DETECTION OF PERMANENT MAGNET DC MOTOR FAILURE DUE TO BRUSH WEAR USING PARAMETER ESTIMATION AND STATISTICAL ANALYSIS

A Thesis

by

COLE CHASE TEEL-JONGEBLOED

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Wei Zhan
Co-Chair of Committee,	Mathew Kuttolamadom
Committee Members,	Byul Hur
	Lan Zhou
Head of Department,	Reza Langari

December 2020

Major Subject: Engineering Technology

ABSTRACT

Failure detection of DC motors is a common study, and could be extremely useful in real world applications. Undiagnosed eminent motor failure could cause a range of effects, and without maintenance will inevitably occur. Motor faults can be classified as electrical or mechanical, both with wide ranges of causes. Electrical failure includes stator or rotor winding faults, inverter faults, position of sensor faults in brushless motors, bearing faults, and brush faults. Mechanical faults include bearing faults, broken rotor bar, rotor eccentricity faults, end ring faults, and load faults. The aim of this study was to observe the effect of brush fault within a permanent magnet DC (PMDC) motor. Carbon contact brushes are used in PMDC motors to transmit electrical current from the stator of the motor to the rotor of the motor, ensuring the rotation of the commutators. Over time, the carbon contact becomes worn down from commutators continually moving across them. As the contacts length is decreased, the spring holding it in place becomes more stretched out, putting in more effort to hold the brush in place. This introduces a resistance, referred to as a contact resistance, that can affect the motor speed and performance. Changes in speed and resistance can be measured and observed, and curves can be fitted to their relationship with statistical significance. We can also create a simulation method using basic differential equations that describe the motor and introducing random noise to the simulation with generation of random numbers for the motor parameters. Finally, a prediction interval is generated, and eminent motor failure can be predicted when values measured values stray from the simulated path. Erratic motor behavior can also be observed at the point of eminent motor failure.

DEDICATION

To my parents, who have given me more love and support than I could have ever asked for.

ACKNOWLEDGEMENTS

I would like to thank my committee chair, Dr. Zhan, my committee members, Dr. Kuttolamadom, Dr. Hur, Dr. Zhou, and my department head, Dr. Leon, for their guidance and support throughout the course of this research.

Thanks also go to my friends and colleagues and the department faculty and staff for making my time at Texas A&M University a great experience.

Finally, thanks to my step-mother and father for their encouragement.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supervised by a thesis committee consisting of Dr. Zhan [advisor], Dr. Kuttolamadom [co-advisor], and Dr. Hur [member] of the Department of Engineering Technology and Industrial Distribution and Dr. Zhou of the Department of Statistics.

The circuit board analyzed and described in Chapter 3 was provided by Professor Zhan.

All other work conducted for the thesis (or) dissertation was completed by the student independently.

Funding Sources

Graduate study was funded entirely independently.

NOMENCLATURE

ANFIS	Adaptive Neuro-Fuzzy Inference System
EMD	Empirical Mode Decomposition
FFT	Fast-Fourier Transform
HMM	Hidden Markov Model
IMFS	Intrinsic Mode Functions
PMDC	Permanent Magnet DC Motor
PM	Permanent-Magnet
PMSM	Permanent-Magnet Synchronous Motors
RUL	Remaining Useful Life
SVM	Support Vector Machine

TABLE OF CONTENTS

Page

LIST OF FIGURES

Page

1. INTRODUCTION

DC motors were the first electrically powered motor, and was the most widely used motor type until the invention of the induction motor and the transformer. Onward, 60 hertz AC power systems became the standard. With the introduction of AC power, DC motors had to utilize speed control systems, increasing their cost, giving them a disadvantage within the market. Standard DC brushes also require more service and maintenance. With the introduction of solid-state electronics came more readily ways, such as thyristor converters, of interfacing DC motors to an AC line with greater control and mobility and lower cost. Power transistors and SCRs make up a solid-state switch that can be used to control the motor from a battery supply. This enabled DC motors to have applications to industrial lift trucks, golf carts, recreational vehicles, marine crafts, and many more [1]. Even though the introduction of AC power may have forced DC motors to share their spotlight, they still have many reasons why they are chosen in specific scenarios. These reasons include lower cost, larger installed base, simple and efficient design, easy service and maintenance, easy speed control, full torque at zero speed, higher motor power density, less inertia, and smaller converters and drives [2]. There are many kinds of DC motors such as series, shunt, compound, permanent magnet, brushless, servo, servo tachometer motors, the first 5 being the most popular. All of these motors are designed differently and will be covered later in this paper. Applications for series motors include products such as elevators, traction systems, cranes, air compressors, vacuum cleaners, sewing machines, and hair dryers. Applications for shunt motors include products such as lather machines, centrifugal

pumps, fans, blowers, conveyors, lifts, weaving machines, drills, shapers, spinning machines. Applications for compound motors include products such as presses, shears, conveyors, elevators, rolling mills, reciprocating machines and heavy planners.

Motors are comprised of multiple components, and these components can have both similarity and variability between motor types. Since motors are constantly subjected to forces such as load, torque, stresses, and heat as well as a variety of other outside forces contributed to by the environment the motor is subjected to, these components will inevitably be deformed, eroded, or corroded. Component failure means motor failure, which can lead to consequences with a range of severity from inconvenient to fatally catastrophic if the failure is entirely unexpected. Motor failure in consumer products could hurt a company's revenue by discouraging customers from returning, to inviting lawsuits due to bodily harm resulting from product failure. Motor failure in industrial products could lead to loss of revenue due to halting the production process, replacement of the failed motor, or lawsuits from employee injury resulting from a machine failure. Knowing when and why a failure is predicted to occur can give companies and consumers the ability to avoid problems caused by motor failure.

The aim of this research is to diagnose probable failure or future failure of a Permanent-Magnet of a DC motor due to brush wear. To begin to understand how to determine when and why motor failure is inevitable, the design of the motor must be understood.

1.1. DC Machine Construction

There are four basic parts of a dc machine. One part of the machine must establish a strong magnetic field necessary for energy conversion, normally consisting of field windings. The field assembly is made up of a core of ferrous material, normally iron or steel, to establish the magnetic field, the coil winding to provide current for the main field, insulating material, and a structure to hold the core and complete the circuit from its poles. Conductors must be interconnected to allow current flow and be able to move relative to the magnetic field, to create what is called the armature. The armature is made up of a steel core to minimize hysteresis and eddy-current losses. The armature winding varies from motor to motor depending on the desired functionality, and the number of turns and wire size determine the voltage and current characteristics of the motor. Finally, the commutator is connected to the conductors, normally made of copper. To keep the torque from reversing every time the coil moves through the plane perpendicular to the magnetic field commutators which ring the armature are used to reverse the current at that point. This is done with the use of either spring-loaded carbon contracts, named brushes, or a solid-state switching device, which will reverse current flow through the armature as they move from one field to the next. Finally, all these components must be held together properly, including a bearing and shaft to allow armature movement [1]. When a current is running through the conductor, a magnetic field is generated. When it is subsequently placed within the external magnetic field generated by the field windings, the armature experiences a force proportional to the current in the conductor and to the strength of the external magnetic field. The rotating

components are normally called the rotor, while the stationary components make up the stator, or field assembly. It is necessary for all conductors opposite a particular magnet polarity to have currents flowing in the same direction, and current direction must change as the polarity of the magnet opposite it changes.

1.2. Permanent Magnet Motor

In a permanent-magnet (PM) motor, permanent magnets are used in place of field windings to generate a magnetic field, and armature windings located in the rotor. The air-gap flux is constant due to the permanent magnet, resulting in a straight-line torque-speed characteristic, comparable to a shunt motor. Permanent magnets are extremely popular in commercial applications, due to the ease of access to ceramic magnets, a lower cost than wound field motors. Though ceramic magnets have a higher frequency of use due to their high coercivity, magnetization resistance, and smaller size, alnico magnets may be and are sometimes used instead, due to their high flux densities and resulting motor performance, but have high costs and are susceptible to demagnetization. Motor sizes may vary from sub fractional to small integral horsepower sized, and are used in a range of products from toys to space and computer applications. Thrust motors based on pm brushed motors are typically used in propulsion systems of small electric-powered boats, and PMDC motors are used in variable speed and torque applications such as antenna positioning, medical equipment, agricultural equipment, door openers, and nuclear power plants [4].

Disadvantages of permanent-magnet motors include a fixed air-gap flux resulting in a lack of speed control, possible demagnetization from a high armature current pulse, and brush sparking due to a lack of interpoles used to improve commutation. Brush sparking can indicate poor brush-life, and other factors such as current density and heat generation may be contributing factors as well.

Permanent-magnet motors have popularity due to their low-cost relative to wound-field motors resulting from their smaller number of poles, which in turn results in a larger diameter motor. Pm motors also do not need an external power supply to generate their magnetic field, resulting in a lack of need for the controller to provide field winding voltage. The constantly enabled magnetic field also increases its reliability factor, as the function is not 3 affected by the field voltage supply or field windings, and provides a detent torque, allowing for the exclusion of a holding brake, as well as a lack of heat generation during quiescent periods within field windings.

The voltage of the system can be determined with the following equation:

$$V - V_R - V_L - V_b = 0$$

The voltage across the resistor can be determined with:

$$V_R = i_a R$$

$$V_L = L\frac{di_a}{dt}$$

The back emf can be represented and reduced to:

$$V_b = K_a \emptyset_d W_m$$

$$V_b = K_m W_m$$

$$(K_m = K_a \emptyset_d)$$

where substitution leads to the completion of the first equation:

$$V_a = I_a R + L \frac{dI_a}{dt} + K_m W_m$$

where V_R is the voltage across the resistor, V_L is the voltage across the armature coil, V_b is the back emf, i_a is the armature current, L is the inductance across the armature coil, R is the resistance, Φ_d is the net flux, K_a is a geometric constant, K_m is the back emf, and W_m is the rotation speed.

The resistance in this last equation can be expanded to:

$$R = R_{Nom} + R_{Temp} + R_{contact}$$

Where R_{Nom} is the nominal resistance, R_{Temp} is the resistance introduced in the motor coils due to temperature increase, and $R_{contact}$ is the resistance introduced by the contact between the worn brushes and the commutators.

The torque can be determined with the equations:

$$T_m = J \frac{d\omega}{dt}$$

$$T_d - T_f - T_L - T_m = 0$$

$$T = K_t i_a$$

which will be represented as:

$$J\frac{d\omega}{dt} = T - T_f - T_L$$

Where T_m is the moment torque, T is the electromagnetic torque, T_w, is the torque due to rotational acceleration of the rotor, T_f is the friction torque, T_L is the load torque, and K_t is the torque gain, and J is the moment inertia of the rotor.

2. LITERATURE REVIEW

2.1. Specific Problems of DC Motors

Faults are characterized by causes of a change in the behavior of a system due to manufacturing error, environmental change, and human control action error that lead a system to lose functionality [4]. Fault analysis is the process of feature extraction, reduction, and categorization. Reasons for motor failure include exceeding the standard lifetime, abnormal power/voltage/current, overload, unbalanced load, mechanical/dynamic/thermal stress, electrical stress from fast switching inverters or unstable ground, residual stress from manufacturing, and harsh application environment, including dust, water, vibration, chemicals, and temperature. Dc motor faults are categorized as either electrical or mechanical.

Electrical faults include stator or rotor winding faults, inverter faults, position of sensor faults in brushless motors, bearing faults, and brush faults. Electrical faults are normally caused by frequency variation and unbalanced voltage. Stator or rotor winding faults are caused by either winding of inverter switch open or short circuits [8]. Winding short circuits result in increased harmonic generation and increased coil current, leading to failure. Inverter switch faults due to short circuits are due to thermal stresses caused by high switching frequency and excessive loading. Inverter switch faults due to open circuits are due to a change in terminal voltage. Hall effect position sensor failures are due to sensor misalignment due to corrosion, cracks, residual magnetic fields and core breakage, current change, or magnetic field orientation change, which is caused by

mechanical shocks [4]. The process of commutating the coil current through the use of sliding contacts or brushes result in a disadvantage of high wear rate and limited life. This introduces the need for strict maintenance schedules, else the brush wears too much, resulting in the brush shunt being pushed into the commutator causing severe damage and repair costs [1].

Mechanical faults include bearing faults, broken rotor bar, rotor eccentricity faults, end ring faults, and load faults. Eccentricity faults are due to unbalance, rotor misalignment, improper mounting or a bent rotor shaft producing an output torque oscillation, and can be indicated by mechanical vibration, temperature ununiformed air-gap, torque increase, and changes in voltage and line current. Broken rotor bars and end ring faults are due to thermal stresses from overload, magnetic stresses from electromagnetic forces, inherent stresses from to manufacturing, and mechanical stresses from lost laminations, fatigued parts, and bearing failure. Unbalanced load leads to lifespan reduction of bearings, shafts, and gears, and can be indicated by stator current time frequency, torque oscillation, and vibration. Bearing faults can occur due to many factors. Distributed bearing defects normally occur due to design and manufacturing errors, improper mounting, wear, and corrosion. Localized bearing defects such as cracks, pits, and spalls on the rolling surface normally occur due to plastic deformation and material fatigue. Winding and bearing faults make up the majority of causes for electrical motor failure. [4] gear and other mechanical faults manifest themselves as mechanical vibrations, acoustic noise, and current transients [8].

2.2. Previous Methods Summary

Many other previously researched methods of motor diagnostic testing have been reviewed to gain a better understanding of the process needed.

Zhang [3] used current signals from a brushed dc motor to diagnose the severity of short circuit resistance faults within windings with the hidden markov method and various algorithms. Glowacz [5] diagnoses short circuits of a motor by recording acoustic signals, extracting features to the time and frequency domain with coiflet wavelet transform, and using the k-nearest neighbor classification method for fault analysis. In [24], short-circuits within winding turns in Permanent-Magnet Synchronous Motors (PMSM) are detected for steady-state conditions and speed transients in motor operation, and the stator current is decomposed by empirical mode decomposition (EMD), generating intrinsic mode functions (IMFS). Smoothed pseudo-Wigner-Ville and Zhao-atlas-marks are the quadratic time-frequency distributions applied to the most significant IMFS for fault detections. In [14], winding short circuit and pole displacement are diagnosed by taking acoustic signals and vibrational signals converted to the frequency domain via fast-Fourier transform (FFT) and analyzing the results of healthy and defective motors. This method does not employ algorithms or machine learning and does not seem entirely reliable, especially so for prognosis and prediction. In [9], cage-winding defects, broken rotor bars [mechanical], and air-gap eccentricity faults of a dc six-pole flue gas compressor motor were detected by analyzing frequency spectrum of the stator current signal. In [13], air-gap eccentricity in a dc shunt motor is diagnosed by taking features such as torque developed at the slot harmonic frequency in armature

current, speed, and armature rms current as features, normalizing, feeding to a bayes classifier, and applying a discriminant function. In [10], [11], and [12], MOSFET/switch fault of a brushless PMDC motor is detected by extracting the motor current using wavelet transform, and adaptive neuro-fuzzy inference system (ANFIS)-based intelligent agent, specifically a zero-order sugeno-type ANFIS with three gaussian membership functions per input, was trained using a set of indexes from multiple operating conditions. The faulty switch was then characterized. In [6], stator turn fault is diagnosed by obtaining measurements of 3-phase voltage and current, calculating the negative sequence impedance and cross-couple impedance, using a generic Fourier transform, and using training and monitoring algorithms in MATLAB to predict fault based on cross-coupled impedance. In [6], stator turn fault is diagnosed by obtaining measurements of 3-phase voltage and current, calculating the negative sequence impedance and cross-couple impedance, using a generic Fourier transform, and using training and monitoring algorithms in MATLAB to predict fault based on cross-coupled impedance. In [29], harmonics of the stator currents induced by the fault conditions of demagnetization were analyzed for prognosis and condition monitoring of a permanent magnet synchronous motor in various non-stationary conditions involving speed and load variation. Simulation was conducted with a 2d finite-element analysis. Continuous wavelet transform and discrete wavelet transform were used to detect and classify different faults.

