This is to certify that the

thesis entitled

THE USE OF WAVELET ANALYSIS FOR THE PROGNOSIS OF FAILURES IN ELECTRIC MOTORS

presented by

Wes Zanardelli

has been accepted towards fulfillment
of the requirements for

Masters degree in Electrical Eng

E G Strangas
Major professor

Date 13 Dec 2000

The Use of Wavelet Analysis for the Prognosis of Failures in Electric Motors

By

Wesley G. Zanardelli

A THESIS

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Department of Electrical and Computer Engineering

2000

ABSTRACT

The Use of Wavelet Analysis for the Prognosis of Failures in Electric Motors

By

Wesley G. Zanardelli

The ability to give a prognosis for failure of a system is an invaluable tool. In this work, four wavelet-based methods have been developed for use with DC motors used in automotive applications that achieve this goal. Wavelet and filter bank theory is reviewed, as well as the nearest neighbor rule, the Minkowski p metrics and linear discriminant functions. The framework for the development of a fault detection and classification algorithm is described. Additionally, an experimental setup based on RT-Linux, and results from testing are presented, verifying the analysis.

To my parents, Virgil and Loretta Zanardelli

ACKNOWLEDGMENTS

I would like to express my gratitude and thanks to my thesis advisor Professor Elias Strangas for his guidance, support, and his professional as well as personal example. I would also like to thank Professors Hassan Khalil and Hayder Radha for their time and effort in being part of my committee. I would like to thank Professor Hassan Khalil, in particular, for his guidance as well as the insight he was able to provide throughout this work.

I would like to extend special thanks to Professor John Miller for providing the funding which made this project possible. I would like to thank Larry Castle and David Price for providing the fuel pumps and wiper motors which were used in this project. I would also like to thank Kevin Houle whose semester-long commitment greatly helped with the development of one of the algorithms which was implemented.

Special words of thanks go to my lab colleagues Ali Khurram, Andres Diaz, Fida Khan, Bader Aloliwi, Bilal Malik and John Kelly for all of their support. I would also like to thank the members of the department whose help was much appreciated, including Brian Wright, Roxanne Peacock, Marilyn Shriver and Vanessa Mitchner.

Finally, I owe special thanks to my parents, Virgil and Loretta Zanardelli, whose support and encouragement helped make my graduate studies possible.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

In recent years, industries have focused much attention in methods of analysis to determine the state of health of electrical systems. The ability to get a prognosis of a system is very useful, because attention can be brought to any problems a system may exhibit before they cause the system to fail. The electric motor is a prime example of a system where failure occurring at an inopportune time can be inconvenient and expensive. The ability to give a prognosis to an electric motor is crucial in correcting problems before they cause the motor to fail.

To address these concerns, specifically in cases of electrical motors, research has been done in the area of examining signatures from the current waveforms of unhealthy electric motors. In general, it can be considered a trivial problem to detect when an electric motor is no longer functional. It is more complicated, however, to measure the state of health of a functional motor. The state of health can include information such as imminent problems with the motor and an estimation of the remaining life expectancy of the motor. In this thesis, this problem is approached by measuring the voltage, current, torque and speed of motors with known defects that are considered to significantly shorten their life and analyzing this data using wavelets. The signatures that indicate the presence of a fault are often minor transient effects in their current waveforms. These waveforms are non-stationary signals whose

characteristics make Fourier methods unsuitable for analysis.

Wavelet analysis is ideal for these types of applications. Unlike traditional frequency domain analysis methods, wavelet analysis has the key advantage of being able to localize information in time. When non-stationary information is transformed into the frequency domain, in the case of the Fourier transform, most of the information about the transient components of the signal is lost. The multiresolution property of wavelet analysis allows for both good time resolution at high frequencies and good frequency resolution at low frequencies. Even techniques such as the short-time Fourier transform (STFT), where a nonstationary signal is divided into short pseudostationary segments and then analyzed, are not suitable for the analysis of signals with complex time-frequency characteristics. If the time-domain analysis window in the STFT is made too short, frequency resolution will suffer, and lengthening it could invalidate the assumption of stationarity within the window.

Implementing wavelets to give a prognosis for an electrical system was initially motivated by research done in the biomedical community [1]. In that research, wavelets were used to analyze heart sounds, which are correlated to the turbulence of blood flow in the cardiovascular system. Instruments were then developed using this information that were capable of detecting coronary ischemia, or the reduction of blood flow caused by clogged arteries, in its early stages. Coronary ischemia, left untreated, is one of the causes of coronary artery disease.

Both the current waveforms from electric motors and signals generated from heart sounds are nonstationary and contain information about faults present in each system. The early detection and classification of faults present in either of these systems can provide valuable information so that steps can be taken to prevent these systems from failing. This thesis focuses on the approaches developed using wavelet analysis to obtain a prognosis for electrical motors.

CHAPTER 2

Wavelets and Filter Banks

2.1 Introduction to Wavelets

We will begin by defining some of the notation that we will be using. $L^p(\mathbf{R})$ denotes the Hilbert space of measurable functions $f(x)$ (2.1):

$$\int_{-\infty}^{+\infty} |f(x)|^p dx < +\infty \tag{2.1}$$

The Hilbert space of measurable, square-integrable functions, $f(x) \in L^2(\mathbf{R})$, (2.2) is a subset of (2.1):

$$\int_{-\infty}^{+\infty} |f(x)|^2 dx < +\infty \tag{2.2}$$

A basis for a space $\mathcal{V}$ is defined as a set of linearly independent functions that span the space. That is, any function in $\mathcal{V}$ can be written as a linear combination of the basis functions. This can be illustrated by a linear decomposition (2.3), where $f(t)$ represents any function in the space $\mathcal{V}$, $\psi_\ell(t)$ are the basis functions, and a_ℓ are the scaling coefficients.

$$f(t) = \sum_\ell a_\ell \psi_\ell(t) \tag{2.3}$$

We can now begin an introduction of wavelets. A wave can be considered to be a function that is periodic in time. An example of a wave is the sinusoid, shown in

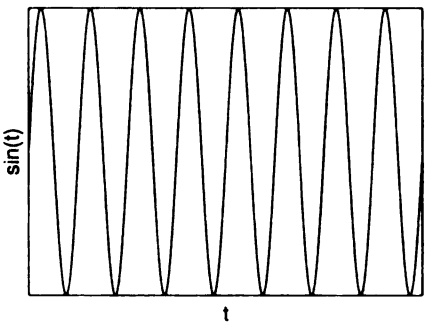

Figure 2.1. Sinusoidal wave

Figure 2.1. Sinusoids are often used in the decomposition and analysis of periodic signals. The Fourier series is an example of this. The Fourier series is a basis for the set of $L^2(\mathbf{R})$ functions. The trigonometric form of the Fourier series of functions $x_p(t)$ is shown in (2.4) [2]:

$$x_p(t) = a_0 + \sum_{k=1}^{\infty} a_k \cos(k\omega_0 t) + b_k \sin(k\omega_0 t) \tag{2.4}$$

A wavelet system [4] is a set of scaling functions and wavelet functions and is also a basis for the set of $L^2(\mathbf{R})$ functions. We will be defining the scaling function and the wavelet function in the next section, however it is appropriate to discuss some of their basic characteristics in this introduction. One of the unique properties of a wavelet system is that its basis functions, the scaling function and the wavelet function, have finite energy, which is concentrated around a point. The basis functions of the Fourier series have an infinite amount of energy, which spreads out on $-\infty < t < \infty$. This property gives a wavelet system the ability to localize a signal in both time and frequency. The Fourier series however, can only localize a signal in frequency. An example of a scaling function and its corresponding wavelet function from the Daubechies' family are shown in Figure 2.2.

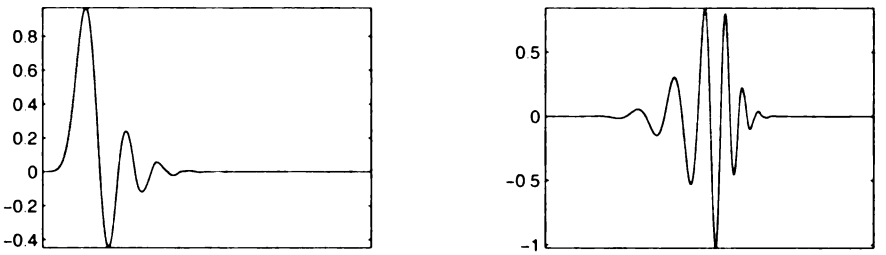

Figure 2.2. Daubechies' D20 scaling and wavelet functions

2.2 The Continuous Wavelet Transform

Although the implementation of the detection algorithms is based on the discrete wavelet transform, a basic understanding of the continuous wavelet transform is helpful. First, we will define more of the notation that we will be using.

The convolution of two functions f and g is shown in (2.5):

$$f * g(x) = \int_{-\infty}^{+\infty} f(u)g(x-u)du \tag{2.5}$$

The exponential form of the Fourier transform of a function f is denoted as $\hat{f}$ in (2.6):

$$\hat{f}(\omega) = \int_{-\infty}^{+\infty} f(x)e^{i\omega x}dx \tag{2.6}$$

Finally, for any function $f(x)$, $f_s(x)$ denotes the dilation of $f(x)$ by the scale factor s in (2.7):

$$f_s(x) = \frac{1}{s}f\left(\frac{x}{s}\right) \tag{2.7}$$

We can now describe the properties of the continuous wavelet transform [7]. As mentioned in the introduction, wavelets are a basis for the $L^2(\mathbf{R})$ functions (2.2). A function $\psi(x)$ is said to be a wavelet if and only if its Fourier transform $\hat{\psi}(x)$ satisfies

(2.8):

$$\int_0^{+\infty} \frac{|\hat{\psi}(\omega)|^2}{\omega} d\omega = \int_{-\infty}^{0} \frac{|\hat{\psi}(\omega)|^2}{|\omega|} d\omega = C_\psi < +\infty \tag{2.8}$$

This implies that the area under the wavelet function is zero (2.9):

$$\int_{-\infty}^{+\infty} \psi(u) du = 0 \tag{2.9}$$

In general, we will denote the continuous wavelet transform of a function by $Wf(s,x)$, which is a function of both scale s and position x, or in this case time. We can say that the continuous wavelet transform is defined for the scale-space plane. The value of $Wf(s,x_0)$ depends on the values of $f(x)$ in an area near x_0, which is proportional to the scale s. We can define a wavelet function for a specific scale s as $\psi_s(x) = (1/s)\psi(x/s)$ and we can define the continuous wavelet transform of a function $f(x)$ at that scale (2.10):

$$Wf(s,x) \triangleq f * \psi_s(x) \tag{2.10}$$

At the scale $s = 1$, $\psi(x)$ is often referred to as the mother wavelet.

The concepts and ideas in the continuous implementation of wavelet transform help in understanding the theory behind wavelets; however, all of the signals used in this work are sampled by A/D hardware and a personal computer and are therefore discrete in nature. We will now go on to discuss the discrete wavelet transform.

2.3 The Discrete Wavelet Transform

We will define the discrete wavelet transform using the idea of multiresolution by starting with the scaling function and defining the wavelet function in terms of it [4]. A basic one-dimensional scaling function can be defined to translate a function in

time (2.11) where $\mathbf{Z}$ is the set of all integers.

$$\varphi_k(t) = \varphi(t-k) \quad k \in \mathbf{Z} \quad \varphi \in L^2 \tag{2.11}$$

Wavelet systems are two-dimensional, so we will define a scaling function $\varphi_{j,k}(t)$ that both scales and translates a function $\varphi(t)$ (2.12):

$$\varphi_{j,k}(t) = 2^{j/2}\varphi(2^j(t-2^{-j}k)) \quad j,k \in \mathbf{Z} \quad \varphi \in L^2, \tag{2.12}$$

where j is the $\log_2$ of the scale and $2^{-j}k$ represents the translation in time. We can define a subspace of the $L^2(\mathbf{R})$ functions as the scaling function space $\mathcal{V}$. We note that $\varphi_{j,k}(t)$ spans the space $\mathcal{V}_j$, meaning that any function in $\mathcal{V}_j$ can be represented by a linear combination of functions of the form $\varphi_{j,k}(t)$.

When discussing scaling functions in terms of multiresolution analysis we need to see the relationship between the span of scaling functions with different indicies (2.13-2.14):

$$\cdots \subset \mathcal{V}_{-2} \subset \mathcal{V}_{-1} \subset \mathcal{V}_0 \subset \mathcal{V}_1 \subset \mathcal{V}_2 \subset \cdots \subset L^2 \tag{2.13}$$

$$\mathcal{V}_{-\infty} = \{0\}, \quad \mathcal{V}_{\infty} = L^2 \tag{2.14}$$

Another subspace of the $L^2(\mathbf{R})$ functions is the wavelet vector space $\mathcal{W}$. A wavelet spans the space $\mathcal{W}_j$, which represents the difference between two scaling function spaces, $\mathcal{V}_j$ and $\mathcal{V}_{j+1}$. We can see (2.15) which extends to (2.16):

$$\mathcal{V}_1 = \mathcal{V}_0 \oplus \mathcal{W}_0 \tag{2.15}$$

$$L^2 = \mathcal{V}_0 \oplus \mathcal{W}_0 \oplus \mathcal{W}_1 \oplus \cdots \tag{2.16}$$

The relationship between the scaling function and wavelet vector spaces is illustrated in Figure 2.3.

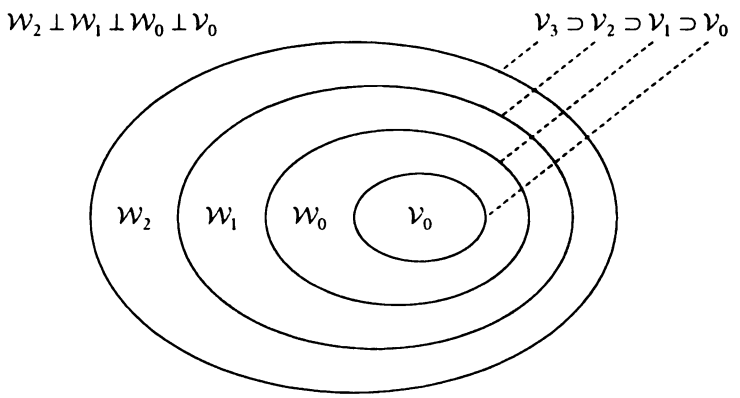

Figure 2.3. Scaling Function and Wavelet Vector Spaces

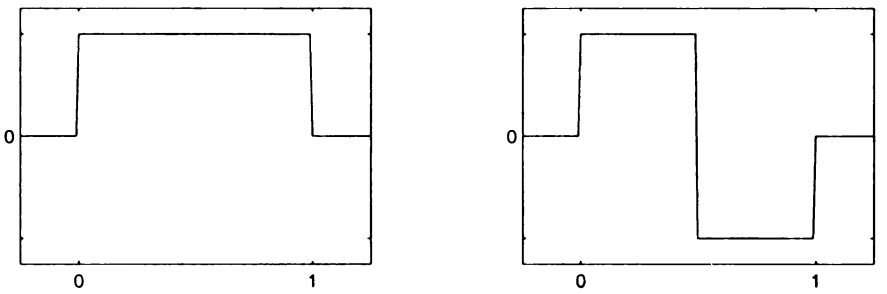

Figure 2.4. Haar scaling and wavelet functions

The scale of the initial space V_j can be chosen arbitrarily, but is usually chosen to be the coarsest detail of interest in a signal. It can even be chosen as $j = -\infty$ where L^2 can be reconstructed in terms of only wavelet functions (2.17):

$$L^2 = \cdots \oplus W_{-2} \oplus W_{-1} \oplus W_0 \oplus W_1 \oplus W_2 \oplus \cdots \tag{2.17}$$

A very basic wavelet system with a scaling function and a wavelet function to make up the detail between one level of decomposition and the next is the Haar system shown in Figure 2.4.

We can now say that any function in $L^2(\mathbf{R})$ can be written as an expansion of a

scaling function and wavelets (2.18), where $c_{j_0}(k)$ are the scaling function coefficients, $\varphi_{j_0,k}(t)$ is the scaling function at the initial scale j_0, $d_j(k)$ are the wavelet function coefficients and $\psi_{j,k}(t)$ are the wavelet functions spanning the space between $\mathcal{V}_{j_0}$ and L^2.

$$f(t) = \sum_{k=-\infty}^{\infty} c_{j_0}(k)\varphi_{j_0,k}(t) + \sum_{k=-\infty}^{\infty}\sum_{j=j_0}^{\infty} d_j(k)\psi_{j,k}(t) \tag{2.18}$$

2.4 Filter Banks

In order to perform the Discrete Wavelet Transform on a computer, computational methods must be developed. The DWT can be performed without using calculus, but rather additions and multiplications in the form of convolutions [4].

