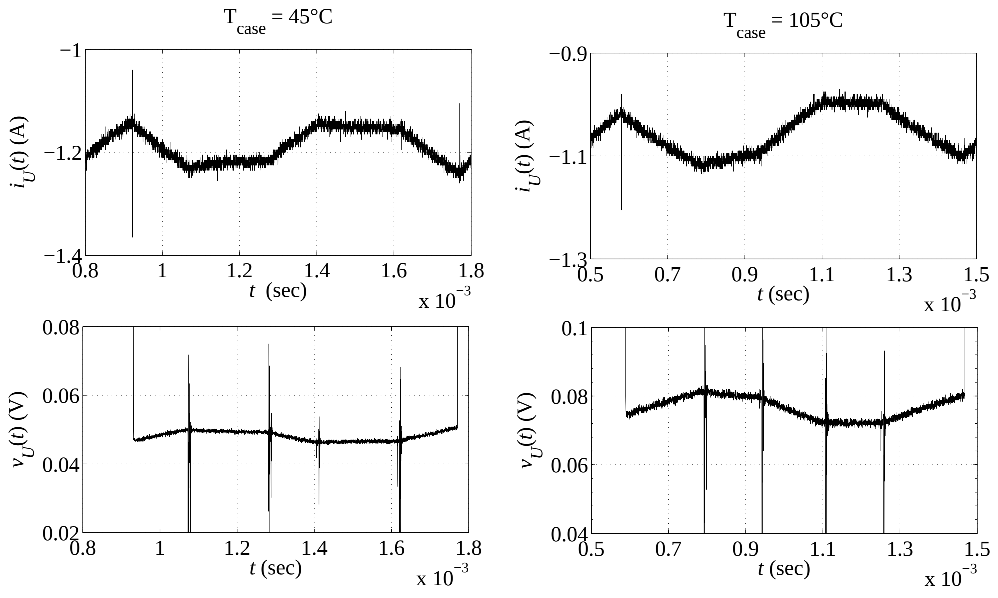


Figure 3.31: Measured $v_U(t)$ and $v_U(t)$ at 45°C (left) and at 105°C (right).

into a temperature controlled chamber. A thermocouple was connected to the case of MOSFET Q_4 to record the temperature during testing. $R_{DS,ON}$ was calculated while the drive was heated from a case temperature of 45° C to 105° C as seen in Fig. 3.31. The value of $R_{DS,ON}$ was computed using the method described above and it was found to vary from 46.2 mΩ at a case temperature of 45°C to 71.6 mΩ at a case temperature of 105°C. Note that a full-scale diagnostic indicator would require one to determine if $R_{DS,ON}$ has changed because of actual faults or simply because of other natural phenomena such as changes in ambient temperature.

CHAPTER 4: FAULT DETECTION IN ELECTRIC MOTORS

A method for the on-line detection of faults in electric motors is presented in this chapter that could be a major stepping stone on the path to fully automated condition-based maintenance (CBM). The first step in this scheme is the fault-detection algorithm that was discussed in Chap. 2 that efficiently encodes a large set of KPIs (i.e. vibration values, spectral quantities, etc.) using PCA and does not require a priori rules. This algorithm can adapt to changes in motor operating conditions, thus significantly reducing the likelihood of false positives and missed faults. The system currently takes a conservative approach to fault detection, as it alerts operators of potential fault conditions and then allows them to investigate the underlying KPI values. The algorithm can be implemented in real-time, and lends itself to applications with drives as well.

The chapter begins with some background on the need for motor fault detection and some of the methods used currently. A description of the fault detection algorithm as it relates to the detection of motor faults is shown and a description of the machinery fault simulator used during testing. In Sec. 4.4 a number of test cases are considered that demonstrate the effectiveness of this approach.

4.1 Background

The failure of critical induction motors in power plants and other industrial facilities costs millions of dollars in reduced output, emergency maintenance costs, and lost revenue. In the power industry and elsewhere the response has been to develop extensive preventative maintenance programs based on regularly scheduled outages. Such programs are expensive. In the nuclear industry, for instance, maintenance budgets can account for as much as 40% of total production costs [63]. Industries are now actively pursuing condition-based maintenance (CBM) programs intended to predict failures before they happen. Such efforts could potentially reduce the length of forced outages. At present, these shutdowns can lead to financial losses in excess of $1 million dollars per day [63]. The huge costs associated with forced shutdowns has increased the desire to develop robust fault detection mechanisms.

Reliable motor fault detection has also become increasingly important on-board naval ships. In OPNAV Instruction 4790.16A, the Chief of Naval Operations (CNO) spells out the Navy's policy on CBM [64]. The CNO makes clear that the proper application of CBM strategies is essential to plans to reduce operating and support costs and manpower requirements. CBM tools such as the Integrated Condition Assessment System (ICAS) use aboard naval ships now monitor various critical Hull, Mechanical, and Electrical (HM&E) systems [65]. ICAS can provide users with fault predictions based on automated algorithms, but much of the performance monitoring capability relies on off-line analysis performed by on-shore experts. The data-transmission capability is also used for integration with other fleet-wide systems [66]. Electric motors are increasingly important components aboard ships and are thus a key target for CBM. This is particularly true in the case of ships using electric motors for propulsion.

The remainder of this section will focus on the failure mechanisms that are associated electric motors and a literature review of the methods for the detection and diagnosis of these faults is presented.

4.1.1 Failure Mechanisms of Electric Motors

The most common electric motor faults include stator faults resulting in the opening or shorting of one or more of a stator phase winding, abnormal connection of stator windings, broken rotor bar or cracked rotor end-rings, static and/or dynamic air-gap irregularities, bent shaft which could cause serious damage to stator core and windings due to rubbing between the rotor and stator, shorted rotor field winding, and faults in the bearing and gearbox [3]. The most prevalent fault mechanisms and thus the most important for monitoring are bearing failures, stator or armature insulation faults, broken rotor bar and end ring faults, and eccentricity faults [3]. The remainder of this section will focus on these key failure mechanisms.

The majority of electrical motor bearings are ball or rolling element and 40% - 50% of all motor failures are bearing related. A bearing consists of two rings one inner and one outer and a set of balls or rolling elements placed in raceways between these rings [67]. Fatigue bearing failures may take place under normal operating conditions with balanced load and good alignment which lead to increased vibration, increased noise levels, and flaking or spalling of bearings. Other than the normal internal operating stresses bearings can also be damaged by external causes. External causes include contamination and corrosion caused by pitting and sanding action of hard and abrasive minute articles or the corrosive action of water, acid, etc., improper lubrication which includes both over and under lubrication causing heating and abrasion, and the improper installation of a bearing either by improperly forcing the bearing onto the shaft or in the housing due to misalignment causing indentations to be formed in the raceways [10]. The ball bearing related defects can be categorized as outer bearing race defect, inner bearing race defect, ball defect, and train defect. For more information about at which vibration frequencies these defects can be detected refer to Appendix D [10]. Sometimes though bearing faults might manifest themselves as rotor asymmetry faults [67] which are usually categorized with eccentricity-related faults.

Stator or armature faults are usually related to insulation failures and make up 30% - 40% of all reported induction motor failures. These faults are more generally known as phase-to-ground or phase-to-phase faults. These faults are believed to start as undetected turn-to-turn faults that finally grow and culminate into major ones [68].

The primary reasons for armature or stator insulation failures are high stator core or winding temperatures; slack core lamination, slot wedges, and joints; loose bracing for end winding; contamination due to oil, moisture, and dirt; short circuit or starting stresses; electrical discharges; and leakage in cooling systems [69].

Rotor bar and end-ring breakage failures account for 5% - 10% of total induction motor failures [67], [70]. These failures can be caused by thermal stresses due to thermal overload and unbalance, hot spots, or excessive losses, sparking (mainly fabricated rotors); magnetic stresses caused by electromagnetic forces, unbalanced magnetic pull, electromagnetic noise, and vibration; residual stresses due to manufacturing problems; dynamic stresses arising from shaft torques, centrifugal forces, and cyclic stresses; environmental stresses caused by for example contamination and abrasion of rotor material due to chemicals or moisture; and mechanical stresses due to loose laminations, fatigued parts, bearing failure, etc.

Machine eccentricity is the condition of unequal air gap that exists between the stator and rotor [3]. A large eccentricity can cause the stator and rotor to rub together which can result in damage to both. The two types of eccentricity are static air-gap eccentricity and the dynamic air gap eccentricity. A static air-gap eccentricity is when the position of the minimal radial air-gap length is fixed in space. Static eccentricity may be due to the ovality of the stator core or by the incorrect positioning of the rotor or stator during commissioning. A dynamic eccentricity is when the center of the rotor is not at the center of the rotation and the position of minimum air-gap rotates with the rotor. This misalignment may be due to several factors such as a bent rotor shaft, bearing wear or misalignment, and mechanical resonance at critical speed[10].

Figure 4.1 shows the distribution of failures by type of failure in electric motors according to a study done by [71]. The failure rates of the components with a “*” by their percentage could be reduced by three-quarters by taking appropriate measures to predict and prevent that failure [71].

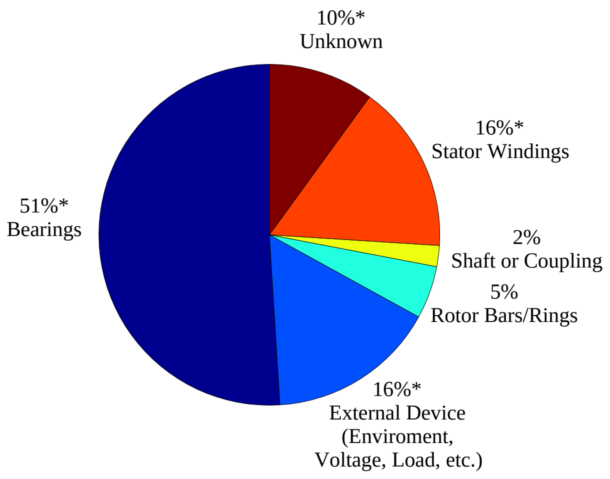


Figure 4.1: Failure distribution of electric motor components. Adapted from [71].

4.1.2 Methods for Motor Fault Detection

In this section the different methods for fault detection in electric motors is discussed. The section begins with a look at rules based approach for fault detection and then takes a look at the different artificial intelligence schemes that are used for fault detection and diagnosis.

4.1.2.1 Rules Based Technique

Motor health can be assessed by extracting certain key performance indicators (KPIs) from various sensors(i.e. current transducers, vibration sensors, etc.). For example, a common approach is to monitor machine vibrations. The methods currently used determine thresholds for each individual parameter and then monitors the parameters to detect if the threshold has been crossed. Trending is used to track fault parameters over a period of time and attempts to detect sudden changes in an individual parameter which could be an indication that failure has occurred. These methods provide the basis for various industry standards, including those in IEC 34-14 [72], ISO 10 816-1 [73], and NEMA MG-1 [74]. Simple rules-based procedures for fault detection are currently used within ICAS [5] and are nominally preferred because of their theoretical simplicity and computational tractability.

There are potential problems with the simple threshold, trending or rules-based systems described above. Consider an example bearing condition-monitoring scheme based on the ISO 10 816 Standard [73]. In this standard, machine vibration is acquired and band-pass filtered over a range of 10 Hz to 1 kHz and then integrated to determine the velocity over that frequency range. Acceptable limits on the absolute magnitude of machine vibration are provided by the standard. It also suggests that one monitor for relative changes above the machine's baseline vibration level. Research and operational experience demonstrate, however, that unhealthy bearings may exceed thresholds at certain loading levels, but not all [75].

Furthermore, research has shown that the speed of a machine can have a direct impact on vibration, meaning that variable-speed drives can potentially mask imminent bearing failures [75]. When using simple rules, it is reasonable that there will be certain such conditions that are not detected, as it is difficult to quantify all possible fault conditions a priori. The key issue with these fault detection methods is the sensitivity of the measured fault parameters to machine specific details such as size, power, construction type and loading. In order to develop a reliable fault detection/-classification algorithm, an extensive set of "healthy" and "faulty" reference data is generally required. The final accuracy of the fault detection algorithm is clearly limited by the size, breadth and quality of the reference data which was used to develop it.

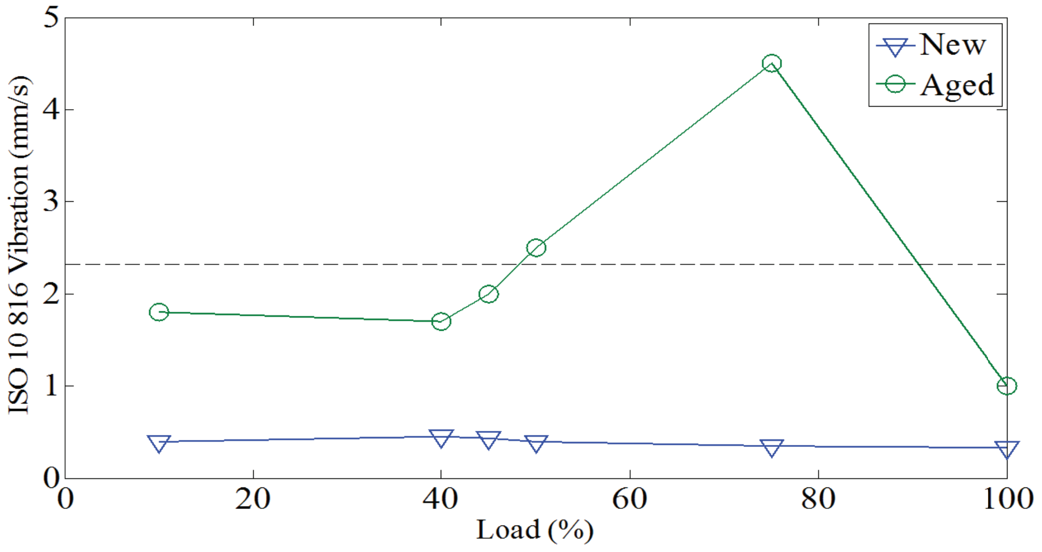


Figure 4.2: Machine vibration versus load for a bearing, where a 0% load corresponds to 1800rpm and 100% load corresponds to 1745 rpm. The ISO 10 816 acceptable vibration limit is shown at 2.3 mm/s. Adapted [75].

4.1.2.2 Artificial Intelligence Techniques

Both research and practical experience have demonstrated that the use of hard-and-fast rules can be misleading. To overcome the limitations of simple rules-based techniques, more advanced artificial intelligence techniques have been developed using techniques such as artificial neural networks, fuzzy logic and expert systems. These techniques have been developed and demonstrated using a variety of sources such vibration data [76], [77], [78], motor current spectral components [79], and detailed motor models [80], [81].

Artificial neural networks have been used for fault detection and diagnosis of electric motors [82]. A neural network is able to formalize the knowledge base of the diagnostic system when inputs and outputs are chosen suitably. The neural network is trained using data gathered from healthy motors and from simulation in of faulted motor and diagnostics are run to determine its condition [83]. For example a neural network based fault diagnosis system utilizing the stator current spectrum is described in [79].

This system uses rules-based frequency filters of the current spectrum to classify its frequency components into four categories with a decreasing level of importance. A neural network based on these rules is trained for all possible operating conditions of the machine which is used to classify the incoming data [10]. If a measured spectral signature falls outside the trained clusters that signature is marked as a potential fault. The system only sends an alarm when a fault signatures has been observed over a period of time to prevent false diagnosis. This type of scheme has been used to diagnose bearing and unbalanced rotor faults of induction motors [79].

Fault detection using fuzzy logic involves making decisions based on classifying signals into a series of bands. A more accurate determination of motor health is possible based on combining these fuzzy values than by just setting thresholds [84]. For example one technique described in [85] monitors signal harmonics from the power spectral density of the current and employs a hybrid fuzzy min-max neural network and classification and regression tree to classify fault conditions. The big issue with this type of scheme is that it is trained to certain failures and are not able to detect all possible faults. Another example of the use of fuzzy-logic-based systems is presented in [86]. In this case a fuzzy logic system is used to classify broken-bar-related faults by categorizing the two sideband components around the fundamental of the induction motor line current by using a set of nine rules [86].

Expert systems are an attempt to emulate the human thought through knowledge representation and inference mechanisms. Within a bound domain of knowledge expert systems are capable of decision making on a quality level comparable to human experts [87]. Some research work has been dedicated to applying expert systems to machine fault diagnosis. The development of an expert system knowledge base for on-line diagnosis for induction machine rotor electrical faults was presented in[88].

All of these methods are focused on the diagnosis of faults in specific components, and none of them fuse all of the data now available when monitoring motor driven loads. Several recognize that the successful use of such schemes requires operator involvement [77], [78]. In the case of motors and their driven loads, it is clear that the intelligent application of enabling technologies for CBM is still an open question.

4.2 Algorithm Development for Electric Motors

This dissertation proposes a scheme intended to lead to more advanced levels of automation by first leveraging the power of collecting data in a centralized location. Figure 4.3 shows the proposed health-monitoring scheme for monitoring of electric motors. Inputs include various raw sensor signals. Specifically, the inputs are the phase currents, the phase voltages, and acceleration signals from sensors mounted on the bearings. The first step is to extract relevant key-performance indicators (KPIs) or features. Examples include the root-mean-squared (rms) value of the velocity measured by individual vibration sensors and the amplitudes of appropriate signals obtained via spectral analysis. The feature-extraction block thus contains various processing steps such as Fast Fourier Transforms (FFTs) and rms calculations. Note that raw signals provided to the feature-extraction block must be sampled at a high rate. The processed features are output at a much lower rate, as multiple data samples are typically required for processing. In this implementation, raw data is collected at 6 kHz and features are calculated at 5 Hz. 186 KPI values are calculated and 149 are used for fault detection. Appendix D has a detailed description on how many of these KPI values are extracted.

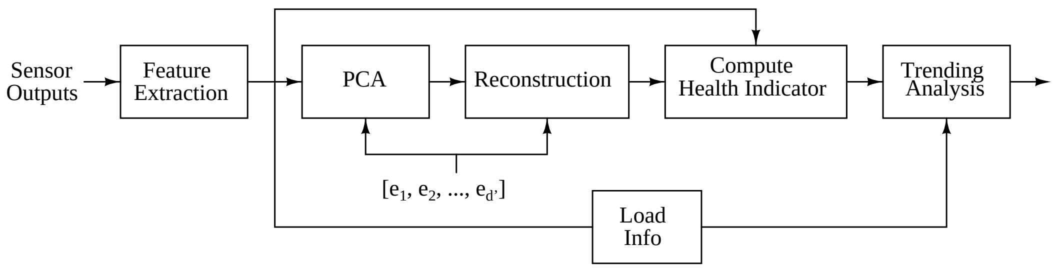


Figure 4.3: Generalized condition-monitoring algorithm for an operational motor based on principal-components analysis. The e_i are based on healthy conditions. New input signals and parameters can be added as needed.

The heart of the fault-detection algorithm is the principal-component analysis (PCA) block. This unit provides information about meaningful trends within the monitored data. In order to provide such information, the features measured at each time step are first grouped into a column vector $\boldsymbol{x}_k$. The length of this vector is equivalent to the number of features; for generality, the vector is said to have a length d. The basic approach of the principal-component-based algorithms to compare each measurement $\boldsymbol{x}_k$ to an expectation. This expected vector is computed by projecting $\boldsymbol{x}_k$ onto a vector space created using “healthy” features. These healthy values are learned during a training phase in which the motor is assumed to be fault-free. During training, the healthy vectors are decomposed into a small set of characteristic vectors that best describe the distribution of the healthy parameters. During operation, each measured vector is projected onto this space.

The training space includes feature vectors recorded under normal operating conditions. In all, there are M such feature vectors and they are denoted as $\Gamma_1, \Gamma_2, \Gamma_3, , \Gamma_M$. These training vectors are subject to a principal-component analysis in which one seeks a set of orthonormal vectors $\boldsymbol{e}_i$ that best describe the distribution of the data. The $j - th$ training vector can thus be expressed as

$$\boldsymbol{\Gamma}_j = \boldsymbol{m} + \sum_{i=1}^{d'} a_{j,i}\boldsymbol{e}_i, \tag{4.1}$$

where $\boldsymbol{m}$ is the sample mean, i.e.

$$\boldsymbol{m} = \frac{1}{M}\sum_{j=1}^{M}\boldsymbol{\Gamma}_j. \tag{4.2}$$

Note that the distribution of the data is best described using $d' \leq d$ orthonormal vectors [18, 16]. During the training phase, one calculates these vectors by minimizing the squared-error criterion function

$$J = \sum_{j=1}^{M}\left\|\left(\boldsymbol{m} + \sum_{i=1}^{d'} a_{j,i}\boldsymbol{e}_i\right) - \boldsymbol{\Gamma}_j\right\|^2. \tag{4.3}$$

[18] and [16] show that the $\boldsymbol{e}_i$ correspond to the eigenvectors of the sample covariance matrix which is

$$\boldsymbol{S} = \sum_{j=1}^{M}\left(\boldsymbol{\Gamma}_j - \boldsymbol{m}\right)\left(\boldsymbol{\Gamma}_j - \boldsymbol{m}\right)^T. \tag{4.4}$$

The actual $\boldsymbol{e}_i$ are the eigenvectors corresponding to the d' largest eigenvalues of $\boldsymbol{S}$ [18, 16]. Ultimately, this process yields a compact basis that efficiently encodes the relevant features of a healthy motor during normal operating conditions. Further details on the PCA algorithm are included in various references, including [18] and [16].