In [6], broken rotor bars are detected by acquiring the 3-phase voltage and current, using a modified fast Fourier transform for extraction of frequency components, and once

again using algorithms to determine the fault indexes. [15] also diagnoses broken rotor bars in PMBLDC motors by taking the motors current signal, converting to frequency variations using windowed Fourier ridges and Wigner Ville based distributions. [16] diagnoses broken rotor bars in induction motors by using multirate signal processing to improve performance of Fourier transform based analysis. In [28], diagnostics take place of the signatures of broken rotor bars, given by the spectrum modulus of line current, when a squirrel-cage induction motor is fed or not by an unbalanced line voltage. A genetic algorithm is used to record the amplitude of faulty lines and a fuzzy logic approach gives the load level operating system, as well as the rotor fault severity. This system requires steady-state operating conditions. In [30], rotor fault detection is conducted on induction motors. Motor current signature analysis is used to obtain grid frequency and machine slip tracked by statistical time-domain methods. These variables are then used to influence the parameters of a fast Fourier transform algorithm to increase frequency resolution with unchanged computational cost, or lowering computation cast with unchanged frequency resolution. In [7], multiple faults are detected, such as two bearing faults, two misalignment faults, and one inter-turn fault [electrical]. Dual tree complex wavelet packet transform was used to extract handcrafted features from measurements of current, then use a support vector machine (SVM) based classifier, training a convolutional neural network and recurrent neural network. In [4], bearing faults are detected and diagnosed by converting both vibrational and current signals of a brushless PMDC motor to the time-frequency domain with discrete wavelet transform, reduces features and trains his model with the orthogonal fuzzy neighborhood

method, and uses a dynamic neural network for classification. This method was run under stationary and non-stationary operating conditions. In [17], bearing defects such as ball defect, inner race defect, and outer race defect, are diagnosed for induction motors. Vibrational signal features are extracted to the time domain, frequency domain, and time-frequency domain as well as statistical measures of the features through the use of methods such as fast Fourier transforms, wavelet packet transform, empirical mode decomposition, singular spectrum analysis, and local mean decomposition. Features are then selected and sorted by calculating the Pearson's correlation, and fed additively into a classifier until performance accuracy is unchanging to avoid further complexity. In [21], bearing defects in induction motors are diagnosed by using a class imbalanced learning technique. In [22], bearing faults are detected by taking vibration signals from an induction motor with normal and defective bearings, applying wavelet transform to generate features, and an adaptive neural-fuzzy inference system (ANFIS) was trained and used as a diagnostic classifier to reliably separate different fault conditions. In [23], vibration signals taken from a laboratory setup for normal and defective bearings were decomposed into wavelet packets and the node energies of the decomposition tree were used as features. Features extracted from normal bearing vibration signals were used to train a Hidden Markov Model (HMM) to model normal bearing operating conditions, and this model was then used to make predictions/probabilities to track the condition of the bearings. In [26], HMMs were used for the diagnostics and prognostics of machining processes such as drilling. Thrust force and torque were measured with a data acquisition device, and features were selected. Multiple types of HMMs were used and the best case

was chosen for study. The final product predicted the remaining useful life of the drill-bit. In [27] acoustic vibration signal evolution is modeled with HMMs to estimate the state of wear at the tools edge for three different time scales, characterizing the machining tools efficiency of metal removal. In [18], a wide range of faults are diagnosed such as buckling restrained brace, rotor bending, bearing failure, stator winding fault, and rotor imbalance in induction motors. Vibration signal was applied to a convolution neural network, pooled, and then used in a support vector machine classifier. The method of using the neural network proposed in this paper is called convolutional discriminative feature learning, which feeds data back into the neural network and is robust and discriminative. [19] goes on to change the method of [18] for the better, introducing a sparse deep stacking network to increase accuracy and robustness. [20] uses 2-d filters by appropriately arranging the time series data in industrial vibration signals for use in a dislocated time series convolutional neural network to diagnose multiple faults in an induction motor. Zaidi [8] outlines the prognosis of gear failures, by using undecimated wavelet transform to extract features to the frequency domain, computing linear discriminant classifiers, and training a hidden markov method to make predictions of failure states. In [25], both recurrent neural networks and neuro-fuzzy systems are analyzed, and it is found that the neuro-fuzzy system has a better performance and training efficiency, leading to an adoption of the neuro-fuzzy system for on-line machine fault prognosis of gear wear defects including worn gear, chipped gear, and cracked gear, as well as previous data sets for gear pitting damage and shaft misalignment.

No research papers were found on the study of remaining useful life of PMDC carbon contact brushes.

3. METHODS

3.1. Simulation

Taking select equations described in the introduction we can create a model derived from differential equations to simulate motor performance with the use of MATLAB and Simulink. The equations to be used are as follows:

$$\frac{di}{dt} = \frac{V}{L} - \frac{Ri}{L} - \frac{K_m \omega}{L}$$

$$R = R_{Nom} + R_{Temp} + R_{contact}$$

Where R_{temp} is calculated using the equation $R_{Temp} = 0.00393 T_{Change}$, with 0.00393 being the temperature coefficient of copper.

$$\frac{d\omega}{dt} = \frac{T}{J} - \frac{T_{friction}}{J} - \frac{T_{Load}}{J}$$

$$T = K_t i$$

Using these equations to create a model in Simulink, the model can be seen below.

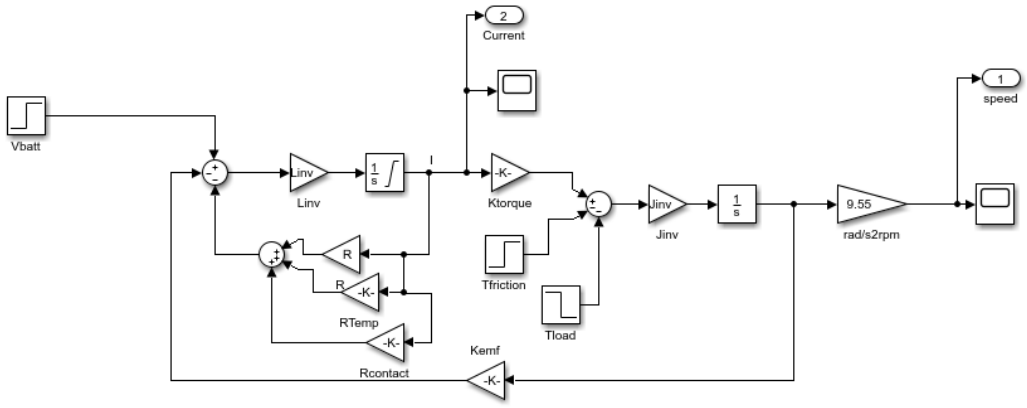

Figure 3.1: Simulink Model

Using MATLAB, we can obtain curves for speed over time, current over time, speed vs torque, speed vs resistance, speed vs voltage, as well as the corresponding curve equations and correlations. By increasing the $R_{contact}$ in the MATLAB code, we can estimate at what resistance the speed will drop to zero for the nominal set of data. Next, we can assume that most of the parameters will have some sort of noise or variation from motor to motor. Random numbers are generated along lognormal distributions for resistance, inductance, polar inertia, back emf, and temperature change using the nominal values and a standard deviation of 5% the nominal value. There are a few reasons for generating random numbers on lognormal curves. First, if was found that certain parameters such as resistance generally follow a lognormal distribution. Secondly, lognormal distributions cannot be negative, and neither can any of the parameters needed for this simulation. Random numbers are generated along a uniform distribution for voltage of +/-5% the nominal value. These sets of random numbers are then run through the simulation, with the contact resistance increasing three to four times for each set. A line is then fit to each of the data sets and the entire line of speed vs resistance is generated. Next the data is averaged at every single point corresponding to resistance change, and a prediction interval is fit around the averaged data. This gives us an estimate of what the speed of the motor should be for the specific resistance of the motor. If the motor speed being measured is not within the prediction interval, we can claim the motor is defective. Knowing the estimated resistance at which the motor will be a certain decreased speed gives us the opportunity to gauge the remaining brush life of the motor being measured.

3.2. Parameter Estimation

When the motor is running in steady state, we have the following equation:

$$\mathbf{V = IR + K\omega}$$

where V is the voltage applied, I is the current, R is the resistance, K is the back emf gain, and ω is the motor speed (in rad/sec). By measuring V, I and ω at different values of V, one can use regression to find the resistance and back emf gain K. If the power is disconnected from the motor while it is spinning, the motor will work as a generator. The voltage generated by the motor is proportional to the speed. The gain is defined as the back emf gain.

Using a LabVIEW VI, the back emf gain can be determined by switching the motor from 100% duty cycle to 0% duty cycle. Back emf gain can be found after switching the motor to 0% duty cycle and determining the voltage at which it begins to fluctuate in a downward slope. Back emf gain and resistance can then be found with the following equations:

$$\boldsymbol{K = \frac{V}{\omega}}$$

$$\boldsymbol{R = \frac{V}{I}}$$

Resistance may also be found using a multimeter. When the motor is powered, we have the following equation, which was determined earlier:

$$\mathrm{T} = \boldsymbol{K_t i}$$

where T is the motor torque, I is the current, and K is the torque gain. The torque gain is equal to the back emf gain. By using the back emf gain you measured the current reading

we can calculate the motor torque (Nm) using the above equation. Using the following equation derived earlier:

$$J\frac{d\omega}{dt} = T - T_f - T_L$$

where J is the motor inertia, ω is the speed, T is the motor torque, T_L is the load torque, and T_f is the friction torque. If we run the motor at 100% duty cycle without a load in steady state, we have

$$0 = T - T_f$$

that is $T_f = T = KI$. This allows us to calculate the friction torque. If we then turn The motor off, to 0% duty cycle, the initial speed will be the motor maximum speed, the load torque is 0, and the motor torque is 0 since the current is 0. The equation then becomes

$$J\frac{d\omega}{dt} = -T_f$$

or equivalently

$$J = -T_f \div \frac{d\omega}{dt}$$

Therefore, we can calculate the motor inertia using the friction torque and the motor deceleration. With no resistance attached, we should have a torque load of zero. Two fans are being used to estimate the load torque, one being 8 inches in diameter with 5 blades, one being 6 inches in diameter with 4 blades. We can calculate the load torque of these with the following equation:

$$T_L = K_t i - T_f$$

Next, we go about calculating the inductance (L) of a motor using an AC voltage divider. The following voltage divider circuit can be used to measure the inductance of the motor. When the motor is not spinning, it can be modeled as a resistor and an inductor connected in series. However, the voltage at the point between the motor resistor and motor inductance is not accessible. Measuring the voltage across the motor (V_{out}), we can find a function of the AC voltage (V_{in}), the frequency (f), the motor resistance previously calculated (R_2), voltage across the motor, and L. From this equation, you can solve for the inductance (L) as a function of other variables.

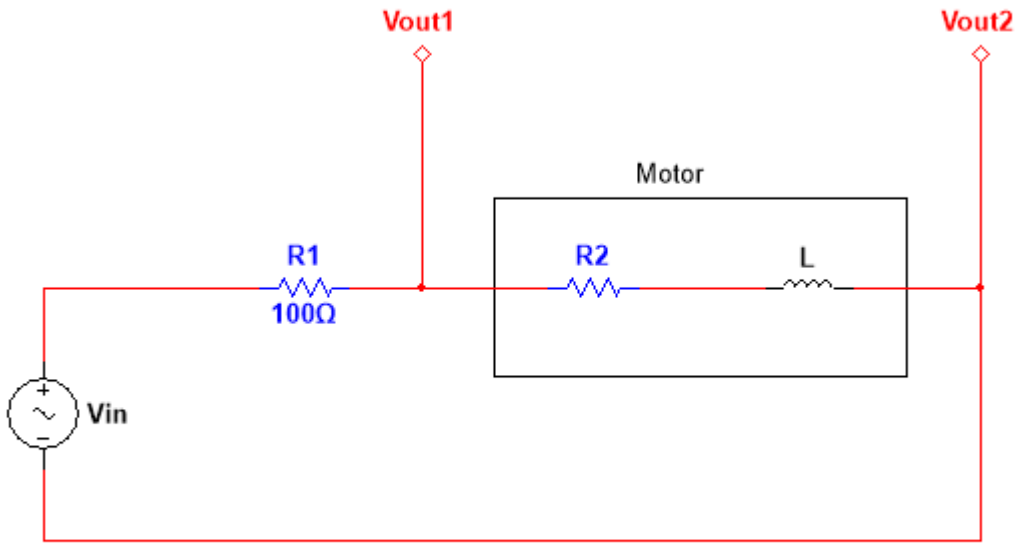

Figure 3.2: Inductance Calculation Circuit

The equation derived can be seen below.

$$L = \frac{\sqrt{\dfrac{2R_2R_1 + {R_1}^2 - \left(1 - \dfrac{{V_{in}}^2}{{V_{out}}^2}\right){R_2}^2}{\left(\dfrac{{V_{in}}^2}{{V_{out}}^2} - 1\right)(2\pi)^2}}}{f}$$

3.3. PCB Design

The following Multisim and Ultiboard files were used in the creation of the PCB. Acknowledgements are made to Dr. Zhan for providing the initial motor control board design.