If we consider the linear decomposition in (2.3), and if the basis functions are orthogonal (2.19),

$$\langle \psi_k(t), \psi_\ell(t) \rangle = \int \psi_k(t)\psi_\ell(t)dt = 0, \quad k \neq \ell \tag{2.19}$$

we can determine the coefficients of the decomposition, a_k, by calculating the inner product (2.20):

$$a_k = \langle f(t), \psi_k(t) \rangle = \int f(t)\psi_k(t)dt \tag{2.20}$$

In the two-dimensional case of the wavelet transform, we can use the same techniques to calculate the scaling coefficients (2.21) and the wavelet coefficients (2.22):

$$c_j(k) = \langle f(t), \varphi_{j,k}(t) \rangle = \int f(t)\varphi_{j,k}(t)dt \tag{2.21}$$

$$d_j(k) = \langle f(t), \psi_{j,k}(t) \rangle = \int f(t)\psi_{j,k}(t)dt \tag{2.22}$$

We can finally define the scaling function coefficients for a coarse scale from the scaling function coefficients at the next finer scale by convolving the coefficients at

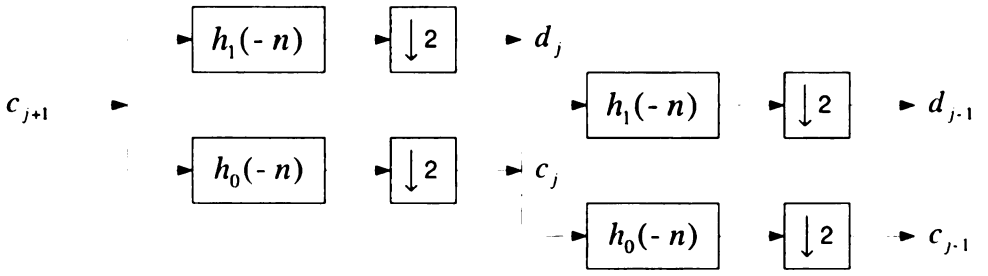

Figure 2.5. Two-Stage Filter Bank Analysis Tree

the finer scale with the recursion coefficients $h_0(n)$ and then down-sampling (2.23):

$$c_j(k) = \sum_m h_0(m - 2k)c_{j+1}(m) \tag{2.23}$$

We can do the same in the case of the wavelet coefficients using the recursion coefficients $h_1(n)$ (2.24) where $h_1(n) = (-1)^n h(1 - n)$.

$$d_j(k) = \sum_m h_1(m - 2k)c_{j+1}(m) \tag{2.24}$$

The decomposition lowpass filter coefficients, $h_0(n)$, and the decomposition highpass filter coefficients, $h_1(n)$ corresponding to each mother wavelet used in this thesis are listed in Appendix C. An example of a filter bank analysis tree is illustrated in Figure 2.5.

The down-sampling operation does not result in the loss of signal information. In the filter bank structure shown in Figure 2.5, there is enough information to reconstruct c_{j+1} in either the combination of c_j and d_j, or the combination of c_{j-1}, d_{j-1} and d_j. Despite down-sampling, either of these combinations of coefficients will have approximately the same number of values as c_{j+1}. Signal reconstruction from DWT coefficients is not used in this thesis, however it is discussed in detail in [4].

CHAPTER 3

Clustering and Discriminant Functions

3.1 Nearest Neighbor Rule

In order to categorize a sample point in d-dimensional space into a set of previously classified points, we use the nearest neighbor rule (1-NN). We assume that observations which are close to each other (in some appropriate metric) will have the same classification [5]. We could approach this problem in two different ways. First, by assuming that we have some given statistical distribution for the data, and second, by assuming no knowledge of a distribution except for what can be concluded from the samples. We will focus on the second method, where we assume no probabilistic model of a distribution.

In calculating the minimum distance, we need to use some appropriate measure. Any dissimilarity measure (3.1) would be applicable, however the most commonly used dissimilarity measures are the Minkowski p metrics (3.2) where the d in the summation is defined to be the dimensionality of the vectors $\mathbf{X}_m$ and $\mathbf{X}_n$ [6].

$$d(\mathbf{X}_m, \mathbf{X}_n) = g\left[\sum_{i=1}^{d} f_i(X_{im}, X_{in})\right] \tag{3.1}$$

$$d(\mathbf{X}_m, \mathbf{X}_n) = \left[\sum_{i=1}^{d} |X_{im} - X_{in}|^p\right]^{\frac{1}{p}} \quad (p \geq 1) \tag{3.2}$$

The three most often used Minkowski metrics are the taxi-cab distance (3.3) where $p = 1$, the Euclidean metric (3.4) for which $p = 2$ and the maximum coordinate distance (3.5) where $p = \infty$. This work focuses on the use of the Euclidean metric.

$$d(\mathbf{X}_m, \mathbf{X}_n) = \sum_{i=1}^{d} |X_{im} - X_{in}| \tag{3.3}$$

$$d(\mathbf{X}_m, \mathbf{X}_n) = \left[\sum_{i=1}^{d} (X_{im} - X_{in})^2\right]^{\frac{1}{2}} \tag{3.4}$$

$$d(\mathbf{X}_m, \mathbf{X}_n) = \max_{1 \leq i \leq d} \{|X_{im} - X_{in}|\} \tag{3.5}$$

3.2 Linear Discriminant Functions

A second approach to categorizing points in a d-dimensional space relies on the use of discriminant functions. In the implementation of discriminant functions, we assume no knowledge of a probability distribution among the sample points. The space is separated into K disjoint regions, each having its own weighting coefficients. In this work, we focus on the use of linear discriminant functions (3.6) [9],

$$D_k(\mathbf{x}) = x_1\alpha_{1k} + x_2\alpha_{2k} + , \ldots , + x_N\alpha_{Nk} + \alpha_{N+1,k} \quad k = 1, 2, \ldots, K \tag{3.6}$$

where x is the N-dimensional sample vector and α are the normalized weighting coefficients for the k-th class. A sample vector belongs to a particular class if its discriminant function is greater for that class than for any other class, i.e., $\mathbf{x}_i$ belongs to class C_j if

$$\boldsymbol{\alpha}_j^T \mathbf{x}_i > \boldsymbol{\alpha}_k^T \mathbf{x}_i \quad \text{for every } k \neq j.$$

The weighting coefficients are adjusted from their initial guess through a training procedure. The algorithm for this procedure makes adjustments to the weighting coefficients until each known sample vector is correctly classified. Young and Calvert [9] prove that this training algorithm will converge in a finite number of steps. When a known sample vector is correctly classified, no adjustment to the weighting coefficients is made. When one of the known sample vectors is incorrectly classified, or

$$\boldsymbol{\alpha}_j^T \mathbf{x}_i \leq \boldsymbol{\alpha}_l^T \mathbf{x}_i$$

where

$$\boldsymbol{\alpha}_l^T \mathbf{x}_i = \max_{l \neq j} \left[\boldsymbol{\alpha}_1^T \mathbf{x}_i, \ldots, \boldsymbol{\alpha}_K^T \mathbf{x}_i \right],$$

adjustments are made to $\boldsymbol{\alpha}_j$ (3.7) and $\boldsymbol{\alpha}_l$ (3.8) only,

$$\boldsymbol{\alpha}_j(i+1) = \boldsymbol{\alpha}_j(i) + a\mathbf{x}_i \tag{3.7}$$

$$\boldsymbol{\alpha}_l(i+1) = \boldsymbol{\alpha}_l(i) - a\mathbf{x}_i \tag{3.8}$$

where a is a gain constant.

CHAPTER 4

Analysis Methods

4.1 Introduction

In this chapter, the theory discussed in the previous chapters including wavelets, filter banks, the nearest neighbor rule, and linear discriminant functions is used in the development of four wavelet-based fault detection and classification algorithms. These algorithms are applied to the current waveforms of brush DC motors used in automotive applications, in particular HVAC fan motors, windshield wiper motors and fuel pump motors. Experimental setups used to obtain these waveforms are discussed in Section 5.1. The algorithms are presented in the order they were developed, and in general they increase in complexity. The first algorithm makes decisions based on the output of the discrete wavelet transform directly. The second algorithm goes a step further and makes decisions based on the modulus maxima of the discrete wavelet transform. This greatly reduces the number of calculations required in the algorithm. The third algorithm adds a normalization step to the modulus maxima of the discrete wavelet transform coefficients and employs a more statistical decision making process based on Euclidean distance calculations. This algorithm is considered to be the best balance between the deterministic approach used in the first two algorithms and the statistical framework of the fourth algorithm. Decisions made in the fourth algorithm

are based on linear discriminant functions, however an additional training procedure is employed. The training procedure is used to fit each motor used in the development of the algorithm with a specified fault classification.

At this point, one might ask why it is not possible to detect the types of faults that are present by simply applying a threshold to the original signal. This would not be effective for several reasons. First, changes in the load on the motor would not be allowed since the load is proportional to the average value of the current. In order to set the threshold value close enough to the signal to accurately detect discontinuities, the load would have to be almost identical in every test. If this were the case, it would not be possible to test a system with a dynamic load such as a windshield wiper motor under normal operating conditions.

Second, it is often the case that visual inspection of an unprocessed signal does not help one to determine whether or not a fault exists. In cases where it can be determined that a fault exists, visual inspection of the signal does not usually help in classifying it.

An sample of raw data taken from several windshield wiper motors running at low speed on a wet windshield is shown in Figure 4.1. A zoomed section of the data is shown in Figure 4.2. It is clear that it is not possible to detect and classify each of the faults shown without a more sophisticated approach than looking at the current directly.

4.2 Discrete Wavelet Transform

When a slight discontinuity is present in a signal, depending on the mother wavelet chosen, its location is usually obvious after inspection of the output of the wavelet transform. With some experience, one can often determine the nature of the fault as well. Different mother wavelets will help to extract different types of discontinuities

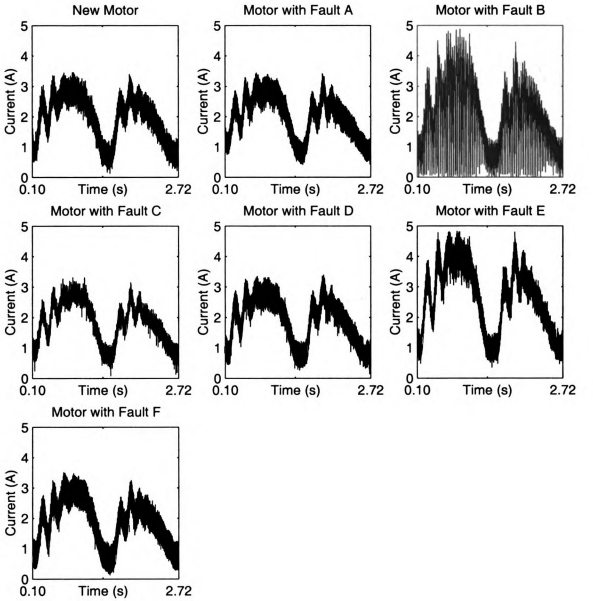

Figure 4.1. Wiper Motor Current Waveforms – Low Speed Wet Windshield

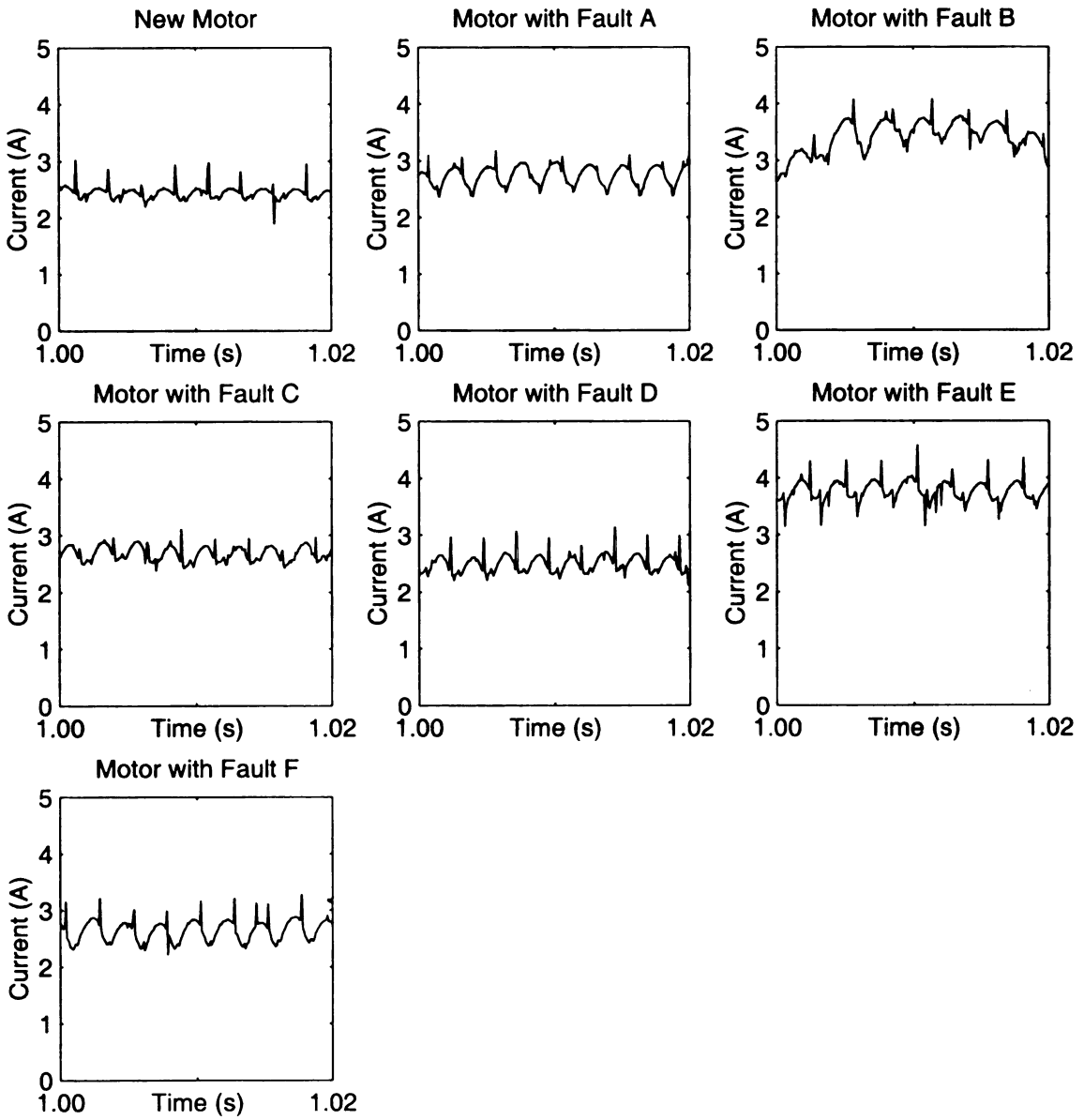

Figure 4.2. Wiper Motor Current Waveforms – Low Speed Wet Windshield

from a signal. The choice of mother wavelet is one of the most critical steps in developing an algorithm to detect and classify faults using wavelets.

The first approach at using wavelets to detect and classify faults was implemented on HVAC fan motors. These motors are discussed in detail in Section 5.1.1. This initial algorithm was motivated by a publication in the biomedical community [1] where it was shown that abnormal cardiac cycles could be detected by abnormally high wavelet coefficients in certain scales using specific mother wavelets.

The algorithm has both a detection phase and a classification phase. The criterion for the detection phase was the comparison of the coefficients of the Discrete Wavelet Transform (DWT) using the Biorthogonal 1.3 mother wavelet at level 9 with a threshold which was determined experimentally. This threshold was set to 2.75, and if exceeded, would indicate the presence of some type of fault.

The DWT coefficients using the Biorthogonal 1.3 mother wavelet at level 11 were used for the classification phase in the analysis. Thresholding was also used on these coefficients, so that if the value of any of the coefficients was greater than or equal to 3 when the first criterion was met, the existence of a step fault would be recognized at that point in the decomposition. Otherwise, if the first criterion was met and the value of the level 11 coefficients was less than 3, the presence of a ramp fault would be recognized at that point in the decomposition.

4.3 Modulus Maxima of the DWT

The second approach to the fault detection and classification problem was implemented on windshield wiper motors. These motors are discussed in detail in Section 5.1.2. The algorithm uses an if-then-else set of rules on the modulus maxima of the wavelet coefficients from the first ten different levels of decomposition. Daubechies' D8 and the C18 Coiflet were used as mother wavelets for decomposition. Wiper mo-

tor data from both low speed dry windshield as well as low speed wet windshield testing was used. For this approach, the goal was to detect and classify all of the faults (including all variations of Fault E) with the exception of Fault F which had the faulty parking mechanism. Initially, it was not believed to be possible to detect the presence of Fault F since in the testing procedure, data is only analyzed while the motors are running. In later testing however, it was discovered that it was possible to properly classify motors with this fault as well.

The method used to detect irregularities in the system was to apply a threshold to the wavelet transform coefficients of a measurement of the current through a motor being tested. To select a threshold, the original signal is compared with the wavelet transform modulus maxima using different mother wavelets and the results are observed at various scales. A local maximum of the wavelet transform modulus is defined at a point x_0 where $\partial Wf(s,x)/\partial x$ has a zero-crossing at $x = x_0$ and $|Wf(s,x)| < |Wf(s,x_0)|$ when x belongs to the neighborhood around x_0. In general, the number of wavelet maxima increase proportionally to the number of irregularities in the signal. Also, the number of maxima at a given scale often increase linearly with the number of vanishing moments in the wavelet. We should, ideally, have the minimum number of maxima necessary to detect the desired irregularity in the signal.

A wavelet is said to have n vanishing moments if and only if for all positive integers k where $k < n$, (4.1) is satisfied.

$$\int_{-\infty}^{+\infty} x^k \psi(x) dx = 0 \tag{4.1}$$

When the irregularities in a signal that are being searched for are sharp, it is desirable to choose wavelets with fewer vanishing moments.

When analyzing a signal in the presence of noise, many additional modulus maxima are created in the finer scales. The maxima due to light noise disappear in higher

scales where only edges relevant to the signal remain.

In building the detection and classification algorithm, fifteen mother wavelets were used. The modulus maxima of the coefficients from the discrete wavelet transform over ten levels of decomposition for each motor were analyzed. For the detection phase of the algorithm, it was considered that a fault may be present when one of the modulus maxima exceeded a threshold. The threshold for each level of decomposition was set to be 5% above the maximum of the modulus maxima observed on the motors said to be either new or having Fault F.