When the motor is operational and the algorithm is applied to monitor its health, the first step is to project the features measured at time t_k onto the space spanned by the $\boldsymbol{e}_i$. This projection is performed by the block labeled PCA, which computes the coefficients

$$a_{k,i} = \boldsymbol{e}_i^T\left(\boldsymbol{x}_k - \boldsymbol{m}\right). \tag{4.5}$$

The next block uses these coefficients to reconstruct an approximation of $\boldsymbol{x}_k$. The resulting estimate is thus denoted as

$$\hat{\boldsymbol{x}}_k = \boldsymbol{m} + \sum_{i=1}^{d'} a_{k,i}\boldsymbol{e}_i. \tag{4.6}$$

Following reconstruction, the algorithm calculates the two-norm of the residual vector $\boldsymbol{r} = \hat{\boldsymbol{x}}_k - \boldsymbol{x}_k$. This quantity, which we term the health indicator (HI), is computed at each time t_k. HI is thus

$$\mathrm{HI}_k = \boldsymbol{r}^T\boldsymbol{r} = (\hat{\boldsymbol{x}}_k - \boldsymbol{x}_k)^T(\hat{\boldsymbol{x}}_k - \boldsymbol{x}_k). \tag{4.7}$$

This quantity represents the error between the measured features $\boldsymbol{x}_k$ and their expected value for a healthy motor. In a very basic sense if the error is small, the motor is operating under normal conditions; if the error grows, a problem may be developing. The final block, labeled "Trending Analysis," monitors for such variations.

At first, this highly mathematical derivation of the principal components seems relatively unintuitive. For this reason, lets step back to place PCA in the context of the problem at hand. Consider that each vector $\boldsymbol{x}_k$ is a set of 149 KPIs recorded at intervals of 0.2 seconds. In general, we do not know if a machine has a fault, and thus we do not know what is indicated by $\boldsymbol{x}_k$. One could develop a set of rules that would monitor each of these 149 values. In a rules-based context, one is likely to assume that as long as each feature is useful in its own right as some sort of fault indicator that it is best to jointly monitor all such fault indicators simultaneously. In what is perhaps a counterintuitive result, this is not necessarily the case. Generally speaking, features are not independent and thus classification accuracy does not necessarily improve as one monitors more features. In fact, there can actually be a diminishing point of returns stemming from the fact that as more features are included, it becomes all the more difficult to develop a meaningful set of rules that encompasses all possible trends in the data [18]. The proposed algorithm combines features in such a way that it reduces the size of the feature set and reveals trends that best separate the features. This is demonstrated in Sec. 4.4.

One additional note is needed about the effect of loading on the proposed Health Indicator. As the load on the motor changes, this can certainly affect quantities such as vibrations [75]. In order to account for such changes, the "Trending Analysis" block also monitors the motor load by monitoring both the real power and the speed. If there is a change in motor load, the "Trending Analysis" block will detect this and re-learn the $\boldsymbol{e}_i$. Specific details are presented in Sec. 4.4.

4.3 Experimental Setup

The fault-detection algorithm has been tested in the laboratory using a Machinery Fault Simulator from SpectraQuest. This unit, which consists of a $\frac{1}{2}$ hp three-phase induction motor, can be configured to drive various different mechanical loads, including pumps, fans, compressors, and constant-torque loads. Figure 4.4 shows the setup with an adjustable brake connected to the shaft via a belt. Note that various different mechanical and electrical faults can be introduced by replacing various components throughout the system. For instance, the normal bearings can be replaced by bearings with faults in their inner or outer races, or the motor can be replaced with a similar motor having a broken rotor bar.

The setup includes thirteen different sensors, and each is sampled via a data-acquisition system at 6kHz. The list of monitored sensors includes the rotational speed; the three phase currents; the three line-to-line voltages; and the axial, vertical and horizontal acceleration on both of the two bearing blocks. Code on-board the computer calculates 186 different features from this data set, 149 of which are included in the actual feature vector $\boldsymbol{x}_k$. The features can be broken up into three categories.The first are load-dependent KPIs (i.e. currents, speed, etc.), which are not included in the feature vector but are useful for. The quantities included in the feature vector are standard vibration-related measurements (i.e. rms velocity, amplitudes at multiples of the electrical and mechanical frequencies, etc.) and the amplitudes of specific bearing, eccentricity, and rotor-bar related frequencies in the vibration and current spectra as described in [10]. Load-dependent data has not been included in the feature vector since it is expected to change and could easily cause one to classify load changes as faults. Such features are used, however, in computing

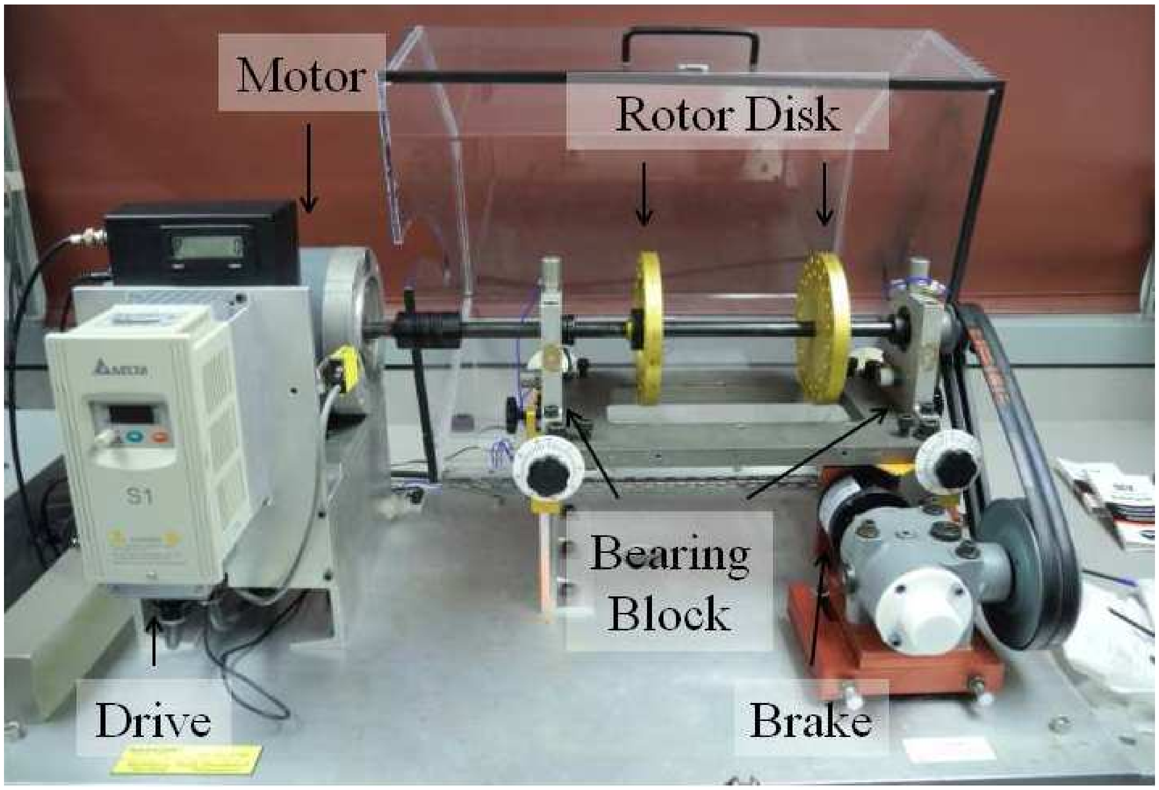


Figure 4.4: Photograph of the Machinery Fault Simulator in the laboratory. Note that the six vibration sensors are placed on the two bearing blocks. The adjustable brake provides a constant-torque load, and the pump can be replaced with other loads (i.e. pump, compressor, etc.).

other features and in understanding the loading condition of the motor. Appendix D list each of the 186 KPIs and also shows how many of them are calculated.

It is important to note that real-time computation of the features and health indicators has been tested and successfully demonstrated in a laboratory at Converteam Naval Systems. For development purposes, however, data is recorded and all analysis performed off-line. There is no reason, however, why any of the computation could not be performed in real-time using market-available computational assets.

4.4 Experimental Results

Several experiments indicative of potential operating conditions have been considered. This section describes the results of those tests.

4.4.1 Introducing faults in an off-line pump/motor setup

Many ship systems use off-line motors to drive centrifugal turbo machines such as pumps and fans. Examples include Auxiliary Sea Water (ASW) pumps, lube-oil (LO) pumps, and generator cooling fans. To simulate these common systems, the motor was coupled to a centrifugal pump as shown in Fig. 4.5. The pump drives fluid throughout a small distribution system that includes a large reservoir and several throttling valves.

As an initial test condition, several different faults were inserted into the system. The faults included the following:

- Bearing with inner-race damage (BPFI)
- Bearing with outer-race damage (BPFO)
- Bearing with a ball-spin fault (BSF)
- Bearing with a combination of the above faults (COMB)
- An unbalanced mechanical load
- An eccentric rotor disk
- A cocked rotor disk
- A faulted pump (with rubbing)

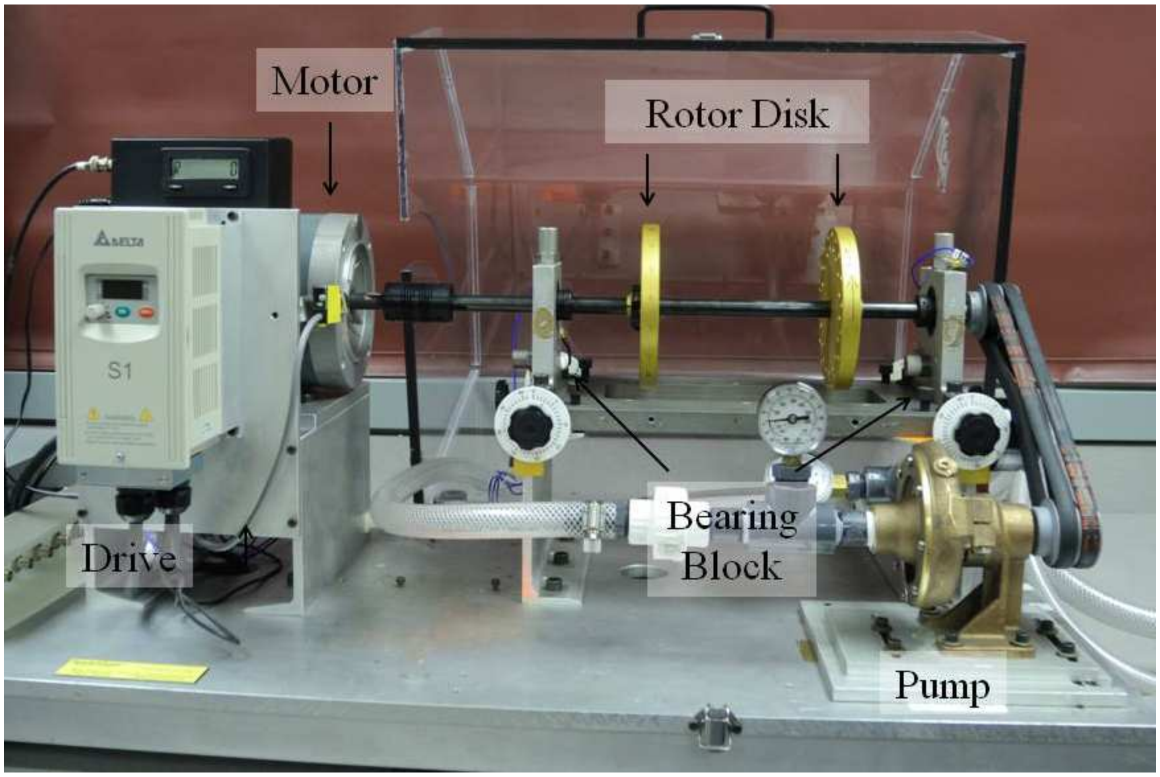
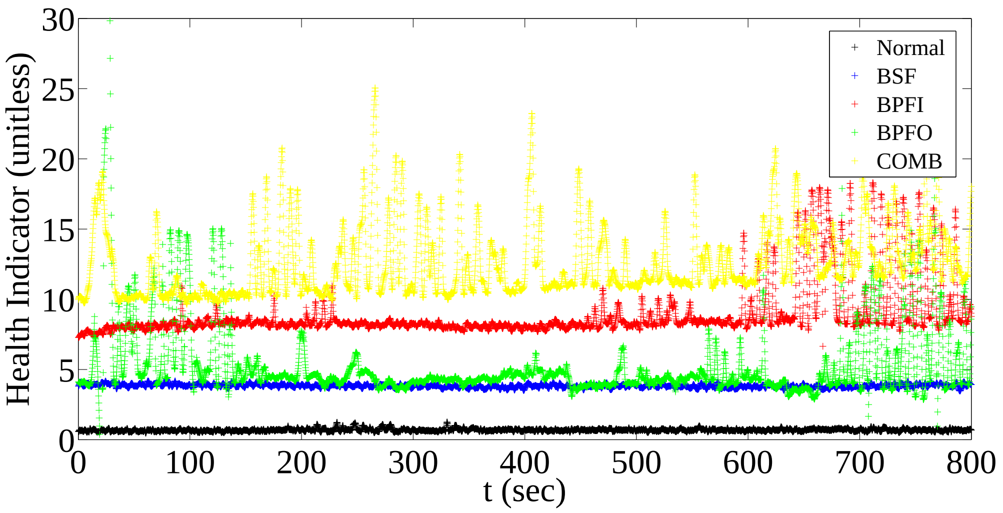


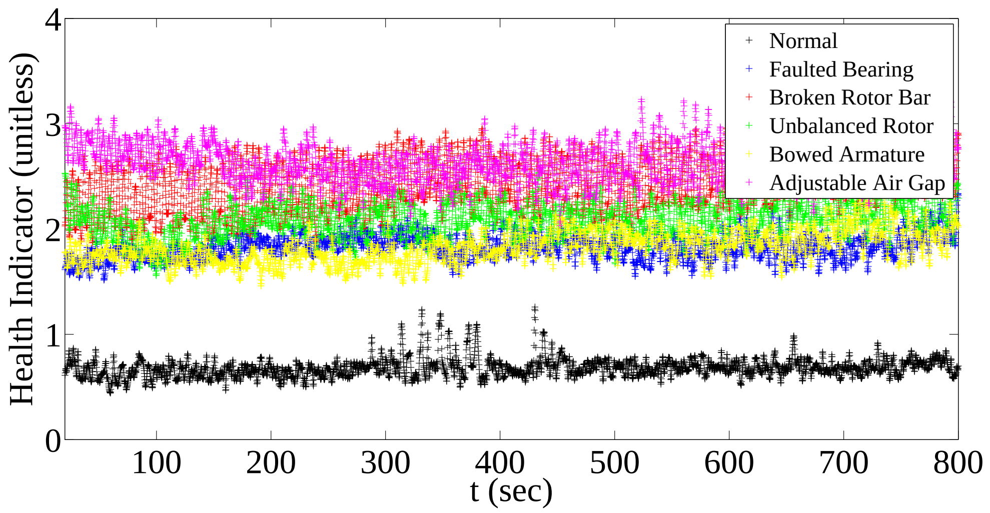
Figure 4.5: Photograph of the Machinery Fault Simulator with the pump coupled to the shaft. A fluid system with a large reservoir and throttling valves is connected.

- A centrally bent shaft
- Motor with a broken rotor bar
- Motor with an unbalanced rotor
- Motor with a faulted machine bearing
- Motor with a bowed armature
- Motor with highly eccentric air gap

Figures 4.6 shows the results of various tests. Note that each plot shows the Health Indicator value as a function of time for both normal and faulted conditions. Note that the indicator value generally increases by a relatively large amount for each of these relatively early-stage fault conditions.

To imagine a practical implementation of the proposed scheme, consider the following. During normal operation at a fixed load, the algorithm initially learns an appropriate set of vectors $\boldsymbol{e}_i$. In reality, these vectors are learned once the motor has reached a steady-state operating condition, which is defined using the motors thermal time constant. In general, we know that the load on the motor will change slightly as the machine heats up. The time scale for such changes can be quantified using the motors thermal time constant [89]. In general, however, most machines operate for much longer than a single thermal time constant and are thus in a quasi-thermal steady state over most of their operating lifetime. It thus makes sense to train the algorithm once the load is steady. Once a fault develops, various KPI values will change, and the HI value will change as well. Note that the HI value indicates only the existence of a fault, and not its exact cause. Thus, HI is not used for diagnosis. In this case, later diagnosis can be performed via a specific analysis of individual KPIs. This step can be performed automatically or manually. In practice, it has been found to be extremely helpful to manually observe the data. One could imagine that such a process could be performed at a remote location if the KPIs were routinely uploaded to the cloud. An example of such analysis is considered below.





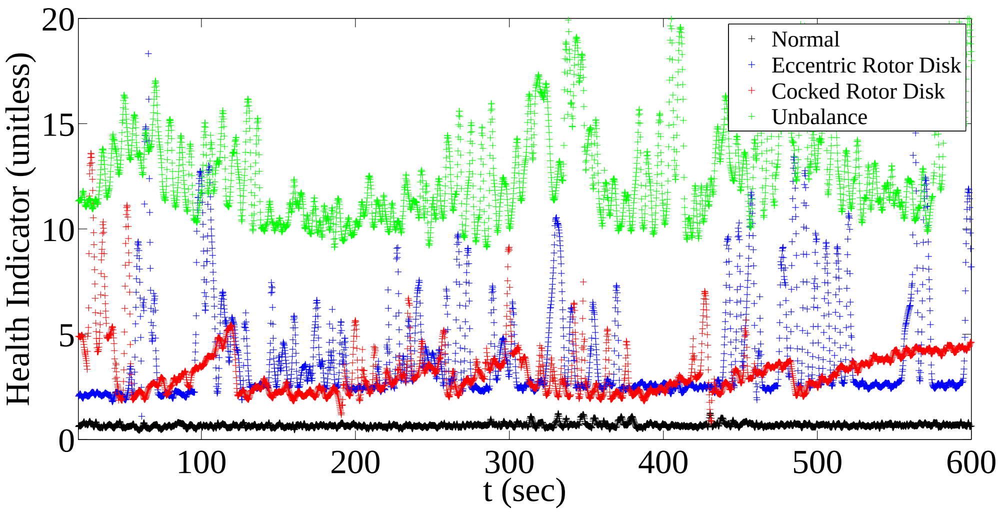


Figure 4.6: HI value versus time for various faults as well as for a setup with no faults (normal).

4.4.2 Effects of loading in the off-line pump/motor setup

In many systems, pumps run at one or two different operating speeds. An example is the lube-oil pump in a DDG-51 naval destroyer. Different speeds are achieved by changing the number of poles. At any given pole setting, however, the motor speed does not change significantly even if torque does. This is, in fact, a fundamental property of off-line induction motors [89]. In a pump application, one reason for a slight speed change is a change in valve settings. In typical flow regimes, the motor torque τ_L tends to be related to flow rate Q according to a power-law relationship of the form [90]

$$\tau_L = \beta_1 Q^2, \tag{4.8}$$

where β_1 is an empirical constant. As the valve is throttled, the flow rate drops and so, too, does the required torque. As a result, the motor speed increases closer to synchronous. Given the operational characteristics of off-line induction motors, speed is thus one indicator of load.

To simulate a load change, the valve in the fluid system was throttled from fully open, to 50% closed, to 75% closed. During the process, thespeed of the motor increased from 3448.2 RPM (fully open), to 3462.6 RPM (50% closed), to 3466.2 RPM (75% closed). Figure 4.7 shows the value of the Health Indicator for each of the various valve settings with a healthy system as well as the Health Indicator value for various faulty conditions in a system with a fully open valve.

The behavior or the Health Indicator in the various throttling scenarios may at first appear to be troubling, particularly at 50% throttling. A closer observation, however, actually serves to demonstrate the power of the algorithm. Given that the Health Indicator value is approximately the same in both the normal, throttled tests and in the case with a faulty outer race (BPFO), it is meaningful to investigate the underlying KPIs. Figure 4.8 shows the velocity recorded by the x-axis, y-axis, and

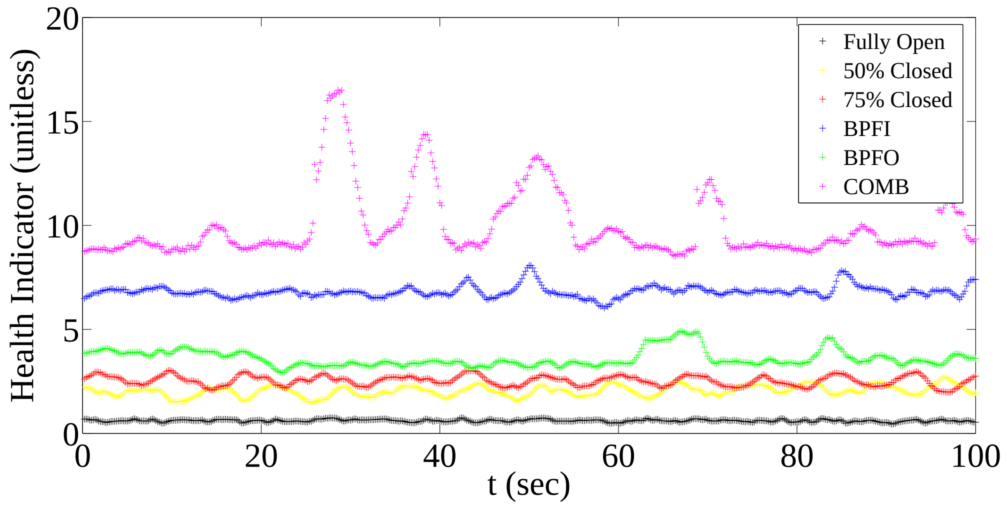


Figure 4.7: HI value versus time for several different valve settings with no faults, and for several different bearing faults with a fully open valve.

z-axis accelerometers, respectively, in each of the following three cases:

- Healthy bearing, fully open valve (labeled as Normal)
- Healthy bearing, 50% throttled valve (labeled as Throttle 50%)
- Bearing with early-stage outer race defect, fully open valve (labeled as BPFO)

Several immediate observations arise. First, it is clear that all each sensor observed very high velocity vibrations for about the first 1200 seconds in the case of the 50% throttled valve. This is completely consistent with the higher Health Indicator value observed over the same time frame. Additionally, it is clear that the velocity changes with both load and fault conditions. When faulted, the velocity is higher on two axes (x and z), and when the valve is simply throttled, it is higher on all three axes. Note that the speed dependence noted in [75] is clearly evident.