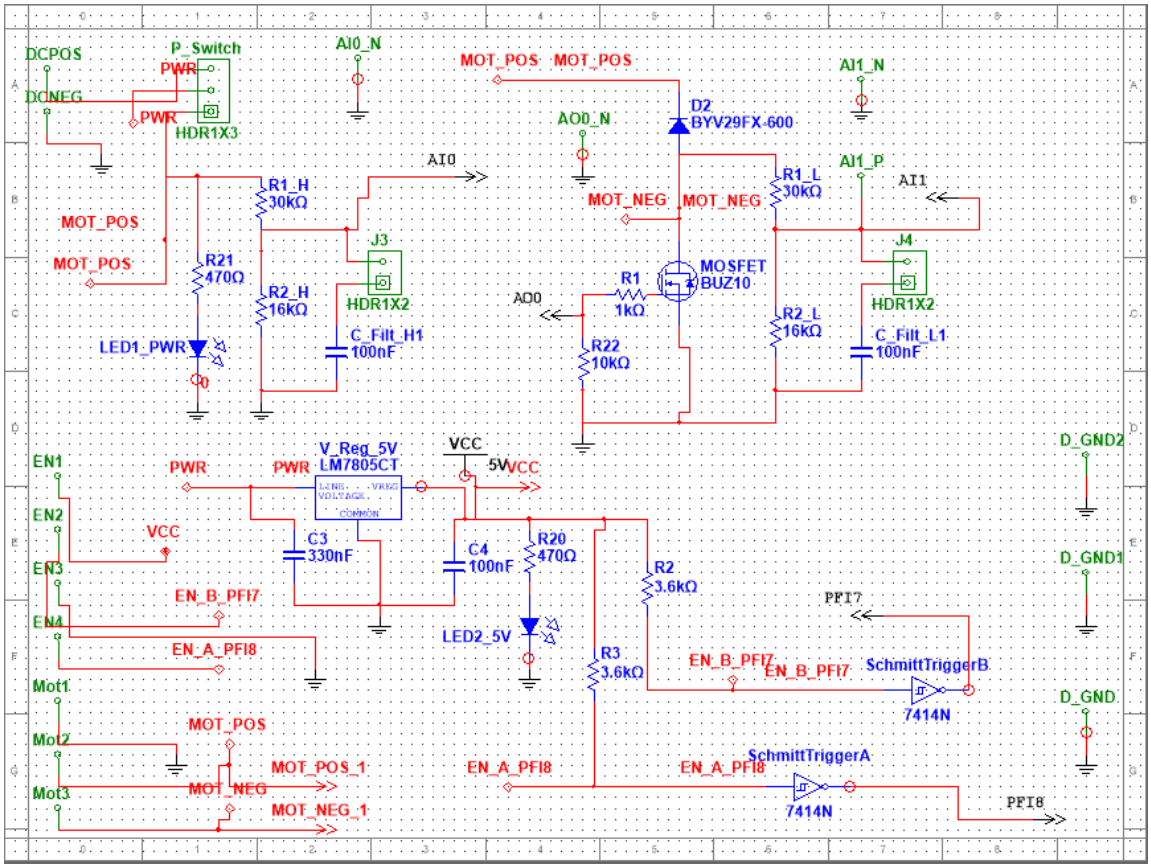

Figure 3.3: Multisim Page 1

The circuit design has inputs for dc voltage that sends power to the positive input for the motor, as well as an led light and a voltage divider for measuring the voltage on the positive side of the motor. Power is alco sent through a 5V voltage regulator to reduce to voltage for powering of certain components. AO0 sends the PWM output to the MOSFET causing the motor to turn at the desired PWM. There is also a voltage divider on the negative side of the motor to allow measuring of the voltage across the motor. There is also a current sensing device which may be seen below. The ina169na/3k is created by Texas Instruments and outputs the current throughput, or a voltage proportionate to an equation given in the datasheet. The encoder is connected to the 5V supply voltage, ground, and outputs from the encoder.

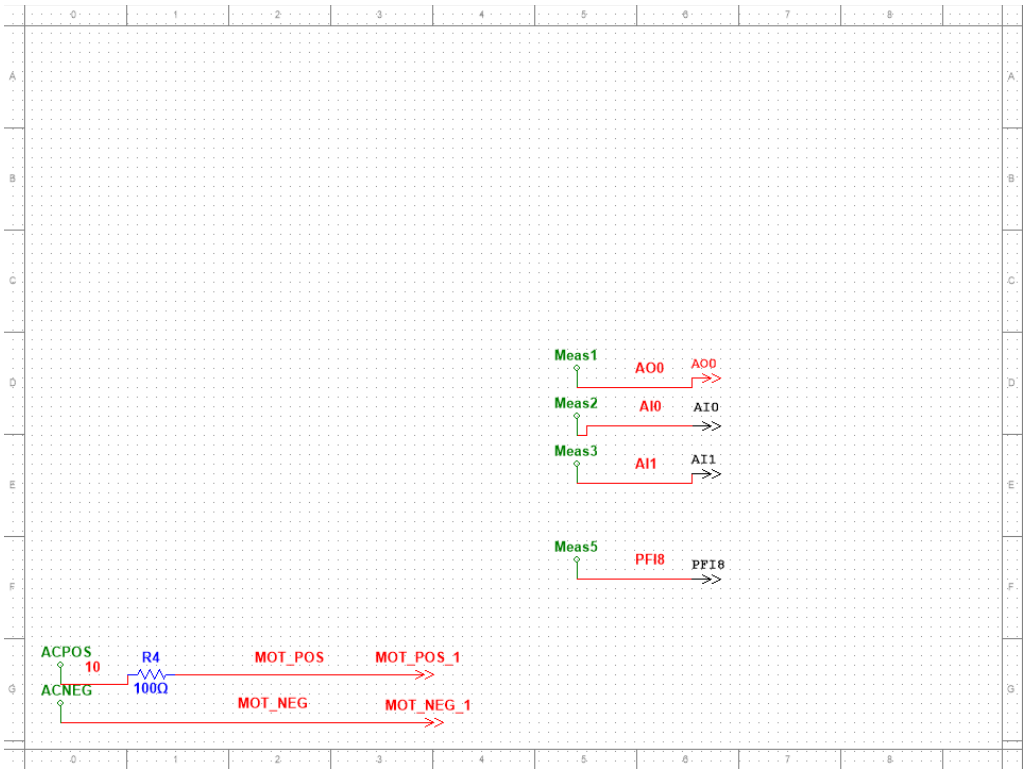

Figure 3.4: Multisim Page 2

The bottom left is the inductance calculation circuit which needs to be connected to an AC generator. The pins Meas1 – Meas8 are the test pins to be connected to the DAQ. The functions of these pins can be found below:

- Meas1 = PWM output
- Meas2 = Positive Side Voltage
- Meas3 = Negative Side Voltage
- Meas5 = Encoder Output

Unfortunately, this current sensing method was deemed unusable due to the source resistor R_s on the current sensor and its effects of probable performance reduction. Board revisions could not be made in time of the completion of this project. Thankfully, data was still able to be collected, and conclusions were achieved. The following figures 3.5 and 3.6 are a picture of the test bench and a diagram of the test bench, respectively.

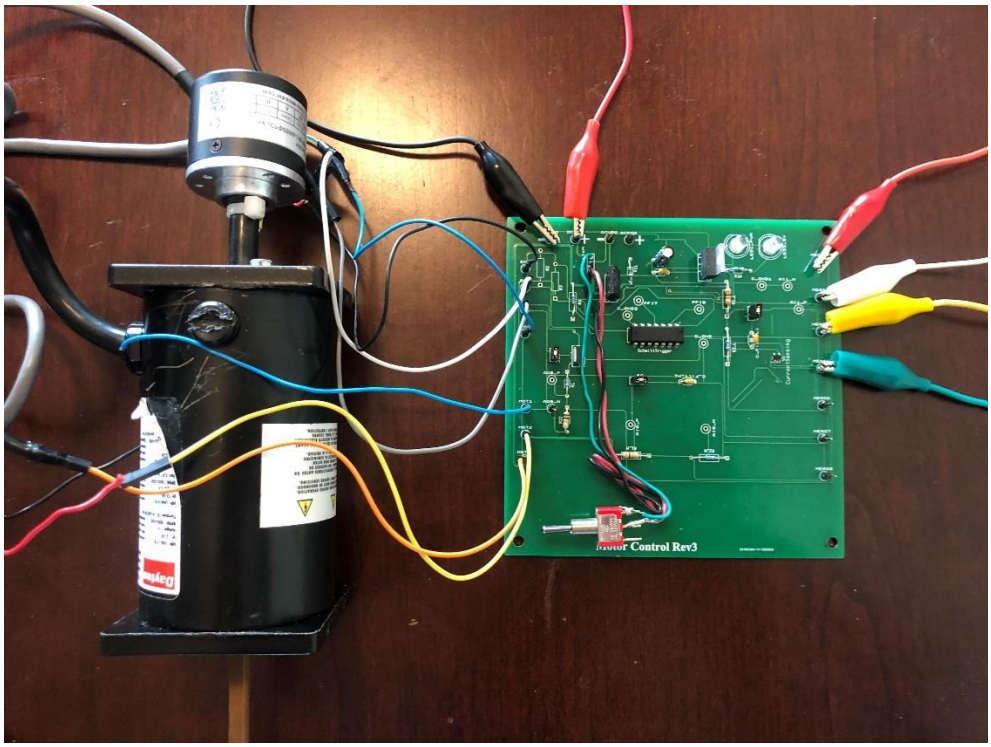

Figure 3.5: Test Bench

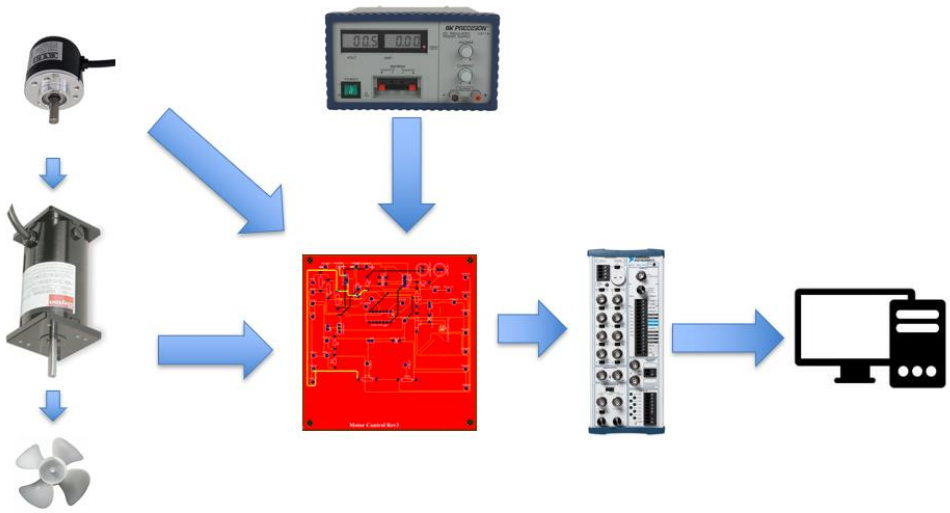

Figure 3.6: Test Bench Diagram

3.4. Overall Method

Originally it was desired to estimate the parameters of the motor with the push of a button in LabVIEW, combining the measurement of voltage, speed, and resistance, estimation of back emf, and inductance, calculations of torque, friction torque, and moment of inertia, and finally the simulation of the predicted motor to resistance change, and placement of the actual motor to resistance change on the simulation graph to display the simulations accuracy.

The motor to be used for experimentation is the DAYTON 3XE19, which has replaceable brushes. This motor was chosen due to the presence of replaceable brushes, and its low full load current ratings. Unfortunately, this motor does not have a built-in encoder, so an external encoder is fixed to the motor. To simulate brush wear, multiple sets of brushes are cut to specific lengths and experimentation is conducted on the motor for each set of brushes. The Brushes were cut or sanded down to percent changes of 18,

35, 44, 54, 58, 76, 98, and 100. These values were acquired with calipers for precision. The DAQ being used is the NI BNC-2120, as well as an NI MyRIO. Software used, as previously mentioned is MATLAB and LabVIEW.

4. RESULTS

The MATLAB results for a simulation and the experiments may be seen in figures 4.1 and 4.2.

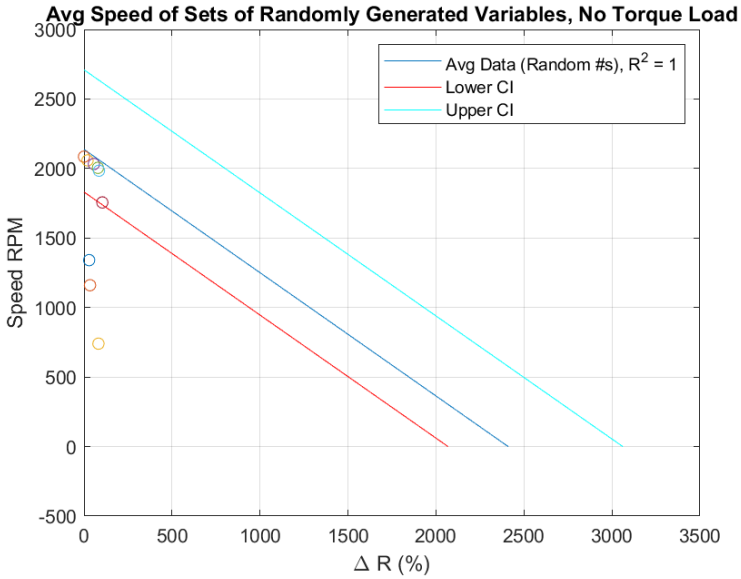

Figure 4.1: No Torque Load Results 12V

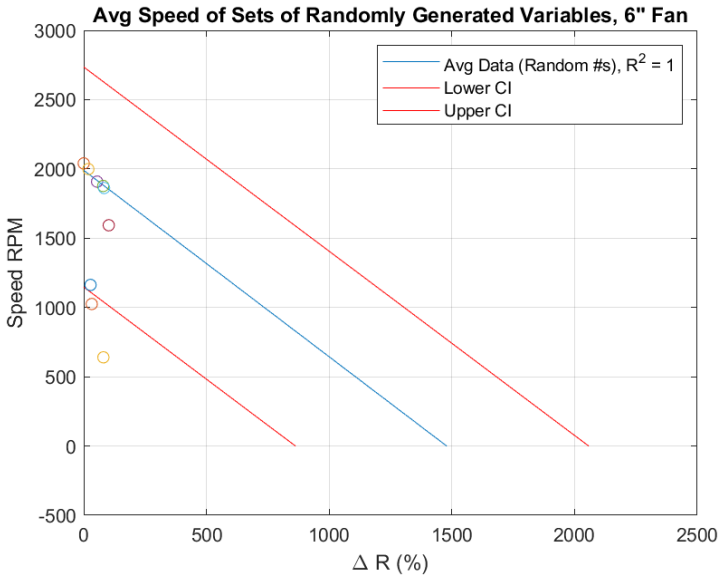

Figure 4.2: Added Torque Load Results 12V

We can see the same general relationships when simulations and tests are conducted at 8V and 16V. The results for the 8V simulations and tests for no torque load and with the added torque load may be found in figures 4.3 and 4.4, respectively. The results for the 16V simulations and tests for no torque load and with the added torque load may be found in figures 4.5 and 4.6, respectively. The presence of less data points for the 8V and 16V experiments is due to a lack of brushes after initial testing at 12V. It can be determined that a high voltage and an added torque load makes it impossible to determine eminent motor failure.