For the classification phase of the algorithm, a parameter α, named the localization parameter, was developed to give an estimate for the level of decomposition exceeding its corresponding threshold most (4.2),

$$\alpha = \frac{\sum_{i=1}^{10}(i \times (d_i - \hat{d}_i))}{\sum_{i=1}^{10}(d_i - \hat{d}_i)} \quad d_i - \hat{d}_i > 0 \tag{4.2}$$

where d_i is defined as the modulus maxima of the ith level of decomposition and $\hat{d}_i$ is the threshold at that level. The parameter α is only defined if the criterion for detection is met, that is if for at least one level of decomposition $d_i - \hat{d}_i > 0$. Then the classification strategy was to run the decomposition coefficients as well as the parameter α through a decision tree to reduce the number of possible faults.

Tables 4.1–4.5 show the modulus maxima as well as the localization parameters for a sample of motors having each of Faults A–E. The bold values represent the coefficients that exceed the corresponding detection threshold. The modulus maxima manifest themselves in a unique way for each of the faults. The localization parameter remains relatively constant for each fault as well.

	Level										
	1	2	3	4	5	6	7	8	9	10	
db1 = [	0.487	0.501	0.699	0.989	1.430	1.216	2.144	2.971	6.583	14.11	] α = N/A
db4 = [	0.437	0.421	0.441	0.801	**1.813**	**1.016**	0.951	1.601	2.395	11.17	] α = 5.457
db7 = [	0.437	0.408	0.472	0.906	**1.817**	0.807	0.872	1.562	1.677	11.25	] α = 5.000
db10 = [	0.422	0.397	0.384	0.907	**1.819**	0.725	1.008	1.438	1.493	9.863	] α = 5.000
bior2.2 = [	0.398	0.551	0.796	1.195	**2.098**	1.815	**2.627**	1.738	**5.654**	11.57	] α = 8.172
bior2.8 = [	0.398	0.611	0.668	1.072	2.098	0.914	1.411	2.043	**4.715**	12.16	] α = 9.000
bior3.5 = [	0.365	0.580	0.747	1.219	2.816	2.153	**3.258**	2.442	4.274	7.351	] α = 7.000
bior6.8 = [	0.430	0.525	0.505	0.900	**1.833**	0.793	1.123	1.443	2.223	10.42	] α = 5.000
coif1 = [	0.490	0.525	0.706	1.068	**1.723**	**1.279**	1.269	1.642	5.084	13.77	] α = 5.541
coif3 = [	0.448	0.432	0.627	1.014	**1.786**	0.878	1.098	1.465	**2.162**	11.33	] α = 5.943
coif5 = [	0.440	0.408	0.510	**1.018**	**1.813**	0.701	1.063	1.436	1.821	5.614	] α = 4.521
sym2 = [	0.530	0.526	0.666	0.948	1.619	**1.242**	1.303	1.614	3.929	15.72	] α = 6.000
sym4 = [	0.465	0.557	0.556	0.918	**1.811**	**1.074**	1.118	1.592	**2.939**	11.84	] α = 6.304
sym6 = [	0.461	0.424	0.647	0.986	**1.768**	0.899	1.086	1.590	**2.228**	10.95	] α = 8.606
sym8 = [	0.458	0.510	0.466	0.913	**1.793**	0.768	1.057	1.516	1.931	10.52	] α = 5.000

db1 = [	0.646	0.701	0.854	**1.146**	**1.656**	**1.535**	1.827	**4.231**	**9.427**	17.59	] α = 7.015
db4 = [	0.562	0.582	0.648	**1.143**	**1.884**	**1.120**	1.074	**1.974**	**4.638**	15.85	] α = 7.810
db7 = [	0.555	0.490	0.509	0.861	**2.026**	**1.000**	0.818	**1.905**	**3.056**	11.33	] α = 7.836
db10 = [	0.479	0.483	0.496	**1.181**	**2.021**	**0.796**	0.805	**1.869**	**3.143**	8.215	] α = 7.388
bior2.2 = [	0.567	0.714	0.969	1.246	**2.276**	**2.141**	**2.715**	2.110	**5.427**	11.00	] α = 7.069
bior2.8 = [	0.567	0.639	0.894	**1.342**	**2.266**	1.228	1.300	2.101	**5.674**	11.13	] α = 7.491
bior3.5 = [	0.438	0.662	0.942	**1.515**	2.655	**2.813**	2.629	2.189	**7.121**	10.79	] α = 7.345
bior6.8 = [	0.586	0.541	0.683	**1.038**	**2.132**	0.945	0.855	**1.751**	**3.588**	10.35	] α = 7.840
coif1 = [	0.686	0.692	0.864	**1.129**	**1.850**	**1.383**	**1.503**	**2.375**	4.926	13.28	] α = 6.082
coif3 = [	0.612	0.576	0.651	0.993	**2.087**	0.905	0.818	**1.896**	**3.197**	11.99	] α = 7.745
coif5 = [	0.595	0.537	0.592	**1.016**	**2.215**	0.773	0.767	**1.756**	**3.036**	10.83	] α = 7.462
sym2 = [	0.582	0.636	0.741	**1.167**	**1.838**	**1.354**	**1.626**	2.108	4.737	10.96	] α = 5.433
sym4 = [	0.638	0.591	0.771	**1.095**	**1.883**	**1.209**	1.064	1.823	**3.729**	12.99	] α = 7.315
sym6 = [	0.617	0.608	0.672	0.959	**1.983**	0.936	0.884	1.844	**3.014**	11.93	] α = 8.112
sym8 = [	0.605	0.520	0.650	1.015	**2.146**	0.868	0.750	**1.928**	**3.080**	11.72	] α = 7.763

Table 4.1. Modulus maxima of the wavelet coefficients for motors with Fault A (coefficients in bold designate those above the corresponding threshold)

	Level										
	1	2	3	4	5	6	7	8	9	10	
db1 = [	**0.859**	**1.208**	**2.510**	**4.720**	**11.61**	**28.27**	**39.73**	**102.4**	**83.92**	**134.4**]	$\alpha = 7.173$
db4 = [	0.730	**1.041**	**2.238**	**3.162**	**6.166**	**10.83**	**23.08**	**67.47**	**105.1**	**117.7**]	$\alpha = 7.937$
db7 = [	0.664	**1.338**	**1.652**	**3.228**	**5.201**	**9.597**	**28.32**	**34.43**	**106.7**	**83.85**]	$\alpha = 7.964$
db10 = [	0.668	**1.047**	**1.457**	**4.374**	**4.567**	**10.83**	**21.64**	**54.08**	**68.98**	**53.83**]	$\alpha = 7.677$
bior2.2 = [	0.691	**1.393**	**2.029**	**3.939**	**5.944**	**13.02**	**37.36**	**58.31**	**164.8**	**151.4**]	$\alpha = 8.188$
bior2.8 = [	0.691	**1.195**	**2.714**	**6.224**	**7.466**	**17.52**	**36.86**	**57.36**	**181.5**	**94.12**]	$\alpha = 7.843$
bior3.5 = [	0.560	1.189	**2.355**	**3.951**	**11.07**	**13.30**	**29.35**	**119.8**	**139.5**	**241.5**]	$\alpha = 8.316$
bior6.8 = [	0.687	**1.014**	**2.196**	**4.965**	**5.822**	**13.07**	**30.25**	**43.57**	**142.9**	**72.68**]	$\alpha = 7.931$
coif1 = [	**0.897**	**1.431**	**2.286**	**3.711**	**6.443**	**12.74**	**37.78**	**57.90**	**152.2**	**112.4**]	$\alpha = 7.763$
coif3 = [	0.744	**1.322**	**1.994**	**5.441**	**6.153**	**12.48**	**32.71**	**43.75**	**141.6**	**64.67**]	$\alpha = 7.933$
coif5 = [	0.696	**1.314**	**1.756**	**3.151**	**5.038**	**12.52**	**28.66**	**42.70**	**135.7**	**65.67**]	$\alpha = 7.934$
sym2 = [	0.712	**1.251**	**2.301**	**4.065**	**6.080**	**19.56**	**35.05**	**54.85**	**107.6**	**88.37**]	$\alpha = 7.525$
sym4 = [	0.776	**1.165**	**1.862**	**4.121**	**5.536**	**11.76**	**35.64**	**54.75**	**138.4**	**95.47**]	$\alpha = 7.960$
sym6 = [	0.732	**1.347**	**1.833**	**5.180**	**6.146**	**11.29**	**34.22**	**47.93**	**141.6**	**70.78**]	$\alpha = 7.981$
sym8 = [	0.702	**1.059**	**2.110**	**5.095**	**5.873**	**12.40**	**31.14**	**43.13**	**138.9**	**79.53**]	$\alpha = 7.925$

db1 = [	0.646	**1.546**	**2.377**	**3.909**	**7.443**	**10.22**	**16.39**	**33.94**	**16.06**	**18.39**]	$\alpha = 6.367$
db4 = [	0.746	0.920	**1.757**	**2.741**	**3.422**	**7.089**	**18.60**	**23.70**	**25.83**	13.07]	$\alpha = 7.312$
db7 = [	0.482	**0.827**	**1.966**	**3.459**	**5.603**	**8.660**	**17.60**	**26.66**	**19.85**	12.68]	$\alpha = 7.097$
db10 = [	0.565	**1.078**	**1.446**	**2.666**	**4.595**	**9.146**	**15.65**	**25.61**	**17.66**	13.18]	$\alpha = 7.024$
bior2.2 = [	0.549	0.967	**1.660**	**3.137**	**6.590**	**10.81**	**20.12**	**28.71**	**34.43**	**15.67**]	$\alpha = 7.376$
bior2.8 = [	0.549	0.982	**2.053**	**3.016**	**7.238**	**10.77**	**19.79**	**33.16**	**22.41**	**14.53**]	$\alpha = 7.049$
bior3.5 = [	0.542	0.887	**2.104**	**4.262**	**5.622**	**12.74**	**24.99**	**50.81**	**27.58**	**16.13**]	$\alpha = 7.286$
bior6.8 = [	0.594	**0.892**	**1.669**	**2.217**	**5.820**	**8.013**	**16.80**	**25.35**	**16.69**	11.07]	$\alpha = 7.138$
coif1 = [	0.640	**0.975**	**1.583**	**2.433**	**4.924**	**9.572**	**19.19**	**21.93**	**24.53**	15.65]	$\alpha = 7.045$
coif3 = [	0.609	**0.897**	**1.599**	**2.239**	**6.218**	**8.195**	**17.56**	**22.49**	**15.77**	11.98]	$\alpha = 7.121$
coif5 = [	0.595	**0.879**	**1.562**	**2.367**	**4.714**	**7.364**	**16.40**	**24.63**	**15.87**	10.10]	$\alpha = 7.112$
sym2 = [	0.531	0.971	**1.453**	**2.659**	**6.707**	**9.533**	**15.56**	**36.14**	**19.62**	17.32]	$\alpha = 7.070$
sym4 = [	0.554	0.894	**1.782**	**2.975**	**4.992**	**8.234**	**16.98**	**20.01**	**26.75**	12.84]	$\alpha = 7.220$
sym6 = [	0.555	**0.904**	**1.656**	**2.634**	**6.206**	**8.347**	**17.38**	**20.93**	**27.26**	12.10]	$\alpha = 7.316$
sym8 = [	0.556	**0.901**	**1.625**	**2.009**	**5.925**	**7.861**	**16.90**	**23.01**	**15.79**	11.29]	$\alpha = 7.104$

Table 4.2. Modulus maxima of the wavelet coefficients for motors with Fault B (coefficients in bold designate those above the corresponding threshold)

	Level										
	1	2	3	4	5	6	7	8	9	10	
db1 = [	0.540	0.463	0.580	0.660	0.943	0.842	**2.557**	**4.007**	**8.067**	16.00]	$\alpha = 7.701$
db4 = [	0.467	0.425	0.535	0.678	1.151	0.594	0.942	1.521	**3.646**	**16.67**]	$\alpha = 9.021$
db7 = [	0.509	0.417	0.421	0.712	1.086	0.444	0.636	0.979	**2.710**	**18.86**]	$\alpha = 9.580$
db10 = [	0.455	0.403	0.339	0.732	1.254	0.531	0.846	0.991	1.889	8.781]	$\alpha = N/A$
bior2.2 = [	0.496	0.571	0.688	0.870	1.445	1.014	1.455	1.644	**5.638**	**18.96**]	$\alpha = 9.705$
bior2.8 = [	0.496	0.589	0.665	0.825	1.552	0.602	0.760	1.908	**5.166**	**20.28**]	$\alpha = 9.735$
bior3.5 = [	0.381	0.654	0.653	1.004	2.128	1.163	1.888	2.240	3.183	9.867]	$\alpha = N/A$
bior6.8 = [	0.531	0.516	0.527	0.632	1.172	0.483	0.622	1.242	**3.104**	**18.33**]	$\alpha = 9.615$
coif1 = [	0.598	0.497	0.590	0.805	1.037	0.695	0.848	1.481	**6.245**	**21.18**]	$\alpha = 9.674$
coif3 = [	0.557	0.478	0.473	0.688	1.162	0.476	0.645	1.155	**3.342**	**20.27**]	$\alpha = 9.470$
coif5 = [	0.542	0.467	0.439	0.722	1.084	0.431	0.644	1.158	**2.779**	**19.29**]	$\alpha = 9.643$
sym2 = [	0.500	0.530	0.490	0.791	1.112	0.688	0.888	1.639	4.041	16.19]	$\alpha = N/A$
sym4 = [	0.559	0.539	0.488	0.778	1.031	0.579	0.683	1.150	**4.418**	**19.13**]	$\alpha = 9.273$
sym6 = [	0.549	0.463	0.504	0.703	1.114	0.505	0.673	1.097	**3.505**	**20.04**]	$\alpha = 9.396$
sym8 = [	0.542	0.488	0.499	0.640	1.154	0.459	0.638	1.137	**3.043**	**20.04**]	$\alpha = 9.538$

db1 = [	0.620	0.651	0.775	0.861	1.206	1.158	1.848	**3.700**	7.751	15.58]	$\alpha = 8.000$
db4 = [	0.530	0.505	0.705	0.713	**1.804**	0.852	0.556	1.591	**3.436**	**17.67**]	$\alpha = 9.194$
db7 = [	0.541	0.578	0.596	0.776	1.407	0.734	0.509	1.325	**3.218**	12.81]	$\alpha = 9.000$
db10 = [	0.514	0.417	0.607	0.920	1.532	0.635	0.409	1.166	**3.139**	10.68]	$\alpha = 9.000$
bior2.2 = [	0.514	0.831	0.933	1.157	1.785	1.950	1.394	1.759	**5.638**	12.27]	$\alpha = 9.000$
bior2.8 = [	0.514	0.757	0.846	1.126	1.717	1.167	0.861	1.394	**5.660**	13.35]	$\alpha = 9.000$
bior3.5 = [	0.414	0.697	1.086	1.262	2.121	**2.319**	1.883	1.679	**6.805**	7.630]	$\alpha = 8.931$
bior6.8 = [	0.550	0.658	0.700	0.839	1.528	0.858	0.550	1.170	**3.329**	**11.76**]	$\alpha = 9.038$
coif1 = [	0.631	0.787	0.788	1.023	1.304	1.079	0.769	2.056	**6.445**	**16.51**]	$\alpha = 9.039$
coif3 = [	0.574	0.690	0.670	0.986	1.435	0.767	0.535	1.290	**3.010**	**14.15**]	$\alpha = 9.100$
coif5 = [	0.560	0.654	0.590	0.777	1.432	0.555	0.495	1.272	**3.360**	**12.46**]	$\alpha = 9.099$
sym2 = [	0.625	0.601	0.617	0.848	1.251	0.980	0.823	2.017	**6.080**	13.79]	$\alpha = 9.000$
sym4 = [	0.599	0.705	0.890	0.825	1.413	0.943	0.500	1.394	**4.585**	**16.45**]	$\alpha = 9.102$
sym6 = [	0.588	0.692	0.711	0.981	1.392	0.791	0.503	1.330	**3.152**	**14.99**]	$\alpha = 9.147$
sym8 = [	0.581	0.650	0.670	0.848	1.483	0.664	0.512	1.268	**3.237**	**13.70**]	$\alpha = 9.126$

Table 4.3. Modulus maxima of the wavelet coefficients for motors with Fault C (coefficients in bold designate those above the corresponding threshold)