These results are somewhat interesting, and they highlight the power of the Health Indicator to aggregate the underlying data into a meaningful value. Note that once the initial large variations die out on the x and z axes in the throttled case, the overall Health Indicator drops closer to normal and below the faulted case. Even if the change in valve setting were to cause an alarm in this case, corresponding flow

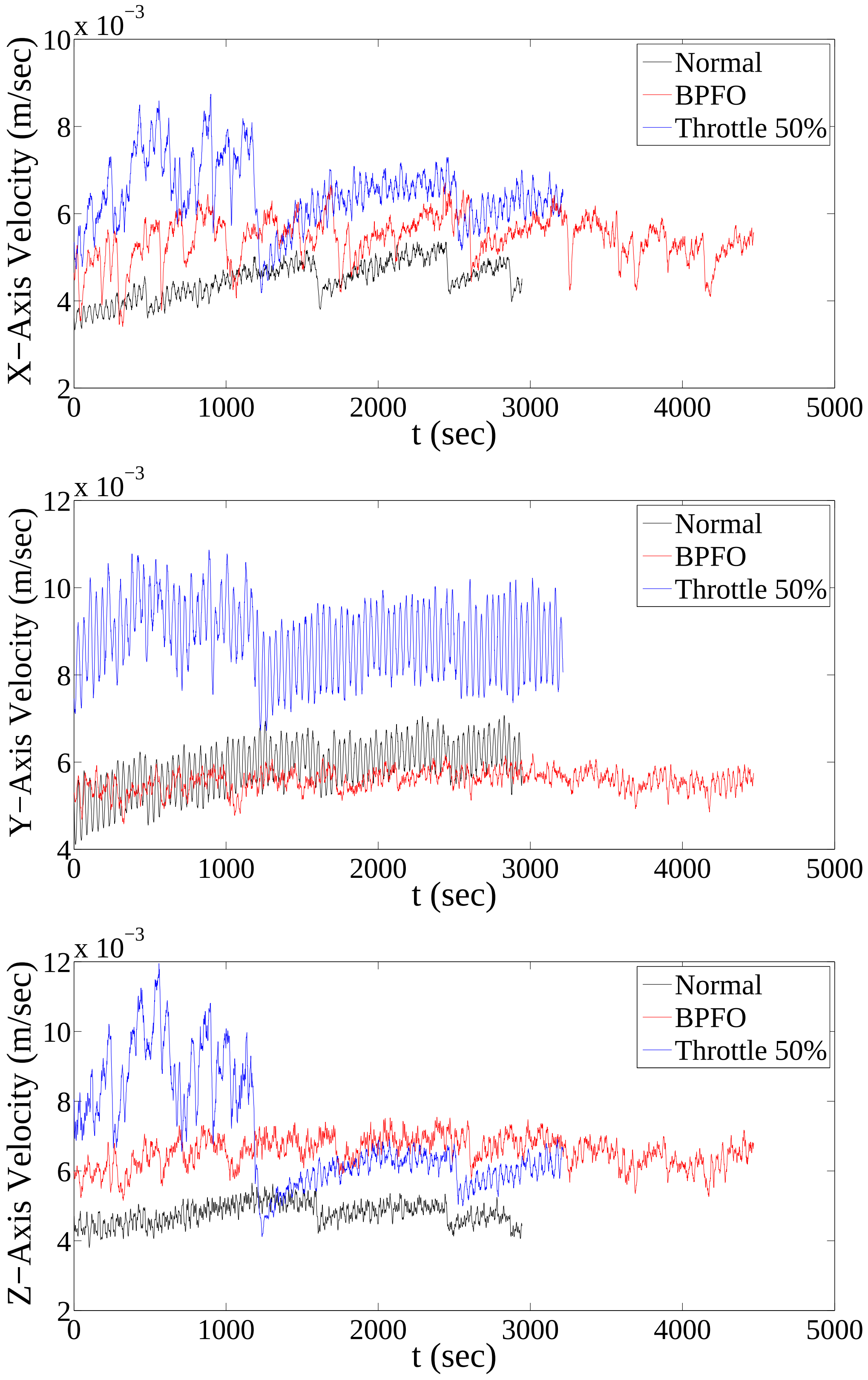


Figure 4.8: X-axis, Y-axis, and Z-axis velocity versus time for three of the cases extracted from Fig. 4.7.

measurements would indicate that there had been a change in motor load and thus would provide warning that some change in vibration might be expected. This data set could be stored on shore for later development of a more robust expert system.

Note that the Health Indicator value clearly increases as the motor load changes. Even though the Health Indicator value is clearly lower in those cases than it is when faults are in place, it is easy to see the potential difficulty in distinguishing load changes from early-stage faults.

The algorithm is able to be retrained if it is known that the load has changed. In this case, the change in speed is a clear indicator that the load setting has changed and that a re-training is needed. Such a retraining is relatively easy to perform on-the-fly and has been tested in real-time.

At first, it may seem as though on-the-fly re-training is inappropriate. Recall, however, that vibrations and other fault indicators are dependent on loading conditions [75]. Since heavily load-dependent quantities (i.e. current amplitude, speed, etc.) have been removed from the feature vector, the corresponding changes in the Health Indicator resulting from load changes are clearly the result of changes in various fault indicator values (i.e. vibrations). It is thus to be expected that individual KPI values also change with loading. Therefore, even a rules-based monitoring scheme would need to account for such activities if it were to be appropriately robust. The proposed method uses a single, succinct indicator to track such changes and is thus potentially better in this regard. The benefits over a rules-based analysis of all KPI values are considered more carefully in Sec 4.4.3.

4.4.3 Detection of an actual fault

An early-stage fault very similar to what is expected in the field was accidently developed during the course of laboratory tests. At one point, one of the "healthy" bearings became stuck on the shaft, and a laboratory assistant had to use reasonable force to remove it. In the process, the bearing fell to the floor. No visible damage

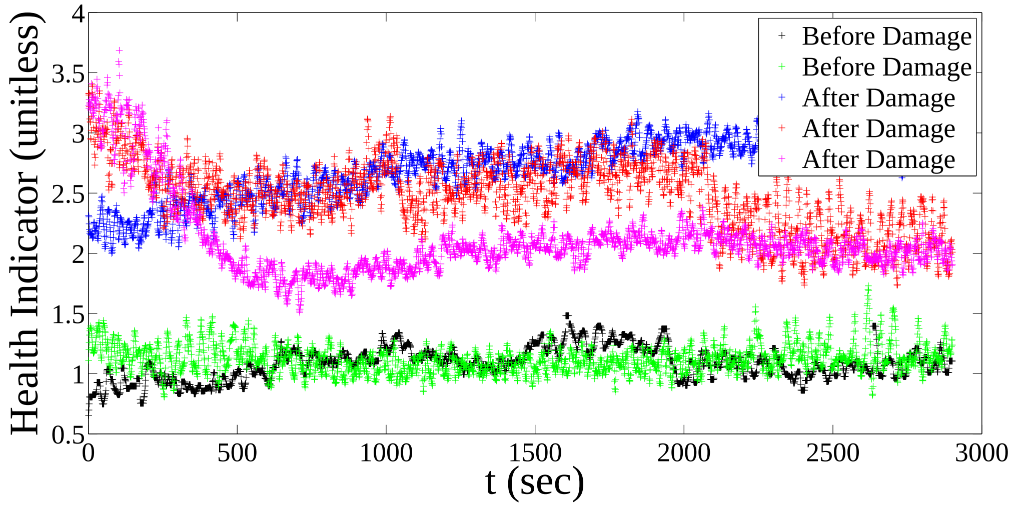


Figure 4.9: Health Indicator value versus time for the normal bearing before and after it was damaged in the laboratory.

was detected, and the bearing seemed completely normal. During subsequent testing, however, the Health Indicator jumped as shown in Fig. 4.9.

A detailed analysis of the KPI values recorded before and after this fault provides strong support for the use of the proposed algorithm over a rules-based approach. Consider, for instance, Fig. 4.10 which shows the value of the velocity signal recorded from the y-axis accelerometer on the bearing housing. A small change has clearly occurred. An investigation of several other fault indicators, however, shows how difficult the fault-detection process can be and how powerful our scheme is at isolating changes.

Figures 4.11 shows two examples, namely the amplitude of the ball-defect signal recorded by the y-axis accelerometer and the amplitude of the inner-race signal recorded by the x-axis accelerometer, respectively. In the case of the former, there is a clear change in mean; in the case of the latter, all of the data appears to be about the same. A change in steady-state speed was also observed before and after the maintenance was performed, and this is likely due to increased rolling friction. The flow rate, however, was unchanged.

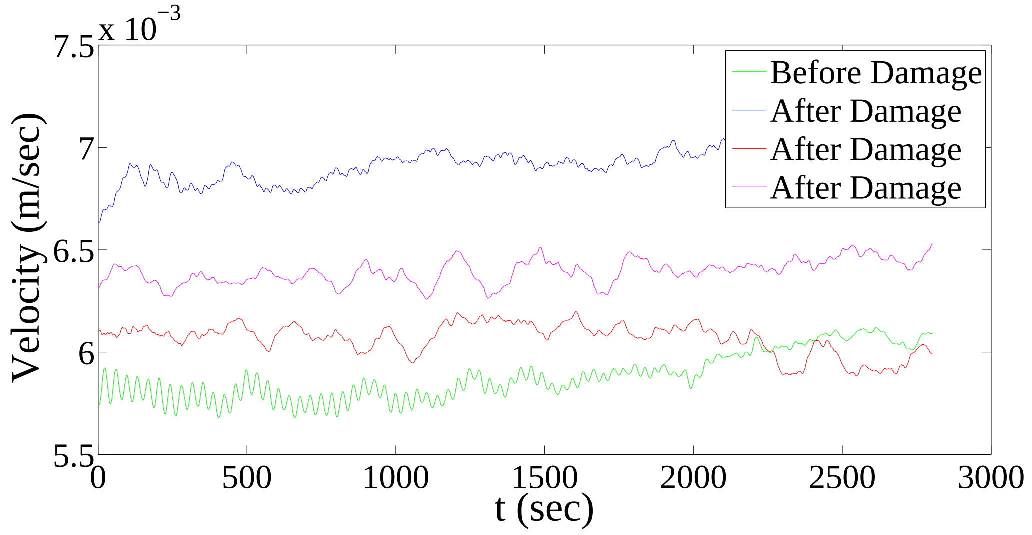


Figure 4.10: Velocity versus time for the normal bearing before and after it was damaged in the laboratory.

This example demonstrates the potential benefits of the proposed fault-detection algorithm over more traditional rules-based approaches. For instance, it is unlikely that a sufficient set of automated rules would have noticed the subtle changes in the various fault indicators. For instance, it is clear that the variance of several indicators has changed, but such metrics are unlikely to be recorded in a rules-based system.

The Health Indicator, however, clearly shows that there has been some sort of a change away from the previously believed normal conditions. This Health Indicator could easily alert a human operator of the need to perform a more detailed investigation of the individual KPIs which would indeed show changes to the naked eye. This approach is thus conservative, as it simply gives the operator the sense that there has been a change and that he must investigate. It does not attempt to make any particularly determinations, and thus potentially reduces the opportunity for a missed fault or an improper diagnosis. In this case, an operator would likely note that the bearing had changed and would consider the new condition to be normal. The algorithm would be re-trained, and the operator would flag the occurrence, noting that it is possible that further changes would be detected in the future.

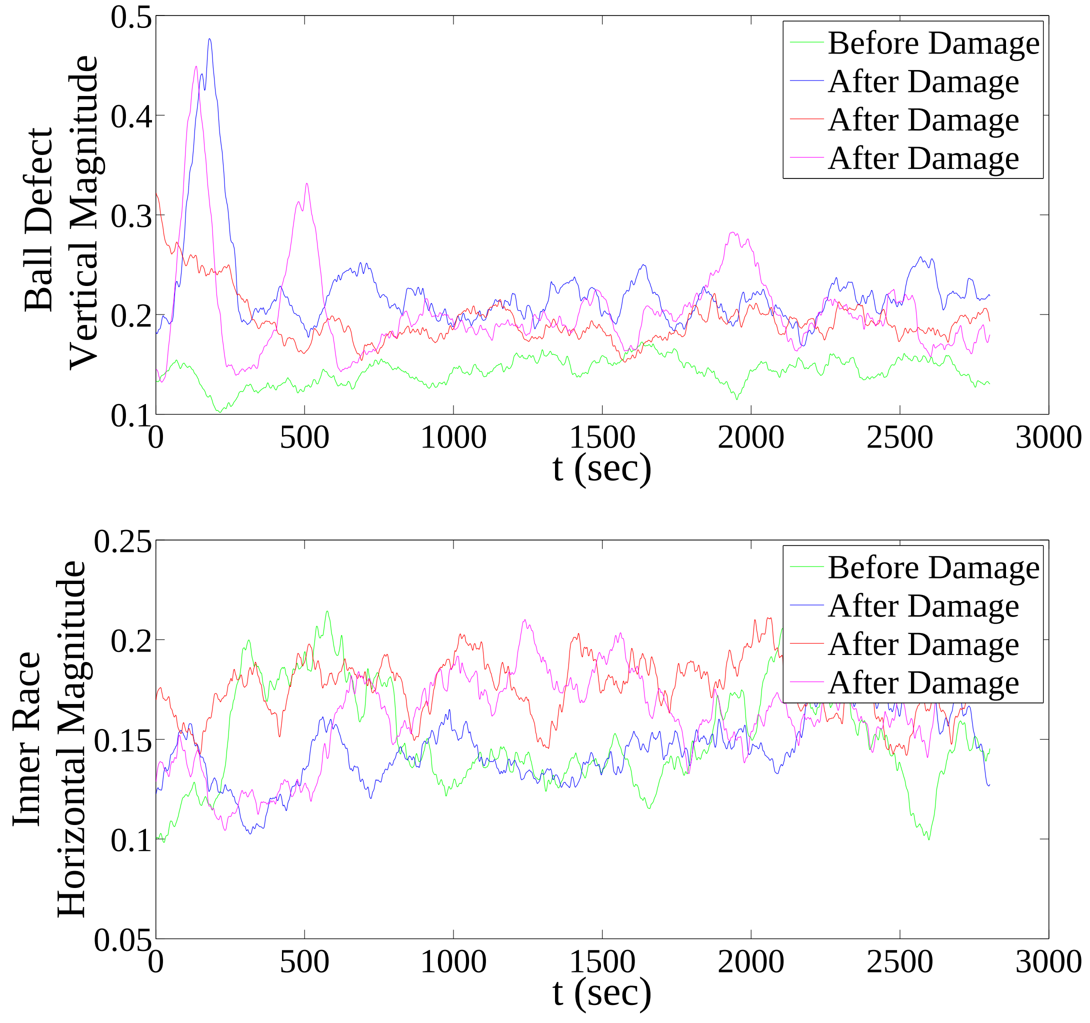


Figure 4.11: Amplitude of the ball-defect signal recorded by the vertical accelerometer and the inner-race signal recorded by the horizontal accelerometer before and after the normal bearing was damaged.

CHAPTER 5: CONCLUSION

5.1 Summary

This dissertation proposes an approach for robust condition monitoring for power conversion assets that robustly detects faults locally allowing for remote fault diagnosis. The condition monitoring algorithm was developed based on a concept from facial recognition. The algorithm is used for fault detection in two key power conversion assets: power electronic drives, specifically IGBTs, and electric motors. Online methods for early stage fault detection will become essential as systems become more heavily dependent on power electronics. The condition monitoring algorithm for power switches is able to distinguish between true faults and changes in operating conditions and does not require the development of rules to set thresholds for fault detection. This dissertation also develops several methods for the online measurement of key health indicators in several key power electronic drive components: IGBTs, MOSFETs, and dc bus capacitors. This dissertation has also demonstrated the potential of a robust, automated condition monitoring algorithm for electric motors and their driven loads. This method is clearly sensitive to very small changes in load behavior and that those changes can be correlated with the underlying KPI data, as well as maintenance records and other process-variable measurements.

5.2 Future Work

The approach for condition monitoring proposed in this dissertation has several opportunities for future work. Ongoing work focuses on the inclusion of additional features and the completion of more testing for both the motor and switch condition monitoring. These tests include building a larger training set of normal motor and drive features. There is particular interest in testing with variations in load and looking into the effects of changes in ambient temperature.

5.2.1 Future Work for Fault Detection in Power Electronics

One potential improvement for fault detection in power semiconductors is using the advanced gate-drive concept that has been described in several works [27], [91], [92], [93]. Figure 5.1 shows a three-phase, full-bridge inverter capable of sampling the required terminal variables. FPGA-based gate drives have been used to acquire switch terminal variables at rates as high as 100 MHZ with emphasis on their use in optimizing turn-on and turn-off performance [91], [92]. The signals measured by such devices can also be used to extract meaningful health-related features, such as threshold voltage, on-state voltage and on-state resistance. Feature extraction could be performed locally at the gate-drive unit, and features could be transmitted back to a central controller over fiber-optic cables for better noise immunity at a much lower data rate.

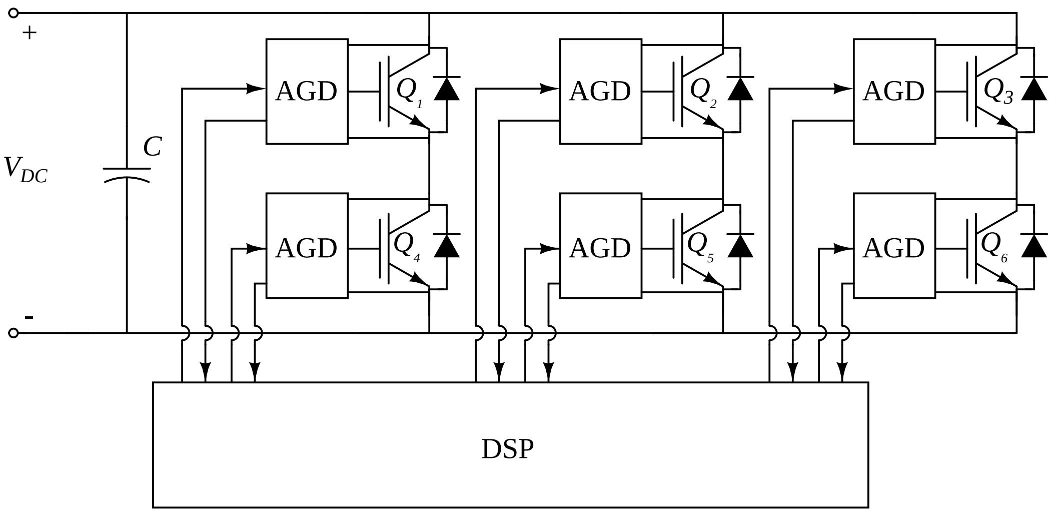


Figure 5.1: A potential smart drive architecture including FPGA-based gate drives. These devices are labeled here as advanced gate drivers (AGDs). The AGDs provide the controller with appropriate measurements. These connections would be fiber optic [71].

The advanced gate-drive concept provides a feasible approach for monitoring key indicators. In this dissertation these signals were obtained without the use of high-speed FPGAs. It would not be difficult to implement the condition monitoring algorithm using an FPGA-based gate drive, so the off-line processing should not be viewed as a limitation of the approach. The algorithm could be implemented by either the controller of the inverter or at each individual gate drive. It was used here simply for ease of demonstration. It should be noted that the limited resolution of the high-speed data converters on the FPGA would likely require some level of analog preprocessing regardless of the format of the implementation.

One major advantage to using the advanced gate-drive concept that it would allow additional features that are not currently being monitored to be such as the threshold voltage, V_T. V_T is the gate voltage at which the IGBT turns on and collector current begins to flow. IGBTs have a negative temperature coefficient for the V_T which leads to a drop in V_T as the temperature increases [59]. This is due to the fact that an increase in temperature leads to a decrease in the band-gap of the silicon, which in turn reduces V_T. Therefore it is easier to turn-on the IGBT at higher ambient temperatures. A reduction in V_T with an increase in temperatures is seen in both new and aged IGBTs. From experiments in [59], it was observed that aged parts have a higher threshold voltage than the new parts across all temperatures.

5.2.2 Future Work for Fault Detection in Electric Motors

Future testing to improve the fault detection scheme proposed in this dissertation is to collect more data across a wide range of operating conditions. This can be done in the laboratory by using the adjustable magnetic brake attached the machinery fault simulator. A larger training set could be recorded to create a better and more extensive feature set. Additional work can also look at adding additional features into the feature set. Some additional measurements include RTDs and thermocouples on the motor and bearing housings for measuring temperatures and adding more accelerometers mounted onto the motor for additional vibration measurements.