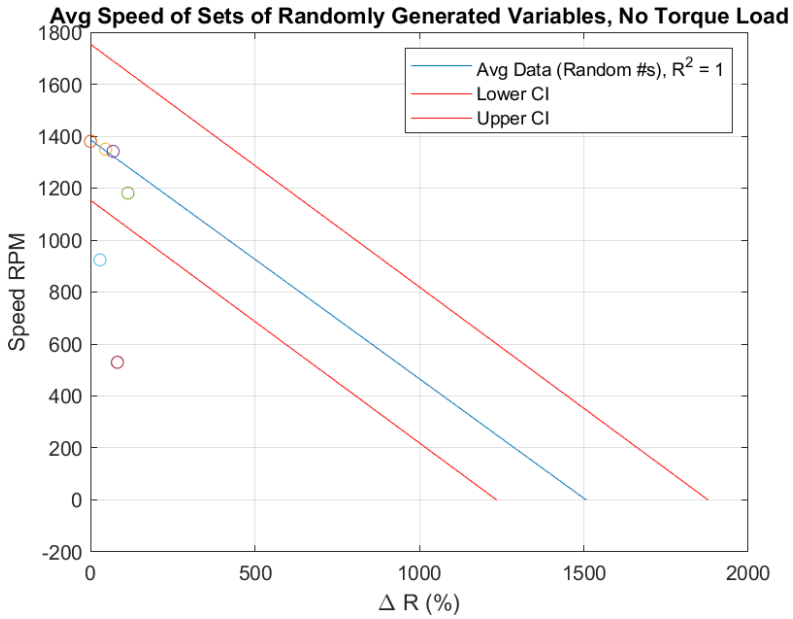

Figure 4.3: No Torque Load Results 8V

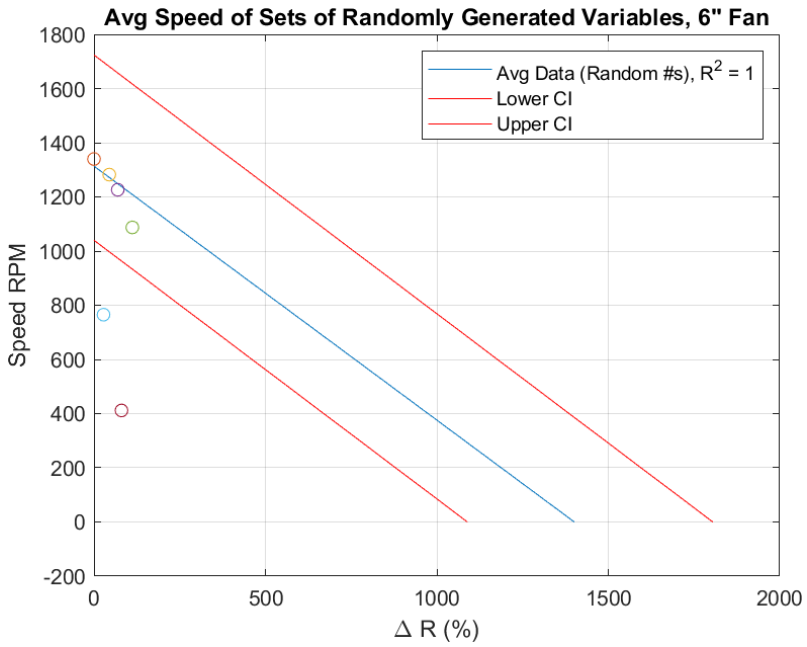

Figure 4.4: Added Torque Load Results 8V

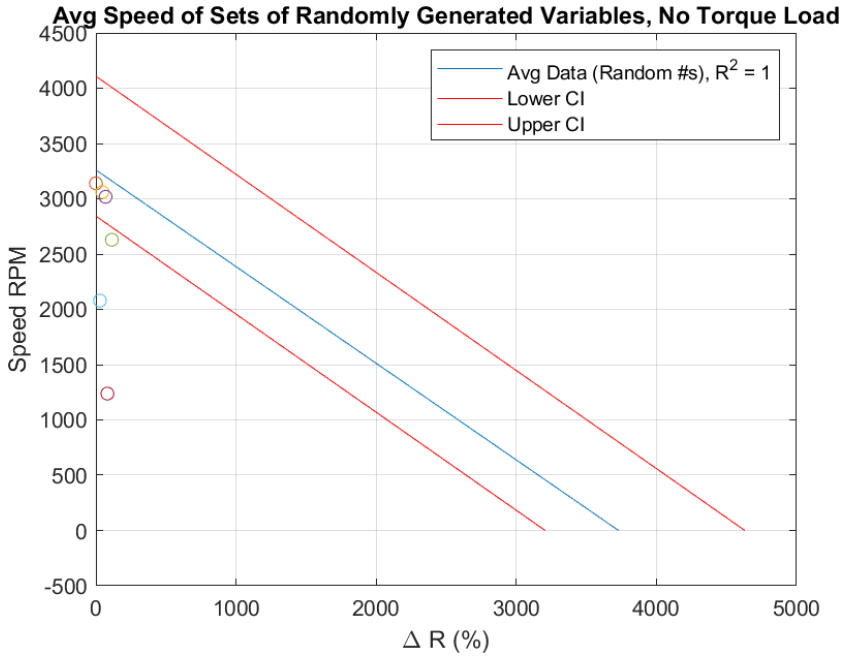

Figure 4.5: No Torque Load Results 16V

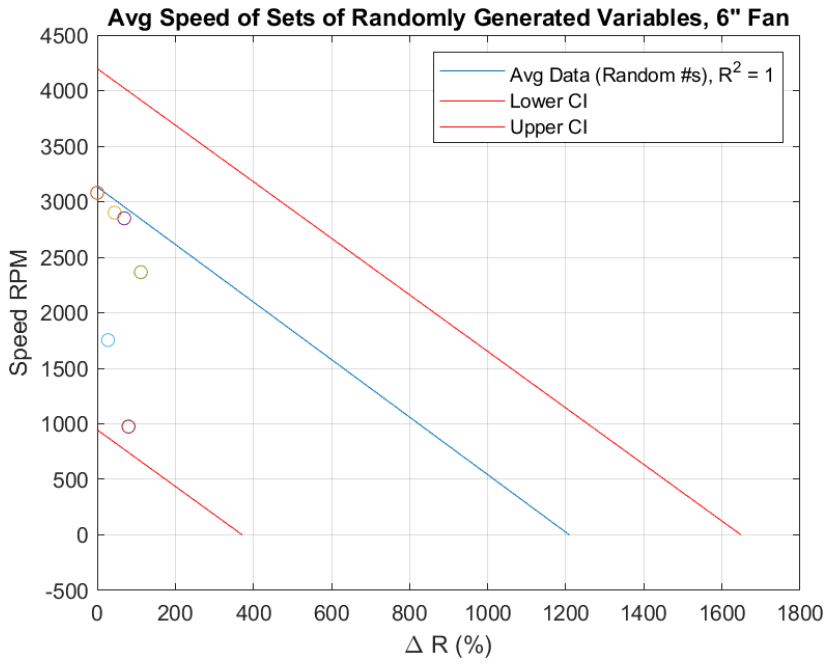

Figure 4.6: Added Torque Load Results 16V

The lines of the graph are the simulation results, and the points are the experimental results. This graph was completed with 5000 randomly generated number sets for multiple variables. Resistance, inductance, polar moment, back emf, and temperature change values are generated along lognormal distributions with a mean of their measured and calculated nominal values and a standard deviation of 5% of their nominal values. Voltage was generated along a uniform distribution of ±5% of the measured value. The prediction interval was generated with the equation as follows:

$$\bar{\boldsymbol{x}} \pm \boldsymbol{t}_{\boldsymbol{\alpha/2,n-1}}\sqrt{\boldsymbol{MSE}(\mathbf{1}+\mathbf{1}/\boldsymbol{n})}$$

Where $\bar{x}$ is the average, t is the student t-distribution, MSE is the mean squared error, and n is the size of the set. The prediction interval was generated piecewise at each point of the outputs of the sets of randomly generated variables. Each point of the averaged

data is calculated by taking the average of the output speeds. We can validate that these data points follow a normal distribution by doing probability plots, which can be seen for no torque load in figures 4.3 and 4.4, and with the added torque load in figures 4.5 and 4.6. All plots have a p-value less than 0.05, giving them statistical significance.

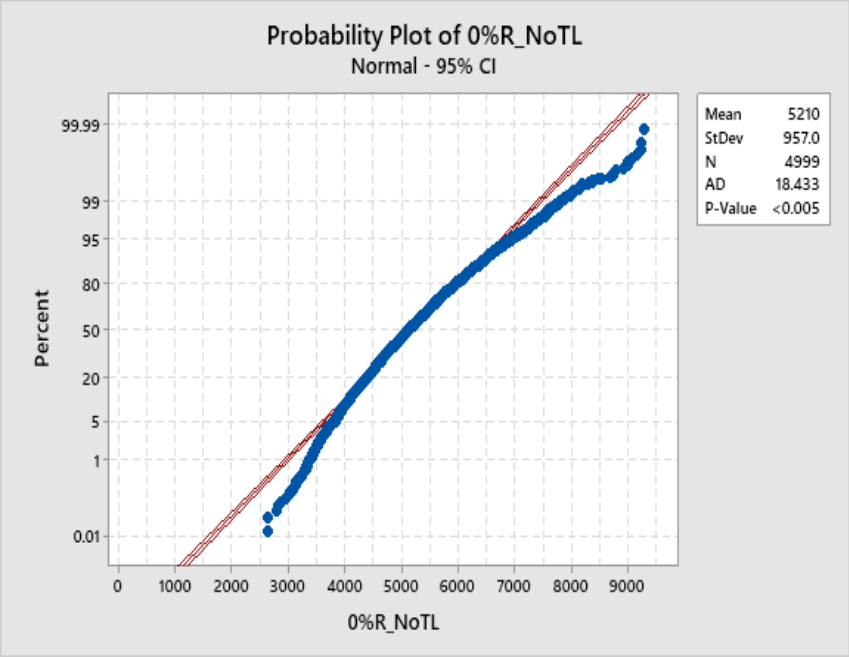

Figure 4.7: Probability Plot No Torque Load or Resistance Change

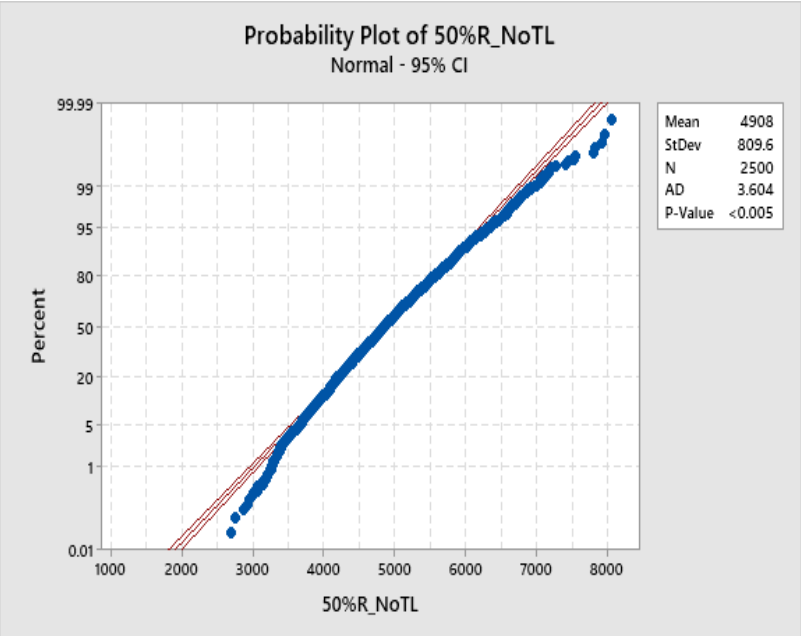

Figure 4.8: Probability Plot No Torque Load, 50% Resistance Change

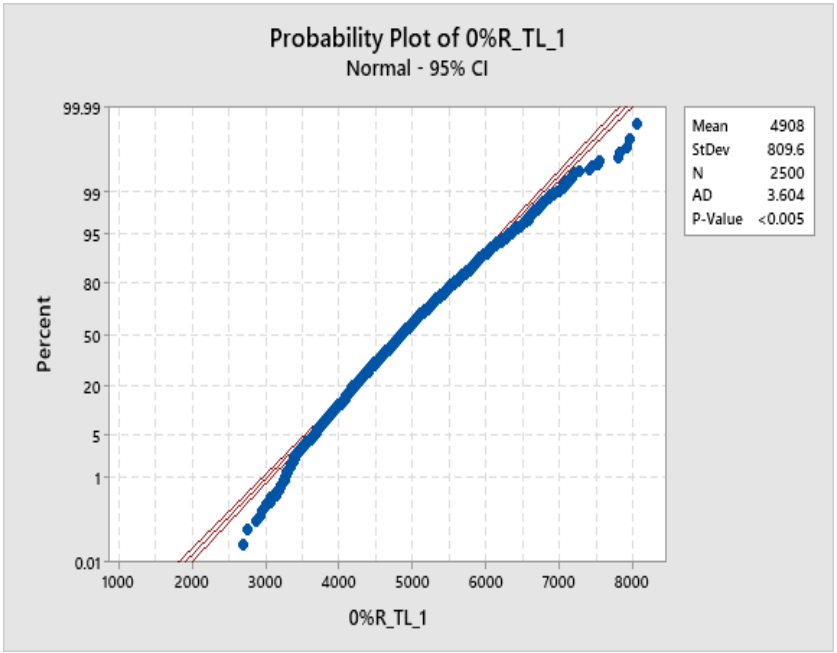

Figure 4.9: Probability Plot Added Torque Load, No Resistance Change

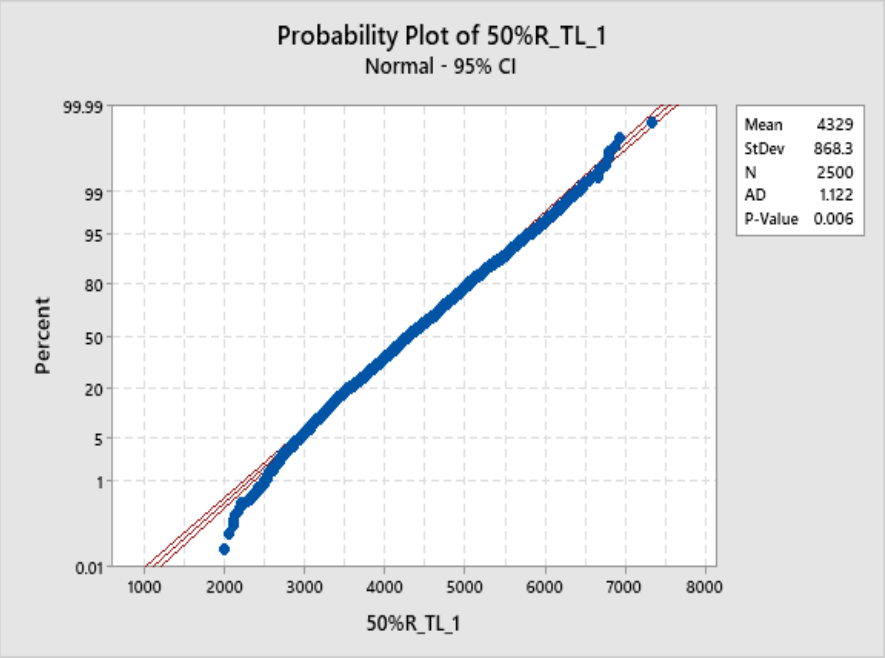

Figure 4.10: Probability Plot Added Torque Load, 50% Resistance Change

These plots are used as examples, and this process may be done at every point on the average speed plot. Eminent failure can be approximated when the speed approaches values outside of the prediction interval. As the brush length decreases, the resistance across the motor increases. This relationship can be seen in figure 4.7, and the fitted line has an R^2 value over 0.95, giving it statistical significance.