	Level										
	1	2	3	4	5	6	7	8	9	10	
db1 = [	0.717	0.820	0.753	0.767	0.901	0.590	0.880	1.505	4.141	9.426]	α = N/A
db4 = [	0.693	0.708	**0.803**	0.654	0.947	0.526	0.542	0.493	0.758	5.921]	α = 3.000
db7 = [	0.570	**0.722**	**0.651**	0.750	0.999	0.476	0.589	0.514	0.655	4.899]	α = 2.211
db10 = [	0.631	0.582	0.607	0.689	1.007	0.528	0.528	0.706	0.515	3.752]	α = N/A
bior2.2 = [	0.514	**0.939**	0.987	0.899	1.422	0.717	1.137	0.986	1.252	5.210]	α = 2.000
bior2.8 = [	0.514	**0.941**	**1.072**	0.863	1.508	0.510	0.703	0.749	1.064	5.567]	α = 2.585
bior3.5 = [	0.474	1.034	**1.240**	0.874	2.026	1.013	1.142	0.938	1.201	4.070]	α = 3.000
bior6.8 = [	0.557	**0.839**	**0.861**	0.733	1.062	0.451	0.581	0.551	0.648	4.844]	α = 2.619
coif1 = [	0.606	**0.906**	0.770	0.719	0.953	0.560	0.640	0.710	1.534	7.228]	α = 2.000
coif3 = [	0.576	**0.844**	**0.787**	0.725	0.978	0.447	0.569	0.508	0.649	5.395]	α = 2.231
coif5 = [	0.564	**0.813**	**0.723**	0.737	1.032	0.436	0.542	0.480	0.594	**4.681**]	α = 2.513
sym2 = [	0.623	0.801	0.747	0.725	0.916	0.547	0.639	0.757	1.788	6.332]	α = N/A
sym4 = [	0.547	**0.832**	**0.756**	0.680	0.972	0.576	0.565	0.480	0.778	5.795]	α = 2.196
sym6 = [	0.544	**0.824**	**0.758**	0.733	0.977	0.511	0.594	0.502	0.679	5.236]	α = 2.342
sym8 = [	0.545	**0.816**	**0.800**	0.732	0.961	0.445	0.571	0.493	0.632	5.053]	α = 2.488

db1 = [	**0.876**	0.826	0.730	0.692	0.799	0.457	0.872	1.588	4.166	10.03]	α = 1.000
db4 = [	**0.739**	0.788	**0.712**	0.592	0.852	0.351	0.429	0.442	0.904	5.300]	α = 1.432
db7 = [	**0.752**	**0.701**	**0.670**	0.628	0.899	0.313	0.454	0.403	0.747	5.274]	α = 1.756
db10 = [	**0.702**	0.622	0.588	0.630	0.911	0.291	0.413	0.413	0.668	4.759]	α = 1.000
bior2.2 = [	0.633	0.834	0.988	0.820	1.317	0.560	0.898	0.855	1.193	5.900]	α = N/A
bior2.8 = [	0.633	**0.876**	0.939	0.729	1.462	0.360	0.515	0.671	1.213	6.108]	α = 2.000
bior3.5 = [	**0.564**	**1.055**	1.146	0.761	1.949	0.609	1.066	0.965	1.068	2.854]	α = 1.292
bior6.8 = [	0.698	**0.779**	**0.763**	0.579	1.069	0.287	0.432	0.446	0.519	4.795]	α = 2.474
coif1 = [	0.734	**0.818**	0.796	0.668	0.891	0.412	0.482	0.620	1.756	7.519]	α = 2.000
coif3 = [	0.719	**0.773**	0.717	0.595	0.989	0.327	0.427	0.425	0.529	5.277]	α = 2.000
coif5 = [	0.712	**0.743**	**0.741**	0.613	0.937	0.265	0.394	0.414	0.604	3.722]	α = 2.607
sym2 = [	0.770	0.852	**0.858**	0.654	0.894	0.340	0.470	0.536	1.627	7.738]	α = 3.000
sym4 = [	**0.755**	**0.852**	0.722	0.583	0.939	0.439	0.459	0.473	0.664	5.862]	α = 1.870
sym6 = [	**0.753**	**0.784**	**0.733**	0.577	0.947	0.375	0.423	0.416	0.529	5.057]	α = 1.461
sym8 = [	**0.748**	**0.796**	0.722	0.580	0.986	0.290	0.412	0.414	0.546	4.896]	α = 1.686

Table 4.4. Modulus maxima of the wavelet coefficients for motors with Fault D (coefficients in bold designate those above the corresponding threshold)

	Level										
	1	2	3	4	5	6	7	8	9	10	
db1 = [	**0.815**	0.895	**1.049**	**1.290**	1.490	**1.823**	1.858	**3.490**	**8.637**	**22.10**]	$\alpha = 6.822$
db4 = [	0.659	0.751	**0.966**	0.895	1.707	**1.899**	0.662	1.088	2.300	**19.14**]	$\alpha = 6.308$
db7 = [	0.626	0.589	0.751	0.766	1.718	**1.441**	0.559	0.934	1.821	**14.70**]	$\alpha = 6.390$
db10 = [	0.550	0.571	**0.871**	0.701	1.694	**1.779**	0.661	0.899	2.017	10.76]	$\alpha = 5.719$
bior2.2 = [	0.589	0.785	1.197	**1.391**	2.070	**2.774**	1.856	2.307	4.260	**15.26**]	$\alpha = 7.122$
bior2.8 = [	0.589	0.808	1.125	**1.374**	**2.208**	**2.563**	0.880	1.426	4.283	**15.93**]	$\alpha = 6.376$
bior3.5 = [	0.531	0.957	1.210	**2.106**	2.030	**4.666**	1.633	2.303	3.266	**12.48**]	$\alpha = 5.765$
bior6.8 = [	0.624	0.697	0.887	0.988	**1.960**	**1.726**	0.608	0.984	2.059	**13.05**]	$\alpha = 6.462$
coif1 = [	0.721	0.715	1.039	**1.270**	**1.891**	**1.583**	0.884	1.380	5.381	**19.73**]	$\alpha = 6.441$
coif3 = [	0.658	0.655	0.811	0.901	**1.918**	**1.497**	0.574	0.917	2.025	**15.73**]	$\alpha = 6.623$
coif5 = [	0.639	0.631	0.867	0.973	**1.977**	**1.610**	0.538	0.960	1.782	9.057]	$\alpha = 5.901$
sym2 = [	0.635	0.794	0.921	**1.047**	**1.721**	**1.582**	0.927	1.842	**5.097**	**20.03**]	$\alpha = 6.587$
sym4 = [	0.691	0.710	0.822	**1.103**	**1.915**	**1.628**	0.647	0.949	2.834	**18.66**]	$\alpha = 6.684$
sym6 = [	0.670	0.671	0.757	0.888	**1.804**	**1.480**	0.576	0.874	2.032	**16.72**]	$\alpha = 7.084$
sym8 = [	0.657	0.686	**0.895**	0.893	**2.030**	**1.442**	0.563	0.916	1.862	**14.98**]	$\alpha = 6.592$

db1 = [	0.806	0.927	**1.182**	**1.299**	1.434	**1.567**	1.607	**3.443**	**9.605**	**21.10**]	$\alpha = 6.618$
db4 = [	0.664	0.826	**0.879**	0.875	**1.870**	**1.585**	0.542	0.734	2.324	13.74]	$\alpha = 5.895$
db7 = [	0.635	0.760	0.765	0.783	1.760	**1.398**	0.496	0.717	1.113	**16.65**]	$\alpha = 7.252$
db10 = [	0.606	**0.706**	**0.774**	0.868	**1.988**	**1.597**	0.482	0.692	1.046	**14.41**]	$\alpha = 5.821$
bior2.2 = [	0.606	0.949	1.207	1.259	**2.206**	**2.833**	1.697	2.057	2.974	**16.90**]	$\alpha = 7.558$
bior2.8 = [	0.606	0.851	**1.186**	**1.465**	1.933	**2.856**	0.668	1.389	2.551	**17.69**]	$\alpha = 6.532$
bior3.5 = [	0.507	1.055	1.311	**1.896**	2.272	**4.397**	1.708	2.238	2.571	8.141]	$\alpha = 5.364$
bior6.8 = [	0.639	0.756	**0.956**	**1.065**	1.792	**1.928**	0.495	0.685	1.247	**14.71**]	$\alpha = 6.649$
coif1 = [	0.749	0.927	0.993	**1.157**	**1.864**	**1.657**	0.866	1.436	3.962	**21.21**]	$\alpha = 7.089$
coif3 = [	0.675	0.856	0.879	1.017	**1.831**	**1.557**	0.522	0.673	1.299	**17.04**]	$\alpha = 7.017$
coif5 = [	0.653	0.820	0.897	0.918	**1.848**	**1.608**	0.459	0.690	1.130	7.976]	$\alpha = 5.957$
sym2 = [	0.741	0.902	0.970	0.938	1.657	**1.672**	0.701	1.420	**5.175**	**20.96**]	$\alpha = 7.220$
sym4 = [	0.705	0.752	0.898	0.931	**1.913**	**1.494**	0.680	0.724	1.753	**16.66**]	$\alpha = 6.379$
sym6 = [	0.683	0.869	0.909	1.002	**1.878**	**1.577**	0.570	0.658	1.300	**16.37**]	$\alpha = 6.778$
sym8 = [	0.669	0.722	**0.926**	0.964	**1.867**	**1.557**	0.462	0.675	1.204	**15.65**]	$\alpha = 6.709$

Table 4.5. Modulus maxima of the wavelet coefficients for motors with Fault E (coefficients in bold designate those above the corresponding threshold)

4.4 Minimum Distance Using the Normalized Modulus Maxima of the DWT

The third approach to the detection and classification problem was implemented on both the wiper and fuel pump motors. These motors are discussed in detail in Sections 5.1.2–5.1.3. The development of this algorithm was motivated by the desire to use a statistical framework rather than the deterministic framework from the algorithms previously implemented. In examining the modulus maxima of the wavelet coefficients from the motors shown in Tables 4.1–4.5, it can be seen that the coefficients from motors with the same fault are not exactly the same, however they follow the same general pattern. In the analysis, these coefficients were represented as ten-dimensional vectors.

In the detection part of the algorithm, it was considered that a fault may be present when one of the modulus maxima exceeded a threshold. The thresholds corresponding to each level of decomposition are defined by the maximum of those observed on the new motors provided for this work. A small number, ϵ, can be added to the corresponding threshold for each level of decomposition to decrease the sensitivity of the detection.

In the classification part of the algorithm the lengths of the vectors from each motor are normalized and the coefficients of the dimensions of the normalized vectors from faults of the same type are averaged. The resultant vectors serve as the centers of each fault cluster. Through this normalization, direction of the coefficient vector provided the determination for the type of fault. Experimental results show this improve the accuracy of the algorithm. It was also found that different faults cluster themselves with different variances, so a maximum radius is defined for the clusters for each type of fault. The classification strategy is then to find the fault, which is now represented by a point on a 10-dimensional unit sphere, having the minimum

Euclidean distance (3.4) from the vector representing the center of each fault cluster that the test motor is included in. If the normalized coefficient vector representing a test motor does not fall into one of the fault clusters, it is said that the motor does not have any of the known faults.

A two-dimensional example of this technique is shown in Figures 4.3 and 4.4. Here an attempt is made to classify the point δ (which belongs to cluster A) into either A or B. The circular shapes belong to cluster A and the square shapes belong to cluster B. The triangles represent the mean of the vectors from each cluster. If δ is said to belong to the cluster having the minimum Euclidean distance between δ and the cluster mean, δ will be classified as part of cluster B rather than cluster A. This is similar to how the motor faults manifest themselves in ten-dimensional space. Experimentation showed that the motor faults could be classified much more accurately by using a normalized Euclidean distance which is shown in Figure 4.4. It is clear, after normalization that δ is closer to the cluster mean of A.

If, however, δ were positioned slightly lower in the figure than it is, its normalized Euclidean distance would again classify it incorrectly within B. This is due to the fact that cluster A has higher variance than cluster B. To remedy this situation, a ball of radius ε was assigned to serve as a valid region for each cluster. This can be seen in Figure 4.4 where ε_1 is the valid region for A and ε_2 is the valid region for B. Each test motor was therefore classified by finding its minimum normalized Euclidean distance within a radius ε from the mean of each fault cluster. If the test motor did not fall within the radius ε from the mean of any of the fault clusters, it was classified as a good motor, or at least free of the faults which were being searched for.

Minimum angle could also be used in the algorithm instead of minimum distance. Since they are proportional, and nearly the same for small angles, this led to the same set of results. For this method, instead of a maximum radius for each cluster, a maximum angle was chosen. The angle between two vectors $\mathbf{x}$ and $\mathbf{y}$ is defined as

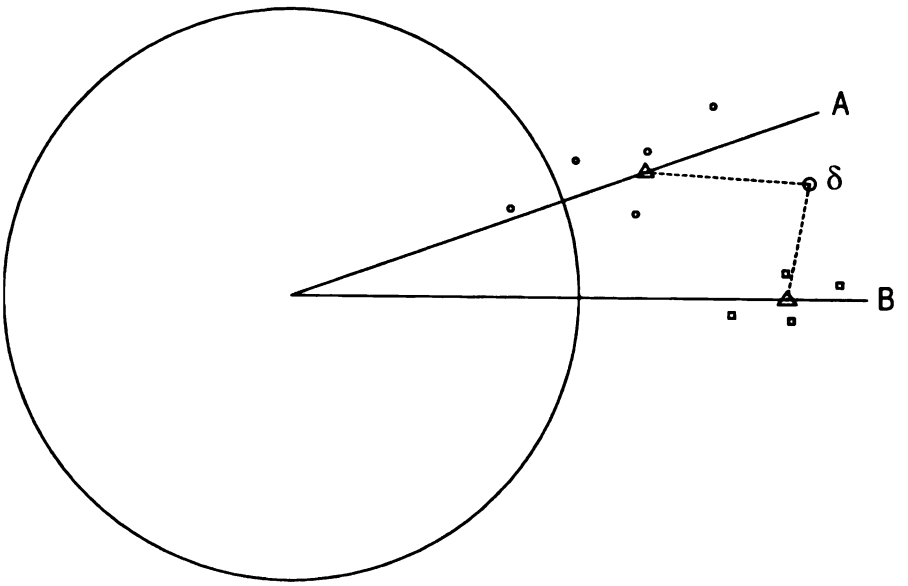

Figure 4.3. Euclidean distance for a non-normalized set of points

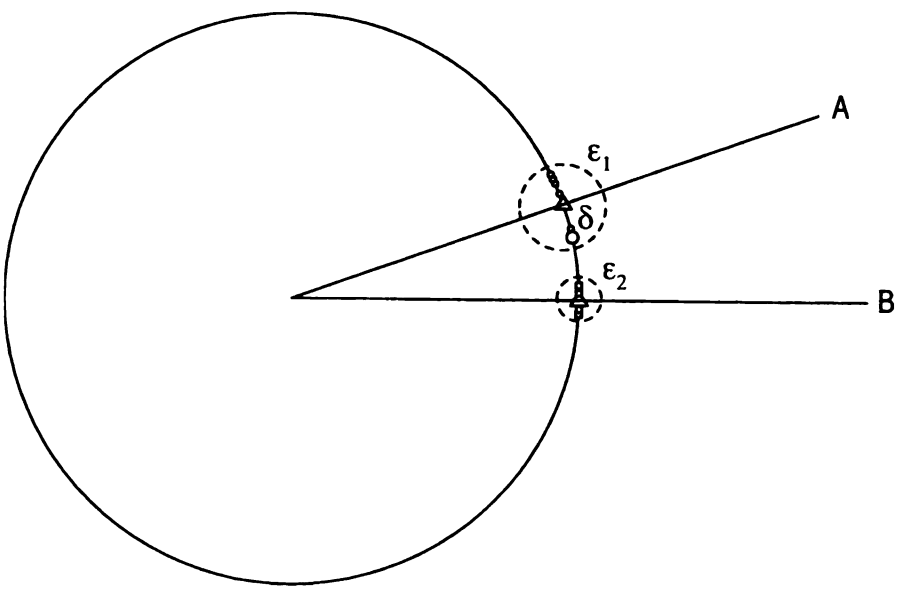

Figure 4.4. Euclidean distance for a normalized set of points

(4.3):

$$\cos\theta = \frac{\mathbf{x}^T\mathbf{y}}{\|\mathbf{x}\|\|\mathbf{y}\|}, \quad 0 \leq \theta \leq \pi \tag{4.3}$$

In the case where $\mathbf{x}$ and $\mathbf{y}$ are normalized, or very close to normalized, (4.3) simplifies to (4.4):

$$\cos\theta \approx \mathbf{x}^T\mathbf{y}, \quad 0 \leq \theta \leq \pi \tag{4.4}$$

Therefore, for small angles, both the cosine function and the inner product are at their maximum values. The use of minimum angle in the algorithm from this section led to the development of the algorithm in the following section using linear discriminant functions.

4.5 Linear Discriminant Functions

The fourth approach to the detection and classification problem was again implemented on both the wiper and fuel pump motors. This algorithm was developed in an effort to fine tune the previous minimum distance algorithm. The detection part of the algorithm remains the same. Thresholds are defined on the modulus maxima of the wavelet transform corresponding to the maximum of those observed on the new motors which were provided for this work. If any one of the modulus maxima of the coefficients from the wavelet transform of a test motor exceed these thresholds, it is considered that a fault may be present.

Instead of relying on minimum distance or minimum angle for the classification step, the maximum linear discriminant was used. A minimum discriminant was also defined for each cluster instead of the maximum distance or maximum angle defined in the previous algorithm. If the test motor did not fall into one of the fault clusters, it was said that it did not have any of the known faults.

The introduction of a training procedure made the classification algorithm more

consistent. The initial guess for the weighting coefficients of each class was defined to be the mean of all of the known sample vectors in that class. A class in this case refers to a particular fault. The initial weighting coefficients are identical to the averaged normalized vectors from faults of the same type from the classification step in the previous algorithm. These coefficients were then adjusted until all of the test motors with known faults were correctly classified. A gain constant of $\alpha = 0.01$ was used to keep adjustments of the weighting coefficients small. With this method, any mother wavelet could be used to achieve perfect classification of the known test motors. The weighting coefficients from the mother wavelet that required the fewest number of corrections to converge were used in the algorithm.

Without the introduction of the training procedure for this method, the results would be the same as the results from the previous algorithm, which made decisions based on minimum distance or minimum angle. This is because the discriminant function is proportional to the cosine of the angle which was used in the previous algorithm. The cosine has its maximum for an angle of 0°, which relates to the case for minimum distance when the distance is zero.

The training procedure, however, could have undesirable effects if one of the motors specified as part of a particular class is considerably different than the others, possibly having multiple faults, or being an outlier in some other way. The weighting coefficients can adjust themselves so that each cluster is much larger than it would have been in the previous algorithm in order to accommodate all of its members.