In a more practical implementation of the condition monitoring algorithm the features could be recorded across many similar assets and sent to a centralized database. By doing this sufficient data can be obtained to help develop expert systems with powerful inference engines. Human operators could also observe the data at a centralized location for diagnosis as well. This requires significant data collection for the development of such system, but since the proposed scheme would be easy to integrate into existing processes, the results would likely be worth the additional marginal effort.

REFERENCES

[1] *Grid 2030: A National Vision for Electricity's Second 100 Years*, United States Department of Energy, Office of Electric Transmission and Distribution, Jul. 2003.

[2] P. Evans and M. Annunziata, *Industrial Internet: Pushing the Boundaries of Minds and Machines*, General Electric, 2012.

[3] P. Vas, *Parameter Estimation, Condition Monitoring, and Diagnosis of Electrical Machines.* Clarendon Press, 1993.

[4] *Fleetwide Monitoring for Equipment Condition Assessment: Final Report.* EPRI, Palo Alto, CA: 2006. 1010266.

[5] J. V. Dyke, J. R. Desgrey, J. Luchs, and M. DiPilla, "An integrated ddg vibration monitoring pilot program," in *Proc. 2009 ASNE Fleet Maintenance and Modernization Symposium (FMMS)*, San Diego, CA, Dec 2009.

[6] C. W. Kang, "Advanced monitoring and advice integrating a comprehensive sensor network for improved operational availability," Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, MA, Feb. 1998.

[7] W. T. Thomson and M. Fenger, "Current signature analysis to detect induction motor faults," *IEEE Indsutry Applications Magazine*, pp. 26–34, July-August 2001.

[8] W. Thomson and R. Gilmore, "Motor current signature analysis to detect faults in induction motor drives fundamentals, data interpretation, and industrial case histories," in *Proc of Thirty-Second Turbomachinery Symposium*, 2003.

[9] V. Wowk, *Machinery Vibration: Measurement and Analysis.* McGraw Hill Inc., 1990.

[10] S. Nandi, H. A. Toliyat, and X. Li, "Condition monitoring and fault diagnosis of electrical motors a review," *IEEE Trans. Energy Convers.*, vol. 20, no. 4, pp. 719–729, Dec. 2005.

[11] S. Braun, *Mechanical Signature Analysis: Theory and Applications.* London: Academic Press, 1986.

[12] *Equipment Condition Assessment, Volume 2: Technology Evaluation and Integration.* EPRI, Palo Alto, CA: 2004. 1009601.

[13] *Diagnostic Advisor Functional Requirements Specification.* EPRI, Palo Alto, CA: 2009. 1018358.

[14] *Transmission Modernization Demonstration.* EPRI, Palo Alto, CA: 2013. 1026865.

[15] *Distribution Modernization Demonstration.* EPRI, Palo Alto, CA: 2013. 1025705.

[16] M. Turk and A. Pentland, "Eigenfaces for recognition," *Journal of Cognitive Neuroscience*, vol. 3, no. 1, pp. 71–86, 1991.

[17] ——, "Face recognition using eigenfaces," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 1991, pp. 586–591.

[18] R. Duda, P. Hart, and D. Stork, *Pattern Classification.* Wiley, 2006.

[19] J. Bay, *Fundamentals of Linear State Space Systems.* Boston, MA: WCB / McGraw-Hill, 1999.

[20] A. M. Imam, T. G. Habetler, R. G. Harley, and D. Divan, "Failure prediction of electrolytic capacitors using dsp methods," in *Proc. of the 20th IEEE Applied Power Electronics Conference (APEC'05)*, vol. 2, 2005, p. 965970.

[21] T. Dalton, A. Boughner, C. Mako, and C. N. Doerry, "LHD 8: A step toward the all electric warship," in *ASNE Day*, 2002.

[22] A. N. Inc., "Variable frequency drives: More reliability, less houseload, more profit," 2008.

[23] J. Rodriguez, J. S. Lai, and F. Z. Peng, "Multilevel inverters: A survey of topologies, controls, and applications," *IEEE Trans. Ind. Electron.*, vol. 49, no. 4, p. 724738, Aug. 2002.

[24] A. Emadi, J. L. Young, and K. Rajashekara, "Power electronics and motor drives in electric, hybrid electric, and plug-in hybrid electric vehicles," *IEEE Power Electron. Lett.*, vol. 55, no. 6, p. 22372245, 2008.

[25] J. M. Anderson, R. W. Cox, and J. Noppakunkajorn, "An on-line fault diagnosis method for power electronic drives," in *Proc of IEEE Electric Ship Technologies Symposium*, Alexandria, VA, Apr. 2011, pp. 492 – 497.

[26] L. Chen, F. Z. Peng, and D. Cao, "A smart gate drive with self-diagnosis for power MOSFETs and IGBTs," in *Proc. of IEEE Applied Power Electronics Conference and Exposition*, Austin, TX, Feb. 2008, pp. 1602–1607.

[27] P. Xiao, G. K. Venayagamoorthy, K. A. Corzine, and R. Woodley, "Self-healing control with multifunctional gate drive circuits for power converters," in *Conference Record of the 2007 IEEE Industry Applications Conference*, New Orleans, LA, Sept. 2007, pp. 1852–1858.

[28] T. Liu, J. Fen, and T. Lipo, "A strategy for improving reliability of field-oriented controlled induction motor drives," *IEEE Trans. Ind. Appl.*, vol. 29, no. 5, pp. 910–918, Sept./Oct. 1993.

[29] R. Ribeiro, C. Jacobina, E. da Silva, and A. Lima, “Fault detection of open-switch damage in voltage-fed pwm motor drive systems,” *IEEE Trans. Power Electron.*, vol. 18, no. 2, pp. 587–593, Mar. 2003.

[30] A. Kharegakar and P. Kumar, “A novel scheme for protection of power semiconductor devices against short circuit faults,” *IEEE Trans. Ind. Electron.*, vol. 42, no. 3, p. 344351, Jun. 1994.

[31] J. Kimball, B. Kuhn, and R. Balog, “Analysis and simulation of five-phase variable-speed induction motor drives under asymmetrical connections,” *IEEE Trans. Power Electron.*, vol. 13, no. 4, pp. 748–756, Jul. 1998.

[32] N. D. Benavides, T. J. McCoy, and M. A. Chrin, “Reliability improvements in integrated power systems with pressure-contact semiconductors,” in *Proc. ASNE Day*, Apr 2009.

[33] M. Ciappa and W. Fichtner, “Lifetime prediction of IGBT modules for traction applications,” in *Proc. IEEE Reliability Physics Symposium*, San Jose CA, 2000, pp. 210–216.

[34] J. Celaya, N. Patil, S. Saha, P. Wysocki3, and K. Goebel, “Towards accelerated aging methodologies and health management of power MOSFETs (technical brief),” in *Proc. of Annual Conference of the Prognostics and Health Management Society*, San Diego, CA, Sept./Oct. 2009.

[35] G. Sonnenfeld, K. Goebel, and J. R. Celaya, “An agile accelerated aging, characterization and scenario simulation system for gate controlled power transistors,” in *Proc of IEEE AUTOTESTCON*, Salt Lake City, UT, Sept. 2008, pp. 208–215.

[36] M. Trivedi and K. Shenai, “Failure mechanisms of IGBTs under short-circuit and clamped inductive switching stress,” *IEEE Trans. Power Electron.*, vol. 14, no. 1, pp. 108–116, Jan. 1999.

[37] J. Morroni, A. Dolgov, R. Zane, and D. Maksimovi?, “Online health monitoring in digitally controlled power converters,” in *Proc. IEEE Power Electron. Specialists Conf.*, Orlando, FL, Jun 2007.

[38] S. Lombardo, J. Stathis, B. Linder, K. Pey, F. Palumbo, and C. Tung, “Dielectric breakdown mechanisms in gate oxides,” *Journal of Applied Physics 98*, 2005.

[39] G. Buh, H. Chung, and Y. Kuk, “Real-time evolution of trapped charge in a sio2 layer: An electrostatic force,” *Applied Physics Letters*, vol. 79, no. 13, pp. 2010–2012, 2001.

[40] Military Handbook 217 F, *Reliability prediction of electronic equipment, revision F*, Notice 1, 10 July 1992, Notice 2, Feb. 28, 1995, ed., Dec. 1991.

[41] C. Kulkarni, G. Biswas, and X. Koutsoukos, “A prognosis case study for electrolytic capacitor degradation in DC/DC converters,” in *Proc. Annual Conf. of the Prognostics and Health Management Society 2009*, San Diego, Oct 2009.

[42] M. L. Gasperi, “Life prediction modeling of bus capacitors in ac variable-frequency drives,” *IEEE Trans. Ind. Appl.*, vol. 41, no. 6, pp. 1430–1435, Nov./Dec. 2005.

[43] J. R. Black, “Electromigration - a brief survey and some recent results,” *IEEE Trans. Electron Devices*, vol. 16, no. 4, pp. 338–347, April 1969.

[44] E. Ameraseka and F. Najm, *Failure Mechanisms in Semiconductor Devices*, 2nd ed. John Wiley & Sons Ltd, 1998.

[45] J. Celaya, A. Saxena, P. Wysocki, S. Saha, and K. Goebel, “Towards prognostics of power MOSFETs: Accelerated aging and precursors of failure,” in *Proc of Annual Conference of the Prognostics and Health Management Society*, Portland, Oregon, Oct. 2010.

[46] M. Ciappa, “Selected failure mechanisms of modern power modules,” *Microelectronics Reliability*, no. 42, pp. 653–667, Jan 2002.

[47] W. Wu, M. Held, P. Jacob, P. Scacco, and A. Birolini, “Investigation on the long term reliability of power IGBT modules,” in *Proceedings of International Symposium on Power Semiconductor Devices & ICs*, Yokohama, Japan, 1995, pp. 443–448.

[48] D. Katsis and J. van Wyk, “Void-induced thermal impedance in power semiconductor modules: Some transient temperature effects,” *IEEE Trans. Ind. Appl.*, vol. 39, no. 5, pp. 1239–1246, Sept. / Oct. 2003.

[49] J. D. Kraus, *Electromagnetics.* McGraw-Hill Book Company, Inc., 1953.

[50] D. Goodman, “Prognostic methodology for deep submicron semiconductor failure modes,” *IEEE Trans. Compon. Packag. Technol.*, vol. 24, no. 1, p. 109111, Mar. 2001.

[51] A. Hamidi, N. Beck, K. Thomas, and E. Herr, “Reliability and lifetime evaluation of different wire bonding technologies for high power IGBT modules,” *Microelectronics Reliability*, no. 39, pp. 1153–1158, 1999.

[52] M. Gasperi, N. Gollhardt, and R. Sladky, “A model for equivalent series resistance in aluminum electrolytic capacitors,” in *Proc. Capacitor and Resistor Technology Symp. (CARTS)*, Jupiter, Fl., Mar. 1997, pp. 71–75.

[53] J. Scholte and W. C. van Geel, “Impedances of the electrolytical rectifier,” Philips Res. Rep, Tech. Rep., 1953.

[54] N. Mohan, T. Undeland, and W. Robbins, *Power Electronics Converters, Applications, and Design, Third Edition.* John Wiley & Sons, 2003.

[55] D. Krug, S. Bernet, S. Fazel, K. Jalili, and M. Malinowski, "Comparison of 2.3-kv medium-voltage multilevel converters for industrial medium voltage drives," *IEEE Trans. Ind. Electron.*, vol. 54, no. 6, p. 29792992, Dec. 2007.

[56] Data Translation, *UM-21336-T DT9816 Series Users Manual*, Sept. 2012.

[57] *HCNR200 and HCNR201: High-Linearity Analog Optocouplers*, Avago Technologies, 2011.

[58] D. Brown, M. Abbas, A. Ginart, I. Ali, P. Kalgren, and G. Vachtsevanos, "Turn-off time as a precursor of gate bipolar transistor latch-up faults in electric motor drives," in *Proc of Annual Conference of the Prognostics and Health Management Society*, Portland, Oregon, Oct. 2010.

[59] N. Patil, D. Das, K. Goebel, and M. Pecht, "Identification of failure precursor parameters for insulated gate bipolar transistors (igbts)," in *Proc of International Conference of Prognostics and Health Management*, Denver, CO, Oct. 2008, pp. 1–5.

[60] B. Baliga, *Fundamentals of Power Semiconductor Devices.* New York, NY: Springer LLC, 2008.

[61] J. Holtz, "Pulsewidth modulation for electronic power conversion," *Proc. IEEE*, vol. 82, no. 8, pp. 1194–1214, Aug 1994.

[62] "IRF540N HEXFET power MOSFET," 2001.

[63] A. Parlos, K. Kim, and R. M. Bharadwaj, "Early detection of plant equipment failures: a case study in just-in-time maintenance," in *Proc. of 2001 American Nuclear Society Annual Meeting*, Milwaukee, WI, Jun. 2001, pp. 1153–1160.

[64] OPNAVINST 4790.16A, "Condition-based maintenance (CBM) policy," Dec. 2007.

[65] R. Caccese and D. Lewis, "Implementing true CBM+ in the fleet," in *Proc. ASNE Intelligent Ships Symposium VIII*, Philadelphia, PA, May 2009.

[66] M. G. Walker, R. Kapadia, B. Sammuli, and M. Venkatesh, "A model-based reasoning framework for condition based maintenance and distance support," in *Proc. ASNE Automation and Controls Symposium*, Biloxi, MS, Dec. 2007.

[67] G. B. Kliman and J. Stein, "Induction motor fault detection via passive current monitoring," in *Proc. Int. Conf. Electrical Machines*, 1990, pp. 13 – 17.

[68] R. A. K. G. B. Kliman, W. J. Premerlani and D. Hoeweler, "A new approach to on-line fault detection in ac motors," in *Proc. IEEE Industry Applications Soc. Annual Meeting Conf.*, San Diego, CA, 1996, p. 687693.

[69] P. Tavner and J. Penman, *Condition Monitoring of Electrical Machines.* Letchworth, U.K.: Res. Studies Press, 1987.

[70] A. H. Bonnett and G. C. Soukup, "Rotor failures in squirrel cage induction motors," *IEEE Trans. Ind. Appl.*, vol. 22, no. 6, p. 11651173, Nov./Dec. 1986.

[71] O. V. Thorsen and M. Dalva, "A survey of faults on induction motors in offshore oil industry, petrochemical industry, gas terminals, and oil refineries," *IEEE Trans. Ind. Appl.*, vol. 31, no. 5, pp. 1186–1196, Sep./Oct. 1995.

[72] *Rotating electrical machinesPart 14: Mechanical vibration of certain machines with shaft heights 56 mm and higher-Measurement, evaluation, and limits of vibration*, International Electrotechnical Commission Std., 1996, iEC 3414.

[73] *Mechanical vibration Evaluation of machine vibration by measurements on non-rotating parts Part 1: General guidelines*, International Organization for Standardization Std., iSO 10 816-1:1995(E).

[74] *Section 1: General standards applying to all machines Part 7: Mechanical vibration- Measurements, evaluation, and limits NEMA MG-1 1993.*, National Electrical Manufacturers Association Std., nEMA MG-1 1993.

[75] J. R. Stack, T. G. Habetler, and R. G. Harley, "Effects of machine speed on the development and detection of rolling element bearing faults," *IEEE Power Electron. Lett.*, vol. 1, no. 1, pp. 19–21, Mar 2003.

[76] B. Li, M. Chow, Y. Tipsuwan, and J. Hung, "Neural-network-based motor rolling bearing fault diagnosis," *IEEE Trans. Ind. Electron.*, vol. 47, no. 5, pp. 1060–1069, Oct. 2000.

[77] B. Yang, D. Lim, and A. Tan, "Vibex: An expert system for vibration fault diagnosis of rotating machinery using decision tree and decision table," *Expert Systems with Applications*, vol. 28, pp. 735–742, 2005.

[78] S. Ebersbach and Z. Peng, "Expert system development for vibration analysis in machine condition monitoring," *Expert Systems with Applications*, vol. 34, pp. 291–299, 2008.

[79] R. R. Schoen, B. K. Lin, T. G. Habetler, H. J. Shlog, and S. Farag, "An unsupervised on-line system for induction motor fault detection using stator current monitoring," *IEEE Trans. Ind. Appl.*, vol. 31, no. 6, pp. 1280–1286, Nov./Dec.

[80] M. Chow, R. Sharpe, and J. Hung, "On the application and design of artificial neural networks for motor fault detection part I," *IEEE Trans. Ind. Electron.*, vol. 40, no. 2, pp. 181–188, Apr. 1993.

[81] ——, "On the application and design of artificial neural networks for motor fault detection part II," *IEEE Trans. Ind. Electron.*, vol. 40, no. 2, pp. 189–196, Apr. 1993.

[82] F. Filippetti, G. Franceschini, and C. Tassoni, "Neural networks aided on-line diagnostics of induction motor rotor faults," *IEEE Trans. Ind. Appl.*, vol. 31, no. 4, pp. 892 – 899, Jul / Aug 1995.

[83] F. Filippetti, G. Franceschini, C. Tassoni, and P. Vas, "AI techniques in induction machines diagnosis including the speed ripple effect," *IEEE Trans. Ind. Appl.*, vol. 34, no. 1, pp. 98–108, Jan./Feb. 1998.

[84] M. Sin, W. Soong, and N. Ertugrul, "Induction machine on-line condition monitoring and fault diagnosis a survey," in *Australasian Universities Power Engineering Conference (AUPEC)*, Christchurch, New Zealand, 2003, pp. 1–6.

[85] M. Seera, C. Lim, D. Ishak, and H. Singh, "Fault detection and diagnosis of induction motors using motor current signature analysis and a hybrid fmm cart model," *IEEE Trans. Neural Netw. and Learning Sys.*, vol. 23, no. 1, pp. 97–108, 2012.

[86] P. Vas, *ArtificialIntelligence-Based Electrical Machines and Drives: Applications of Fuzzy, Neural, Fuzzy-Neural and Genetic Algorithm Based Techniques.* New York: Oxford University Press, 1999.

[87] M. A. Awadallah and M. M. Morcos, "Application of AI tools in fault diagnosis of electrical machines and drives an overview," *IEEE Trans. Energy Convers.*, vol. 18, no. 2, pp. 245–251, Jun. 2003.

[88] F. Filippetti, M. Martelli, G. Franceschini, and C. Tassoni, "Development of expert system knowledge base to on-line diagnosis of rotor electrical faults of induction motors," in *Proc. of IEEE Industrial Applications Society Annual Meeting*, 1992, p. 9299.

[89] A. E. Fitzgerald, C. Kingsley, and S. Umans, *Electric Machinery*, 6th ed. McGraw-Hill, Inc., 2002.

[90] R. Isermann, *Fault-Diagnosis Systems.* Berlin, Germany: Springer, 2006.

[91] Y. Lobsiger and J. W. Kolar, "Voltage, current and temperature measurement concepts enabling intelligent gate drives," in *European Center for Power Electronics (ECPE) Workshop - Electronics Around the Power Switch: Gate Drives, Sensors, and Control*, 2011.

[92] H. Kuhn, T. Koneke, and A. Mertens, "Potential of digital gate units in high power applications," in *Proc. 13th Power Electronics and Motion Conference (EPE-PEMC)*, 2008, pp. 1458–1464.

[93] ——, "Considerations for a digital gate unit in high power applications," in *Proc. 2008 Power Electronics Specialists Conference (PESC)*, 2008, pp. 2784–2790.

[94] A. Merello, A. Rugginenti, and M. Grasso, *Using Monolithic High Voltage Gate Drivers*, International Rectifier.

[95] A. Oppenheim and R. Schafer, *Discrete-time Signal Processing.* Prentice Hall, 1999.

[96] M. Blodt, P. Granjon, B. Raison, and G. Rostaing, “Models for bearing damage detection in induction motors using stator current monitoring,” *IEEE Trans. Ind. Electron.*, vol. 55, no. 4, p. 18131822, Apr. 2007.