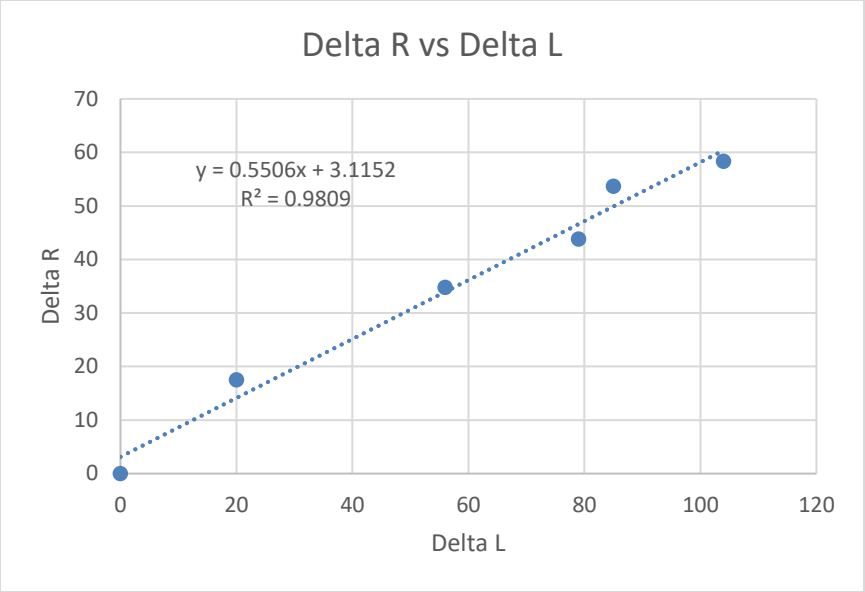

Figure 4.11: Brush Length to Resistance Relationship

Interestingly, upon eminent failure the resistance drops and begins to increase again. This can likely be explained by the connection of the motor commutators with the copper wire that attaches to the brush. This relationship can be seen in figure 4.8.

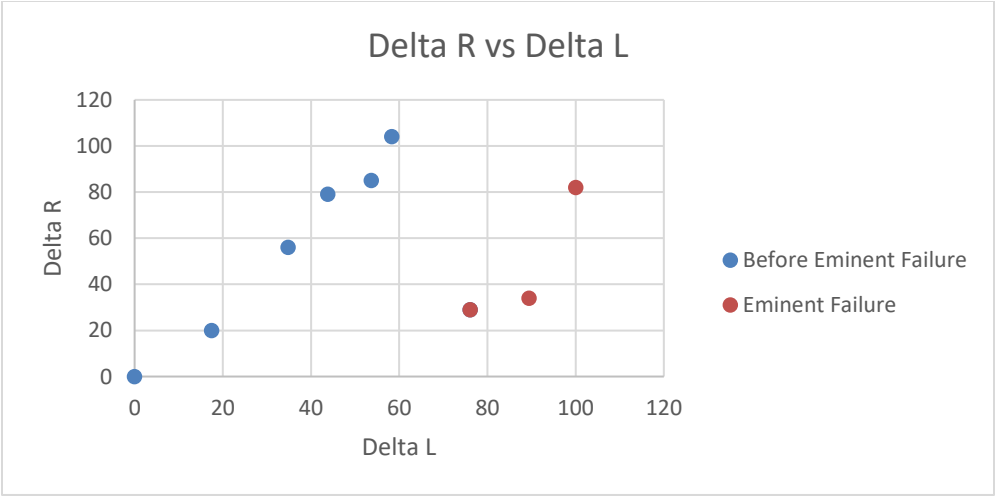

Figure 4.12: Brush Length to Resistance Relationship Eminent Failure

The corresponding speed changes for the resistance and length changes can be seen for no torque load in figures 4.9 and 4.10, and for an added torque load in figures 4.11 and figure 4.12.

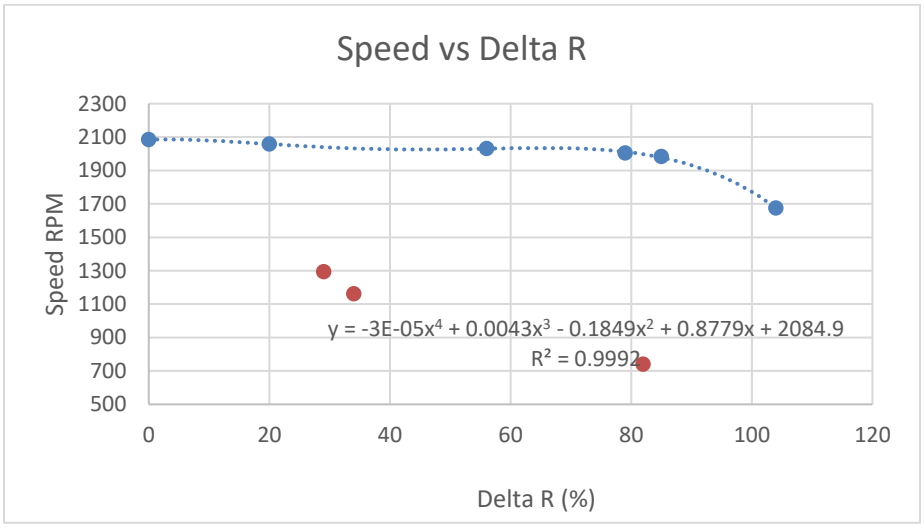

Figure 4.13: Speed to Resistance Relationship, No torque Load at 12V

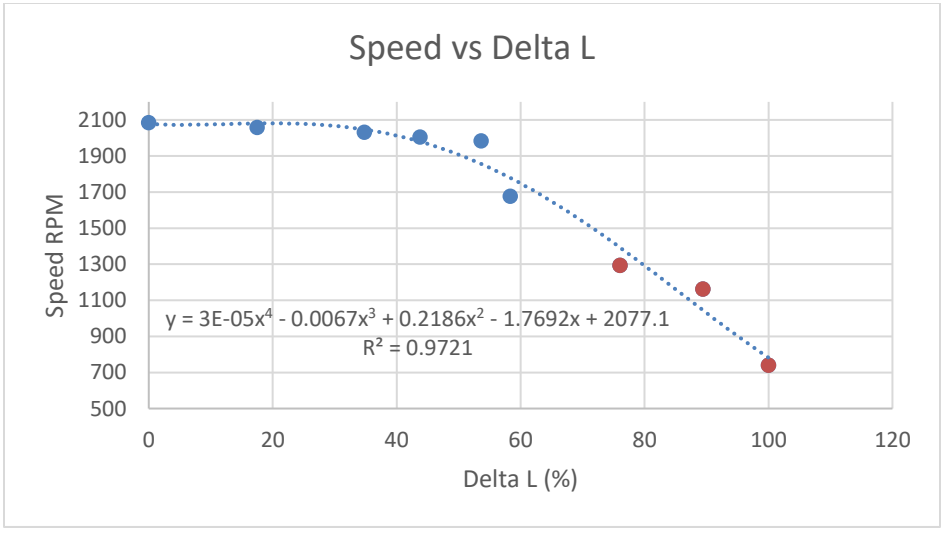

Figure 4.14: Speed to Length Relationship, No Torque Load at 12V

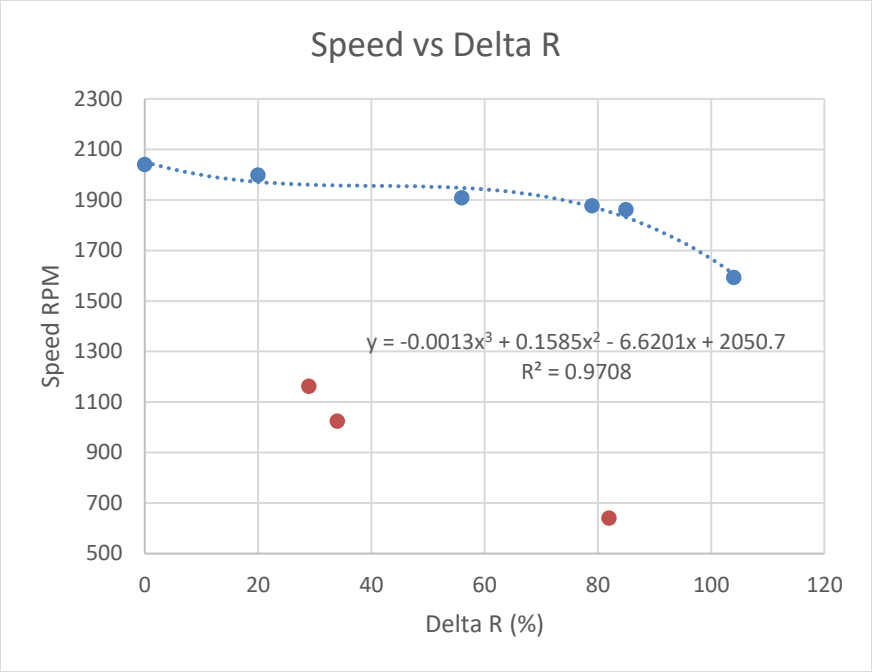

Figure 4.15: Speed to Resistance Relationship, Added Torque Load at 12V

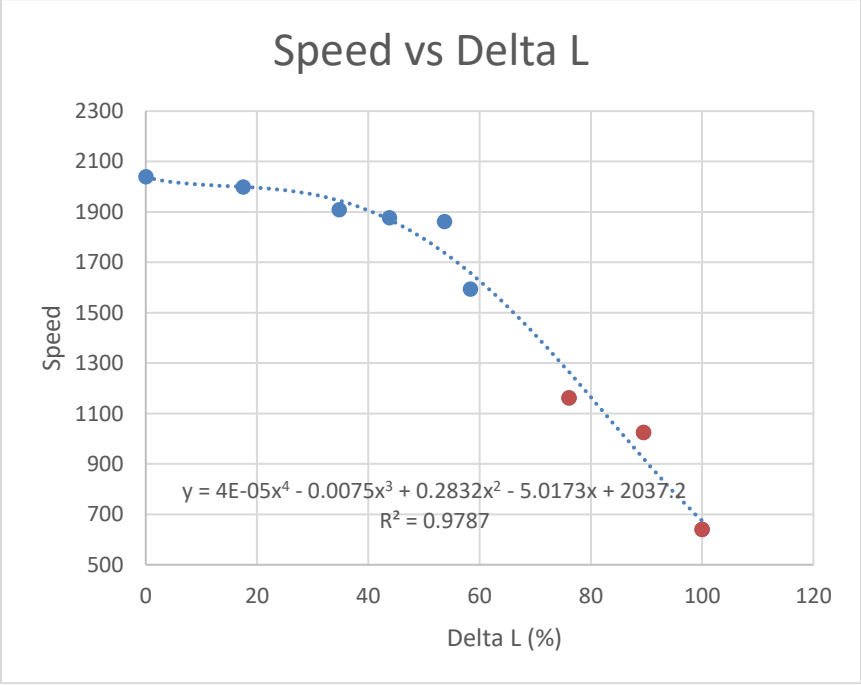

Figure 4.16: Speed to Length Relationship, Added Torque Load at 12V

These fitted curves can be done for the shown 8V and 16V datasets as well. Eminent failure can be estimated as a function of speed versus length change. Upon eminent failure when the resistance drops, the motor begins to behave erratically. Before reaching steady state, the speed spikes to values even greater than the speed for a completely healthy brush. That said, this does not indicate better performance because the speed is not constant, ands decreases upon achieving steady state. The erratic behavior can be visualized in figures 4.13, 4.14, and 4.15, and should be an indicator of eminent failure