CHAPTER 5

Experimental Setup and Results

5.1 Experimental Setup

5.1.1 HVAC Fan Motor Experimental Setup

The experimental setup in the HVAC fan motor testing consisted of an HVAC fan motor, a PC running Real-Time Linux (RT-Linux), and a 12-bit A/D board. The motor was fed by a standard automobile battery and it was loaded by the squirrel cage fan that it is normally coupled to in an automobile. Both voltage and current were simultaneously sampled from the motor at a frequency of approximately 16kHz. Data was recorded in the computer and MATLAB was then used for post-experimental analysis. The experimental setup is shown in Figure 5.1.

In this experiment, it was unknown what physical abnormalities were present in each motor. It was only known that the motors were removed from vehicles for warranty reasons and were faulty. After some analysis of the data obtained from them, two specific types of signatures from the current waveforms were recognized. Some of the data had faults where the current abruptly increased or decreased, remained constant for a short period of time and then quickly returned back to the original state. This was classified as a step fault. Sample data from a motor with a step fault is shown in Figure 5.2.

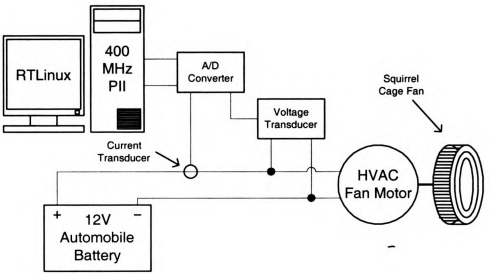

Figure 5.1. HVAC fan motor experimental setup

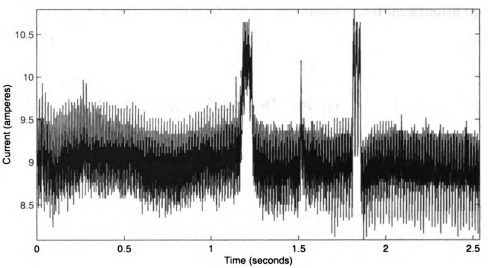

Figure 5.2. Fan motor exhibiting step discontinuities

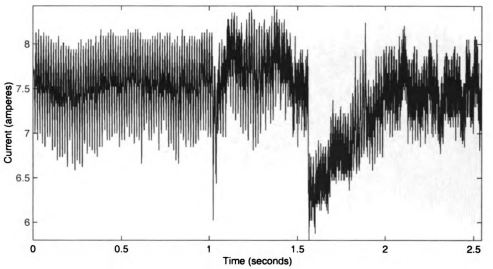

Figure 5.3. Fan motor exhibiting ramp discontinuities

Other motors demonstrated faults where the current abruptly increased or decreased and then slowly ramped back to the original state. These were classified as ramp faults. Sample data from a motor with a ramp fault is shown in Figure 5.3.

In many cases, the current stayed more or less steady through the experiment. These motors were classified as normal. The current waveform from a motor operating normally is in Figure 5.4.

5.1.2 Wiper Motor Experimental Setup

The experimental setup in the wiper motor testing consisted of a wiper motor, a PC running Real-Time Linux (RT-Linux), a 12-bit A/D board and an 8-bit D/A board. The wiper motor has a 50:1 gear ratio and contains 12 commutator bars/slots. It was powered by a standard automobile battery through a controller and was loaded by a brushless DC motor. The RT-Linux system was used to control the torque output of the brushless DC motor. Torque profiles of the wiper system under different environmental conditions were constructed. Torque profiles were developed for the

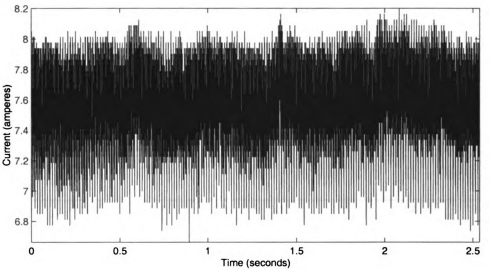

Figure 5.4. Fan motor operating normally

wiper motor at high speed on dry glass (Figure 5.5), at high speed on wet glass (Figure 5.6), at low speed on dry glass (Figure 5.7) and at low speed on wet glass (Figure 5.8). Using the simulated load offered much greater flexibility in the experiment. Voltage, current, torque and speed were simultaneously sampled from the motor and drive at a frequency of 12.5kHz. Data was recorded in the computer and MATLAB was then used for post-experimental analysis. The experimental setup is shown in Figure 5.9.

To develop and test various detection and classification algorithms, both new windshield wiper motors as well as motors that were manufactured to have specific faults known to significantly shorten their lives were analyzed. The faults are referred to by capital letters throughout this thesis. Fault A refers to a condition where one of the springs that normally keep the brushes in contact with the commutator face has become stuck due to excess sealant applied during assembly. This sealant is used to seal the motor's housing so water cannot enter. Fault B indicates a condition where one of the springs that keep the brushes in contact with the commutator face has been kinked at some point during assembly. Fault C refers to the condition where

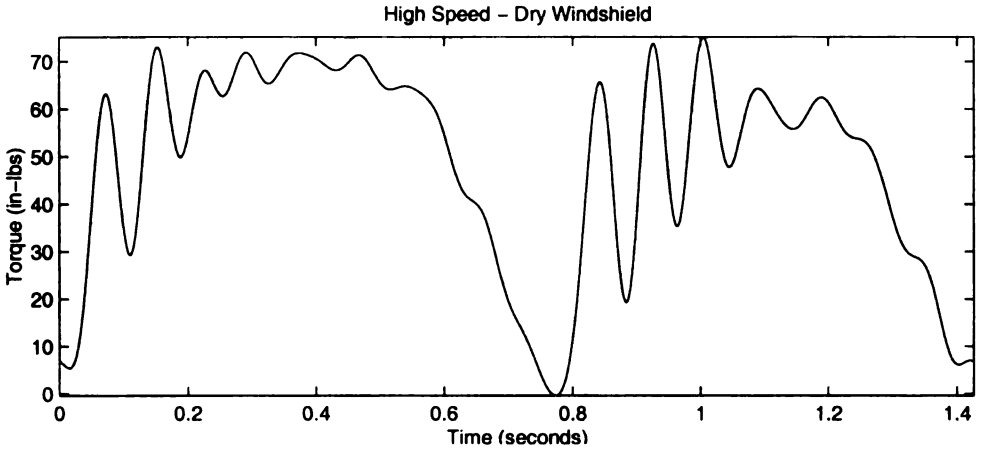

Figure 5.5. Wiper motor profile for high speed operation on dry glass

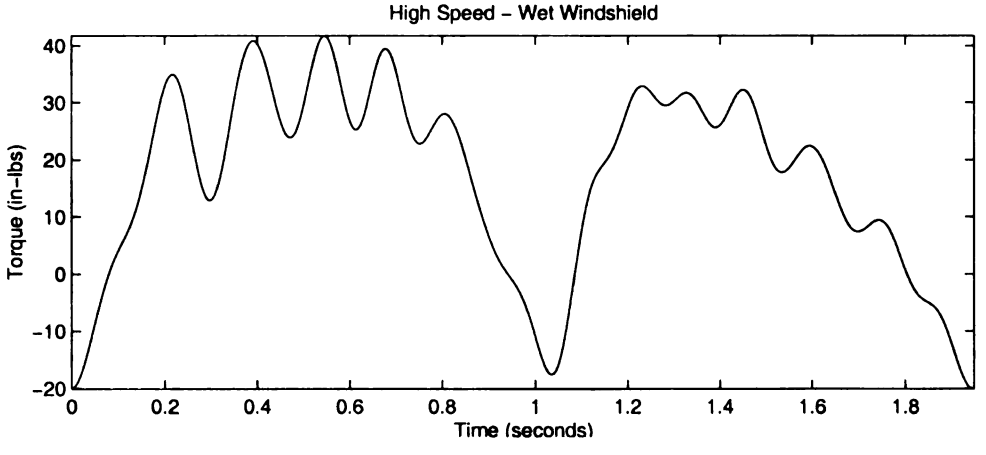

Figure 5.6. Wiper motor profile for high speed operation on wet glass

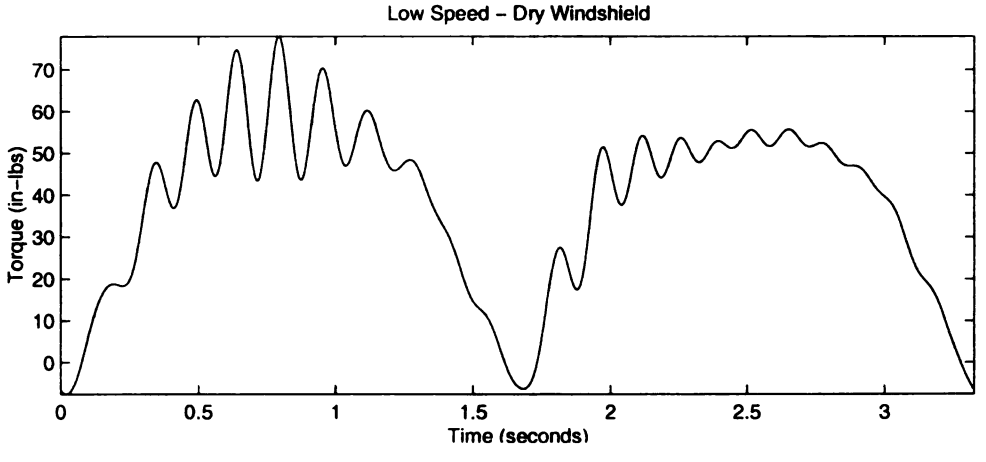

Figure 5.7. Wiper motor profile for low speed operation on dry glass

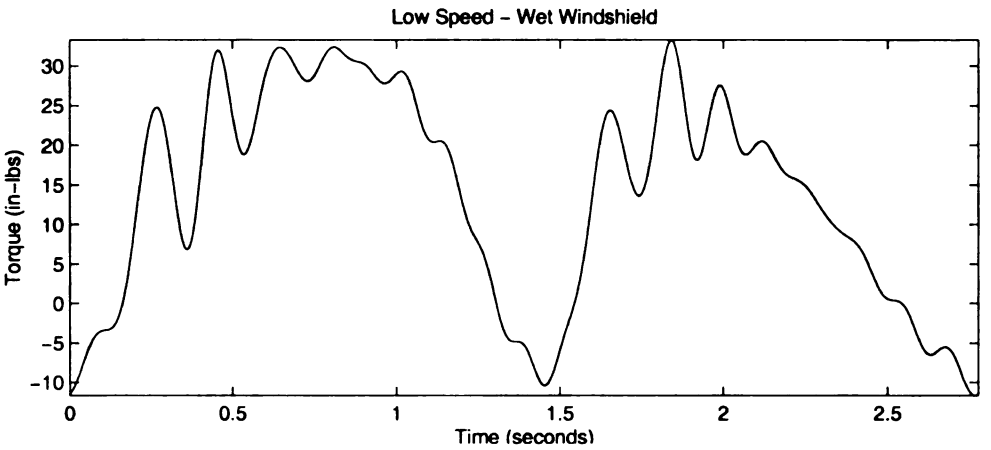

Figure 5.8. Wiper motor profile for low speed operation on wet glass

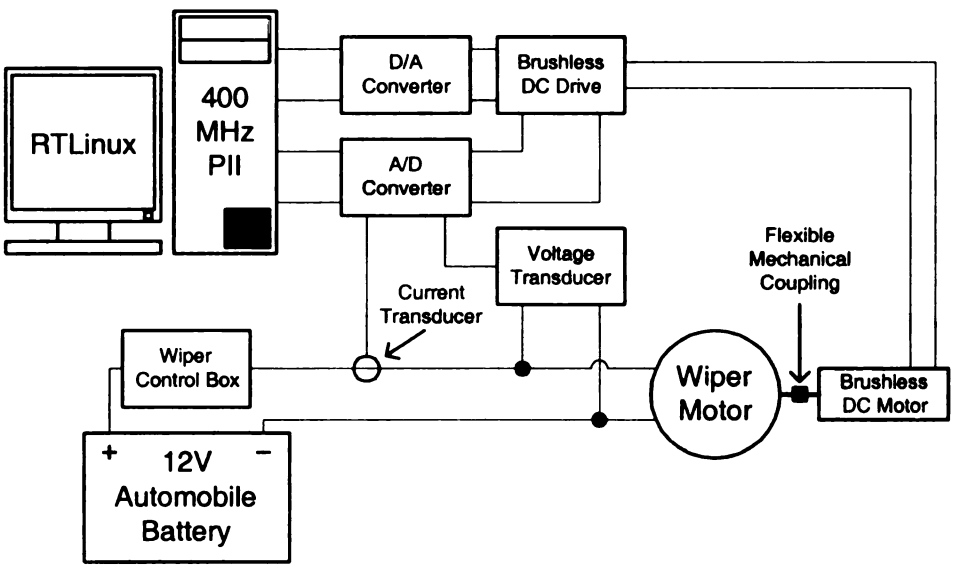

Figure 5.9. Wiper motor experimental setup

one gear tooth is removed from the 50:1 gear reduction mechanism. Fault D indicates shaft misalignment due to the failure to install a bushing during assembly. Faults E_1, E_2 and E_3 were due to increased friction. These were grouped together because of their similarity. Motors with Fault E_1 had no gear grease applied during assembly. In the case of Fault E_2, no thrust ball grease was applied during assembly and Fault E_3 refers to the application of some thrust ball grease during assembly, however less than what the assembly specification calls for. Fault F indicates that the cardboard/copper disk located in the plastic gear cover which is used by the motor to locate the correct parking position has been "punched" at some point during assembly causing the copper to be raised slightly which results in the motor running through the park position for one or more revolutions.

5.1.3 Fuel Pump Experimental Setup

The experimental setup for the fuel pump motor testing consisted of a fuel pump motor, a PC running Real-Time Linux (RT-Linux), and a 12-bit A/D board. Testing was performed in an enclosure which was resistant to the test solvent which was used as a fuel substitute. The motor was fed by a standard automobile battery. Pressure was monitored by a gauge and adjusted by a valve in the fuel line. The solvent was filtered at both the input to the test motor and prior to reentry into the test enclosure. Both voltage and current were simultaneously sampled from the motor at a frequency of approximately 16.7kHz. Data was recorded in the computer and MATLAB was then used for post-experimental analysis. The experimental setup is shown in Figure 5.10.

As with the wiper motors, both new fuel pump motors as well as motors that were manufactured to have specific faults known to significantly shorten their lives were analyzed. The faults are referred to by capital letters throughout this thesis. Faults G and H both refer to cases where the resistance in one coil is above the specification.

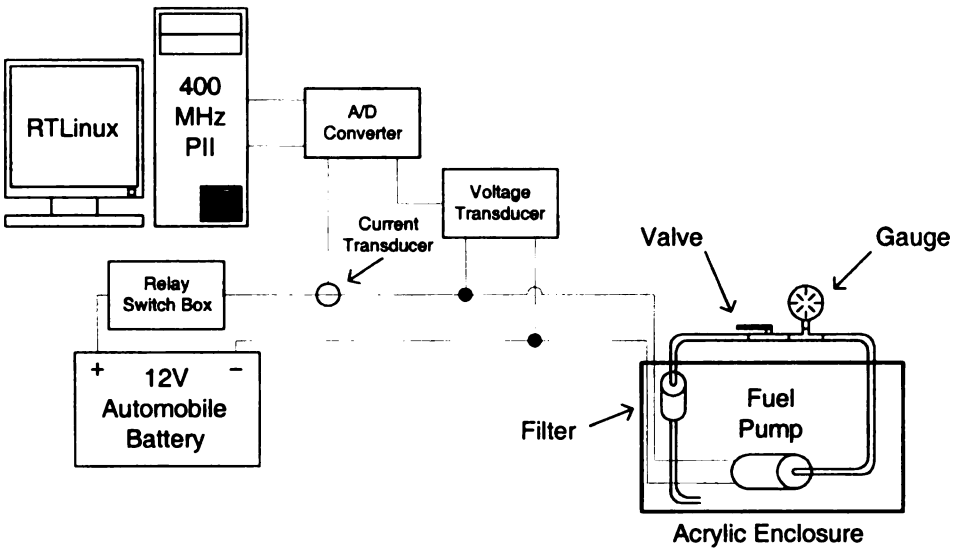

Figure 5.10. Fuel pump experimental setup

In the case of Fault G, one of the coils is poorly fused to the commutator causing its resistance to be increased and in the case of Fault H, the coil is cut entirely making its resistance infinite. Fault I indicates that the commutator face was scored during assembly.

5.2 Experimental Results

5.2.1 Discrete Wavelet Transform

Results from testing show that this algorithm was accurate in that it was not only able to detect that a fault was present, but it could also determine which type it was. We can see, however, that the first threshold from the level 9 decomposition can be modified to make the system more or less sensitive to detecting a fault condition. The second threshold from the level 11 decomposition can also be modified to make the system lean more or less toward a specific type of discontinuity. Performing analysis directly on the DWT coefficients is advantageous in that it is possible to localize the

faults in time, however it is more computationally intensive than the strategies that follow since the number of coefficients is very large.

In the first example with the step discontinuity (Figure 5.2), the algorithm detected three step faults at 1.22s, 1.53s and 1.81s. One of the remarkable qualities of the DWT is that it makes it possible to detect irregularities with different time-frequency characteristics. This is clear in this example where all three faults were considerably different from one another.

In the second example with the ramp discontinuity (Figure 5.3), the algorithm detected three ramp faults at 1.56s, 2.28s and 2.44s. There are clearly ramp type discontinuities at these time periods, however the other fault in the signal just after 1 second was not detected. Perhaps this was because the beginning of the discontinuity was not abrupt enough to satisfy the first criterion in the algorithm. In this case, the threshold for the level 9 decomposition could be lowered or a different mother wavelet or different level of decomposition could be used in the analysis to make the algorithm more sensitive to this type of fault.