APPENDIX A: MATLAB CODE

A.1 Health Monitoring Algorithm Code

KPI Classifier Code

```
%'filename' is used as generic filename in this example of
%the KPI classifier

data = filename(:,:);
[Ureduced,psi,lambda] = defineEigenSpace2(data');
[E_filename, E2_filename] = KPI2E(data28, Ureduced, psi, lambda);
```

Function Used to Calculate Error from KPIs

```
function[E,E2] = KPI2E(data, Ureduced, psi, lambda)
    omegaMatrix = eigenParams2(data',0,Ureduced,psi,lambda);
    data_hat = bsxfun(@plus,omegaMatrix*Ureduced',psi');
    e = data-data_hat;
    E = zeros(size(e,1),1);
    E2 = zeros(size(e,1),1);
    for ii = 1:size(e,1)
        vec = e(ii,:);
        E(ii) = norm(vec);
        E2(ii) = norm(vec)/norm(data(ii,:));
    end
end
```

Function Loads or Calculates Eigenspace Basis Vectors, Ureduced, and the Mean Vector

```
% This fuction loads or calculates the eigenspace basis
% vectors, Ureduced, and also the mean vector, psi. Psi
% must be subtracted from evt transients before being
% projected into the eigenspace defined by Ureduced.
%
% If no inputs are given the function will lookfor and
% load Ureduced and psi from a EIGEN_SPACE.mat file in
% the current directory.
%
% [Ureduced,psi,lambda] = defineEigenSpace();
%
% If inputs are given, Ureduced and psi will be
% calculated and then saved to EIGEN_SPACE.mat in the
% current directory. This will write over any existing
% EIGEN_SPACE_2.mat file.
%
%* USE 'eigenParams.m' for projecting evts into eigenspace.*
%
% function [Ureduced,psi,lambda] = defineEigenSpace2(gammaMatrix)
%
% Inputs:
% gammaMarrix, a the complete set of test sample features
% in a DxM matrix. Where D is the dimensions of the
% feature vectors, and M is the number of sample vectors.
%
% D = number of pts in vector
% M = number of training Vectors
% Ufull, Dx(D or M) matrix of eigenvectors, min(D,M)
% lambda, (D or M)x1 vector of corresponding eigenvalues, min(D,M)
% psi, a Dx1 image mean vector
%
% Nt = number of test vectors
% K = reduced dimensions from D (K<D)
% PhiMatrix, DxM zero-mean traing vectors
% Ureduced, DxK matrix of eigenvectors
% omegaMatrix is MxK matrix, K reduced dimensions,
% projected training vectors => the features for
% each training vector

function [Ureduced,psi,lambda] = defineEigenSpace2(gammaMatrix)

if ~exist('gammaMatrix','var')
    disp('Loaded current EIGEN_SPACE_2.mat')
    load('EIGEN_SPACE_2.mat'); % this loads Ureduced and psi
    return
end

% Setting Full Space
```

```
[Ufull, lambda, psi] = computeFullEigenSpace(gammaMatrix);
PhiMatrix = bsxfun(@minus,gammaMatrix,psi); % zero mean image ↘
→vectors
[omegaMatrix, Ureduced] = reduceEigenSpace(Ufull,lambda,PhiMatrix);

%%% note: omegaMatrix = PhiMatrix1'*Ureduced;

% Saving

save('EIGEN_SPACE_2.mat','Ureduced','psi','lambda');
disp('Saved new EIGEN_SPACE_2.mat')
end

function [Ufull, lambda, psi] = computeFullEigenSpace(gammaMatrix)
% Computes the Eigenspace vectors/values, and the mean
% input vector for a set of M training vectors with D
% dimensions.
%
% [Ufull, lambda, psi] = computeFullEigenSpace(gammaMatrix)
%
% Input,
% gammaMarrix, a the complete set of test images in a
% DxM matrix.
% Where D is the dimensions of the image vectors, and
% M is the number of image vectors.
%
% Output,
% Ufull, a DxM matrix of eigenvectors of the full eigenspace ( ||↘
→Ufull(:,i)|| = 1 ).
% lambda, a Mx1 vector of eigenvalues (in descending order).
% psi, a Dx1 image mean vector.

[D,M]=size(gammaMatrix);
% average the image vectors (2nd dim)
psi = (1/M)*sum(gammaMatrix,2);

% zero mean image vectors
PhiMatrix = bsxfun(@minus,gammaMatrix,psi);

if D>M
    C = (1/M)*PhiMatrix'*PhiMatrix; % if D>M
else
    C = (1/M)*PhiMatrix*PhiMatrix'; % do it this way for M>D
end

[vec,d]=eig(C);
[lambda,index] = sort(diag(d),'descend');
% sorting eigenvectors to correspond with sorted eigenvalues
U= vec(:,index);
% if eigenvalue is <= 0 set corresponding eignevector to zero(↘
→numerical instability)
U(:,lambda<=0)=0;

if D>M
```

```
    %if D>M also makes ||Ufull(:,i)|| = 1
    Ufull = bsxfun(@times,((1./(M*lambda)).^0.5)',PhiMatrix*U);
else
    % do it this way for M>D
    Ufull = U;
end
end


function [omegaMatrix, Ureduced] = reduceEigenSpace(Ufull,lambda,↘
→PhiMatrix)
% Reduces the Eigenspace dimensions from M to K (K<M<D);
% eigenvectors are D dimensions.
%
% [omegaMatrix, Ureduced] = reduceEigenSpace(Ufull,lambda,PhiMatrix↘
→)
% Input,
% Ufull, DxM matrix of eigenvectors for the full eigenspace.
% lambda, Mx1 vector of corresponding eigenvalues.
% PhiMatrix, DxM zero-mean traing vectors, M vectors of D dimension↘
→.
%
% Output,
% omegaMatrix, MxK matrix of reduced eigenspace training
% vectors (K coordinates for each training vector M).
% Ureduced, DxK matrix of eigenvectors associated with
% the K dimensional eigenspace.

[D,M] = size(Ufull);
avgVal = (1./M).*sum(lambda);
Ureduced = Ufull(:,lambda>avgVal)

% must have at least 2 dimensions.
if size(Ureduced,2)<2
    Ureduced = Ufull(:,1:2);
end

% projecting each training vector into reduced eigenspace.
omegaMatrix = PhiMatrix'*Ureduced;

end
```

Function Projects Measurements Into Eigenspace Defined by Ureduced and the Mean Vector

```
% This function projects measurements into an eigenspace
% defined by Ureduced and the mean vector psi. Psi is
% subtracted from evt transients to make them zero-mean
% before being projected into Ureduced.
%
%* USE 'defineEigenSpace.m' to define Ureduced and psi *
%
% If Ureduced and psi not inputted, will load them from
% EIGENSPACE.mat
%
% function omegaMatrix = eigenParams(evtlist,white,Ureduced,psi,↘
→lambda)
%
% white = 1/0, 1 if you want the eigenvector to undergo
% a whitening transform, this makes each of the
% cordinates to have similar values. Otherwise the
% omegaMatrix would be heavily weighted to the first
% few eigenvectors. If white is not given will be set
% to 0.
%
% D = number of pts in vector
% M = number of training Vectors
% Ufull,Dx(D or M) matrix of eigenvectors, min(D,M)
% lambda,(D or M)x1 vector of corresponding eigenvalues, min(D,M)
% psi, a Dx1 image mean vector
%
% Nt = number of test vectors
% K = reduced dimensions from D (K<D)
% PhiMatrix, DxM zero-mean traing vectors
% Ureduced, DxK matrix of eigenvectors
% omegaMatrix is MxK matrix, K reduced dimensions,
% projected training vectors => the features
%for each training vector

function omegaMatrix = eigenParams2(gammaMatrix,white,Ureduced,psi,↘
→lambda)

if ~exist('Ureduced','var')
    load('EIGEN_SPACE_2.mat');
 % this loads Ureduced, psi, and lambda
end

if ~exist('white','var')
    white = 0;
end
% zero mean image vectors
PhiMatrix = bsxfun(@minus,gammaMatrix,psi);
omegaMatrix = PhiMatrix'*Ureduced;
```

```
if white == 1
    unweight_lambda = lambda(1:size(omegaMatrix,2)).^-0.5;
    omegaMatrix = bsxfun(@times,omegaMatrix,unweight_lambda');
end
end
```

A.2 Drive KPI Extraction

Drive KPI Extraction Function Call

```
data = [];

x = load('temp1.raw');

    for n = 1:20
       K = (5000*n-5000)+1:5000*n;
       y = KPIConv(x,K);
       data = [data; y];
    end

save out.kpi data -ascii -tabs -double;
```

Drive KPI Extraction Function

```
function [y] = KPIConv(x,K)

%Scale raw data
w1 = -2.4*(x(K,1)-2.5);
w2 = medfilt1(x(K,2),3);
w3 = medfilt1(x(K,3),3);
w4 = medfilt1(x(K,4),5);
w5 = medfilt1(x(K,5)*100,3);
w6 = medfilt1((x(K,6)-.5)*100,3);

%Edge Detector
x1 = medfilt1(x(K,3),5);
dx1 = filter([1 -1],1,x1);
dx2 = zeros(1,length(dx1));
dx2(dx1 < -.6) = -1;
dx2(dx1 > .6) = 1;
dx3 = filter([1 -1], 1, dx2);
locs = find(dx3 ~= 0);
locs = locs(1:2:end);
slopes = dx3(locs);
start_ind = find(slopes == 1);
end_ind = start_ind+1;

OUTPUT = [];

for m = 1:length(start_ind)-1
N = locs(start_ind(m))+2:locs(end_ind(m))-2;
Ic = w1(N);
VCEon = w2(N);
R = find(VCEon > 0.8 & Ic > 0.4);
Ic2 = Ic(R);
VCEon2 = VCEon(R);
OUTPUT(end+1:end+length(R),[1 2]) = [Ic2(:) VCEon2(:)];
end

OUTPUT2 = sortrows(OUTPUT,1);

M = 1:size(OUTPUT2,1);
px = OUTPUT2(M,1);
py = OUTPUT2(M,2);
P = polyfit(px,py,1);

Irms = sqrt(filtfilt(1/833*ones(833,1), 1, w1.^2));

y1 = mean(Irms(1:length(K)-1500)); %Irms
y2 = P(:,1); %Ron
y3 = P(:,2); %Vd
y4 = mean(w4); %Qi
y5 = mean(w5); %TC
y6 = mean(w6); %TA
```

```
y = [y1,y2,y3,y4,y5,y6];

end
```

A.3 Motor KPI Extraction

Motor KPI Main Code

```
function [KPI] = KPIpro2(rawdata,f_sample)
%Setting up the output KPI matrix
%KPI data is given for every 0.2sec of operation
numsamples = length(rawdata(:,1)); %determining the number of ↘
→samples taken during testing
KPIsampling = f_sample/5; %defines number of samples during a 0.2 ↘
→sec period
vectorref = 1:KPIsampling:numsamples;
time = 0:0.2:numsamples/f_sample; %creates a time vector for plots ↘
→of KPI vs time
KPI = zeros(length(vectorref),187);%preallocating the KPI matrix ↘
→length and writes zeros to all positions
KPI (:,187) = time;
%%
%Scaling the raw data from Labjack
scaled = zeros(length(rawdata),13);
%initializes scaled matrix length and 13 columns, writes zeros to ↘
→all positions

%scale raw line voltage data
scaled(:,1) = ((rawdata(:,1)*(100000/110)*(1/2.5)));
%phase A voltage with Labjack ouputing in voltage
%scaled(:,1) = (100000*2.5/4096*(rawdata(:,1)-2047)/110/2.5);
%phase A voltage
scaled(:,2) = ((rawdata(:,3)*(100000/110)*(1/2.5)));
%phase B voltage with Labjack ouputing in voltage
%scaled(:,2) = (100000*2.5/4096*(rawdata(:,3)-2047)/110/2.5);
%phase B voltage
scaled(:,3) = ((rawdata(:,5)*(100000/110)*(1/2.5)));
%phase C voltage with Labjack ouputing in voltage
%scaled(:,3) = (100000*2.5/4096*(rawdata(:,5)-2047)/110/2.5);
%phase C voltage scale raw current data
scaled(:,4) = (1000*(rawdata(:,2))/110);
%phase A current with Labjack ouputing in voltage
%scaled(:,4) = (1000*1.25/4096*(rawdata(:,2)-2047)/110);
%phase A current
scaled(:,5) = (1000*(rawdata(:,4))/110);
%phase B current with Labjack ouputing in voltage
%scaled(:,5) = (1000*1.25/4096*(rawdata(:,4)-2047)/110);
%phase B current
scaled(:,6) = (1000*(rawdata(:,6))/110);
%phase C current with Labjack ouputing in voltage
%scaled(:,6) = (1000*1.25/4096*(rawdata(:,6)-2047)/110);
%phase C current
%scale raw mechanical speed data
scaled(:,7) = (rawdata(:,7));
%scaling not needed if using labjack and use -v -c in code saving ↘
→data
```

```
%scale raw accelerometer data
scaled(:,8) = rawdata(:,8)*(9.80665/0.1);
%converts the block 1-x accelerometer data from V to m/s^2 based on↘
→ use of 100mv/g accelerometers
scaled(:,9) = rawdata(:,9)*(9.80665/0.1);
%converts the block 1-y accelerometer data from V to m/s^2 based on↘
→ use of 100mv/g accelerometers
scaled(:,10) = rawdata(:,10)*(9.80665/0.1);
%converts the block 1-z accelerometer data from V to m/s^2 based on↘
→ use of 100mv/g accelerometers
scaled(:,11) = rawdata(:,11)*(9.80665/0.1);
%converts the block 2-x accelerometer data from V to m/s^2 based on↘
→ use of 100mv/g accelerometers
scaled(:,12) = rawdata(:,12)*(9.80665/0.1);
%converts the block 2-y accelerometer data from V to m/s^2 based on↘
→ use of 100mv/g accelerometers
scaled(:,13) = rawdata(:,13)*(9.80665/0.1);
%converts the block 2-z accelerometer data from V to m/s^2 based on↘
→ use of 100mv/g accelerometers
%%
%Electrical frequency is determined
%filter current before  finding zero crossing locations
Num = get(FIRfilter3,'Numerator');
%gets filter data from FIR filter design
filtcurrent = filtfilt (Num,1,scaled(:,5));
%filters the noise out off the current signal

%determine electrical frequency by zero crossing
zercrs = crossing(filtcurrent);
% finds and returns index of locations of zero crossings

%frequency calculation
frqcal = zeros (1,length(zercrs));

for d = 1:length (frqcal)
     if d == 1
         frqcal (d) = 0;
     else
         deltazercrs = zercrs(d) - zercrs (d-1);
         %determine number of samples between zero crossings
         if deltazercrs == 1
             frqcal(d) = 0;
         else
             frqcal(d)= (1/(deltazercrs/f_sample))/2;
             %first converts samples to sec, then period length in ↘
             →sec to frequency
         end
     end
end

%create a frequency vector the length of the scaled vectors
frequency = zeros (length(scaled(:,4)),1);
%sets up the size of the vector and writes zeros to all locations
for m = 1:length (frqcal)
```

```
    if m == 1
         frequency (m:zercrs(m)) = 0;
    else
         frequency(zercrs(m-1)+1:zercrs(m))=frqcal(m);
         %writes the calculated frequency to frequency vector index↘
         → matching zercrossing index
    end
end

%RMS voltage calculation
%phase A voltage
KPI(:,4) = KPIrms2( scaled(:,1),vectorref,f_sample );
%phase B voltage
KPI(:,5) = KPIrms2( scaled(:,2),vectorref,f_sample );
%phase C voltage
KPI(:,6) = KPIrms2( scaled(:,3),vectorref,f_sample );

%%
%RMS current calculation
%phase A current
KPI(:,1) = KPIrms2( scaled(:,4),vectorref,f_sample );
%phase B current
KPI(:,2) = KPIrms2( scaled(:,5),vectorref,f_sample );
%phase C current
KPI(:,3) = KPIrms2( scaled(:,6),vectorref,f_sample );
KPI(:,34) = KPItscaling( frequency,vectorref);
% run speed mag  in rpm
KPI(:,41) = run_speed(scaled(:,7),f_sample);
%slip frequency
 slip  = slipratio( KPI(:,34),KPI(:,41),vectorref );
 KPI(:,108) = slip;
 rotorbar = rotorbarsidebands( slip, vectorref );
 bearingvibfreq = bearingfreqs( KPI(:,41), vectorref );
 bearingcurfreq = bearingcurrent(KPI(:,34),bearingvibfreq, ↘
 →vectorref);
%%
%Find the magnitude of current at the 1st, 3rd, 5th, 7th, 9th, 11th↘
→, 13th Harmonics
%phase A current harmonics
rmsharmonA = scaled(:,4)/sqrt(2);
[RMSharmonmagA,RMSharmonmagAb,RMSharmonmagAc] = harmagloc3(↘
→rmsharmonA,f_sample,vectorref,KPI(:,34),rotorbar, bearingcurfreq);
KPI(:,9:15) = RMSharmonmagA;% current harmonics
KPI(:,109:114) = RMSharmonmagAb;% Broken rotorbars
KPI(:,157:166) = RMSharmonmagAc;
%phase B current harmonics
rmsharmonB = scaled(:,5)/sqrt(2);
[RMSharmonmagB,RMSharmonmagBb,RMSharmonmagBc] = harmagloc3(↘
→rmsharmonB,f_sample,vectorref,KPI(:,34),rotorbar, bearingcurfreq);
KPI(:,18:24) = RMSharmonmagB;
KPI(:,115:120) = RMSharmonmagBb;
KPI(:,167:176) = RMSharmonmagBc;
%phase C current harmonics
rmsharmonC = scaled(:,6)/sqrt(2);
```

```
[RMSharmonmagC,RMSharmonmagCb,RMSharmonmagCc] = harmagloc3(↘
→rmsharmonC,f_sample,vectorref,KPI(:,34),rotorbar, bearingcurfreq);
KPI(:,27:33) = RMSharmonmagC;
KPI(:,121:126) = RMSharmonmagCb;
KPI(:,177:186) = RMSharmonmagCc;

%%
%IEEE total harmonic distortion Phase A
IEEETHDA = zeros (length(vectorref),1);
for ha = 1:length(vectorref)
    IEEETHDA(ha) =(sqrt((KPI(ha,1)^2)-(KPI(ha,9)^2)))/KPI(ha,9);
end
KPI(:,7) = real(IEEETHDA)*100;

%IEEE total harmonic distortion Phase B
IEEETHDB = zeros (length(vectorref),1);
for hb = 1:length(vectorref)
    IEEETHDB(hb) =(sqrt((KPI(hb,2)^2)-(KPI(hb,18)^2)))/KPI(hb,18);
end
KPI(:,16) = real(IEEETHDB)*100;

%IEEE total harmonic distortion Phase C
IEEETHDC = zeros (length(vectorref),1);
for hc = 1:length(vectorref)
    IEEETHDC(hc) =(sqrt((KPI(hc,3)^2)-(KPI(hc,27)^2)))/KPI(hc,27);
end
KPI(:,25) = real(IEEETHDC)*100;

%%
% IEC Thd Phase A
IECTHDA = zeros (length(vectorref),1);
for ga = 1:length(vectorref)
    IECTHDA(ga) =(sqrt((KPI(ga,1)^2)-(KPI(ga,9)^2)))/(sqrt(KPI(ga↘
    →,1)^2));
end
KPI(:,8) = real(IECTHDA)*100;

% IEC Thd Phase B
IECTHDB = zeros (length(vectorref),1);
for gb = 1:length(vectorref)
    IECTHDB(gb) =(sqrt((KPI(gb,2)^2)-(KPI(gb,18)^2)))/(sqrt(KPI(gb↘
    →,2)^2));
end
KPI(:,17) = real(IECTHDB)*100;

% IEC Thd Phase C
IECTHDC = zeros (length(vectorref),1);
for gc = 1:length(vectorref)
    IECTHDC(gc) =(sqrt((KPI(gc,3)^2)-(KPI(gc,27)^2)))/(sqrt(KPI(gc↘
    →,3)^2));
end
KPI(:,26) = real(IECTHDC)*100;

%velocity
```

```
%integral of acceleration
KPI(:,35) = velrms( scaled(:,8),vectorref,f_sample );
KPI(:,36) = velrms( scaled(:,9),vectorref,f_sample );
KPI(:,37) = velrms( scaled(:,10),vectorref,f_sample );
KPI(:,38) = velrms( scaled(:,11),vectorref,f_sample );
KPI(:,39) = velrms( scaled(:,12),vectorref,f_sample );
KPI(:,40) = velrms(scaled(:,13),vectorref,f_sample );

%rms band lf,mf,hf
%lf band 10 to 200 hz
%mf band 200 to 2000 hz
%hf band 2000+ hz
KPI(:,42:44) = bands( scaled(:,8),f_sample,vectorref );
KPI(:,53:55) = bands( scaled(:,9),f_sample,vectorref );
KPI(:,64:66) = bands( scaled(:,10),f_sample,vectorref );
KPI(:,75:77) = bands( scaled(:,11),f_sample,vectorref );
KPI(:,86:88) = bands( scaled(:,12),f_sample,vectorref );
KPI(:,97:99) = bands( scaled(:,13),f_sample,vectorref );
%%
%vibration 0.5,1,2,3,4 rotational freq for bearing block 1 x axis
[ vibmag1x,vibelec1x, bearingvib1x] = Vibharmmech2(scaled(:,8),↘
→f_sample,vectorref,KPI(:,41),KPI(:,34),bearingvibfreq);
KPI(:,45:49) = vibmag1x;
KPI(:,50:52) = vibelec1x;
KPI(:,127:131) = bearingvib1x;

%vibration 0.5,1,2,3,4 rotational freq for bearing block 1 y axis
[ vibmag1y,vibelec1y, bearingvib1y] = Vibharmmech2(scaled(:,9),↘
→f_sample,vectorref,KPI(:,41),KPI(:,34),bearingvibfreq);
KPI(:,56:60) = vibmag1y;
KPI(:,61:63) = vibelec1y;
KPI(:,132:136) = bearingvib1y;

%vibration 0.5,1,2,3,4 rotational freq for bearing block 1 z axis
[ vibmag1z,vibelec1z, bearingvib1z] = Vibharmmech2(scaled(:,10),↘
→f_sample,vectorref,KPI(:,41),KPI(:,34),bearingvibfreq);
KPI(:,67:71) = vibmag1z;
KPI(:,72:74) = vibelec1z;
KPI(:,137:141) = bearingvib1z;

%vibration 0.5,1,2,3,4 rotational freq for bearing block 2 x axis
[ vibmag2x,vibelec2x, bearingvib2x] = Vibharmmech2(scaled(:,11),↘
→f_sample,vectorref,KPI(:,41),KPI(:,34),bearingvibfreq);
KPI(:,78:82) = vibmag2x;
KPI(:,83:85) = vibelec2x;
KPI(:,142:146) = bearingvib2x;

%vibration 0.5,1,2,3,4 rotational freq for bearing block 2 y axis
[ vibmag2y,vibelec2y ,bearingvib2y] = Vibharmmech2(scaled(:,12),↘
→f_sample,vectorref,KPI(:,41),KPI(:,34),bearingvibfreq);
KPI(:,89:93) = vibmag2y;
KPI(:,94:96) = vibelec2y;
KPI(:,147:151) = bearingvib2y;
```

```
%vibration 0.5,1,2,3,4 rotational freq for bearing block 2 z axis
[ vibmag2z,vibelec2z ,bearingvib2z] = Vibharmmech2(scaled(:,13),↘
→f_sample,vectorref,KPI(:,41),KPI(:,34),bearingvibfreq);
KPI(:,100:104) = vibmag2z;
KPI(:,105:107) = vibelec2z;
KPI(:,152:156) = bearingvib2z;
end
```