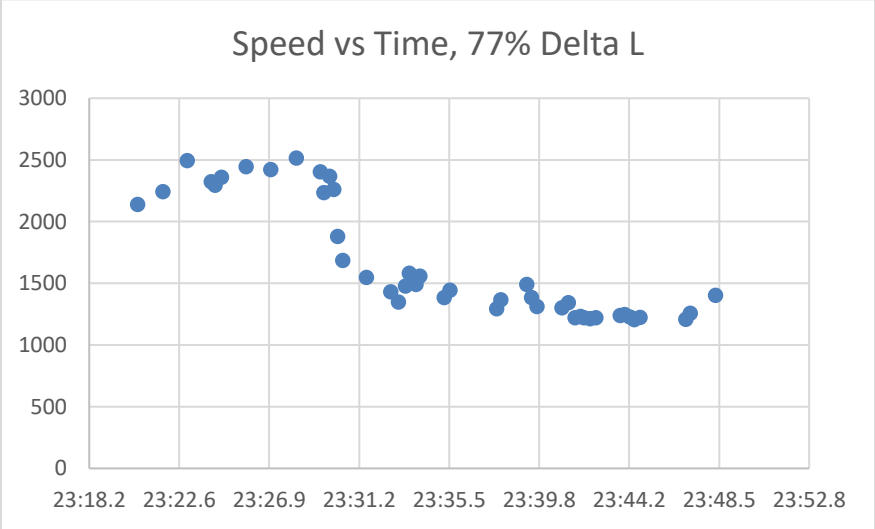

Figure 4.17: Erratic Motor Behavior with No Torque Load at 12 V

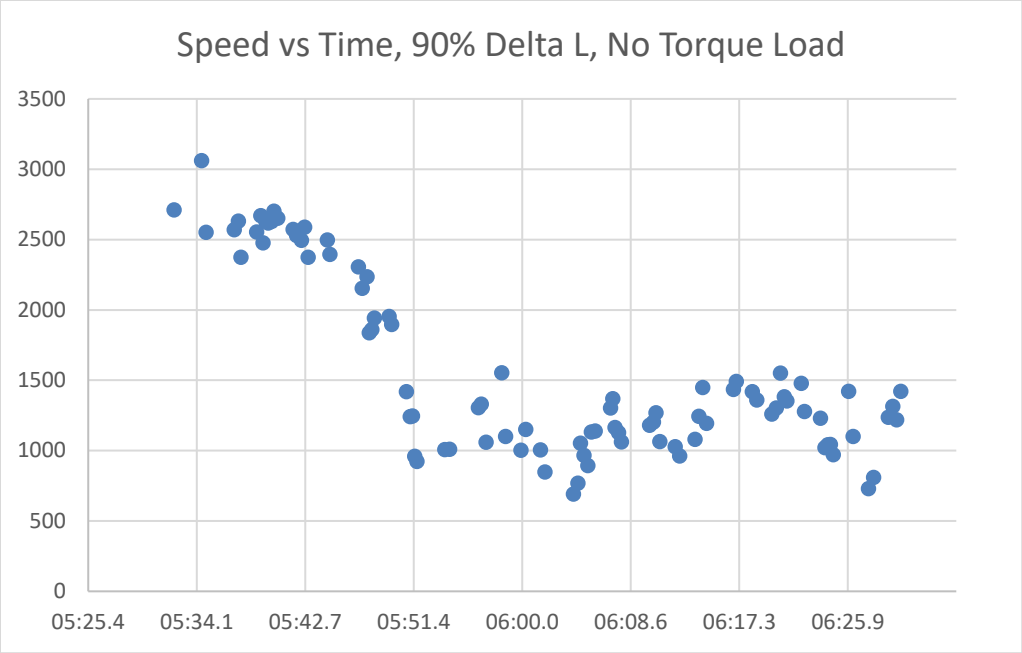

Figure 4.18: Erratic Motor Behavior with No Torque Load at 12V

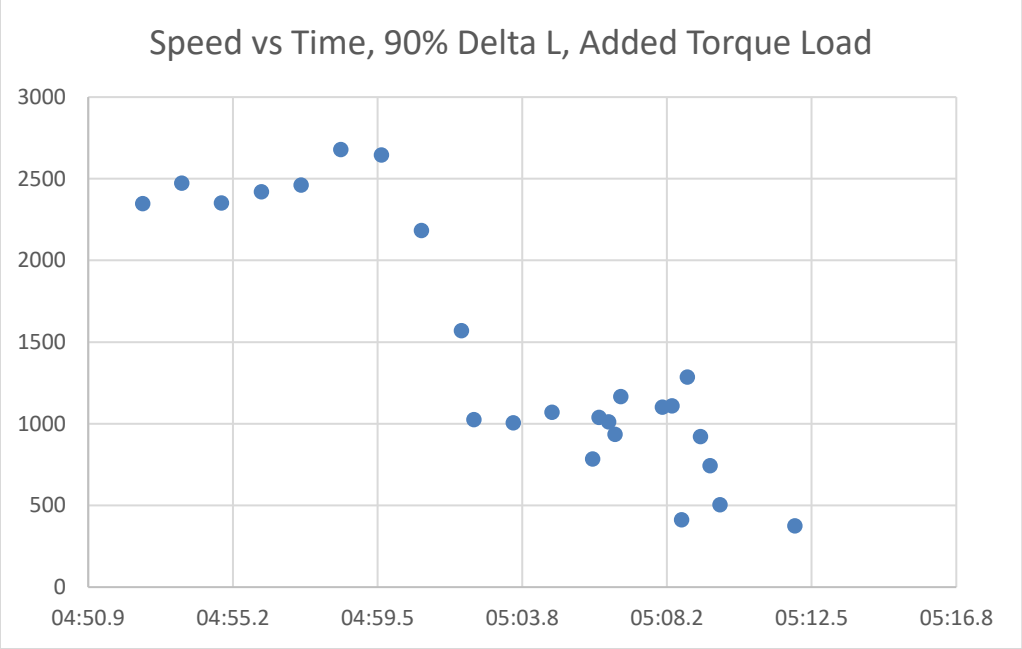

Figure 4.19: Erratic Motor Behavior with No Torque Load at 12V

For the erratic behavior conditions, the steady state speed was taken at the lower end of the speed curves. Interestingly enough, eminent failure is not only associated with a resistance drop but also a voltage drop, as seen in figure 4.16.

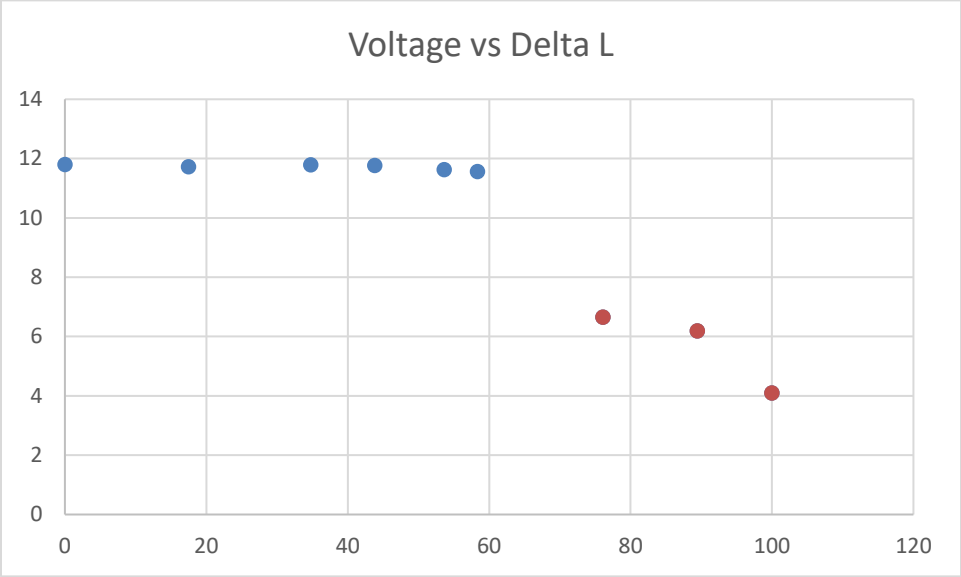

Figure 4.20: Voltage Drop at Eminent Failure at 12V

5. CONCLUSIONS

Carbon contact brushes used in PMDC are subjected to wear and become worn down from commutators continually moving across them. As the contacts length is decreased, the spring holding it in place becomes more stretched out, putting in more effort to hold the brush in place. This introduces a resistance, referred to as a contact resistance, that can affect the motor speed and performance. Changes in speed and resistance can be measured and observed, and curves can be fitted to their relationship with statistical significance. We can also create a simulation method using basic differential equations that describe the motor and introduce random noise to the simulation with generation of random numbers for the motor parameters. Finally, a prediction interval can be generated, and eminent motor failure can be predicted when values measured values stray from the simulated path. Erratic motor behavior can also be observed at the point of eminent motor failure, and the RUL can be estimated using generated equations describing the speed, length, and resistance relationships. This method could theoretically be performed on any permanent magnet DC motor with replaceable brushes, and could have applications to the industry, introducing a way of performing preventative maintenance to streamline the maintenance process and increase productivity.

REFERENCES

[1] P. Walker, “Direct Current Motors: Characteristics & Applications”, 1978

[2] J. Kimbrell “The DC Motor Advantage.” *StackPath*, 3 Nov. 2016, Available: www.processingmagazine.com/pumps-motors-drives/article/15586862/the-dc-motor-advantage.

[3] J. Zhang, W. Zhan, M. Ehsani, “On-line Diagnosis of Inter-turn Short Circuit Fault for DC Brushed Motor”, April 2018

[4] W. Abed, “Robust Fault Analysis for Permanent Magnet DC Motor in Safety Critical Applications”, *EThOS*, 2015, Available: https://ethos.bl.uk/OrderDetails.do?uin=uk.bl.ethos.686063

[5] A. Glowacz, "DC Motor Fault Analysis with the Use of Acoustic Signals, Coiflet Wavelet Transform, and K-Nearest Neighbor Classifier," *Archives of Acoustics*, vol. 40, *(3)*, pp. 321-327, 2015. Available: http://proxy.library.tamu.edu/login?url=https://search.proquest.com/docview/1860866514?accountid=7082. DOI: http://dx.doi.org/10.1515/aoa-2015-0035

[6] P. Ostojic, W. Basu, A. Banerjee, S. Ali, D. Patel and T. Smith, "Advanced motor monitoring and diagnostics," *Conference Record of 2014 Annual Pulp and Paper Industry Technical Conference*, Atlanta, GA, 2014, pp. 197-206

[7] S. Munikoti, L. Das, B. Natarajan and B. Srinivasan, "Data-Driven Approaches for Diagnosis of Incipient Faults in DC Motors," in *IEEE Transactions on Industrial Informatics*, vol. 15, no. 9, pp. 5299-5308, Sept. 2019.

[8] S. S. H. Zaidi, S. Aviyente, M. Salman, K. Shin and E. G. Strangas, "Prognosis of Gear Failures in DC Starter Motors Using Hidden Markov Models," in *IEEE Transactions on Industrial Electronics*, vol. 58, no. 5, pp. 1695-1706, May 2011.

[9] I. Culbert and J. Letal, "Signature Analysis for Online Motor Diagnostics: Early Detection of Rotating Machine Problems Prior to Failure," in *IEEE Industry Applications Magazine*, vol. 23, no. 4, pp. 76-81, July-Aug. 2017.

[10] M. A. Awadallah and M. M. Morcos, "Switch fault diagnosis of PM brushless DC motor drive using adaptive fuzzy techniques," in *IEEE Transactions on Energy Conversion*, vol. 19, no. 1, pp. 226-227, March 2004.

[11] M. A. Awadallah and M. M. Morcos, “Automatic Fault Diagnosis of Electric Machinery: A Case Study in PM Brushless DC Motors”, Taylor & Francis Online Electric Power Components and Systems, vol.33, iss.6, pp. 597-610, Aug. 2006

[12] M. A. Awadallah and M. M. Morcos, "Automatic diagnosis and location of open-switch fault in brushless DC motor drives using wavelets and neuro-fuzzy systems," in *IEEE Transactions on Energy Conversion*, vol. 21, no. 1, pp. 104-111, March 2006

[13] M. Hajiaghajani, H. A. Toliyat and I. M. S. Panahi, "Advanced fault diagnosis of a DC motor," in IEEE Transactions on Energy Conversion, vol. 19, no. 1, pp. 60-65, March 2004.

[14] M. Iorgulescu, M. Alexandru and R. Beloiu, "Noise and vibration monitoring for diagnosis of DC motor's faults," 2012 13th International Conference on Optimization of Electrical and Electronic Equipment (OPTIM), Brasov, 2012, pp. 724-729.

[15] S. Rajagopalan, J. M. Aller, J. A. Restrepo, T. G. Habetler and R. G. Harley, "Detection of Rotor Faults in Brushless DC Motors Operating Under Nonstationary Conditions," in IEEE Transactions on Industry Applications, vol. 42, no. 6, pp. 1464-1477, Nov.-dec. 2006.

[16] R. de Jesus Romero-Troncoso, "Multirate Signal Processing to Improve FFT-Based Analysis for Detecting Faults in Induction Motors," in *IEEE Transactions on Industrial Informatics*, vol. 13, no. 3, pp. 1291-1300, June 2017.

[17] M. Farajzadeh-Zanjani, R. Razavi-Far and M. Saif, "A Critical Study on the Importance of Feature Extraction and Selection for Diagnosing Bearing Defects," 2018 IEEE 61st International Midwest Symposium on Circuits and Systems (MWSCAS), Windsor, ON, Canada, 2018, pp. 803-808.

[18] W. Sun, R. Zhao, R. Yan, S. Shao and X. Chen, "Convolutional Discriminative Feature Learning for Induction Motor Fault Diagnosis," in IEEE Transactions on Industrial Informatics, vol. 13, no. 3, pp. 1350-1359, June 2017.

[19] C. Sun, M. Ma, Z. Zhao and X. Chen, "Sparse Deep Stacking Network for Fault Diagnosis of Motor," in IEEE Transactions on Industrial Informatics, vol. 14, no. 7, pp. 3261-3270, July 2018.

[20] R. Liu, G. Meng, B. Yang, C. Sun and X. Chen, "Dislocated Time Series Convolutional Neural Architecture: An Intelligent Fault Diagnosis Approach for Electric Machine," in IEEE Transactions on Industrial Informatics, vol. 13, no. 3, pp. 1310-1320, June 2017.

[21] R. Razavi-Far, M. Farajzadeh-Zanjani and M. Saif, "An Integrated Class-Imbalanced Learning Scheme for Diagnosing Bearing Defects in Induction Motors," in IEEE Transactions on Industrial Informatics, vol. 13, no. 6, pp. 2758-2769, Dec. 2017.

[22] X. Lou and K. A. Loparo, “Bearing fault diagnosis based on wavelet transform and fuzzy inference,” Mech. Syst. Signal Process., vol. 18, no. 5, pp. 1077–1095, Sep. 2004.

[23] H. Ocak, K. A. Loparo, and F. M. Discenzo, “Online tracking of bearing wear using wavelet packet decomposition and probabilistic modeling: A method for bearing prognostics,” J. Sound Vib., vol. 302, no. 4/5, pp. 951–961, May 2007.

[24] J. A. Rosero, L. Romeral, J. A. Ortega and E. Rosero, "Short-Circuit Detection by Means of Empirical Mode Decomposition and Wigner–Ville Distribution for PMSM Running Under Dynamic Condition," in IEEE Transactions on Industrial Electronics, vol. 56, no. 11, pp. 4534-4547, Nov. 2009.

[25] W. Q. Wang, M. F. Golnaraghi, and F. Ismail, “Prognosis of machine health condition using neuro-fuzzy systems,” Mech. Syst. Signal Process., vol. 18, no. 4, pp. 813–831, Jul. 2004.

[26] P. Baruah and R. B. Chinnam, “HMMs for diagnostics and prognostics in machining processes,” Int. J. Prod. Res., vol. 43, no. 6, pp. 1275–1293, Mar. 15, 2005.

[27] L. Atlas, M. Ostendorf, and G. Bernard, “Hidden Markov models for monitoring machining tool-wear,” in Proc. IEEE Int. Conf. Acoust., Speech, Signal Process., 2000, vol. 6, pp. 3887–3890.

[28] H. Razik, M. de Rossiter Correa, and E. da Silva, “A novel monitoring of load level and broken bar fault severity applied to squirrel-cage induction motors using a genetic algorithm,” IEEE Trans. Ind. Electron., vol. 56, no. 11, pp. 4615–4626, Nov. 2009.