Finally, in the last example with the normally operating motor (Figure 5.4), the algorithm did not detect any irregularities in the signal. The signal essentially remained at steady state.

In developing an algorithm using the DWT coefficients directly, the mother wavelet selection is highly deterministic and is therefore a very crucial step. As the signal processing techniques become more sophisticated in the following sections, the procedure for choosing a mother wavelet is based on more statistical criteria.

5.2.2 Modulus Maxima of the DWT

In the implementation of the second algorithm, many observations can be made from the results shown in Tables 4.1–4.5. The algorithm was developed based on which coefficients exceeded their corresponding thresholds as well as the value of the local-

ization parameter from decompositions using specific mother wavelets.

It can be observed that the modulus maxima from the motors with Fault A exceeded their corresponding thresholds among the mid-to-high levels of decomposition. The average value of α, the localization parameter, from both motors over the fifteen mother wavelets was 6.767. The modulus maxima from the motors with Fault B were greater than their corresponding thresholds for almost all levels of decomposition. The average value of α in this case was 7.490. The modulus maxima from the motors with Fault C exceeding their corresponding thresholds distributed themselves over the high levels of decomposition. The average value of α was 9.153. The modulus maxima from the motors with Fault D were above their corresponding thresholds for the low levels of decomposition. The average value of α was 2.106. The modulus maxima from the motors with Fault E exceeded their corresponding thresholds for the middle levels as well as the highest levels of decomposition but with a gap in between. The average value of α for this fault was 6.524. It is clear from examination of these results how a decision tree based on a series of if-then-else tests was developed to correctly detect and classify faults.

Performing analysis on the modulus maxima of the DWT coefficients was advantageous in that the algorithm was far less computationally intensive than the previous algorithm using the DWT coefficients directly.

5.2.3 Minimum Distance Using the Normalized Modulus Maxima of the DWT

To measure the quality of this and the following clustering methods, the ratio of the average cluster radius to the average distance between clusters is used. An illustration of this is shown in Figure 5.11 where A and B are clusters, x and z are the radii of clusters A and B respectively, and y is the distance between the center of clusters A and B. All measurements are made on the unit circle. The ratio in this case would

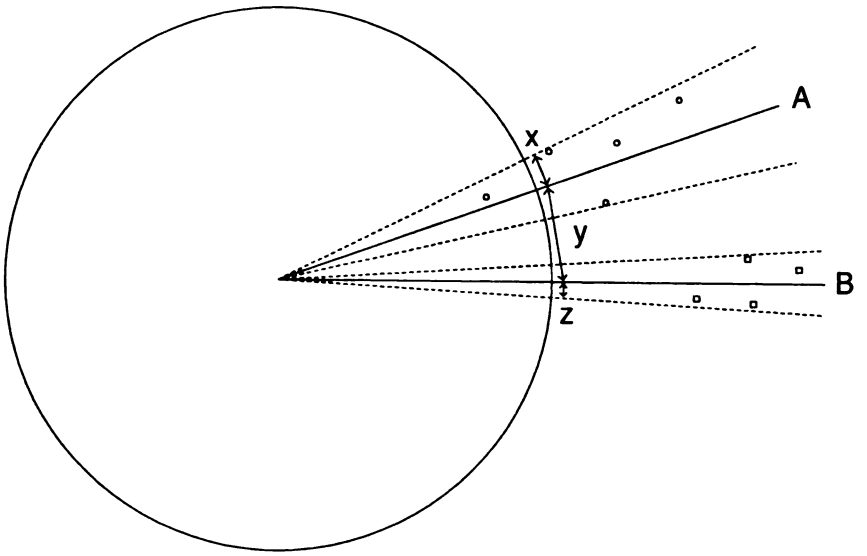

Figure 5.11. Ratio of average cluster radius to average distance between clusters

be defined as in (5.1):

$$\text{ratio} = 1 : \frac{y}{\frac{x+z}{2}} \tag{5.1}$$

For the wiper motor analysis, the ratio was 1 : 3.378, and for the fuel pump motor analysis, the ratio was 1 : 0.703. In any case, a higher ratio indicates more closely spaced points and better separation between clusters.

The only parameters required by the algorithm were the modulus maxima of the coefficients from the decomposition using the Biorthogonal 3.5 mother wavelet. For the windshield wiper motors, data was only required from low speed dry windshield testing and for the fuel pump motors, data was only used from testing at 250 kPa. This was one-fourth the amount of data that was required for the if-then-else approach from the second algorithm.

The application of a preliminary weight to each dimension prior to analysis was experimented with to bring the magnitudes from each dimension closer together. This extra mapping step, however, did not improve the performance of the classification algorithm so this step was not kept in the final version of the algorithm.

	Level									
	1	2	3	4	5	6	7	8	9	10
A = [	0.1848	0.1562	0.1648	0.3548	0.3732	0.1964	0.3011	0.4758	0.3022	0.4497]
B = [	0.0107	0.0153	0.0245	0.0421	0.0873	0.1437	0.4117	0.7108	0.5421	0.0192]
C = [	0.2884	0.2386	0.1600	0.4569	0.3907	0.1676	0.1971	0.2821	0.2587	0.5094]
D = [	0.3709	0.3164	0.2294	0.4330	0.3438	0.1638	0.2007	0.2381	0.2861	0.4454]
E_1 = [	0.2730	0.2530	0.1856	0.4299	0.4479	0.1426	0.2241	0.3038	0.3303	0.4119]
E_2 = [	0.2615	0.2200	0.1888	0.3703	0.3612	0.1514	0.1590	0.4520	0.3956	0.4135]
E_3 = [	0.2291	0.1556	0.1913	0.4550	0.3319	0.1423	0.3582	0.2263	0.2940	0.5323]
F = [	0.2453	0.1838	0.1668	0.4383	0.4144	0.2028	0.3309	0.3532	0.2959	0.3892]

Table 5.1. Initial Weighting Coefficients for Wiper Motor Testing

The addition of a second normalization step on the fault cluster centers after the averaging step was also experimented with to maintain unit length. Results from analysis, however, show that this technique did not improve overall performance of the classification algorithm.

5.2.4 Linear Discriminant Functions

In Table 5.1, the initial weighting coefficients for the wiper motors from low speed testing on a wet windshield are shown. These are the average values of the modulus maxima from the DWT coefficients on all of the motors for each fault using the Coiflet 30 mother wavelet. This mother wavelet was chosen because it required only 401 corrections, the least among all mother wavelets tested, to converge using the training algorithm described in Section 3.2. The weighting coefficients after training are shown in Table 5.2.

In Table 5.3, the initial weighting coefficients for the fuel pump motors tested at 310kPa are shown. These are the average values of the modulus maxima from the DWT coefficients on all of the motors for each fault using the Biorthogonal 2.2 mother wavelet. Using this mother wavelet, 1136 adjustments were required for the training algorithm to converge. The weighting coefficients after training are shown in Table 5.4.

	Level									
	1	2	3	4	5	6	7	8	9	10
A = [	0.1820	0.1539	0.1777	0.3450	0.3707	0.2018	0.2810	0.4821	0.3189	0.4556]
B = [	0.0107	0.0153	0.0245	0.0421	0.0873	0.1437	0.4117	0.7108	0.5421	0.0192]
C = [	0.2986	0.2019	0.1352	0.4930	0.4121	0.1688	0.2182	0.2742	0.2210	0.4989]
D = [	0.3581	0.3621	0.2401	0.4148	0.3170	0.1663	0.2008	0.2771	0.3053	0.4352]
E_1 = [	0.2679	0.2500	0.1755	0.4186	0.4345	0.1227	0.2087	0.2883	0.3550	0.4674]
E_2 = [	0.2615	0.2200	0.1888	0.3703	0.3612	0.1514	0.1590	0.4520	0.3956	0.4135]
E_3 = [	0.2274	0.1538	0.1906	0.4518	0.3280	0.1410	0.3557	0.2223	0.2911	0.5260]
F = [	0.2574	0.1819	0.1788	0.4447	0.4397	0.2149	0.3477	0.3354	0.2759	0.3546]

Table 5.2. Adjusted Weighting Coefficients for Wiper Motor Testing

	Level									
	1	2	3	4	5	6	7	8	9	10
G = [	0.0857	0.1507	0.2844	0.5170	0.2709	0.4351	0.4479	0.2162	0.2680	0.1997]
H = [	0.0838	0.1392	0.2506	0.3626	0.4191	0.4194	0.4484	0.3359	0.3157	0.1191]
I = [	0.1040	0.1613	0.2870	0.5510	0.2925	0.4231	0.4920	0.1214	0.1595	0.1741]

Table 5.3. Initial Weighting Coefficients for Fuel Pump Testing

	Level									
	1	2	3	4	5	6	7	8	9	10
G = [	0.0573	0.1369	0.2810	0.4857	0.3223	0.4388	0.4858	0.1902	0.2593	0.1682]
H = [	0.0721	0.1280	0.2714	0.4445	0.3383	0.4282	0.3789	0.4050	0.3205	0.1227]
I = [	0.1443	0.1863	0.2696	0.5003	0.3220	0.4107	0.5236	0.0784	0.1635	0.2020]

Table 5.4. Adjusted Weighting Coefficients for Fuel Pump Testing

For this algorithm, the same measure of quality used for the previous algorithm is used. The ratio of the average cluster radius to the average distance between clusters for the wiper motor analysis is 1 : 2.599 and the ratio for the fuel pump motor analysis is 1 : 0.768. Using the previous algorithm without training on the same set of coefficients, the ratios were 1 : 2.643 and 1 : 0.773 for the wiper motor analysis and fuel pump analysis respectively. Although the training procedure can increase the number of data points classified correctly, it is also shown to increase the average cluster size.

The choice to use the mother wavelet requiring the fewest number of corrections helps to assure that outliers do not exist in the data. In the case of the Coiflet 30 wiper motor coefficients from the low speed wet windshield testing in this section (Tables 5.1 and 5.2), only slight adjustments were made causing the average cluster radius to increase from 0.1531 to 0.1627 or 6.27% after 401 corrections. Cluster radii are measured using the Euclidean metric (3.4). In the case of the Biorthogonal 2.2 fuel pump coefficients from the 310 kPa testing in this section (Tables 5.3 and 5.4), the average cluster radius increased from 0.3383 to 0.3726 or 10.14% after 1136 corrections. The adjustments made to the coefficients are reasonable and improved the performance of the algorithm.

It can be shown by applying the same procedure to the weighting coefficients from the work in Section 4.4 that the training algorithm does not provide a benefit in all cases. In the case of the Biorthogonal 3.5 wiper motor coefficients from the low speed dry windshield testing, considerable adjustments were required causing the average cluster radius to increase from 0.0744 to 0.2751 or 269.76% after 7273 corrections. In the case of the Biorthogonal 3.5 fuel pump coefficients, the average cluster radius increased from 0.3743 to 0.6616 or 76.76% after 12172 corrections. Although the test motors may still be classified correctly in this case, the validity of the clusters should be questioned after such a significant adjustment from the initial weighting

coefficients. It is likely that at least one outlying data point is present.

CHAPTER 6

Conclusions

The objective of this work was to give a prognosis for the failure of DC motors used in automotive applications and to achieve this goal using a wavelet-based approach. This was accomplished by attempting to detect different faults that lead to a shorter overall life of the motor. The ability to classify the faults into different categories was another objective in this work, that was also met.

The results from this work can be applied to the development of a fault prognosis system as well. In this type of system, the goal would be to not only detect the presence of a fault and correctly classify it, but also determine the severity of it. This could be achieved by having test motors with different degrees of severity of each particular fault. The same techniques used in this work could then be applied, however in terms of the work done for this thesis, faults with different degrees of severity would be considered as separate faults altogether. Then the goal would again be to correctly detect and classify all of the known faults, however many of the faults we would be searching for would actually be different degrees of severity of a smaller set of faults. In this manner, one could monitor the cumulative degradation of a system.

For the prognosis of Fault X, this fault would be divided into three separate faults, X_1, X_2 and X_3. Fault X_1 would be defined as a motor in the earliest detectable stages.

Motors with Fault X_1 would not show much if any performance degradation compared to motors considered to be free of faults. Fault X_2 would be defined as a motor with a moderate case of the fault. Motors with Fault X_2 may show a slight decrease in performance compared to motors considered to be free of faults. Fault X_3 would be defined as a motor in the final stages before failure having the fault. These motors would have a severe degradation in performance compared to motors considered to be free of faults and most likely a severe impact on the system they are interacting with as well. In being able to properly detect and classify motors with Faults X_1, X_2 and X_3, one would be able to give an accurate prognosis for Fault X.

Having this type of information about a variety of faults could be described as a state of health prognosis. With prior knowledge of the typical longevity of motors having each of the faults with different degrees of severity, an idea of the time to fail for a motor can be given as well.

There are some issues that could be explored in the future related to this work. One possibility for future work could be the implementation of a fault prognosis system as described above, in a dedicated system, or a system dedicated to a different task but with available time.

Further research in terms of clustering methodologies could be explored as well. The application of a set of initial weights to each of the test motors may be useful if it could be used to minimize the size of each fault cluster, while at the same time maximize the distance between the vectors, or weighting coefficients, representing the center of each fault cluster.

Research in the area of nonlinear discriminant functions could be explored in the future as well. It is possible that performance in terms of the classification between different types of faults could be improved, especially if work with faults having varying degrees of severity is attempted.

Future work could be done in applying the techniques developed in this work to

a problem where a much greater number of test motors would be available. In this type of problem, issues such as the existence and removal of outliers from the test data as well as consequences arising from having a significantly greater number of test vectors could be explored.

Finally, future work could be done to implement a system capable of detecting and classifying faults in more than one motor connected to a node. With the additive nature of the current, new issues regarding the detection and classification of faults as well the additional step of determining which motor is responsible for a fault would have to be researched. In this case, current would only be measured at the node instead of at each motor individually, so the overall cost of hardware would be reduced.

APPENDICES

APPENDIX A

MATLAB Tools

All computational analysis in this research was done using MATLAB. The MathWorks' Wavelet Toolbox [8] was used initially and custom MATLAB functions followed for increased flexibility.

The main problem with the Wavelet Toolbox is that in performing a discrete wavelet transform some form of signal extension is required. The options given are zero padding, boundary value replication, first order smooth padding, smooth padding of order zero, and periodic padding. This is to assure that the edges of the signal being analyzed are not cut off in the downsampling operation. Unfortunately, this also distorts the Discrete Wavelet Transform (DWT) coefficients at the edges of the analysis. To avoid this problem, we developed a DWT algorithm that does not pad the edges of the signal. Using this method, to prevent the loss of information at the edges of the signal, care must be taken in selecting both the length of the signal as well as the length of the mother wavelet to be used for analysis. Prior to the discovery of this problem, the edge effects made it difficult to form conclusions from the DWT coefficients.

The DWT algorithm is based on the filter bank theory discussed in Section 2.4. A series of convolutions with both lowpass and highpass filters are implemented to get the scaling and wavelet coefficients for the next level of decomposition respectively

and a downsampling operation on each set of output coefficients follows.