KPI Scaling Calculator

```
function [KPIscaled] = KPItscaling(signal,vectorref)
KPIscaled = zeros(length(vectorref),1);

for p = 1:length(vectorref)
     if p == 1
         KPIscaled (p,1) = 0;
     else
         KPIscaled (p,1) = median(signal(vectorref(p-1):vectorref(p↘
         →)-1));
     end
end
end
```

Zero Crossing Caclulation

```
function [ind,t0,s0,t0close,s0close] = crossing(S,t,level,imeth)
% CROSSING find the crossings of a given level of a signal
% ind = CROSSING(S) returns an index vector ind, the signal
% S crosses zero at ind or at between ind and ind+1
% [ind,t0] = CROSSING(S,t) additionally returns a time
% vector t0 of the zero crossings of the signal S. The crossing
% times are linearly interpolated between the given times t
% [ind,t0] = CROSSING(S,t,level) returns the crossings of the
% given level instead of the zero crossings
% [ind,t0] = CROSSING(S,t,level,par) allows additional parameters
% par = {'none'|'linear'}.
% With interpolation turned off (par = 'none') this function always
% returns the value left of the zero (the data point thats nearest
% to the zero AND smaller than the zero crossing).
% [ind,t0,s0] = ... also returns the data vector corresponding to
% the t0 values.
% [ind,t0,s0,t0close,s0close] additionally returns the data points
% closest to a zero crossing in the arrays t0close and s0close.

error(nargchk(1,4,nargin));

% check the time vector input for consistency
if nargin < 2 || isempty(t)
% if no time vector is given, use the index vector as time
    t = 1:length(S);
elseif length(t) ~= length(S)
% if S and t are not of the same length, throw an error
    error('t and S must be of identical length!');
end

% check the level input
if nargin < 3
% set standard value 0, if level is not given
    level = 0;
end

% check interpolation method input
if nargin < 4
    imeth = 'linear';
end

% make row vectors
t = t(:)';
S = S(:)';

% always search for zeros. So if we want the crossing of
% any other threshold value "level", we subtract it from
% the values and search for zeros.
S   = S - level;
```

```
% first look for exact zeros
ind0 = find( S == 0 );

% then look for zero crossings between data points
S1 = S(1:end-1) .* S(2:end);
ind1 = find( S1 < 0 );

% bring exact zeros and "in-between" zeros together
ind = sort([ind0 ind1]);

% and pick the associated time values
t0 = t(ind);
s0 = S(ind);

if strcmp(imeth,'linear')
    % linear interpolation of crossing
    for ii=1:length(t0)
        if abs(S(ind(ii))) > eps(S(ind(ii)))
% interpolate only when data point is not already zero
            NUM = (t(ind(ii)+1) - t(ind(ii)));
            DEN = (S(ind(ii)+1) - S(ind(ii)));
            DELTA =  NUM / DEN;
            t0(ii) = t0(ii) - S(ind(ii)) * DELTA;
% Set the value to zero instead of calculating the perfect number
            s0(ii) = 0;
        end
    end
end
```

Current RMS Calculation

```
function [rms_signal] = KPIrms2(signal,vectorref,f_sample)
rms_signal = zeros(length(vectorref),1);

start_ind = f_sample*3 * ones(1,length(vectorref));
temp = find(vectorref < f_sample*3);
start_ind(temp) = vectorref(temp) - 1;

for xx = 1:length(vectorref)
    if xx == 1
        rms_signal(xx) = 0;
    else
        rms_signal (xx,1) = sqrt(sum(signal(vectorref(1,xx)-↘
        →start_ind(xx):vectorref(1,xx)-1).^2)/start_ind(xx));
    end
end

end
```

Velocity RMS Calculation

```
function [KPI] = velrms(acceldata,vectorref,f_sample)
Num = get(Highpass1,'Numerator');
filtaccel = filtfilt (Num,1,acceldata);
temp1 = fft(filtaccel);
temp1(1) = 0;
integ1 = 1/6000*cumtrapz(real(ifft(temp1)));
KPI = KPIrms2( integ1,vectorref,f_sample );
end
```

Velocity RMS Bands Calculation

```
function [band_rms] = bands(signal,f_sample,vectorref)
lfband_rms = zeros(length(vectorref),1);
mfband_rms = zeros(length(vectorref),1);
hfband_rms = zeros(length(vectorref),1);
band_rms = zeros(length(vectorref),3);
start_ind = f_sample*3 * ones(1,length(vectorref));
temp = find(vectorref < f_sample*3);
start_ind(temp) = vectorref(temp) - 1;
for xx = 1:length(vectorref)
    if xx == 1
        lfband_rms(xx) = 0;
        mfband_rms(xx) = 0;
        hfband_rms(xx) = 0;
    else
        y = signal(vectorref(1,xx)-start_ind(xx):vectorref(1,xx)-1)↘
        →;
        N = length (y);
        NumUniquePts = ceil((N+1)/2);
        freq = (0:NumUniquePts-2)*f_sample/(N-1);
        accelfft = fft(y);
        accelmag = abs (accelfft);
        ind_vec = find(freq>10 & freq<200);
        lfband = (accelmag(ind_vec));
        lfband_rms (xx) = sqrt(sum((abs(lfband)/(N)).^2));
        ind_vec2 = find(freq>200 & freq<2000);
        mfband = (accelmag(ind_vec2));
        mfband_rms (xx) = sqrt(sum((abs(mfband)/(N)).^2));
        ind_vec3 = find(freq>2000);
        hfband = (accelmag(ind_vec3));
        hfband_rms (xx) = sqrt(sum((abs(hfband)/(N)).^2));
    end
    band_rms(xx,1)= lfband_rms (xx);
    band_rms(xx,2)= mfband_rms (xx);
    band_rms(xx,3)= hfband_rms (xx);
end
end
```

Run Speed Calculation Function

```
function [speedmag] = run_speed(speeddata, f_sample)
signal = speeddata;
vectorref = 1:f_sample/5:length(signal);
speed = zeros (length(vectorref),1);
run = zeros (length(vectorref),1);
speedmag = zeros (length(vectorref),1);
start_ind = f_sample*3 * ones(1,length(vectorref));
temp = find(vectorref < f_sample*3);
start_ind(temp) = vectorref(temp) - 1;
for xx = 1:length(vectorref)
   if xx == 1
       speed (xx) = 0;
   else
       run (xx) = mean(signal(vectorref(1,xx)-start_ind(xx):↘
       →vectorref(xx)-1));
       speed (xx) = run (xx)/(15*(0.0097*10^-6)*99200*.9668);
   end
   speedmag (xx) = speed (xx);%*60; % speed magnitude in RPM
   %speed needs to be in Hz for vibration calculations
end

end
```

Slip Ratio Calculation

```
function [slipr] = slipratio(elecfreq,mechfreq,vectorref)
slipr = zeros(length(vectorref),1);
for yy = 1:length(vectorref)
    if yy == 1
        slipr(yy,:) = 0;
    else
        slipr(yy) = (elecfreq(yy) - mechfreq(yy))/elecfreq(yy);
    end
end

end
```

Vibration Mechanical Harmonics Calculation

```
function [vibharmonmag,vibharelec,vibharbearing] = Vibharmmech2(↘
→signal,f_sample,vectorref,KPI,elecfq,bearingvibfq)

N=2^16;
f=[0:1:N/2-1]*(f_sample/N);
max_Y = zeros (1,5);
max_YY = zeros (1,5);
max_YX = zeros (1,3);
vibharmonmag = zeros(length(vectorref),5);
vibharbearing = zeros(length(vectorref),5);
vibharelec = zeros(length(vectorref),3);
h = [1,0.5,2,3,4];
hh = [1,2,6];
start_ind = f_sample*3 * ones(1,length(vectorref));
temp = find(vectorref < f_sample*3);
start_ind(temp) = vectorref(temp) - 1;
for s = 1:length(vectorref)
    if s == 1
        for r=1:1:5
            max_Y(r) = 0;
            max_YY(r) = 0;
        end
        for rr=1:1:3
            max_YX(rr) = 0;
        end
    else if KPI(s) == 0
            for r=1:1:5
                max_Y(r) = 0;
                max_YY(r) = 0;
            end
            for rr=1:1:3
                max_Y(rr) = 0;
            end
        else
        y = (signal(vectorref(1,s)-start_ind(s):vectorref(1,s)-1));
        har=4/length(y)*fft(y.*hann(length(y)),N);
        harmag = abs(har);

        for r=1:1:5
            ind_vec = find(f>h(r)*KPI(s)-0.5 & f<h(r)*KPI(s)+0.5);
            [~, ind] = max(harmag(ind_vec));
            ind = ind_vec(ind);
            max_Y(r) = harmag(ind);
        end
         for rr=1:1:5
            ind_vec = find(f>bearingvibfq(s,rr)-1 & f<bearingvibfq(↘
            →s,rr)+1);
            [~, ind] = max(harmag(ind_vec));
            ind = ind_vec(ind);
            max_YY(rr) = harmag(ind);
        end
        for rf=1:1:3
            ind_vec = find(f>hh(rf)*elecfq(s)-0.5 & f<hh(rf)*elecfq↘
            →(s)+0.5);
            [~, ind] = max(harmag(ind_vec));
            ind = ind_vec(ind);
            max_YX(rf) = harmag(ind);
        end
        end
    end

    vibharmonmag(s,:) = max_Y;
    vibharbearing(s,:) = max_YY;
    vibharelec(s,:) = max_YX;

end

end
```

Rotor Bar Sidebands Calculation

```
function [rotorfreq] = rotorbarsidebands(slip,vectorref)
rotorfreq = zeros(length(vectorref),6);
for xx = 1:length(vectorref)
        rotorfreq(xx,1) = (1 - 2*3*slip(xx));
        rotorfreq(xx,2) = (1 - 2*2*slip(xx));
        rotorfreq(xx,3) = (1 - 2*1*slip(xx));
        rotorfreq(xx,4) = (1 + 2*1*slip(xx));
        rotorfreq(xx,5) = (1 + 2*2*slip(xx));
        rotorfreq(xx,6) = (1 + 2*3*slip(xx));
end

for yy = 1:length(vectorref)
    for rr=1:1:6
        if rotorfreq(yy,rr)<0
            rotorfreq(yy,rr) = abs(rotorfreq(yy,rr)) ;
        end
    end
end

end
```

Bearing Current Calculation

```
function [fbng] = bearingcurrent(elecfrq,bearingfreq,vectorref)
fbng = zeros(length(vectorref),10);
fbnga = zeros(length(vectorref),5);
fbngb = zeros(length(vectorref),5);
for xx = 1:length(vectorref)
    for yy = 1:1:5
        fbnga(xx,yy) = abs(elecfrq(xx) - 1*bearingfreq(xx,yy));
        fbngb(xx,yy) = abs(elecfrq(xx) + 1*bearingfreq(xx,yy));
    end
    fbng(xx,1:5) = fbnga(xx,:);
    fbng(xx,6:10) = fbngb(xx,:);
end
end
```

Bearing Frequency Calculation

```
function [bearingfreq] = bearingfreqs(Mechspeed,vectorref)
bearingfreq = zeros(length(vectorref),5);
for xx = 1:length(vectorref)
        bearingfreq(xx,1) = (0.0332*60*Mechspeed(xx));
        %Ball spin bearing fault
        bearingfreq(xx,2) = (0.0063*60*Mechspeed(xx));
        %Fundamental Train fault
        bearingfreq(xx,3) = (0.0825*60*Mechspeed(xx));
        %Inner ring defect fault
        bearingfreq(xx,4) = (0.0508*60*Mechspeed(xx));
        %Outer ring defect fault
        bearingfreq(xx,5) = (0.0663*60*Mechspeed(xx));
        %Ball defect fault
end
 for yy = 1:length(vectorref)
    for rr=1:1:5
        if bearingfreq(yy,rr)<0
            bearingfreq(yy,rr) = abs(bearingfreq(yy,rr)) ;
        end
    end
 end
end
```

Harmonic Magnitude Locator

```
function  [harmonmag3,harmonmag3b,harmonmag3c] = harmagloc3(signal,↘
→f_sample,vectorref,KPI,rotorbar,bearing)
%Harmonic magnitude locator
max_Y = zeros (1,7);
max_YY = zeros (1,6);
max_YX = zeros (1,10);
harmonmag3 = zeros(length(vectorref),7);
harmonmag3b = zeros(length(vectorref),6);%broken rotor bars
harmonmag3c = zeros(length(vectorref),10);%bearing faults
start_ind = f_sample*3 * ones(1,length(vectorref));
temp = find(vectorref < f_sample*3);
start_ind(temp) = vectorref(temp) - 1;

for s = 1:length(vectorref)
    if s == 1
        for r=1:2:13
            max_Y((r+1)/2) = 0;
        end
        for rr=1:1:6
            max_YY(rr) = 0;
        end
        for ss=1:1:10
            max_YX(ss) = 0;
        end

    else if KPI(s) <= 1
            for r=1:2:13
                max_Y((r+1)/2) = 0;
            end
            for rr=1:1:6
                max_YY(rr) = 0;
            end
            for ss=1:1:10
                max_YX(ss) = 0;
            end
 else
        y = (signal(vectorref(1,s)-start_ind(s):vectorref(1,s)-1));
        N=2^16;
        f=[0:1:N/2-1]*(f_sample/N);
        har=4/length(y)*fft(y.*hann(length(y)),N);
        harmag = abs(har);

        for r=1:2:13
            ind_vec = find(f>r*KPI(s)-1 & f<r*KPI(s)+1);
            [~, ind] = max(harmag(ind_vec));
            ind = ind_vec(ind);
            max_Y((r+1)/2) = harmag(ind);
        end
        for rr=1:1:6
            ind_vec2 = find(f>rotorbar(s,rr)*KPI(s,1)-2 & f<↘
            →rotorbar(s,rr)*KPI(s,1)+2);
            [~, ind2] = max(harmag(ind_vec2));
            ind2 = ind_vec2(ind2);
            max_YY(rr) = harmag(ind2);
        end
        for ss=1:1:10
            ind_vec3 = find(f>bearing(s,ss)-2 & f<bearing(s,ss)+2);
            [~, ind3] = max(harmag(ind_vec3));
            ind3 = ind_vec3(ind3);
            max_YX(ss) = harmag(ind3);
        end
        end
    end

    harmonmag3(s,:) = max_Y;
    harmonmag3b(s,:) = max_YY;
    harmonmag3c(s,:) = max_YX;
end
end
```

FIR Filter

```
function Hd = FIRfilter3
%FIRFILTER3 Returns a discrete-time filter object.
% Equiripple Lowpass filter designed using the FIRPM function.

% All frequency values are in Hz.
Fs = 6000;  % Sampling Frequency

Fpass = 70;              % Passband Frequency
Fstop = 400;             % Stopband Frequency
Dpass = 0.057501127785;  % Passband Ripple
Dstop = 0.01;            % Stopband Attenuation
dens  = 20;              % Density Factor

% Calculate the order from the parameters using FIRPMORD.
[N, Fo, Ao, W] = firpmord([Fpass, Fstop]/(Fs/2), [1 0], [Dpass, ↘
→Dstop]);

% Calculate the coefficients using the FIRPM function.
b  = firpm(N, Fo, Ao, W, {dens});
Hd = dfilt.dffir(b);
```

High Pass Filter

```
function Hd = Highpass1
%HIGHPASS1 Returns a discrete-time filter object.
% Equiripple Highpass filter designed using the FIRPM function.
% All frequency values are in Hz.
Fs = 6000;  % Sampling Frequency

Fstop = 3;               % Stopband Frequency
Fpass = 55;              % Passband Frequency
Dstop = 1e-007;          % Stopband Attenuation
Dpass = 0.057501127785;  % Passband Ripple
dens  = 20;              % Density Factor

% Calculate the order from the parameters using FIRPMORD.
[N, Fo, Ao, W] = firpmord([Fstop, Fpass]/(Fs/2), [0 1], [Dstop, ↘
→Dpass]);

% Calculate the coefficients using the FIRPM function.
b  = firpm(N, Fo, Ao, W, {dens});
Hd = dfilt.dffir(b);
```

APPENDIX B: MOTOR DRIVE DESIGN

This appendix will cover the design of the custom motor drive that is used for the testing in Chapter 3.

B.1 Component Selection

Transistor Selection

The IGBT must be sized so that its voltage rating is higher than the dc bus voltage. A rule of thumb is to choose the IGBT to be rated at twice the dc bus voltage or higher. As for current the IGBT should be rated to carry at least twice the rated current but in reality it is normally sized for much more than this.

Gate Drive Selection

Figure B.1 shows IR2112 gate driver connected to a totem pole set of IGBTs. The gate driver was chosen based on the IGBT collector to emitter voltage rating.

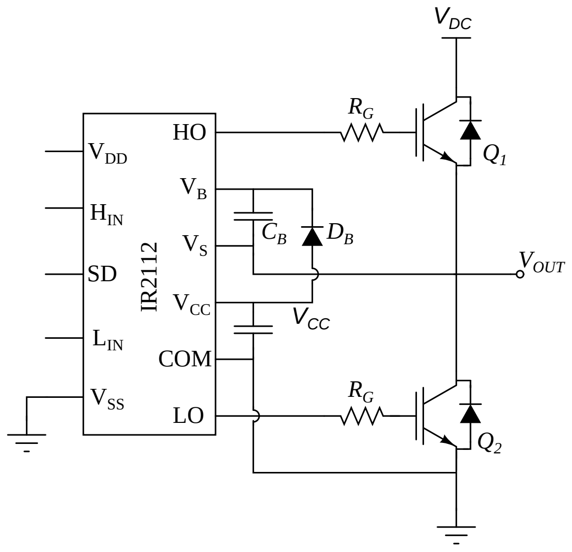


Figure B.1: Detailed schematic of a single phase of the custom motor drive.

Bootstrap Capacitor Sizing

The first step to sizing the bootstrap capacitor is to establish the minimum voltage drop (ΔV_{BS}) to guarantee that the high side IGBT is on. The minimum voltage drop

Table B.1: Motor Fault Detection KPIs

Quantity	Value	Notes
V_{CC}	15 V	Voltage supply of gate driver
V_F	1.4 V	Bootstrap Diode Datasheet (BYG20J)
$V_{CE,ON,max}$	3.7 V	IGBT Datasheet (IRGP20B60PDPbF)
$V_{GE,min}$	5 V	IGBT Datasheet (IRGP20B60PDPbF)

is [94]

$$\Delta V_{BS} \leq V_{CC} - V_F - V_{GE,min} - V_{CE,on}, \tag{B.1}$$

as long as

$$V_{GE,min} > V_{BS,UV-} \tag{B.2}$$

where V_{CC} is the gate driver IC supply voltage, V_F is the bootstrap diode forward voltage drop, $V_{GE,min}$ is the minimum gate to emitter voltage to maintain conduction, $V_{CE,on}$ is the collector to emitter voltage of the low side IGBT, and $V_{BS,UV-}$ is the high-side supply undervoltage negative going threshold. For the custom designed motor drive the values used are shown in Table B.1. The value of ΔV_{BS} for the custom motor drive is 4.9 V. The following are the influencing factors contributing to a decrease in V_{BS} [94]

- IGBT turn on required gate charge (Q_G)
- Charge required by the internal level shifters (Q_{LS})
- IGBT gate-emitter leakage current ($I_{LK,GE}$)
- Floating section quiescent current ($I_{Q,BS}$)
- Floating section leakage current (I_{LK})
- Bootstrap diode leakage current ($I_{LK,DIODE}$)
- Desaturation diode bias current when on (I_{DS-})
- Bootstrap capacitor leakage current ($I_{LK,CAP}$)
- High side on time ($T_{H,ON}$)

For the custom designed motor drive the values used are shown in Table B.2. The

Table B.2: Motor Fault Detection KPIs

Quantity	Value	Notes
Q_G	68 nC	IGBT Datasheet (IRGP20B60PDPbF)
Q_{LS}	20 nC	IGBT Datasheet (IRGP20B60PDPbF)
$I_{LK,GE}$	5 μA	IGBT Datasheet (IRGP20B60PDPbF)
$I_{Q,BS}$	25 μA	Gate Driver Datasheet (IR2112)
I_{LK}	50 μA	Gate Driver Datasheet (IR2112)
I_{DS-}	150 μA	Gate Driver Datasheet (IR2112)
$I_{LK,DIODE}$	1 μA	Bootstrap Diode Datasheet (BYG20J), with $t_{rr} < 100$ ns
$I_{LK,CAP}$	0	using a non electrolytic capacitor
$T_{H,ON}$	1 ms	

total gate charge Q_{TOT} is calculated as [94]

$$Q_{TOT} = Q_G + Q_{LS} + (I_{LK,GE} + I_{Q,BS} + I_{LK} + I_{LK,DIODE} + I_{LK,CAP} + I_{DS-})T_{H,ON}. \quad \text{(B.3)}$$

The total gate charge for the custom motor drive is 319 nC. Therefore, the minimum size of the bootstrap capacitor C_B is

$$C_B > \frac{Q_{TOT}}{\Delta V_{BS}}. \quad \text{(B.4)}$$

For the custom motor drive the booststrap capacitor must be greater than 65 nF.