[29] J.-R. Ruiz, J. Rosero, A. Espinosa, and L. Romeral, “Detection of demagnetization faults in permanent-magnet synchronous motors under nonstationary conditions,” IEEE Trans. Magn., vol. 45, no. 7, pp. 2961–2969, Jul. 2009.

[30] A. Bellini, F. Filippetti, C. Tassoni, and G.-A. Capolino, “Advances in diagnostic techniques for induction machines,” IEEE Trans. Ind. Electron., vol. 55, no. 12, pp. 4109–4126, Dec. 2008.

APPENDIX A

MATLAB

```
clear all
close all

%Motor Values
VALUES = xlsread('test10.xlsx');
[Sig, TStr, Raw] = xlsread('test10.xlsx',1,'A2:A50');
VALUESC = xlsread('current10.xlsx');
[SigC, TStrC, RawC] = xlsread('current10.xlsx',1,'A2:A500');
DC0 = find(VALUES(:,2)>0);
DC0C = find(VALUESC(:,2)>0);
MotorV = (mean(VALUES(DC0(end-8:end-3),1)))/50;
MotorW = (mean(VALUES(DC0(end-8:end-3),2)))/10;
MotorRNom = (mean(VALUES(DC0(end-8:end-3),3)));
MotorL = (mean(VALUES(DC0(end-8:end-3),4)));
MotorI = (mean(VALUESC(DC0C(end-62:end-1),1)));
MotorKV = (mean(VALUES(DC0(end-8:end-3),5)));
MotorK = (MotorKV)/(MotorW/9.55);
MotorTf = MotorK*MotorI;
MotorR = (MotorV-MotorKV)/MotorI;
MotorT = MotorTf;
MotorTL = MotorT - MotorTf;
Wchange = (VALUES(DC0(end)+1,2)-VALUES(DC0(end)-3,2));
timechange = datetime(Raw(DC0(end)+1))-datetime(Raw(DC0(end)-3));
[Year, Month, Day, Hour, Min, Secs] = datevec(timechange);
SecChange = (Hour*3600+Min*60+Secs);
dWdT = ((VALUES(DC0(end)+1,2)-MotorW)/9.55)/SecChange;
MotorJ = -MotorTf/dWdT;

%Increase .5
VALUES3 = xlsread('test40.xlsx');
[Sig3, TStr3, Raw3] = xlsread('test40.xlsx',1,'A2:A50');
VALUES3C = xlsread('current10.xlsx');
[Sig3C, TStr3C, Raw3C] = xlsread('current10.xlsx',1,'A2:A500');
DC3 = find(VALUES3(:,3)>0);
DC3C = find(VALUES3C(:,2)>0);
MotorV3 = (mean(VALUES3(DC3(end-24:end),2)))/50;
MotorW3 = (mean(VALUES3(DC3(end-24:end),3)))/10;
MotorRNom3 = (mean(VALUES3(DC3(end-24:end),4)));

%Increased Brush Resistance Values .75
```

```
VALUES6 = xlsread('test70.xlsx');
[Sig6, TStr6, Raw6] = xlsread('test70.xlsx',1,'A2:A50');
VALUES6C = xlsread('Current20.xlsx');
[Sig6C, TStr6C, Raw6C] = xlsread('Current20.xlsx',1,'A2:A500');
DC6 = find(VALUES6(:,2)>0);
DC6C = find(VALUES6C(:,2)>0);
MotorV6 = (mean(VALUES6(DC6(end-46:end-1),1)))/50;
MotorW6 = (mean(VALUES6(DC6(end-46:end-1),2)))/10;
MotorRNom6 = (mean(VALUES6(DC6(end-46:end-1),3)));

%Increased Brush Resistance Values 1
VALUES1 = xlsread('test20.xlsx');
[Sig1, TStr1, Raw1] = xlsread('test20.xlsx',1,'A2:A50');
VALUES1C = xlsread('Current20.xlsx');
[Sig1C, TStr1C, Raw1C] = xlsread('Current20.xlsx',1,'A2:A500');
DC1 = find(VALUES1(:,2)>0);
DC1C = find(VALUES1C(:,2)>0);
MotorV1 = (mean(VALUES1(DC1(end-7:end),1)))/50;
MotorW1 = (mean(VALUES1(DC1(end-7:end),2)))/10;
MotorRNom1 = (mean(VALUES1(DC1(end-7:end),3)));


%Increase 1.5
VALUES4 = xlsread('test50.xlsx');
[Sig4, TStr4, Raw4] = xlsread('test50.xlsx',1,'A2:A50');
VALUES4C = xlsread('current20.xlsx');
[Sig4C, TStr4C, Raw4C] = xlsread('current20.xlsx',1,'A2:A500');
DC4 = find(VALUES4(:,3)>0);
DC4C = find(VALUES4C(:,2)>0);
MotorV4 = (mean(VALUES4(DC4(end-15:end-1),2)))/50;
MotorW4 = (mean(VALUES4(DC4(end-15:end-1),3)))/10;
MotorRNom4 = (mean(VALUES4(DC4(end-15:end-1),4)));


%Increased Brush Resistance Values 2
VALUES2 = xlsread('test30.xlsx');
[Sig2, TStr2, Raw2] = xlsread('test30.xlsx',1,'A2:A50');
VALUES2C = xlsread('current30.xlsx');
[Sig2C, TStr2C, Raw2C] = xlsread('current30.xlsx',1,'A2:A500');
DC2 = find(VALUES2(:,2)>0);
DC2C = find(VALUES2C(:,2)>0);
MotorV2 = (mean(VALUES2(DC2(end-7:end-2),1)))/50;
MotorW2 = (mean(VALUES2(DC2(end-7:end-2),2)))/10;
MotorRNom2 = (mean(VALUES2(DC2(end-7:end-2),3)));
```

```
%Increase 2.5
VALUES5 = xlsread('test60.xlsx');
[Sig5, TStr5, Raw5] = xlsread('test60.xlsx',1,'A2:A50');
VALUES5C = xlsread('current30.xlsx');
[Sig5C, TStr5C, Raw5C] = xlsread('current30.xlsx',1,'A2:A500');
DC5 = find(VALUES5(:,3)>0);
DC5C = find(VALUES5C(:,2)>0);
MotorV5 = (mean(VALUES5(DC5(end-30:end-4),2)))/50;
MotorW5 = (mean(VALUES5(DC5(end-30:end-4),3)))/10;
MotorRNom5 = (mean(VALUES5(DC5(end-30:end-4),4)));

%Increased Brush Resistance Values 3
VALUES7 = xlsread('test80.xlsx');
[Sig7, TStr7, Raw7] = xlsread('test80.xlsx',1,'A2:A50');
VALUES7C = xlsread('Current20.xlsx');
[Sig7C, TStr7C, Raw7C] = xlsread('Current20.xlsx',1,'A2:A500');
DC7 = find(VALUES7(:,2)>0);
DC7C = find(VALUES7C(:,2)>0);
MotorV7 = (mean(VALUES7(DC7(end-61:end-3),1)))/50;
MotorW7 = (mean(VALUES7(DC7(end-61:end-3),2)))/10;
MotorRNom7 = (mean(VALUES7(DC7(end-61:end-3),3)));

%Increased Brush Resistance Values last
VALUES8 = xlsread('test90.xlsx');
[Sig8, TStr8, Raw8] = xlsread('test90.xlsx',1,'A2:A50');
DC8 = find(VALUES8(:,2)>0);
MotorV8 = (mean(VALUES8(DC8(end-37:end-1),1)))/50;
MotorW8 = (mean(VALUES8(DC8(end-37:end-1),2)))/10;
MotorRNom8 = (mean(VALUES8(DC8(end-37:end-1),3)));


%Setting Nominal Values
RNom = MotorRNom;                   %Nominal R (Ohm)
LNom = MotorL;                %Nominal L (H)enry
JNom = MotorJ;          %Nominal J (kg-m^2)
KNom = MotorK;                  %Back EMF, conversion (Nm/A)
Tload=MotorTL;                  %Nominal Tload (N-m)
Tfriction = MotorTf;              %Average Friction Force, conversion, (N-m)
VbattNom = MotorV;                    %Nominal Vbatt (V)
TempChangeNom = 0.01;           %Nominal Temperature Change (C)
Period = 1/500;               %PWM Generator Period, (s)(1/Hz)
DC = 100;                     %Duty Cycle (%)
```

```
%Simulation Inputs
multi=1000                  %UN COMMENT IF NOT FINDING ZERO SPEED
RESISTANCE
%Rchange=0.5    %Stepside of Resistance Increase
Q=3;       %Number of Resistance Increases for actual simulations
S=5*multi;      %Number of Randomly Generated Numbers
B=200;       %Number of Bins in Histograns
P=.001;      %Stepsize of Simulation
F=0.025;     %Stepsize of Zero Speed Estimation Resistance Increase
Time=2;     %Length of Simulation Time in Seconds
Inc=100;      %Percent Increase Of Estimate
IncReset=Inc;

%Setting Random Generation Values
Rmean = log(RNom);                          %Nominal R Ohm
if Rmean < 0;
   Rstd = -Rmean*0.05;                      %R stdev
else
   Rstd = Rmean*0.05;
end
Lmean = log(LNom);                          %Nominal L (H)enry
if Lmean < 0;
   Lstd = -Lmean*.05;                       %L stdev
else
   Lstd = Lmean*0.05;
end
Jmean=log(JNom);                            %Nominal J kg-m^2
if Jmean < 0;
   Jstd = -Jmean*.05;                       %J stdev
else
   Jstd = Jmean*0.05;
end
Kemfmean  = log(KNom);                      %Back EMF, conversion (Nm/A)
if Kemfmean < 0;
   Kemfstd = -Kemfmean*0.05;                %Back EMF Standard Deviation
else
   Kemfstd = Kemfmean*0.05;
end
Tempchangemean = log(TempChangeNom);        %Nominal Temperature
Change (C)
if Tempchangemean < 0
   Tempchangestd = -Tempchangemean*0.05;    %Temperature Change stdev
else
   Tempchangestd = Tempchangemean*0.05;
```

```
end                                                    %PWM Generator Duty
Cycle, (%)
Vstd = VbattNom*0.05;

%Random Number Generation
Rmean_Rand=lognrnd(Rmean,Rstd,[S,1]);              %Random Generation for R
L_Rand=lognrnd(Lmean,Lstd,[S,1]);                  %Random Generation for L
J_Rand=lognrnd(Jmean,Jstd,[S,1]);                  %Random Generation for J
Kemfmean_Rand=lognrnd(Kemfmean,Kemfstd,[S,1]);     %Random Generation
for K
Tempchange_Rand=lognrnd(Tempchangemean,Tempchangestd, [S,1]); %Random
Generation for Temperature Change
Vbatt_Rand=(VbattNom-Vstd)+(2*Vstd).*rand([S,1]);          %Random
Generation for Vbatt

MaxR=1;     %Max Resistance Change to determine Rchange
Rchange=MaxR/(Q);              %Resistance Increase Interval

%Actual Simulation of Nominal Set
for i=0:Q
    RtempConv = TempChangeNom.*0.00393;      %Convert Temperature Change to
Resistance
    Rtemp = RtempConv;                 %Setting Temperature Resistance
    R = RNom;                  %Setting Resistance
    Rcontact = Rchange*i;
    Linv = 1/(LNom);                   %Setting Inductance Inverse
    Jinv = 1/(JNom);                   %Setting Inertia Inverse
    Kemfmean = KNom;                   %Setting Kemf
    Tload = Tload;                 %Setting Tload as Constant
    Vbatt = VbattNom;                  %Setting Voltage
    Kemf=Kemfmean;                     %Back EMF
    Ktorque=Kemf;                  %Back EMF
    [T,X,Y]=sim('DCPMmotorSteady.slx', [0:P:Time]);    %0 to 2s, intervals of .001,
switch to DCPMmotor.mdl for PWM
    Speed=Y(:,1);                      %Acquire speed
    Speed0(:,1) = Speed;                   %Create array
    Speed00(1,i+1) = max(Y(:,1));  %Finding max speed
%    %UNCOMMENT THESE TO MONITOR THE SPEED PLOTS BEING
PRODUCED
%        figure((11));                  %Setting figure 1
%        plot(T,Speed)                      %Plot speed vs time
%        title({strcat('Speed Over Time, Constant Tload,
R=Rnom+',num2str(Rchange),'*(0:',num2str(i),')')});        %Set title
%        xlabel('Time');                    %label x axis
```

```
%         ylabel('Speed RPM');                    %label y axis
%         hold on                          %Hold for continuous graphing to monitor
end
hold off

%Estimate Speeds of Nominal Set
Rchangeperc = ((Rchange)/RNom)*100 ;
Rchangeend = (((MaxR))/RNom)*100;
U1 = [0:Rchangeperc:Rchangeend];                    %Setting the y interval
Speed00neg(1,:) = (find(Speed00(1,:)<=0));          %Find the location of the zero or neg
speed in matrix
Speed00new = Speed00;
Speed00new(Speed00neg) = 0;
c1 = polyfit(U1,Speed00new,1);                      % Here 'c' contains the 'm' and 'b'
disp(['Line Equation of Nominal Set is y = F(x) = ',num2str(c1(1)),'*x +
','(',num2str(c1(2)),')']);
disp(['Zero Speed Resistance of Nominal Set = ',num2str(-c1(2)/c1(1)),'% ', newline]);
Speed00_est=polyval(c1,U1);
for x = 0:Inc
   Speed00_estnew(1,x+1)=c1(1)*x+c1(2);
   while Speed00_estnew(end) > 0
      Inc=Inc+1;
      Speed00_estnew(1,Inc+1)=c1(1)*Inc+c1(2);
   end
   if Speed00_estnew(end) < 0;
      Speed00_estnew(end) =0;
   end
end
Speed00zero = min(find(Speed00_estnew(1,:)<=0));
Speed00_estnew(Speed00zero) = 0;
Speed00_estnew = Speed00_estnew(1:(Speed00zero));
U11 = [0:(Speed00zero-1)];
U11(end) = -c1(2)/c1(1);
%UNCOMMENT THESE TO MONITOR THE PLOTS BEING PRODUCED
figure(1);
Inc=IncReset;
plot(U1,Speed00new)
xlabel('\Delta R (%)');                             %Setting x label
ylabel('Speed RPM');                                %Setting y label
title({strcat('Speed vs Resistance of Nominal Values')});          %Set title
grid on
hold on
plot(U11,Speed00_estnew)
hold off
```

```
%Actual Simulation of Random Number Sets
for i=0:Q
   for k=1:S                                %for loop for each random number
       RtempConv = Tempchange_Rand.*0.00393;      %Convert Temperature Change to
Resistance
       Rtemp = RtempConv(k,1);                %Setting Temperature Resistance
       R = RNom;          %Setting Resistance
       Rcontact = Rchange*i;
       R_s(k,i+1)=R;
       Linv = 1/(L_Rand(k,1));                %Setting Inductance Inverse