A few examples of our MATLAB code are included. First, the detection algorithm for the HVAC fan motors (fault5.m) is listed. This was written before we were aware of the edge effect problems in the toolbox and could be revised to eliminate this problem. Our wiper motor and fuel pump testing algorithms were created after we resolved these issues. Second, our DWT algorithm (waveanalyze.m) is listed. This algorithm performs all of the convolutions and downsampling necessary for the DWT and does not rely on the wavelet toolbox except to get the filter coefficients. Third, the DWT summary algorithm (wavesummarize.m) is listed. This function is used to summarize the maximum of the absolute value of the DWT coefficients using various mother wavelets at the first ten levels of decomposition. This is helpful in choosing threshold values in fault analysis.

```
% Function:       fault5
% Author:         Wes Zanardelli
% Last Modified:  02/10/99
%
% Usage:
%   fault5(y)
%
%     y=original signal
%
%   example:  fault5(y)
function fault5(y)
y=y(:,3);
yflip=flipud(y);
y=[y;yflip];
[c1,l1]=wavedec(y,9,'bior1.3');
[c2,l2]=wavedec(y,11,'bior1.3');
d1=detcoef(c1,l1,9);
d1=abs(d1(1:fix(length(d1)/2)));
d2=detcoef(c2,l2,11);
d2=d2(1:fix(length(d2)/2));
flag=0;
for n=1:1:length(d1),
   if flag>0,
      flag=flag-1;
   end;
```

```
if(d1(n)>=2.75)
   if((flag==0)&(length(d2)>=(fix(n/4)+2))&
      (abs(d2(fix(n/4)+2)>=0)))
      flag=5;
      if(d2(fix(n/4)+2)>5)
         fprintf('\nThere is a +2 step discontinuity near
                 %.2f seconds',n*2.5/80);
      else
         fprintf('\nThere is a +2 ramp discontinuity near
                 %.2f seconds',n*2.5/80);
      end;
   end;
   if((flag==0)&(length(d2)>=(fix(n/4)+3))&
      (abs(d2(fix(n/4)+3)>=0)))
      flag=5;
      if(d2(fix(n/4)+3)>0)
         fprintf('\nThere is a +3 step discontinuity near
                 %.2f seconds',n*2.5/80);
      else
         fprintf('\nThere is a +3 ramp discontinuity near
                 %.2f seconds',n*2.5/80);
      end;
   end;
   if((flag==0)&(length(d2)>=(fix(n/4)+4))&
      (abs(d2(fix(n/4)+4)>=0)))
      flag=5;
      if(d2(fix(n/4)+4)>0)
         fprintf('\nThere is a +4 step discontinuity near
                 %.2f seconds',n*2.5/80);
      else
         fprintf('\nThere is a +4 ramp discontinuity near
                 %.2f seconds',n*2.5/80);
      end;
   end;
   if((flag==0)&(length(d2)>=(fix(n/4)+1))&
      (abs(d2(fix(n/4)+1)>=0)))
      flag=5;
      if(d2(fix(n/4)+1)>3)
         fprintf('\nThere is a +1 step discontinuity near
                 %.2f seconds',n*2.5/80);
      else
         fprintf('\nThere is a +1 ramp discontinuity near
                 %.2f seconds',n*2.5/80);
      end;
   end;
   if((flag==0)&(n>=4)&(abs(d2(fix(n/4))>=0)))
      flag=5;
      if(d2(fix(n/4))>2)
```

```
        fprintf('\nThere is a 0 step discontinuity near
                %.2f seconds',n*2.5/80);
      else
        fprintf('\nThere is a 0 ramp discontinuity near
                %.2f seconds',n*2.5/80);
      end;
    end;
    if((flag==0)&(n>=8)&(abs(d2(fix(n/4)-1))>=0))
      flag=5;
      if(d2(fix(n/4)-1)>1)
        fprintf('\nThere is a -1 step discontinuity near
                %.2f seconds',n*2.5/80);
      else
        fprintf('\nThere is a -1 ramp discontinuity near
                %.2f seconds',n*2.5/80);
      end;
    end;
    if((flag==0)&(n>=12)&(abs(d2(fix(n/4)-2))>=0))
      flag=5;
      if(d2(fix(n/4)-2)>1)
        fprintf('\nThere is a -2 step discontinuity near
                %.2f seconds',n*2.5/80);
      else
        fprintf('\nThere is a -2 ramp discontinuity near
                %.2f seconds',n*2.5/80);
      end;
    end;
    if((flag==0)&(n>=16)&(abs(d2(fix(n/4)-3))>=0))
      flag=5;
      if(d2(fix(n/4)-3)>1)
        fprintf('\nThere is a -3 step discontinuity near
                %.2f seconds',n*2.5/80);
      else
        fprintf('\nThere is a -3 ramp discontinuity near
                %.2f seconds',n*2.5/80);
      end;
    end;
    if((flag==0)&(n>=20)&(abs(d2(fix(n/4)-4))>=0))
      flag=5;
      if(d2(fix(n/4)-4)>1)
        fprintf('\nThere is a -4 step discontinuity near
                %.2f seconds',n*2.5/80);
      else
        fprintf('\nThere is a -4 ramp discontinuity near
                %.2f seconds',n*2.5/80);
      end;
    end;
end;
```

```
end;
fprintf('\nAnalysis is complete\n\n');



% Function:       waveanalyze
% Author:         Wes Zanardelli
% Last Modified:  10/05/99
%
% Usage:
%   waveanalyze(y,'wavelet',min,max,ymax,'identifier')
%
%     y=original signal
%     wavelet=predefined mother wavelet
%     min=first displayed scale
%     max=last displayed scale
%     ymax=maximum y-axis value
%     identifier=identifier for motor being tested
%
%   example:  waveanalyze(y,'coif1',1,10,20,'N1')
function waveanalyze(y,wavelet,min,max,ymax,identifier) y=y(:,3)';
subplot(max+2-min,2,1:2),plot(y); title(strcat('Motor #',identifier,' -
Decomposition using the ''',wavelet,''' Mother Wavelet'));
[lo_d,hi_d]=wfilters(wavelet,'d'); for n=1:max,
   % Convolution computed without the zero-padded edges
   d=conv2(y,hi_d,'valid');
   y=conv2(y,lo_d,'valid');
   % Dyadic downsampling
   d=d(2:2:length(d));
   y=y(2:2:length(y));
   if((n>=min)&(n<=max))
      % Save desired levels of decomposition to base workspace
      assignin('base',strcat('d',num2str(n)),d);
      subplot(max+2-min,2,2*(max-n+2)-1),plot(d);
      axis tight;
      lim=ylim;
      axis_max=ymax;
      axis_min=-ymax;
      data_max=lim(2);
      data_min=lim(1);
      if(data_max<axis_max)
         axis_max=data_max;
      end
      if(data_min>axis_min)
         axis_min=data_min;
      end
      ylim('manual');
      ylim([axis_min axis_max]);
```

```
      set(gca,'XTick',[1 length(d)])
      set(gca,'XTickLabel',{'1';num2str(length(d))})
      title(strcat('Details Coefficients at level ''',
                  num2str(n),''''));
      subplot(max+2-min,2,2*(max-n+2)),plot(y);
      axis tight;
      set(gca,'XTick',[1 length(y)])
      set(gca,'XTickLabel',{'1';num2str(length(y))})
      title(strcat('Reconstruction at level ''',num2str(n),''''));
   end
end;



% Function:       wavesummarize
% Author:         Wes Zanardelli
% Last Modified:  10/18/99
%
% Usage:
%   wavesummarize(y,min_scale,max_scale,'identifier')
%
%     y=original signal
%     min_scale=first displayed scale
%     max_scale=last displayed scale
%     identifier=identifier for motor being tested
%
%   example:  wavesummarize(y,1,10,'N1')
function wavesummarize(y,min_scale,max_scale,identifier) wavelet=[{'db1'}
{'db4'} {'db7'} {'db10'} {'bior2.2'} {'bior2.8'} {'bior3.5'} {'bior6.8'}
{'coif1'} {'coif3'} {'coif5'} {'sym2'} {'sym4'} {'sym6'} {'sym8'}];
summary=sprintf('Motor #%s\nlevel\t',identifier);
for n=1:max_scale,
   if((n>=min_scale)&(n<max_scale))
      summary=strcat(summary,sprintf('%-8d\t',n));
   elseif(n==max_scale)
      summary=strcat(summary,sprintf('%-8d',n));
   end
end; for loop=1:length(wavelet)
   sig=y(:,3)';
   summary=strcat(summary,sprintf('\n%s\t',char(wavelet(loop))));
   [lo_d,hi_d]=wfilters(char(wavelet(loop)),'d');
   for n=1:max_scale,
      % Convolution computed without the zero-padded edges
      d=conv2(sig,hi_d,'valid');
      sig=conv2(sig,lo_d,'valid');
      % Dyadic downsampling
      d=d(2:2:length(d));
      sig=sig(2:2:length(sig));
```

```
      if((n>=min_scale)&(n<max_scale))
         summary=strcat(summary,sprintf('%.6f\t',max(abs(d))));
      elseif(n==max_scale)
         summary=strcat(summary,sprintf('%.6f',max(abs(d))));
      end
   end;
end;
sprintf('%s',summary)
```

APPENDIX B

RT-Linux System

As a basic platform for data acquisition and control for the experiment, Real-Time Linux [3] was used with a Pentium II 400MHz PC. The system was connected to a 4 channel 12-bit A/D data acquisition system and a single channel 8-bit D/A analog output system through the parallel port on the motherboard as well as two additional parallel ports connected via the ISA bus. We used the A/D system to measure voltages and currents in each experiment and additionally torque and speed for the wiper motor experiment. The D/A system was used to give a torque command to the brushless DC drive which controls the load used in the wiper motor experiment.

We chose RT-Linux as opposed to a DSP system for several reasons. RT-Linux is an excellent development platform just as standard Linux is. Compilers, debugging tools and editors all come standard with most Linux distributions. When writing real-time software, the C language is used and all of the C libraries including those which add additional math functionality are available. RT-Linux is an excellent environment to prototype a system. Parameters and equations governing the system being developed can be changed and tested very quickly.

The difference between standard Linux and RT-Linux is that RT-Linux has a real-time operating system running underneath the standard Linux kernel. First, a standard Linux distribution is installed. Debian 2.2 was used for this work. Then the RT-Linux patch was applied to a fresh kernel source and finally the kernel was recompiled. Linux then became

a task in the real-time part of the operating system that runs when there is no real-time task waiting to use the processor. RT-Linux is a hard real-time operating system. In a hard real-time operating system, all deadlines are guaranteed. It is not acceptable for interrupts to be missed or data to be lost as it would be in a soft real-time operating system. Linux is pre-empted whenever a real-time task requires use of the processor. RT-Linux provides the ability for either software timers or external hardware devices to trigger interrupts in the system which can then force interrupt service routines to run. The A/D data acquisition system used in this work is triggered by software timers. Software timers were found to have superior periodicity compared with the use of the hardware interrupt on the parallel port from the A/D data acquisition system.