Bootstrap Diode Sizing

The bootstrap diode must have a blocking voltage greater than the dc bus voltage and a fast recovery time (t_{rr} ¡ 100 ns) to minimize the amount of charge fed back from the bootstrap capacitor to V_{CC} supply.

Gate Resistor Sizing

Gate resistance may be chosen either to fix the switching-time or the output voltage slope. For the matters of the calculation included hereafter, the switching time t_{SW} is defined as the time spent to reach the end of the plateau voltage (a total Q_{gc} + Q_{ge} has been provided to the IGBT gate). To obtain the desired switching time the gate resistance can be sized starting from Q_{gc} + Q_{ge}, V_{CC}, and the desired V_{ge} [94].

The average current, I_{avg} is found to be [94]

$$I_{avg} = \frac{(Q_{gc} + Q_{ge})}{t_{SW}}, \tag{B.5}$$

and the turn-on gate resistor, $R_{G,on}$, is

$$R_{G,on} = \frac{V_{CC} + V_{ge}}{I_a vg - R_D R_p} then \tag{B.6}$$

where $R_D R_p$ is the drivers equivalent on-resistance.

Figure B.2 shows a complete schematic of a single phase of the three phase inverter that is used in the custom motor drive. Note that ther should be no earth ground reference on the high voltage side.

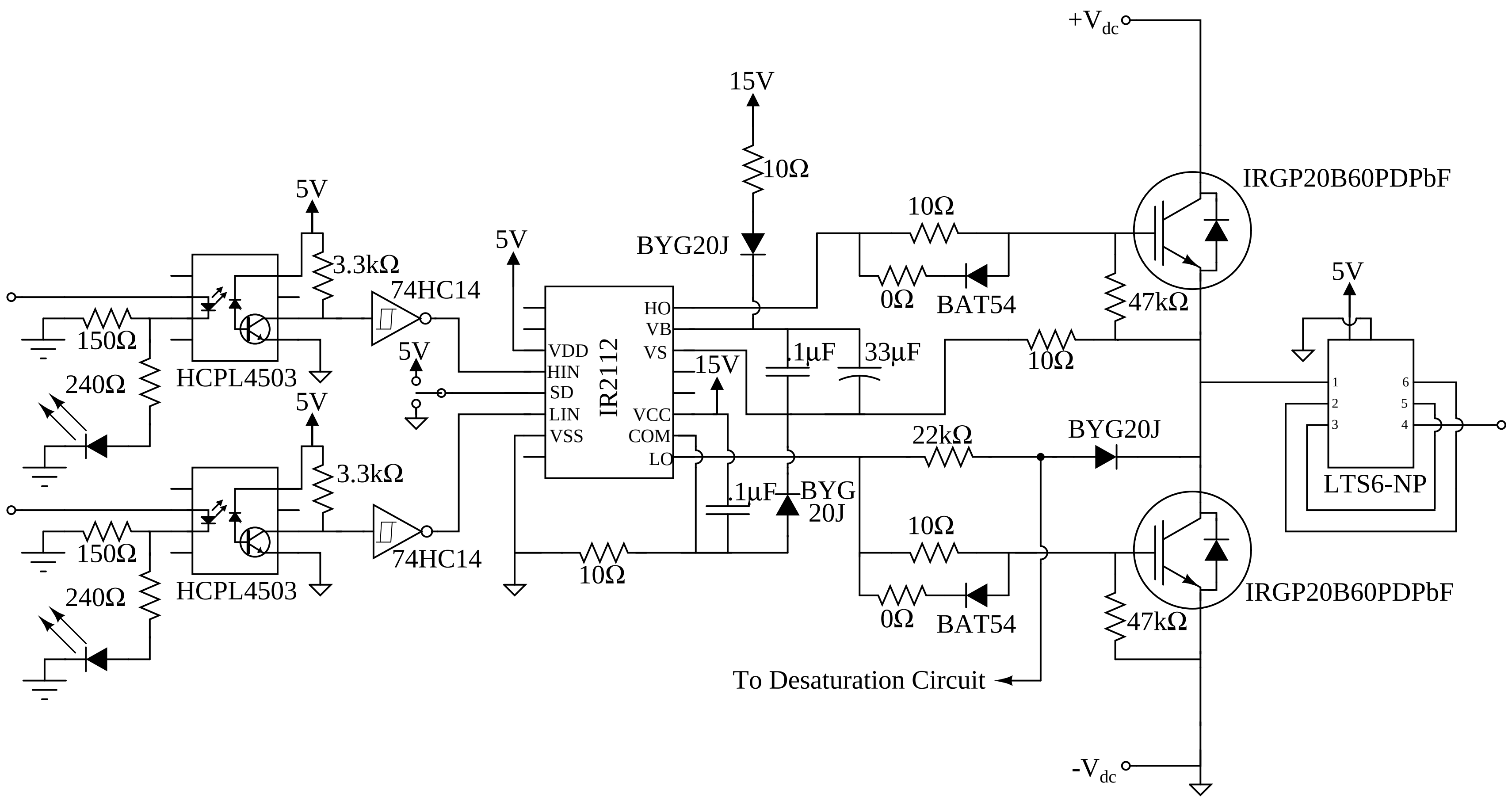


Figure B.2: Detailed schematic of a single phase of the custom desinged inverter.

B.2 Motor Drive PCB Layouts

This section of the appendix shows the layout of the PCB for the inverter and rectifier portions of the custom motor drive.

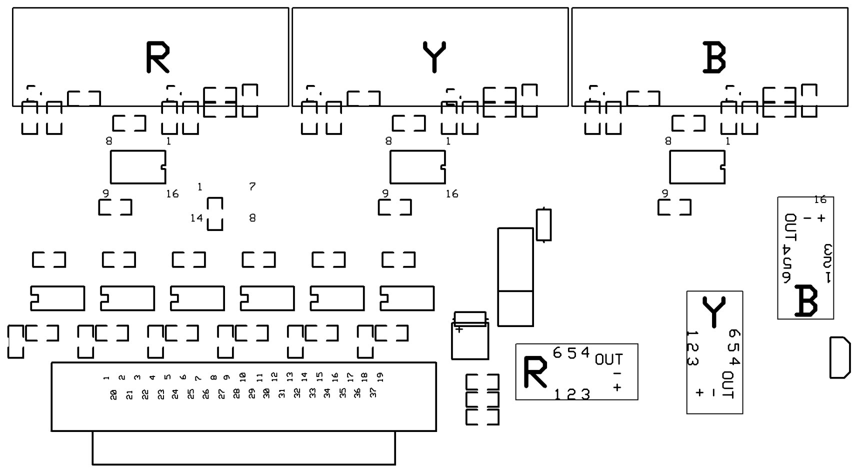


Figure B.3: Silk screen layer for inverter.

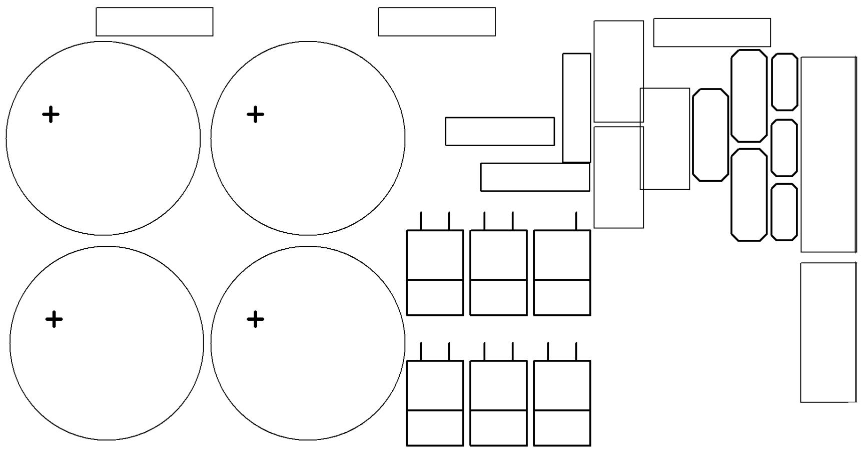


Figure B.4: Silk screen layer for rectifier.

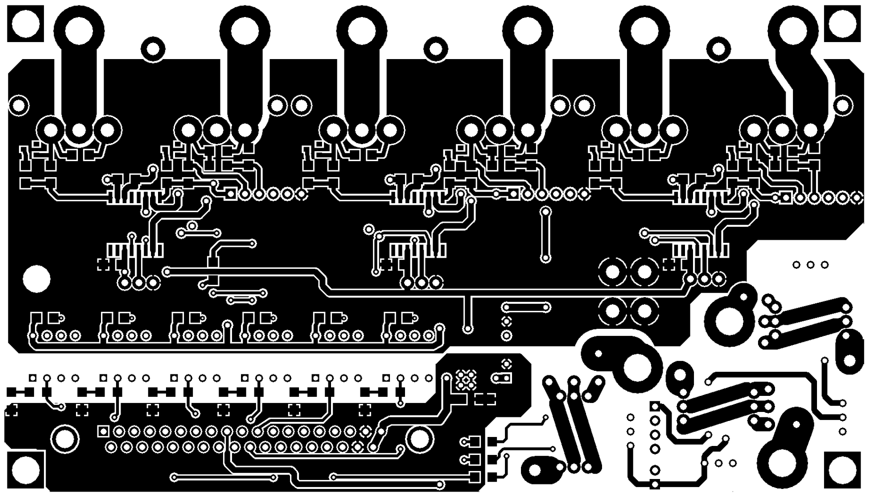


Figure B.5: Top copper layer for inverter.

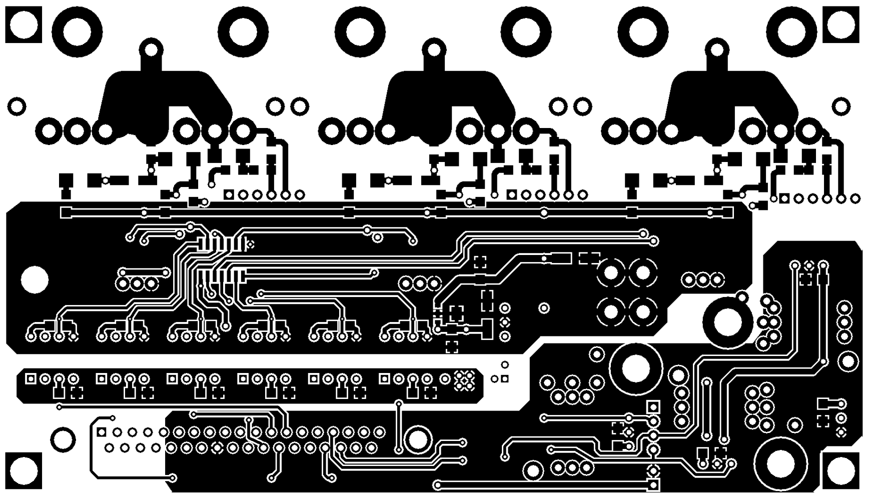


Figure B.6: Bottom copper layer for inverter.

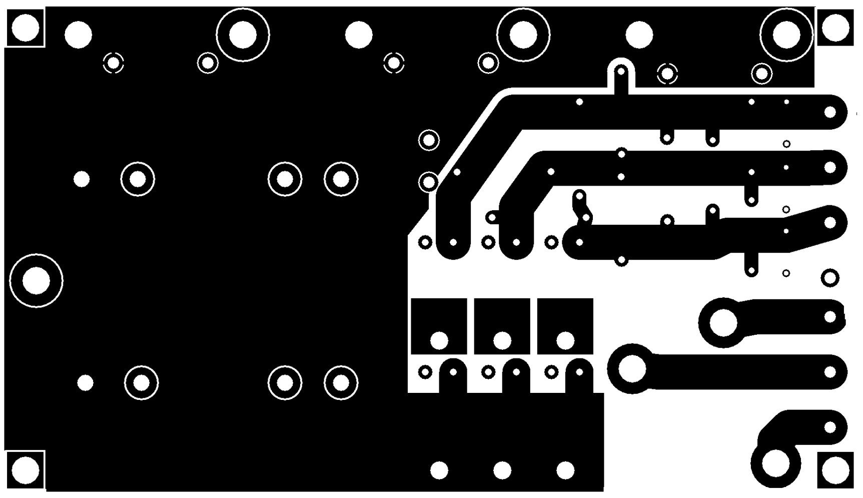


Figure B.7: Top copper layer for rectifier.

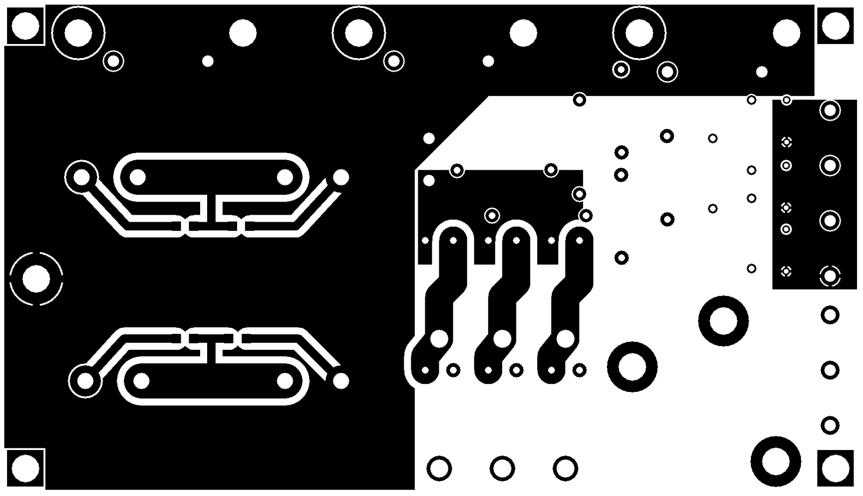


Figure B.8: Bottom copper layer for rectifier.

B.3 Motor Drive PWM Code

```
;*********************************************************
;
; Filename: acim_vhz.s
;
;*********************************************************
;    Notes:
;    ======
; The A/D is enabled to sample VR1pots on the dsPICDEM-MC1
; demo board.  VR1 is used to vary the V/Hz ratio of the
; modulation.
;*********************************************************
;---------------------------------------------------------

.equ __30F6010A, 1
.include "p30f6010a.inc"

.global __reset

;---------------------------------------------------------
;Configuration bits:
;---------------------------------------------------------

;Turn off clock switching and
;fail-safe clock monitoring and
;use the XT osc and 4x PLL as
;system clock
config __FOSC, CSW_FSCM_OFF & XT_PLL4

;Turn off Watchdog Timer
config __FWDT, WDT_OFF

;Set Brown-out Reset voltage and
;and set Power-up Timer to 16msecs
config __FBORPOR, PBOR_ON & BORV27 & PWRT_16 & MCLR_EN

;Set Code Protection Off for the
;General Segment
config __FGS, CODE_PROT_OFF
```

```
;----------------------------------------------------------
;Uninitialized variables in Near data memory
;(Lower 8Kb of RAM)
;----------------------------------------------------------

          .section .nbss, "b"

;This variable is added to the 16-bit sine wave table
;pointer at each PWM period.  A value of 246 will provide
;60 Hz modulation frequency with 16 KHz PWM.
Frequency: .space 2

;This variable is used to set the modulation amplitude
;and scales the value retrieved from the sine wave table.
;Valid values range from 0 to 32767.
Amplitude: .space 2

;This variable is the pointer to the sinewave table.
;It is incremented by the value of the Frequency variable
;at each PWM interrupt.
Phase: .space 2

;----------------------------------------------------------
;Constants stored in Program space
;----------------------------------------------------------

        .section .sine_table, "x"
        .align 256
;This is a 64 entry sinewave table covering 360 degrees
;of the sine function. These values were calculated
;using Microsoft Excel and pasted into this program.

SineTable:
.hword 0,3212,6393,9512,12539,15446,18204,20787,23170,25329
.hword 27245,28898,30273,31356,32137,32609,32767,32609
.hword 32137,31356,30273,28898, 27245,25329,23170,20787
.hword 18204,15446,12539,9512,6393,3212,0,-3212,-6393,-9512
.hword -12539,-15446,-18204,-20787,-23170,-25329,-27245,
.hword -28898,-30273, -31356,-32137,-32609,-32767,-32609
.hword -32137,-31356,-30273,-28898,-27245,-25329,-23170,
.hword -20787,-18204,-15446,-12539,-9512,-6393,-3212

;----------------------------------------------------------
; Constants for this application
;----------------------------------------------------------
```

```
;This constant is used to scale the sine lookup value to the
;the valid range of PWM duty cycles.  This is based on the
;value written to PTPER.  We will PTPER = 230 for this
;application, which allows duty cycles between 0 and 460.
;The sine table data is signed, so we will multiply the table
;data by 230, then add a constant offset to scale the lookup
;data to positive values.

.equ PWM_Scaling, 3686 ;3680 Points for 50% duty ratio of 1kHz PWM

;The pointer to the sign wave table is 16 bits.  Adding
;0x5555 to the pointer will provide a 120 degree offset
;and 0xAAAA will give a 240 degree offset.  These offsets
;are used to get the lookup values for phase 2 and phase
;3 of the PWM outputs.

.equ Offset_120, 0x5555

;----------------------------------------------------------
;Code Section in Program Memory
;----------------------------------------------------------

.text                              ;Start of Code section
__reset:

MOV  #__SP_init, W15     ;Initalize the Stack Pointer
MOV  #__SPLIM_init, W0   ;Initialize the Stack Pointer Limit Register
MOV  W0, SPLIM
        NOP                        ;Add NOP to follow SPLIM initialization

        CALL  _wreg_init       ;Call _wreg_init subroutine
;Optionally use RCALL instead of CALL


call Setup ; Call the routine to setup I/O and PWM
;----------------------------------------------------------
; Variable initialization
;----------------------------------------------------------
clr Frequency
clr Amplitude

;----------------------------------------------------------
```

```
; Main loop code
; The PWM interrupt flag is polled in the main loop
;----------------------------------------------------------
Loop:

btss IFS2,#PWMIF ;poll the PWM interrupt flag
bra CheckADC ;if it is set, continue

call Modulation ;call the sinewave modulation routine
bclr IFS2, #PWMIF ;Clear the PWM interrupt flag

CheckADC:
btss IFS0,#ADIF
bra Loop

call ReadADC

bra Loop

;----------------------------------------------------------
; ADC processing subroutine
;----------------------------------------------------------

ReadADC:

push.d W0
push.d W4

mov #983,W0 ;983*4 = 3932, 3932 for 60 Hz
;modulation frequency at 1Khz PWM frequency.
mov ADCBUF1,W1 ;and W1.

sl W0,#2,W4 ;Trying to fix this code to allow
mov W4,Frequency ;60Hz fixed frequency modulation frequency.

sl W1,#5,W4 ;Left shift AN7 and AN12 values to get
sl W0,#5,W5 ;1.15 fractional data.
mpy W4*W5,A ;multiply frequency by V/Hz gain to get
sac A,W0 ;mod. amplitude.  Store result in W0
mov #32000,W1
cp W1,W0 ;dead-time induced distortion in PWM
bra GE,NoLimit ;modulation.
mov W1,W0

NoLimit:
```

```
mov W0,Amplitude

pop.d W4
pop.d W0

return

;---------------------------------------------------------
; PWM sine wave modulation subroutine
;---------------------------------------------------------
Modulation:
push.d W0 ;Save off working registers
push.d W2
push.d W4
push.d W6
push.d W8
push.d W10

;The next three instructions initialize the TBLPAG
;and pointer register for access to the sinewave
;data in program memory using table reads.

mov #tblpage(SineTable),W0
mov W0,TBLPAG
mov #tbloffset(SineTable),W0

; The next block of instructions loads various constants
; and variables used in the sinewave modulation routine.

mov Phase,W1 ;Load the sinewave table pointer
mov #Offset_120,W4 ;This is the value for a 120 degree offset
mov Amplitude,W6 ;Load the Amplitude scaling factor
mov #PWM_Scaling,W7 ;Load the PWM scaling value
mov Frequency,W8 ;Load the Frequency constant that will
;be added to the table pointer at each
;interrupt.

;This is the pointer adjustment code.  The Frequency
;value is added to the sine pointer to move through
;the sine table.  Then, offsetsare added to this pointer
;to get the phase 2 and phase 2 pointers.Note:  If
;different phase offsets are desired, other constant
;values can be used here.   Add 0x4000 to get a 90
;degree offset, 0x8000 will provide a 180 degree offset.
;Here, 0x5555 has been loaded to W4 to provide 120 degrees.
```

```
add W8,W1,W1 ;Add the Frequency value to the sine pointer
add W1,W4,W2 ;Add 120 degree offset value for phase 2
add W2,W4,W3 ;Add another 120 degree offset for phase 3

; The sine table has 64 entries, so the pointers are right shifted
; to get a 6-bit pointer value.

lsr W1,#10,W9 ;Shift the phase 1 pointer right to get the upper 6 bits
sl W9,#1,W9 ;Left shift by one to convert to byte address
lsr W2,#10,W10 ;Shift the phase 2 pointer right to get the upper 6 bits
sl W10,#1,W10 ;Left shift by one to convert to byte address
lsr W3,#10,W11 ;Shift the phase 3 pointer right to get the upper 6 bits
sl W11,#1,W11 ;Left shift by one to convert to byte address

;Now, the pointer for each phase is added to the base table pointer
;to get the absolute table address for the lookup value.  The lookup
;value is then scaled for the correct amplitude and for the range
;of valid duty cycles.  The next block of instructions calculates
;the duty cycle for phase 1.  The phase 2 and phase 3 code is the same.

add W0,W9,W9 ;Form the table address for phase 1
tblrdl [W9],W5 ;Read the lookup value for phase 1
mpy W5*W6,A ;Multiply by the amplitude scaling
sac A,W5 ;Store the scaled result
mpy W5*W7,A ;Multiply by the PWM scaling factor
sac A,W8 ;Store the scaled result
add W7,W8,W8 ;Add the PWM scaling factor to produce 50% offset
mov W8,PDC1 ;Write the PWM duty cycle

;The next block of code calculates the duty cycle for phase 2.

add W0,W10,W10 ;Form the table address for phase 2
tblrdl [W10],W5 ;Read the lookup value for phase 2
mpy W5*W6,A ;Multiply by the amplitude scaling
sac A,W5 ;Store the scaled result
mpy W5*W7,A ;Multiply by the PWM scaling factor
sac A,W8 ;Store the scaled result
add W7,W8,W8 ;Add the PWM scaling factor to produce 50% offset
mov W8,PDC2 ;Write the PWM duty cycle

;The next block of code calculates the duty cycle for phase 3.

add W0,W11,W11 ;Form the table address for phase 3
tblrdl [W11],W5 ;Read the lookup value for phase 3
```

```
mpy W5*W6,A ;Multiply by the amplitude scaling
sac A,W5 ;Store the scaled result
mpy W5*W7,A ;Multiply by the PWM scaling factor
sac A,W8 ;Store the scaled result
add W7,W8,W8 ;Add the PWM scaling factor to produce 50% offset
mov W8,PDC3 ;Write the PWM duty cycle

;Now, save off the adjusted sinewave table pointer so it
;can be used during the next iteration of this code.

mov W1,Phase

; restore working registers
pop.d W10
pop.d W8
pop.d W6
pop.d W4
pop.d W2
pop.d W0

; return from the subroutine
return

;---------------------------------------------------------
; PWM and ADC setup code
;---------------------------------------------------------
Setup:

;The first thing we need to do before enabling the PWM is to
;configure the I/O.  The control board has a driver IC that
;buffers the PWM control lines. The active low output enable
;for this buffer is on port RD11.

clr PORTD
clr PORTE
mov #0xF7FF,W0 ;Make RD11 an output to drive PWM buffer
mov W0,TRISD ;output enable.
mov #0xFDFF,W0
mov W0,TRISE ;Make RE9 an output for power module reset

;Now, ensure the power module is reset by
;driving the reset line for a few usec.

bset PORTE,#9
repeat #39
```

```
nop
bclr PORTE,#9

;Setup the ADC

mov #0x0404,W0 ;scan inputs
mov W0,ADCON2 ;2 sample/converts per interrupt
mov #0x0003,W0
mov W0,ADCON3 ;Tad is 2*Tcy
clr ADCHS
clr ADPCFG ;all A/D pins Analog mode
clr ADCSSL
bset ADCSSL,#7 ;enable scan of AN7
bset ADCSSL,#12 ;enable scan of AN12
mov #0x8066,W0 ;enable A/D, PWM trigger, auto sample
mov W0,ADCON1
bclr IFS0,#ADIF ;clear A/D interrupt flag

;Now, setup the PWM registers

mov #0x0077,W0 ;complementary mode, #1, #2, and #3
mov W0,PWMCON1 ;pairs are enabled
mov #0x000F,W0 ;2usec deadtime at 7.38 MIPS
mov W0,DTCON1
mov #PWM_Scaling, W0;set period for 16KHz PWM at 7.38 MIPS
mov W0,PTPER
mov #0x0001,W0
mov W0,SEVTCMP ;setup the special event trigger for the ADC
mov #0x0F00,W0 ;set the special event postscaler to 1:16
mov W0,PWMCON2
mov #0x8002,W0 ;PWM timebase enabled, center aligned mode
mov W0,PTCON

return ;return from the Setup routine

;----------------------------------------------------------
;Subroutine: Initialization of W registers to 0x0000
;----------------------------------------------------------
_wreg_init:
CLR W0
    MOV W0,W14
    REPEAT #12
    MOV W0,[++W14]
    CLR W14
    RETURN
```

```
;--------End of All Code Sections --------------------

.end ;End of program code in this file
```