       Jinv = 1/(J_Rand(k,1));                %Setting Inertia Inverse
       Kemfmean = Kemfmean_Rand(k,1);          %Setting Kemf
       %Tload = Tload_Rand(k,1);               %Setting Tload for Random
       Tload = Tload;                         %Setting Tload as Constant
       Vbatt = Vbatt_Rand(k,1);               %Setting Voltage
       Kemf=Kemfmean;                         %Back EMF
       Ktorque=Kemf;                          %Back EMF
       [T,X,Y]=sim('DCPMmotorSteady.slx', [0:P:Time]);     %0 to 2s, intervals of .001,
switch to DCPMmotor.mdl for PWM
       Speed=Y(:,1);                          %Acquire speed
       Speed1(:,1) = Speed;                   %Create array
       Speed2(k,i+1) = max(Y(:,1));   %Finding mean speed
%        %UNCOMMENT THESE TO MONITOR THE SPEED PLOTS BEING
PRODUCED
%        figure((i+3));                        %Setting figure 1
%        plot(T,Speed)                         %Plot speed vs time
%        title({strcat('Speed Over Time, Constant Tload,
R=Rrand+',num2str(Rchange*i))});          %Set title
%        xlabel('Time');                       %label x axis
%        ylabel('Speed RPM');                  %label y axis
%        hold on                          %Hold for continuous graphing to monitor
   end                                  %End random number for loop
   Speed3 = mean(Speed2,1);
   hold off
   Ravg= mean(Rmean_Rand(:,1),1);
end
steps = 2;
stepsinv = 1/steps;
%Estimate Speeds of Random Number Sets
for n=1:S
   Speed2_estnew = [];
   U22 = [];
   x = [];
```

```
    Inc=IncReset;
    Rchangeperc = ((Rchange)/Rmean_Rand(n,1))*100;
    Rchangeend = (((Rchange*Q))/Rmean_Rand(n,1))*100;
    U2 = [0:Rchangeperc:Rchangeend];                    %Setting the y interval
    c2(n,:) = polyfit(U2,Speed2(n,:),1);                % Here 'c' contains the 'm' and 'b'
    Speed2_est(n,:)=polyval(c2(n,:),U2);
    for x = 0:stepsinv:Inc
        Speed2_estnew(1,x*steps+1)=c2(n,1)*x+c2(n,2);
        while Speed2_estnew(end) > 0
            Inc=Inc+stepsinv;
            Speed2_estnew(1,Inc*steps+1)=c2(n,1)*Inc+c2(n,2);
        end
    end
    Speed2zero = min(find(Speed2_estnew(1,:)<=0));
    Speed2_estnew1 = Speed2_estnew;
    Speed2_estnew1 = Speed2_estnew(1:(Speed2zero));
    U22 = [0:(Speed2zero-1)];
    figure((i+4));                          %Setting figure 1
    lengths(n,:) = length(Speed2_estnew);
    M1{n,1} = Speed2_estnew;
    M11{n,1} = Speed2_estnew1;
    M2{n,1} = U22;
%    %UNCOMMENT THESE TO MONITOR THE SPEED PLOTS BEING
PRODUCED
%     plot(U22,Speed2_estnew1(1,:))                       %Plot speed vs time
%     title({strcat('Speed vs Resistance Estimated of Random Numbers')});          %Set
title
%     xlabel('\Delta R (%)');                       %Setting x label
%     ylabel('Speed RPM');                    %label y axis
%     grid on
%     hold on                           %Hold for continuous graphing to monitor
end

% hold off
maxlength = max(lengths);
v = [0:maxlength-1];
v1 = repelem(v,[S],[1]);
%Estimate Average Speeds of Random Number Sets
Rchangeperc = ((Rchange)/Ravg)*100;
Rchangeend = (((Rchange*i))/Ravg)*100;
U3 = [0:Rchangeperc:Rchangeend];                    %Setting the y interval
Speed3neg(1,:) = (find(Speed3(1,:)<=0));          %Find the location of the zero or neg
speed in matrix
Speed3new = Speed3;
```

```
Speed3new(Speed3neg) = 0;
c3 = polyfit(U3,Speed3new,1);                    % Here 'c' contains the 'm' and 'b'
disp(['Line Equation Weird is y = F(x) = ',num2str(c3(1)),'*x + ','(',num2str(c3(2)),')']);
Speed3_est=polyval(c3,U3);
SSE1 = sum((Speed3new-Speed3_est).^2);
SSyy1 = sum((Speed3new-mean(Speed3)).^2);
Rsq1 = 1-SSE1/SSyy1;
for x = 0:stepsinv:Inc
   Speed3_estnew(1,x*steps+1)=c3(1)*x+c3(2);
   while Speed3_estnew(end) > 0
      Inc=Inc+stepsinv;
      Speed3_estnew(1,Inc*steps+1)=c3(1)*Inc+c3(2);
   end
   if Speed3_estnew(end) < 0;
      Speed3_estnew(end) =0;
   end
end
Speed3zero = min(find(Speed3_estnew(1,:)<=0));
Speed3_estnew(Speed3zero) = 0;
Speed3_estnew = Speed3_estnew(1:(Speed3zero));
U33 = [0:(Speed3zero-1)];                    %Setting the y interval
U33(end) = -c3(2)/c3(1);              % setting y to zero, solving for x
disp(['Zero Speed Resistance Weird = ',num2str(-c3(2)/c3(1)),'%', newline]);
% %UNCOMMENT THESE TO MONITOR THE SPEED PLOTS BEING
PRODUCED
% figure(i+5);
% Inc=IncReset;
% plot(U3,Speed3new)
% xlabel('\Delta R (%)');                              %Setting x label
% ylabel('Speed RPM');                                 %Setting y label
% title({strcat('Speed vs Resistance Estimated Average')});          %Set title
% grid on
% hold on
% plot(U33,Speed3_estnew)
% hold off
% SSE1 = sum((Speed3-Speed3_est).^2);
% SSyy1 = sum((Speed3-mean(Speed3)).^2);
% Rsq1 = 1-SSE1/SSyy1;

%Prediction Interval
[~,I] = sort(cellfun(@length,M1),'descend');
M3 = M1(I);
lengths1 = sort(lengths,'descend');
for n = 1:maxlength
```

```
    for k=1:S
        M4(k,:) = M3{k}(n);
        if n == 1
            M6(k,:) = M3{k}(n);
        end
        if n == 50
            M7(k,:) = M3{k}(n);
        end
        if n==lengths1(S);
            S=S-1;
        end
    end
    M5(:,n) = M4;
    N1 = S;                          % Number of 'Experiments' In Data Set
    N2(n,1) = N1;
    if N1 > 1
        CI95notnan = [1:n];
    end
    SpeedMean(1,n) = mean(M4);
    SpeedMSE(1,n) = sqrt(((sum(M4-SpeedMean(1,n)^2)/N1))*(1+1/N1));
% Compute 'Standard Error Of The Mean' Of All Experiments At Each Value Of 'x'
    CI95(:,n) = tinv([0.025 0.975], N1-1);          % Calculate 95% Probability
Intervals Of t-Distribution
    if CI95 == NaN
        CI95(:,n) = 0;
    end
    CI951 = CI95(:,CI95notnan);
    SpeedMean1 = SpeedMean(:,CI95notnan);
    SpeedMSE1 = SpeedMSE(:,CI95notnan);  %Standard error sigma/sqrt(n)
    CI95notnanlength=length(CI95notnan);
    if n-1<CI95notnanlength
        SpeedCI95 = bsxfun(@times, SpeedMSE1, CI951(:,n)); %Piecewise multiplication
of t(alpha/2) and sigma/sqrt(n)
    end
end
% %UNCOMMENT THESE TO MONITOR THE SPEED PLOTS BEING
PRODUCED
% figure(9);
% U44 = [0:CI95notnan(end)-1];
% line(U44, SpeedCI95act,'Color','r')
% title({strcat('Speed vs Resistance Confidence Interval')});          %Set title
% xlabel('\Delta R (%)');                          %Setting x label
% ylabel('Speed RPM');                             %Setting y label
% grid on
```

```
%Length to Resistance
L0 = 15.84;
L3 = 13.07;
L6 = 10.33;
L1 = 8.9;
L4 = 7.34;
L2 = 6.6;
L5 = 3.79;
L7 = 1.67;
L0PERC = ((L0-L0)/L0)*100;
L1PERC = ((L0-L1)/L0)*100;
L2PERC = ((L0-L2)/L0)*100;
L3PERC = ((L0-L3)/L0)*100;
L4PERC = ((L0-L4)/L0)*100;
L5PERC = ((L0-L5)/L0)*100;
L6PERC = ((L0-L6)/L0)*100;
L7PERC = ((L0-L7)/L0)*100;
L8PERC = 100;
figure(2)
R0PERC = ((MotorRNom-MotorRNom)/MotorRNom)*100;
R1PERC = ((MotorRNom1-MotorRNom)/MotorRNom)*100;
R2PERC = ((MotorRNom2-MotorRNom)/MotorRNom)*100;
R3PERC = ((MotorRNom3-MotorRNom)/MotorRNom)*100;
R4PERC = ((MotorRNom4-MotorRNom)/MotorRNom)*100;
R5PERC = ((MotorRNom5-MotorRNom)/MotorRNom)*100;
R6PERC = ((MotorRNom6-MotorRNom)/MotorRNom)*100;
R7PERC = ((MotorRNom7-MotorRNom)/MotorRNom)*100;
R8PERC = ((MotorRNom8-MotorRNom)/MotorRNom)*100;
figure(3)
scatter(L0PERC, R0PERC)
hold on
scatter(L3PERC, R3PERC)
hold on
scatter(L6PERC, R6PERC)
hold on
scatter(L1PERC,R1PERC);
hold on
scatter(L4PERC,R4PERC);
hold on
scatter(L2PERC,R2PERC)
hold on
scatter(L5PERC,R5PERC)
hold on
```

```
scatter(L7PERC, R7PERC)
hold on
scatter(L8PERC, R8PERC)
title('Length vs Resistance');  %Setting Title
xlabel('\Delta Length (%)');                              %Setting x label
ylabel('\Delta R (%)');
hold off

figure(11)
scatter(R0PERC, MotorW);
hold on
scatter(R3PERC, MotorW3);
hold on
scatter(R6PERC, MotorW6);
hold on
scatter(R1PERC, MotorW1);
hold on
scatter(R4PERC, MotorW4);
hold on
scatter(R2PERC, MotorW2);
hold on
scatter(R5PERC, MotorW5);
hold on
scatter(R7PERC, MotorW7);
hold on
scatter(R8PERC, MotorW8);
title('Length vs Resistance Ecperimental Data');  %Setting Title
xlabel('\Delta R (%)');                                %Setting x label
ylabel('Speed RPM');
hold off


figure(10);                  %Setting the y interval
U66 = [0:size(SpeedMean1,2)-1];
U666 = U66.';
SpeedMean1fit = SpeedMean1.';
c6 = polyfit(U66(1:20),SpeedMean1(1:20),1);
Speedaverages_est=polyval(c6,U66);
%U444(end) = -c4(2)/c4(1);
U66(end) = -c6(2)/c6(1);
disp(['Line Equation of Average is y = F(x) = ',num2str(c6(1)),'*x +
','(',num2str(c6(2)),')']);
disp(['Zero Speed Resistance of Averages = ',num2str(-c6(2)/c6(1)),'Ohms', newline]);
Speedaverages_estneg = min(find(Speedaverages_est(1,:)<=0));
```

```
maxs = max(M5);
U7 = [0:size(maxs,2)-1];
mins = min(M5);
U8zero = min(find(mins(1,:)<=0));
U8 = [0:U8zero-1];
M5=M5(:,1:size(SpeedMean1,2));
v1 = v1(:,1:size(SpeedMean1,2));
M5linear = M5(:);
v1linear = v1(:);
fitresult = fit(v1linear,M5linear,'poly2');
p12 = predint(fitresult,v1linear,0.95,'observation','on');
plot(U66(1:Speedaverages_estneg)./steps,Speedaverages_est(1:Speedaverages_estneg));
hold on
grid on
SpeedCI95act = SpeedCI95+Speedaverages_est;
SpeedCI95act1L = p12(:,1);
SpeedMean1zero = min(find(SpeedMean1(1,:)<=0));
SpeedMean1act = SpeedMean1;
SpeedMean1act(SpeedMean1zero) = 0;
SpeedMean1act = SpeedMean1act(1:(SpeedMean1zero));
SpeedCI95act1U = p12(:,2);
U444 = [0:size(SpeedCI95act1L)-1];                    %Setting the y interval
U4444 = U444./(10);
U555 = [0:size(SpeedCI95act1U)-1];
U5555 = U555./10;
c4 = polyfit(U4444(1:20000),SpeedCI95act1L(1:20000).',1);
StraightL = polyval(c4,U4444);
straightzeroL = min(find(StraightL<=0));
UStraightL = [0:straightzeroL-1]./(10*multi);
UStraightL(end)=(-c4(2)/c4(1))/((10/10)*multi);
StraightL(straightzeroL)=0;
StraightL = StraightL(1:straightzeroL);
c5 = polyfit(U5555(1:20000),SpeedCI95act1U(1:20000).',1);
StraightU = polyval(c5,U5555);
UStraightU = [0:straightzeroL-1]./(10*multi);
StraightU(straightzeroL)=0;
StraightU = StraightU(1:straightzeroL);
UStraightU(end)=(-c5(2)/c5(1))/((10/10)*multi);
line(UStraightL, StraightL,'Color','r');
hold on
line(UStraightU, StraightU,'Color','c');
hold on
SpeedMean1act_est=polyval(c6,U66);
```

```
SSE2 = sum((SpeedMean1(1:5)-SpeedMean1act_est(1:5)).^2);
SSyy2 = sum((SpeedMean1-mean(SpeedMean1)).^2);
Rsq2 = 1-SSE2/SSyy2;
scatter(R0PERC, MotorW);
hold on
 scatter(R3PERC, MotorW3);
 hold on
scatter(R6PERC, MotorW6);
hold on
scatter(R1PERC, MotorW1);
hold on
 scatter(R4PERC, MotorW4);
 hold on
scatter(R2PERC, MotorW2);
hold on
 scatter(R5PERC, MotorW5);
 hold on
 scatter(R7PERC, MotorW7);
 hold on
scatter(R8PERC, MotorW8);
hold on
legend11 = sprintf('Avg Data (Random #s), R^{2} = %g', Rsq2);                    %Set
legend
legend10 = sprintf('Lower CI');
legend16 = sprintf('Upper CI')
legend(legend11,legend10,legend16,'Location','northeast');
title('Avg Speed of Sets of Randomly Generated Variables, No Torque Load');  %Setting
Title
xlabel('\Delta R (%)');                              %Setting x label
ylabel('Speed RPM');                                 %Setting y label
hold off
```