RT-Linux also provides the ability for real-time tasks to communicate with standard user tasks via either first-in first-out buffers known as FIFOs or shared memory which can be accessed through the POSIX mmap calls. This allows data acquisition, control calculations, and the issuing of output commands to be performed in real-time and less critical tasks such as writing data to disk to be performed in a user process. This ability makes the system more flexible and allows it to handle more complex tasks. Included is the software used in the wiper motor analysis. The first file (sched.c) is the real-time part of the system, which handles data acquisition, controls the brushless DC load and processes data to be sent to the user process. The second file (user.c) is the user process, which takes the data from the real-time process and archives it on the hard disk. This is necessary so that the data can be analyzed later in MATLAB.

```
//
// Wes Zanardelli
// Machines Lab
// Michigan State University
// Last Modified: 07/07/2000
//
// sched.c
//

#include <rtl.h>
#include <rtl_sched.h>
```

```
#include <pthread.h>
#include <rtl_fifo.h>
#include <asm/io.h>
#include <math.h>

#define LPT 0x378
#define LPTS LPT+1
#define LPTC LPT+2
#define LPT2 0x278
#define LPT2S LPT2+1
#define LPT2C LPT2+2
#define LPT3 0x268
#define LPT3S LPT3+1
#define LPT3C LPT3+2
#define LPT4 0x280
#define LPT4S LPT4+1
#define LPT4C LPT4+2

pthread_t thread;

int voltage_msb,voltage_lsb,current_msb,current_lsb;
int torque_msb,torque_lsb,velocity_msb,velocity_lsb;
int voltage_d,current_d,torque_d,velocity_d,brushless_d;
int control,control_default,i;
double seconds,voltage_a,current_a,torque_a,velocity_a,brushless_a=0;
double pi=3.14159265359;
hrtime_t zero_ticks,ticks;

void * sched_task(void *arg)
{
  // Begin infinite loop
  while(1)
  {
    // Return control to the Linux kernel
    pthread_wait_np();

    // After software interrupt we begin here
    control=control|0x01;              // Pulse RD
    outb(control,LPTC);                // Pulse RD
    voltage_msb=inb(LPT);              // Read 8-bit MSB voltage from LPT
    voltage_lsb=inb(LPT2);             // Read 8-bit LSB voltage from LPT2
    control=control&0xFE;              // Un-Pulse RD
    outb(control,LPTC);                // Un-Pulse RD

    control=control|0x01;              // Pulse RD
    outb(control,LPTC);                // Pulse RD
    current_msb=inb(LPT);              // Read 8-bit MSB current from LPT
    current_lsb=inb(LPT2);             // Read 8-bit LSB current from LPT2
```

```
control=control&0xFE;              // Un-Pulse RD
outb(control,LPTC);                // Un-Pulse RD

control=control|0x01;              // Pulse RD
outb(control,LPTC);                // Pulse RD
torque_msb=inb(LPT);               // Read 8-bit MSB torque from LPT
torque_lsb=inb(LPT2);              // Read 8-bit LSB torque from LPT2
control=control&0xFE;              // Un-Pulse RD
outb(control,LPTC);                // Un-Pulse RD

control=control|0x01;              // Pulse RD
outb(control,LPTC);                // Pulse RD
velocity_msb=inb(LPT);             // Read 8-bit MSB velocity from LPT
velocity_lsb=inb(LPT2);            // Read 8-bit LSB velocity from LPT2
control=control&0xFE;              // Un-Pulse RD
outb(control,LPTC);                // Un-Pulse RD

control=control|0x02;              // Enable CONVST
outb(control,LPTC);                // Enable CONVST
control=control&0xFD;              // Disable CONVST
outb(control,LPTC);                // Disable CONVST

ticks=gethrtime()-zero_ticks;          // Get time in ticks
seconds=(double)ticks/NSECS_PER_SEC;   // Convert ticks to seconds

// Multiply MSB by 16 and add on the least sig. 4 bits of the LSB value
voltage_d=(16*voltage_msb)+(voltage_lsb&0x0F);
if ((voltage_d&0x800)==0x800)
  voltage_d=-(voltage_d^0xFFF)-1;

// Multiply MSB by 16 and add on the least sig. 4 bits of the LSB value
current_d=(16*current_msb)+(current_lsb&0x0F);
if ((current_d&0x800)==0x800)
  current_d=-(current_d^0xFFF)-1;

// Multiply MSB by 16 and add on the least sig. 4 bits of the LSB value
torque_d=(16*torque_msb)+(torque_lsb&0x0F);
if ((torque_d&0x800)==0x800)
  torque_d=-(torque_d^0xFFF)-1;

// Multiply MSB by 16 and add on the least sig. 4 bits of the LSB value
velocity_d=(16*velocity_msb)+(velocity_lsb&0x0F);
if ((velocity_d&0x800)==0x800)
  velocity_d=-(velocity_d^0xFFF)-1;

// voltage_a=(double)voltage_d;
// current_a=(double)current_d;
// torque_a=(double)torque_d;
```

```
// velocity_a=(double)velocity_d;

voltage_a=((double)voltage_d+3.443)*0.00842434551611;      // volts
current_a=(((double)current_d+8.3511)*0.0375600961539)/3;  // amps
torque_a=((double)torque_d+0.0285)*0.099609375;            // lb-in
velocity_a=((double)velocity_d+0.1570)*0.09765625;         // RPM

// Analog commands for the brushless motor
// brushless_a=80*sin(2*pi*0.5*seconds);

// CHANGE TO HIGH SPEED OPERATION

// High Speed - Dry Windshield
brushless_a=48.376449+7.823853*sin(pi*seconds/0.713)-22.788756*
            cos(2*pi*seconds/0.713)-2.470633*sin(2*pi*seconds/
            0.713)+4.994901*sin(3*pi*seconds/0.713)-9.392711*
            cos(4*pi*seconds/0.713)+4.092408*sin(5*pi*seconds/
            0.713)-3.955176*cos(6*pi*seconds/0.713)+2.717524*
            sin(7*pi*seconds/0.713)-1.946637*cos(8*pi*seconds/
            0.713)-1.515678*sin(8*pi*seconds/0.713)-1.753906*
            cos(9*pi*seconds/0.713)+2.351623*sin(9*pi*seconds/
            0.713)-1.504302*sin(10*pi*seconds/0.713)-3.365590*
            cos(11*pi*seconds/0.713)+1.721990*sin(11*pi*seconds/
            0.713)-3.939613*cos(13*pi*seconds/0.713)-1.826908*
            cos(15*pi*seconds/0.713)-5.012247*sin(15*pi*seconds/
            0.713)-3.032478*cos(16*pi*seconds/0.713)+2.074728*
            sin(16*pi*seconds/0.713)+4.785437*cos(17*pi*seconds/
            0.713)-2.219884*sin(17*pi*seconds/0.713)-4.452630*
            sin(18*pi*seconds/0.713)+1.817154*cos(19*pi*seconds/
            0.713)+1.638729*sin(19*pi*seconds/0.713)+3.874182*
            cos(20*pi*seconds/0.713)+1.902562*sin(22*pi*seconds/
            0.713);

// High Speed - Wet Windshield
// brushless_a=16.981963-5.298248*cos(pi*seconds/0.974)+4.940973*
               sin(pi*seconds/0.974)-17.836616*cos(2*pi*seconds/
               0.974)+2.201684*sin(2*pi*seconds/0.974)+2.398844*
               cos(3*pi*seconds/0.974)+3.286658*sin(3*pi*seconds/
               0.974)-7.759975*cos(4*pi*seconds/0.974)-1.159234*
               sin(4*pi*seconds/0.974)+2.497423*sin(5*pi*seconds/
               0.974)-4.026412*cos(6*pi*seconds/0.974)-2.036235*
               sin(6*pi*seconds/0.974)-1.407880*cos(7*pi*seconds/
               0.974)+0.857107*sin(7*pi*seconds/0.974)-1.079281*
               cos(8*pi*seconds/0.974)-2.581217*sin(8*pi*seconds/
               0.974)-0.733904*sin(9*pi*seconds/0.974)+1.371537*
               cos(10*pi*seconds/0.974)+1.075152*cos(11*pi*seconds/
               0.974)+2.111872*sin(11*pi*seconds/0.974)+1.732163*
               sin(12*pi*seconds/0.974)-3.742143*cos(13*pi*seconds/
```

```
                0.974)-1.345360*sin(13*pi*seconds/0.974)-2.068130*
                sin(14*pi*seconds/0.974)-0.799167*cos(15*pi*seconds/
                0.974)+1.181509*sin(15*pi*seconds/0.974)-0.934461*
                sin(17*pi*seconds/0.974)+0.859892*sin(18*pi*seconds/
                0.974);

// CHANGE TO LOW SPEED OPERATION

// Low Speed - Dry Windshield
// brushless_a=38.170087-0.796044*cos(pi*seconds/1.663)+1.980773*
                sin(pi*seconds/1.663)-25.513824*cos(2*pi*seconds/
                1.663)-0.929707*cos(3*pi*seconds/1.663)-2.256987*
                sin(3*pi*seconds/1.663)-9.811351*cos(4*pi*seconds/
                1.663)+1.120521*cos(5*pi*seconds/1.663)-0.839671*
                sin(5*pi*seconds/1.663)-3.700725*cos(6*pi*seconds/
                1.663)-1.362421*cos(8*pi*seconds/1.663)-0.744313*
                cos(10*pi*seconds/1.663)-0.920556*cos(12*pi*seconds/
                1.663)-1.793031*cos(14*pi*seconds/1.663)-1.619333*
                cos(16*pi*seconds/1.663)-1.085740*sin(16*pi*seconds/
                1.663)+1.183431*cos(17*pi*seconds/1.663)-1.347943*
                cos(18*pi*seconds/1.663)-1.298173*cos(19*pi*seconds/
                1.663)+1.618273*cos(20*pi*seconds/1.663)-3.978682*
                sin(20*pi*seconds/1.663)+2.600308*cos(21*pi*seconds/
                1.663)+1.859519*sin(21*pi*seconds/1.663)+0.861204*
                cos(22*pi*seconds/1.663)+2.709995*sin(22*pi*seconds/
                1.663)-3.125911*cos(23*pi*seconds/1.663)+0.840724*
                cos(25*pi*seconds/1.663)-1.556771*sin(25*pi*seconds/
                1.663);

// Low Speed - Wet Windshield
// brushless_a=13.677002-4.872948*cos(pi*seconds/1.389)+5.399117*
                sin(pi*seconds/1.389)-15.155698*cos(2*pi*seconds/
                1.389)+1.969676*sin(2*pi*seconds/1.389)+3.196256*
                cos(3*pi*seconds/1.389)+1.514894*sin(3*pi*seconds/
                1.389)-4.969694*cos(4*pi*seconds/1.389)+0.705146*
                cos(5*pi*seconds/1.389)+0.858481*sin(5*pi*seconds/
                1.389)-2.259268*cos(6*pi*seconds/1.389)-0.653241*
                sin(6*pi*seconds/1.389)-0.983701*cos(8*pi*seconds/
                1.389)-1.296708*sin(8*pi*seconds/1.389)+0.447498*
                cos(10*pi*seconds/1.389)-1.632106*sin(10*pi*seconds/
                1.389)+2.115891*cos(12*pi*seconds/1.389)+0.464924*
                sin(13*pi*seconds/1.389)+2.835643*sin(14*pi*seconds/
                1.389)-1.011456*cos(15*pi*seconds/1.389)-2.305862*
                cos(16*pi*seconds/1.389)-0.820942*sin(16*pi*seconds/
                1.389)+0.551730*cos(18*pi*seconds/1.389)-1.177618*
                sin(18*pi*seconds/1.389)-0.519896*cos(19*pi*seconds/
                1.389)+0.964000*sin(20*pi*seconds/1.389)-0.596129*
                cos(23*pi*seconds/1.389)+0.523948*cos(24*pi*seconds/
```

```
                1.389);

    // Quantize Analog Torque Command for 8-Bit D/A
    brushless_d=(int)(brushless_a*127/80)+128;

    // Send analog command to brushless
    outb(brushless_d,LPT3);

    // Push seconds into fifo
    rtf_put(0,&seconds,sizeof(double));
    // Push analog voltage into fifo
    rtf_put(1,&voltage_a,sizeof(double));
    // Push analog current into fifo
    rtf_put(2,&current_a,sizeof(double));
    // Push analog torque into fifo
    rtf_put(3,&torque_a,sizeof(double));
    // Push analog velocity into fifo
    rtf_put(4,&velocity_a,sizeof(double));
    // Push torque command into fifo
    rtf_put(5,&brushless_a,sizeof(double));
  }
  return 0;
}

int init_module(void)
{
  // Initialize thread
  struct sched_param p;

  rtf_create(0,16*sizeof(double));        // Create fifos
  rtf_create(1,16*sizeof(double));
  rtf_create(2,16*sizeof(double));
  rtf_create(3,16*sizeof(double));
  rtf_create(4,16*sizeof(double));
  rtf_create(5,16*sizeof(double));

  control=control_default=inb(LPTC);      // Store initial values

  control=control|0x20;                   // Enable input direction on LPT
  outb(control,LPTC);                     // Enable input direction on LPT
  outb(inb(LPT2C)|0x20,LPT2C);            // Enable input direction on LPT2

  control=control&0xF0;                   // Clear control bits
  outb(control,LPTC);                     // Clear control bits

  control=control|0x02;                   // Enable CONVST
  outb(control,LPTC);                     // Enable CONVST
  control=control&0xFD;                   // Disable CONVST
```

```
  outb(control,LPTC);                    // Disable CONVST

  pthread_create(&thread,NULL,sched_task,0);
  pthread_make_periodic_np(thread,gethrtime()+0.1*NSECS_PER_SEC,80000);
  pthread_setfp_np(thread,1);
  p.sched_priority=1;
  pthread_setschedparam(thread,SCHED_FIFO,&p);

  zero_ticks=gethrtime();                // Get initial time in ticks

  return 0;
}

void cleanup_module(void)
{
  pthread_delete_np(thread);

  rtf_destroy(5);                        // Destroy fifos
  rtf_destroy(4);
  rtf_destroy(3);
  rtf_destroy(2);
  rtf_destroy(1);
  rtf_destroy(0);

  outb(control_default,LPTC);            // Replace initial values
}



//
// Wes Zanardelli
// Machines Lab
// Michigan State University
// Last Modified: 07/07/2000
//
// user.c
//

#define NUM_SAMPLES 32785
#define BASE_ADDRESS (127 * 0x100000)
#include <stdio.h>
#include <sys/resource.h>
#include <fcntl.h>
#include <unistd.h>
#include <rtl_fifo.h>
#include <rtl_time.h>

int main(void)
```

```
{
  int rt_to_user_0,rt_to_user_1,rt_to_user_2;
  int rt_to_user_3,rt_to_user_4,rt_to_user_5;
  int i;
  double stor_seconds[NUM_SAMPLES];
  double stor_voltage[NUM_SAMPLES];
  double stor_current[NUM_SAMPLES];
  double stor_torque[NUM_SAMPLES];
  double stor_velocity[NUM_SAMPLES];
  double stor_brushless[NUM_SAMPLES];
  FILE *fp;
  char datafile_name[20];
  double seconds,voltage_a,current_a,torque_a,velocity_a,brushless_a;

  setpriority(PRIO_PROCESS,0,-20);           // Increase priority

  // Open fifo
  if ((rt_to_user_0 = open("/dev/rtf0", O_RDONLY)) < 0)
  {
    fprintf(stderr, "Error opening /dev/rtf0\n");
    exit(1);
  }

  // Open fifo
  if ((rt_to_user_1 = open("/dev/rtf1", O_RDONLY)) < 0)
  {
    fprintf(stderr, "Error opening /dev/rtf1\n");
    exit(1);
  }

  // Open fifo
  if ((rt_to_user_2 = open("/dev/rtf2", O_RDONLY)) < 0)
  {
    fprintf(stderr, "Error opening /dev/rtf2\n");
    exit(1);
  }

  // Open fifo
  if ((rt_to_user_3 = open("/dev/rtf3", O_RDONLY)) < 0)
  {
    fprintf(stderr, "Error opening /dev/rtf3\n");
    exit(1);
  }

  // Open fifo
  if ((rt_to_user_4 = open("/dev/rtf4", O_RDONLY)) < 0)
  {
    fprintf(stderr, "Error opening /dev/rtf4\n");
```

```
  exit(1);
}

// Open fifo
if ((rt_to_user_5 = open("/dev/rtf5", O_RDONLY)) < 0)
{
  fprintf(stderr, "Error opening /dev/rtf5\n");
  exit(1);
}

for (i=0;i<NUM_SAMPLES;i++)
{
  // Read seconds from fifo
  while(read(rt_to_user_0,&seconds,sizeof(double))==0);
  // Store seconds in vector
  stor_seconds[i]=seconds;
  // Read voltage from fifo
  read(rt_to_user_1,&voltage_a,sizeof(double));
  // Store analog voltage in vector
  stor_voltage[i]=voltage_a;
  // Read current from fifo
  read(rt_to_user_2,&current_a,sizeof(double));
  // Store analog current in vector
  stor_current[i]=current_a;
  // Read torque from fifo
  read(rt_to_user_3,&torque_a,sizeof(double));
  // Store analog torque in vector
  stor_torque[i]=torque_a;
  // Read velocity from fifo
  read(rt_to_user_4,&velocity_a,sizeof(double));
  // Store analog velocity in vector
  stor_velocity[i]=velocity_a;
  // Read torque command from fifo
  read(rt_to_user_5,&brushless_a,sizeof(double));
  // Store torque command in vector
  stor_brushless[i]=brushless_a;
}

close(rt_to_user_5);                    // Close fifo
close(rt_to_user_4);                    // Close fifo
close(rt_to_user_3);                    // Close fifo
close(rt_to_user_2);                    // Close fifo
close(rt_to_user_1);                    // Close fifo
close(rt_to_user_0);                    // Close fifo

// Get name for datafile
printf("Enter the name for the datafile: ");
scanf("%s",datafile_name);
```

```
  while (fopen(datafile_name,"r")!=NULL)
  {
    printf("Already Exists!  Enter a new name for the datafile: ");
    scanf("%s",datafile_name);
  }

  fp=fopen(datafile_name,"w");              // Open datafile

  // Write seconds, V, I, torque, velocity and torque command to file
  for (i=17;i<NUM_SAMPLES;i++)
  {
    seconds=stor_seconds[i];
    voltage_a=stor_voltage[i];
    current_a=stor_current[i];
    torque_a=stor_torque[i];
    velocity_a=stor_velocity[i];
    brushless_a=stor_brushless[i];
    fprintf(fp,"%f\t%f\t%f\t%f\t%f\t%f\n",seconds,voltage_a,current_a,
            torque_a,velocity_a,brushless_a);
  }

  fclose(fp);                               // Close datafile

  return 0;
}
```

APPENDIX C

DWT Filter Coefficients

For the DWT scaling and wavelet function coefficients to be realized using filter banks, both the decomposition lowpass filter coefficients, $h_0(n)$, and the decomposition highpass filter coefficients, $h_1(n)$, must be defined.

The scaling function coefficients at the scale j are defined as the convolution of $h_0(n)$ with the scaling function coefficients at the scale $j+1$ (2.23). Similarly, the wavelet function coefficients at the scale j are defined as the convolution of $h_1(n)$ with the scaling function coefficients at the scale $j+1$ (2.24).

Tables C.1–C.4 list the decomposition filter coefficients, $h_0(n)$ and $h_1(n)$, corresponding to each mother wavelet used in this thesis from the Daubechies, Biorthogonal, Coiflet and Symlet families respectively.

db1

$h_0(n) = [$ 0.7071 0.7071]
$h_1(n) = [$ -0.7071 0.7071]

db4

$h_0(n) = [$ -0.0106 0.0329 0.0308 -0.1870 -0.0280 0.6309 0.7148 0.2304]
$h_1(n) = [$ -0.2304 0.7148 -0.6309 -0.0280 0.1870 0.0308 -0.0329 -0.0106]

db7

$h_0(n) = [$ 0.0004 -0.0018 0.0004 0.0126 -0.0166 -0.0380 0.0806 0.0713
-0.2240 -0.1439 0.4698 0.7291 0.3965 0.0779]
$h_1(n) = [$ -0.0779 0.3965 -0.7291 0.4698 0.1439 -0.2240 -0.0713 0.0806
0.0380 -0.0166 -0.0126 0.0004 0.0018 0.0004]

db10

$h_0(n) = [$ -0.0000 0.0001 -0.0001 -0.0007 0.0020 0.0014 -0.0107 0.0036
0.0332 -0.0295 -0.0714 0.0931 0.1274 -0.1959 -0.2498 0.2812
0.6885 0.5272 0.1882 0.0267]
$h_1(n) = [$ -0.0267 0.1882 -0.5272 0.6885 -0.2812 -0.2498 0.1959 0.1274
-0.0931 -0.0714 0.0295 0.0332 -0.0036 -0.0107 -0.0014 0.0020
0.0007 -0.0001 -0.0001 -0.0000]

Table C.1. Daubechies wavelet decomposition filter coefficients

bior2.2

$h_0(n) = [$ 0 -0.1768 0.3536 1.0607 0.3536 -0.1768]
$h_1(n) = [$ 0 0.3536 -0.7071 0.3536 0 0]

bior2.8

$h_0(n) = [$ 0 0.0015 -0.0030 -0.0129 0.0289 0.0530 -0.1349 -0.1638
0.4626 0.9516 0.4626 -0.1638 -0.1349 0.0530 0.0289 -0.0129
-0.0030 0.0015]
$h_1(n) = [$ 0 0 0 0 0 0 0 0.3536
-0.7071 0.3536 0 0 0 0 0 0
0 0]

bior3.5

$h_0(n) = [$ -0.0138 0.0414 0.0525 -0.2679 -0.0718 0.9667 0.9667 -0.0718
-0.2679 0.0525 0.0414 -0.0138]
$h_1(n) = [$ 0 0 0 0 -0.1768 0.5303 -0.5303 0.1768
0 0 0 0]

bior6.8

$h_0(n) = [$ 0 0.0019 -0.0019 -0.0170 0.0119 0.0497 -0.0773 -0.0941
0.4208 0.8259 0.4208 -0.0941 -0.0773 0.0497 0.0119 -0.0170
-0.0019 0.0019]
$h_1(n) = [$ 0 0 0 0.0144 -0.0145 -0.0787 0.0404 0.4178
-0.7589 0.4178 0.0404 -0.0787 -0.0145 0.0144 0 0
0 0]

Table C.2. Biorthogonal wavelet decomposition filter coefficients

coif1

$h_0(n) = [$ -0.0157 -0.0727 0.3849 0.8526 0.3379 -0.0727]
$h_1(n) = [$ 0.0727 0.3379 -0.8526 0.3849 0.0727 -0.0157]

coif3

$h_0(n) = [$ -0.0000 -0.0001 0.0005 0.0011 -0.0026 -0.0090 0.0159 0.0346
-0.0823 -0.0718 0.4285 0.7938 0.4052 -0.0611 -0.0658 0.0235
0.0078 -0.0038]
$h_1(n) = [$ 0.0038 0.0078 -0.0235 -0.0658 0.0611 0.4052 -0.7938 0.4285
0.0718 -0.0823 -0.0346 0.0159 0.0090 -0.0026 -0.0011 0.0005
0.0001 -0.0000]

coif5

$h_0(n) = [$ -0.0000 -0.0000 0.0000 0.0000 -0.0000 -0.0000 0.0001 0.0003
-0.0006 -0.0017 0.0024 0.0068 -0.0092 -0.0198 0.0327 0.0413
-0.1056 -0.0620 0.4380 0.7743 0.4216 -0.0520 -0.0919 0.0282
0.0234 -0.0101 -0.0042 0.0022 0.0004 -0.0002]
$h_1(n) = [$ 0.0002 0.0004 -0.0022 -0.0042 0.0101 0.0234 -0.0282 -0.0919
0.0520 0.4216 -0.7743 0.4380 0.0620 -0.1056 -0.0413 0.0327
0.0198 -0.0092 -0.0068 0.0024 0.0017 -0.0006 -0.0003 0.0001
0.0000 -0.0000 -0.0000 0.0000 0.0000 -0.0000]

Table C.3. Coiflet wavelet decomposition filter coefficients

sym2

$h_0(n) = [$ -0.1294 0.2241 0.8365 0.4830]
$h_1(n) = [$ -0.4830 0.8365 -0.2241 -0.1294]

sym4

$h_0(n) = [$ -0.0758 -0.0296 0.4976 0.8037 0.2979 -0.0992 -0.0126 0.0322]
$h_1(n) = [$ -0.0322 -0.0126 0.0992 0.2979 -0.8037 0.4976 0.0296 -0.0758]

sym6

$h_0(n) = [$ 0.0154 0.0035 -0.1180 -0.0483 0.4911 0.7876 0.3379 -0.0726
-0.0211 0.0447 0.0018 -0.0078]
$h_1(n) = [$ 0.0078 0.0018 -0.0447 -0.0211 0.0726 0.3379 -0.7876 0.4911
0.0483 -0.1180 -0.0035 0.0154]

sym8

$h_0(n) = [$ -0.0034 -0.0005 0.0317 0.0076 -0.1433 -0.0613 0.4814 0.7772
0.3644 -0.0519 -0.0272 0.0491 0.0038 -0.0150 -0.0003 0.0019]
$h_1(n) = [$ -0.0019 -0.0003 0.0150 0.0038 -0.0491 -0.0272 0.0519 0.3644
-0.7772 0.4814 0.0613 -0.1433 -0.0076 0.0317 0.0005 -0.0034]

Table C.4. Symlet wavelet decomposition filter coefficients

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] M. Akay. Wavelet applications in medicine. *IEEE Spectrum*, pages 50–56, May 1997.

[2] A. Ambardar. *Analog and Digital Signal Processing.* PWS Publishing Company, 1995.

[3] M. Barabanov. A linux-based real-time operating system. Master's thesis, New Mexico Institute of Mining and Technology, 1997.

[4] C. S. Burrus, R. A. Gopinath, and H. Guo. *Introduction to Wavelets and Wavelet Transforms a Primer.* Prentice Hall, 1998.

[5] T. M. Cover and P. E. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, IT-13(1):21–27, 1967.

[6] J. H. Friedman, F. Baskett, and L. J. Shustek. An algorithm for finding nearest neighbors. *IEEE Transactions on Computers*, pages 1000–1006, October 1975.

[7] S. Mallat and W. L. Hwang. Singularity detection and processing with wavelets. *IEEE Transactions on Information Theory*, 38:617–643, 1992.

[8] M. Misiti, Y. Misiti, G. Oppenheim, and J. M. Poggi. *Wavelet Toolbox for Use with MATLAB.* The MathWorks, Inc., 1997.

[9] T. Y. Young and T. W. Calvert. *Classification, Estimation and Pattern Recognition.* American Elsevier Publishing Co., Inc., 1974.