APPENDIX C: SPECTRAQUEST MFS SPECIFICATIONS

Table C.1: Mechanical Specifications

Shaft Diameter	5/8" and 1"; Turned, Ground, & Polished (TGP) steel
Bearing	Two sealed rolling element in aluminum horizontally split bracket housing for easy changes, tapped for transducer mount. Bearing mounts can be mounted in five different position for variable rotor span.
Rotor Base	18" long, completely movable using jack bolts for easy horizontal misalignment and standard shims for vertical misalignment. Pinned for easy realignment.
Rotors	Two 6" aluminum with 36 threaded holes at 10 degree intervals for introducing unbalance.
Belt Mechanism	Two double groove "V" belt with one set screw mounting and one bush/key mounting. Positive displacement lever with turnbuckle plus adjustable gearbox platform.
Gearbox	Accessible three-way straight cut bevel gearbox with 1.5:1 ratio (20 gear input).
Brake	Manually adjustable magnetic brake 0.5 - 10 lb-in.
Reciprocating Mechanism	Adjustable spring engagement timing and two stroke settings.
Centrifugal Pump	1/2 hp, 27 PSI at 0 GPM, 25 GPM at 0 PSI with water at 4000 RPM.
Reciprocating Compressor	1/2 hp, 2.6 CFM, 120 PSI belt driven with 5 gallon air tank.
Instrumentation Connectors	16 BNC connector plate under the rotor base linked to BNC connector panel mounted on the edge for the base plate for direct connection to data collectors.
Safety Cover	Lockable clear, impact resistant hinged plastic cover with motor interlock switch to shut down motor when cover is raised.
Foundation	1/2" die cast aluminum base, base stiffener and eight rubber isolators.

Table C.2: Electrial Specifications

Motor	3 Phase, .5 hp motor, pre-wired self-aligning mounting system for easy installation/removal.
Drive	1/2 hp variable frequency AC drive with multi-featured front panel programmable controller.
RPM range	0 to 6000 RPM (short duration) variable speed.
Current Measurement	Power leads accessible for current measurements.
Tachometer	Built-in tachometer with LCD display and one pulse per revolution analog TTL output for DAQ purposes.
Voltage	115/230 VAC, Single phase, 60/50 Hz

Table C.3: Physical Specifications

Weight	Approximately 130 lb
Dimensions	L = 39”(100cm), W = 25”(63cm) , H = 21”(53cm)

APPENDIX D: MOTOR KEY PERFORMANCE INDICATORS

This appendix will look at how some of the KPIs that are used for motor fault detection are calculated and also provides a listing of all 186 KPIs that were calculated.

D.1 Motor KPI Calculations

The KPI for velocity was calculated by taking the FFT of the acceleration. The dc value of the FFT is then set to zero. This is done because the low frequency or dc content of the signal cause an effect which throws out the conversion process and the integration cannot account for this dc content. This error builds up as the signal is integrated and gives this growing or decreasing error effect. The IFFT is then performed on the signal after the removal of the DC component. Finally, the velocity, ν, is found by doing the cumulative trapezoidal numerical integration of the acceleration, a, and is defined as

$$\nu = \frac{1}{F_S}\left(\sum_{i=1}^{N} a(i) - \frac{a(1)+a(N)}{2}\right) \tag{D.1}$$

where F_S is the sampling frequency.

KPIs of the rms bands were also calculated. Since the KPI data is output at 5 Hz, the FFT was performed over a 0.2 s window of the accelerometer data. The three bands are then pulled out and assigned as variables. The frequency bands are defined as:

- Low frequency band from 10 to 200 Hz
- Middle frequency band from 200 to 2000 Hz
- High frequency band greater than 2000 Hz

With the three bands selected from the FFT acceleration data, a rms calculation is performed in the frequency domain to determine each of the three bands. This can be seen from Parseval's relation [95]

$$\sum_{n=0}^{N-1} |x[n]|^2 = \frac{1}{N} \sum_{f=0}^{N-1} |X[f]|^2. \tag{D.2}$$

The rms was derived from this to be [95]

$$sqrt\frac{1}{N}\sum_{n} x^2(t) = sqrt\frac{1}{N^2}\sum_{n} |X(f)|^2 = sqrt\sum_{n}\left|\frac{X(f)}{N}\right|^2. \tag{D.3}$$

Broken rotor bars can be detected by performing a spectrum analysis of the motor current. The harmonic frequencies relating to broken rotor bars, f_{br}, can be predicted at by [83]

$$f_{br} = (1 \pm 2ks)\, f, \tag{D.4}$$

where s is the slip ratio, f is the fundamental frequency of the electrical current, and $k = 1, 2, 3 \ldots$

There are four categories of defects in a ball bearing such as seen in Fig. D.1 that specific KPIs were calculated to detect: outer raceway defect, inner raceway defect, ball defect, train defect. The faults can be detected at vibration frequencies in Hertz (Hz), f_v, specified by:

- Outer Raceway [96]

$$f_v = \left(\frac{N}{2}\right) f_r \left(1 - 2\frac{D_b}{D_c}\cos\beta\right) \tag{D.5}$$

- Inner Raceway [96]

$$f_v = \left(\frac{N}{2}\right) f_r \left(1 + 2\frac{D_b}{D_c}\cos\beta\right) \tag{D.6}$$

- Ball Defect [96]

$$f_v = \left(\frac{D_c}{D_b}\right) f_r \left(1 - \frac{D_b^2}{D_c^2}\cos^2\beta\right) \tag{D.7}$$

- Cage Defect [10]

$$f_v = \left(\frac{f_r}{2}\right) \left(1 - \frac{D_b}{D_c}\cos\beta\right) \tag{D.8}$$

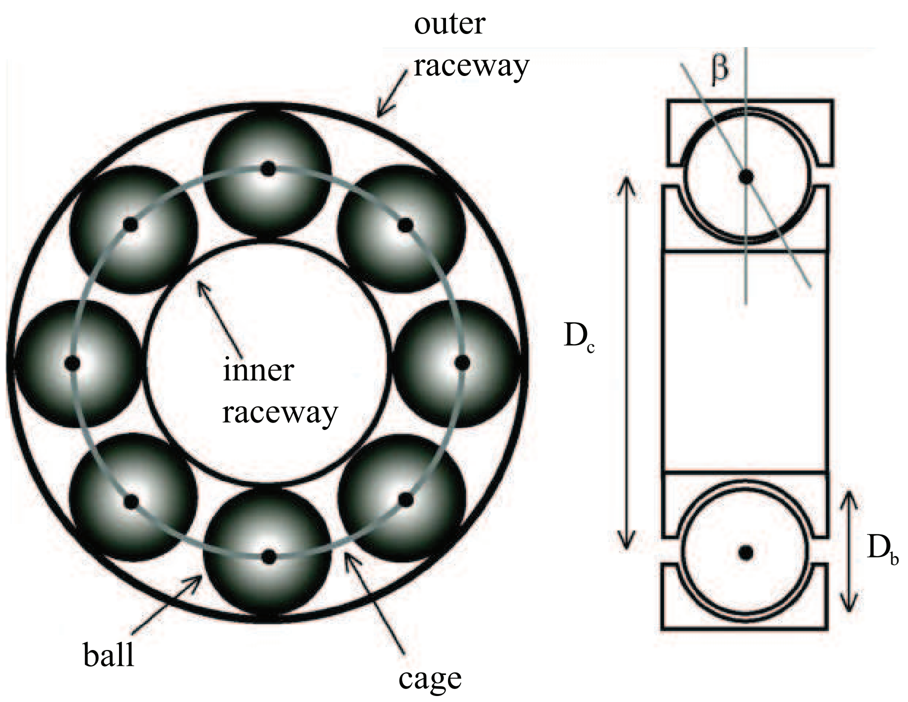


Figure D.1: Geometry of a rolling-element bearing. Adapted from [96].

where N is the number of balls, D_b is the ball diameter, D_c is the pitch or cage diameter, and β is the contact angle of the ball with the races

Rotating eccentricities can occur due to bearing faults, which can lead to variations in the machine inductances. This produces frequencies in the stator current that can be used to detect bearing faults. These vibration frequencies reflected in the current spectrum can be calculated as [10]

$$f_{bng} = |f_1 \pm k f_v| \tag{D.9}$$

where f_1 the electrical supply frequency, f_v is the vibration frequency in Hz, and $k = 1, 2, 3 \ldots$

D.2 Motor KPI Listing

This remainder of this appendix contains a listing of KPIs used for motor fault detection.

Table D.1: Motor Fault Detection KPIs

#	KPI Description	#	KPI Description
1	CurrentA.RMSCalc.Amps	34	Run.Speed.Frequency
2	CurrentB.RMSCalc.Amps	35	AccMeter_1.VelRMS.Calc
3	CurrentC.RMSCalc.Amps	36	AccMeter_2.VelRMS.Calc
4	Voltage_ab.RMSCalc.Volts	37	AccMeter_3.VelRMS.Calc
5	Voltage_bc.RMSCalc.Volts	38	AccMeter_4.VelRMS.Calc
6	Voltage_ca.RMSCalc.Volts	39	AccMeter_5.VelRMS.Calc
7	THD.CurrentA.IEEE	40	AccMeter_6.VelRMS.Calc
8	THD.CurrentA.IEC	41	Run.Speed.Magnitude
9	CurharidA.OneE.Mag	42	RMS_Band1.LF.RMS
10	CurharidA.ThreeE.Mag	43	RMS_Band1.MF.RMS
11	CurharidA.FiveE.Mag	44	RMS_Band1.HF.RMS
12	CurharidA.SevenE.Mag	45	Vibtonid.OneR1.Mag
13	CurharidA.NineE.Mag	46	Vibtonid.HalfR1.Mag
14	CurharidA.ElevenE.Mag	47	Vibtonid.TwoR1.Mag
15	CurharidA.ThirteenE.Mag	48	Vibtonid.ThreeR1.Mag
16	THD.CurrentB.IEEE	49	Vibtonid.FourR1.Mag
17	THD.CurrentB.IEC	49	Vibtonid.FourR1.Mag
18	CurharidB.OneE.Mag	50	Vibtonid.OneE1.Mag
19	CurharidB.ThreeE.Mag	51	Vibtonid.TwoE1.Mag
20	CurharidB.FiveE.Mag	52	Vibtonid.SixE1.Mag
21	CurharidB.SevenE.Mag	53	RMS_Band2.LF.RMS
22	CurharidB.NineE.Mag	54	RMS_Band2.MF.RMS
23	CurharidB.ElevenE.Mag	55	RMS_Band2.HF.RMS
24	CurharidB.ThirteenE.Mag	56	Vibtonid.OneR2.Mag
25	THD.CurrentC.IEEE	57	Vibtonid.HalfR2.Mag
26	THD.CurrentC.IEC	58	Vibtonid.TwoR2.Mag
27	CurharidC.OneE.Mag	59	Vibtonid.ThreeR2.Mag
28	CurharidC.ThreeE.Mag	60	Vibtonid.FourR2.Mag
29	CurharidC.FiveE.Mag	61	Vibtonid.OneE2.Mag
30	CurharidC.SevenE.Mag	62	Vibtonid.TwoE2.Mag
31	CurharidC.NineE.Mag	63	Vibtonid.SixE2.Mag
32	CurharidC.ElevenE.Mag	64	RMS_Band3.LF.RMS
33	CurharidC.ThirteenE.Mag	65	RMS_Band3.MF.RMS

Table D.2: Motor Fault Detection KPIs cont.

#	KPI Description	#	KPI Description
66	RMS_Band3.HF.RMS	100	Vibtonid.OneR6.Mag
67	Vibtonid.OneR3.Mag	101	Vibtonid.HalfR6.Mag
68	Vibtonid.HalfR3.Mag	102	Vibtonid.TwoR6.Mag
69	Vibtonid.TwoR3.Mag	103	Vibtonid.ThreeR6.Mag
70	Vibtonid.ThreeR3.Mag	104	Vibtonid.FourR6.Mag
71	Vibtonid.FourR3.Mag	105	Vibtonid.OneE6.Mag
72	Vibtonid.OneE3.Mag	106	Vibtonid.TwoE6.Mag
73	Vibtonid.TwoE3.Mag	107	Vibtonid.SixE6.Mag
74	Vibtonid.SixE3.Mag	108	Slip
75	RMS_Band4.LF.RMS	109	Broken rotor bars.A -3*s
76	RMS_Band4.MF.RMS	110	Broken rotor bars.A -2*s
77	RMS_Band4.HF.RMS	111	Broken rotor bars.A -1*s
78	Vibtonid.OneR4.Mag	112	Broken rotor bars.A +1*s
79	Vibtonid.HalfR4.Mag	113	Broken rotor bars.A +2*s
80	Vibtonid.TwoR4.Mag	114	Broken rotor bars.A +3*s
81	Vibtonid.ThreeR4.Mag	115	Broken rotor bars.B -3*s
82	Vibtonid.FourR4.Mag	116	Broken rotor bars.B -2*s
83	Vibtonid.OneE4.Mag	117	Broken rotor bars.B -1*s
84	Vibtonid.TwoE4.Mag	118	Broken rotor bars.B +1*s
85	Vibtonid.SixE4.Mag	119	Broken rotor bars.B +2*s
86	RMS_Band5.LF.RMS	120	Broken rotor bars.B +3*s
87	RMS_Band5.MF.RMS	121	Broken rotor bars.C -3*s
88	RMS_Band5.HF.RMS	122	Broken rotor bars.C -2*s
89	Vibtonid.OneR5.Mag	123	Broken rotor bars.C -1*s
90	Vibtonid.HalfR5.Mag	124	Broken rotor bars.C +1*s
91	Vibtonid.TwoR5.Mag	125	Broken rotor bars.C +2*s
92	Vibtonid.ThreeR5.Mag	126	Broken rotor bars.C +3*s
93	Vibtonid.FourR5.Mag	127	bearing.vib.Ball spin.1
94	Vibtonid.OneE5.Mag	128	bearing.vib.Fund Train.1
95	Vibtonid.TwoE5.Mag	129	bearing.vib.Inner ring.1
96	Vibtonid.SixE5.Mag	130	bearing.vib.Outer ring.1
97	RMS_Band6.LF.RMS	131	bearing.vib.Ball defect.1
98	RMS_Band6.MF.RMS	132	bearing.vib.Ball spin.2
99	RMS_Band6.HF.RMS	133	bearing.vib.Fund Train.2

Table D.3: Motor Fault Detection KPIs cont.

#	KPI Description	#	KPI Description
134	bearing.vib.Inner ring.2	161	Cur.bearing.-Ball defect.A
135	bearing.vib.Outer ring.2	162	Cur.bearing.+Ball spin.A
136	bearing.vib.Ball defect.2	163	Cur.bearing.+Fund Train.A
137	bearing.vib.Ball spin.3	164	Cur.bearing.+Inner ring.A
138	bearing.vib.Fund Train.3	165	Cur.bearing.+Outer ring.A
139	bearing.vib.Inner ring.3	166	Cur.bearing.+Ball defect.A
140	bearing.vib.Outer ring.3	167	Cur.bearing.-Ball spin.B
141	bearing.vib.Ball defect.3	168	Cur.bearing.-Fund Train.B
142	bearing.vib.Ball spin.4	169	Cur.bearing.-Inner ring.B
143	bearing.vib.Fund Train.4	170	Cur.bearing.-Outer ring.B
144	bearing.vib.Inner ring.4	171	Cur.bearing.-Ball defect.B
145	bearing.vib.Outer ring.4	172	Cur.bearing.+Ball spin.B
146	bearing.vib.Ball defect.4	173	Cur.bearing.+Fund Train.B
147	bearing.vib.Ball spin.5	174	Cur.bearing.+Inner ring.B
148	bearing.vib.Fund Train.5	175	Cur.bearing.+Outer ring.B
149	bearing.vib.Inner ring.5	176	Cur.bearing.+Ball defect.B
150	bearing.vib.Outer ring.5	177	Cur.bearing.-Ball spin.C
151	bearing.vib.Ball defect.5	178	Cur.bearing.-Fund Train.C
152	bearing.vib.Ball spin.6	179	Cur.bearing.-Inner ring.C
153	bearing.vib.Fund Train.6	180	Cur.bearing.-Outer ring.C
154	bearing.vib.Inner ring.6	181	Cur.bearing.-Ball defect.C
155	bearing.vib.Outer ring.6	182	Cur.bearing.+Ball spin.C
156	bearing.vib.Ball defect.6	183	Cur.bearing.+Fund Train.C
157	Cur.bearing.-Ball spin.A	184	Cur.bearing.+Inner ring.C
158	Cur.bearing.-Fund Train.A	185	Cur.bearing.+Outer ring.C
159	Cur.bearing.-Inner ring.A	186	Cur.bearing.+Ball defect.C
160	Cur.bearing.-Outer ring.A		