INVESTIGATION AND MODELING OF VOLTAGE AND CURRENT UNBALANCE ON A THREE-PHASE SUBMERSIBLE PUMP

by

JAMES ANDREW REED

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE ELECTRICAL ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

May 2013

Acknowledgements

First I would like to thank my advisor Dr.Wetz, for his support and assistance throughout this project. His guidance provided me insights which allowed me to complete this project successfully. I would also like to thank the additional members of my committee Dr.Kenarangui and Dr.Lee, for their review of this thesis and the comments they provided.

A huge thank you to my colleagues and labmates Jay, Travis, Peter, Biju and Caroline, who sat through many brainstorming sessions over the course of this project. Your assistance, advice, comments, and feedback helped me create the program into the best it could be.

I would also like to thank my parents, sister, and family as without them I would not have been able to complete this project.

April 5, 2013

Abstract

INVESTIGATION AND MODELING OF VOLTAGE AND CURRENT UNBALANCE ON A THREE-PHASE SUBMERSIBLE PUMP

James Andrew Reed, M.S. E.E.

The University of Texas at Arlington, 2013

Supervising Professor: David Wetz, Ph.D.

During the summer of 2011, a North Texas subdivision repeatedly experienced unexpected tripping with a submersible pump, which transfers water from an underground well to an above ground water storage tank. The tripping events were random in nature but occurred often. Suspecting that unbalanced phase voltages/currents were to blame, the local electrical cooperative sampled the line voltages/currents for several days in order to understand what the caused the pump to trip. This thesis will discuss the analysis of that data and the development of a Simulink® based computer aided model which can predict the behavior of both the submersible pump and the motor controller which monitors and controls the motor operation. It will be shown through analysis of the data that the motor protection system tripped as a result of current unbalances and that the computer aided model can accurately predict this as well as normal modes of operation.

Table of Contents

List of Illustrations

List of Tables

Chapter 1

Introduction:

Induction motors are part of almost every human's life, whether they realize it or not. A very small subset of their applications include driving motors for ceiling fans, washing machines, dryers, air conditioning systems, assembly lines, small-scale power generation, and water pumping systems. The latter application is of particular interest in the research effort documented here.

A North Dallas, Texas subdivision utilizes three submersible pumps to harvest water from underground wells into an above ground storage and distribution system. Two of the installed motors, which will not be discussed in detail here, are smaller than the larger and run in a near continuous mode of operation to maintain the average water usage of the community. The third pump, which is larger than the former two, is a Franklin 40 Horsepower with a 6 inch discharge and quickly transfers water to the above ground storage when the water capacity goes low and needs refilling. Due to its the larger size and cost investment, a commercially available motor controller, a Symcom Motor Saver (SMS) Model 777-HVR, is manages the larger motor's operation and safety. The smaller motors rely only on circuit breaker and fuse protection.

In the summer of 2011, record heats were recorded for the majority of the summer. The Symcom motor controller experienced numerous unexpected trip events, which left the water reserve extremely low for an unacceptable amount of time [1]. The community and the local electric cooperative launched an investigation into the events causing the controller to trip. A local utility applied several immediate changes that fixed the tripping issues, which will be described later. However, what caused the controller to trip was not easily understood [1]. As a result, the University of Texas at Arlington (UTA) was brought in to help analyze the data, and determine what caused the trip events.

Simultaneously, UTA was asked to develop a computer-aided model, which could predict the controller and pump operation when exposed to random inputs or user defined inputs.

This report documents the steps taken by a local electric utility, the analysis they performed, as well as the analysis and model developed by UTA. The model was developed using Matlab/Simulink® software tools and incorporates the electrical properties of both the motor and the Symcom motor controller.

As will be shown, two unique versions of the model were developed. The first version allows the user to model the system using randomly defined input signals which are the software generates by using maximum and minimum values input by the user. This allows the user to simulate the normal operation of the system including random events which a user may never consider to occur. The second version allows the user to input real system data that is recorded using analog to digital data acquisition devices while the system is operational. This allows the user to evaluate and understand how the system will or should have responded when the data recorded occurred.

Induction Motors:

In 1824, François Arago demonstrated the theory of rotating magnetic fields, which forms the theoretical basis for transformers and induction motors [2]. The credit for these devices are often given to Nikola Tesla since he filed US patents in 1888 for a simplistic induction motor that he demonstrated as a working model in 1887 to the AIEE (American Institute of Electrical Engineers) [3]. History reveals in fact that both Tesla and Galileo Ferraris developed their own versions of a rotating induction machine at approximately the same time [4]. Tesla's work was performed in the United States while Ferraris was based in Europe. At the time, Tesla worked in collaboration with Westinghouse, a US company who played a major role in the development of the AC

(Alternating Current) electrical power distribution grid [5]. As previously mentioned, induction motors and machines are used in a wide variety of applications that are used by nearly every single person on a daily basis. In fact, electric motors make up 57% of the world's electrical energy use [6]. Figure 1 presents a simple exploded view of an induction motor.

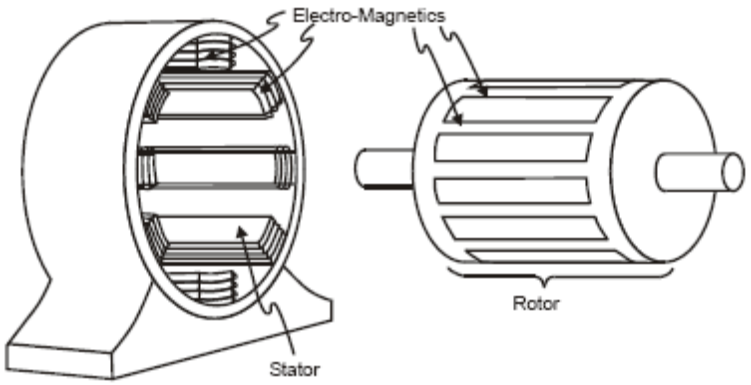

Figure 3-24. Rotor and stator

Figure 1: Stator and Rotor Example [7]

As seen in Figure 1, induction machines are made up of two main components. The first is the rotor that rotates during operation and the second is the stator, which encompasses the rotor and remains stationary. Induction motors are essentially rotating transformers. The introduction of a time varying sinusoidal voltage into the windings of the stator induces voltage into the rotor. Interactions between the magnetic and electric fields present in both the rotor and stator results in the electromagnetic forces which cause the rotor, and therefore the shaft attached to it, to spin. This process is seen graphically in Figure 2. As shown, the induced current in the rotor generates an opposing magnetic field to that created in the stator, thereby inducing the rotation in the form of a push pull. Unlike a synchronous motor in which electric fields in the stator and the rotor rotate at the same rate, an induction motor is asynchronous, meaning that the rotor's

fields spin slightly slower than the stator's fields, resulting a process referred to as the slip [8] [9]. As the motor works to match the output torque to the required load torque, the slip starts at 100% (complete slip), this also results in high induced currents or inrush currents. As the motor begins to provide enough torque to the load, the slip is reduced until it reaches a steady state of approximately 3 – 4%.

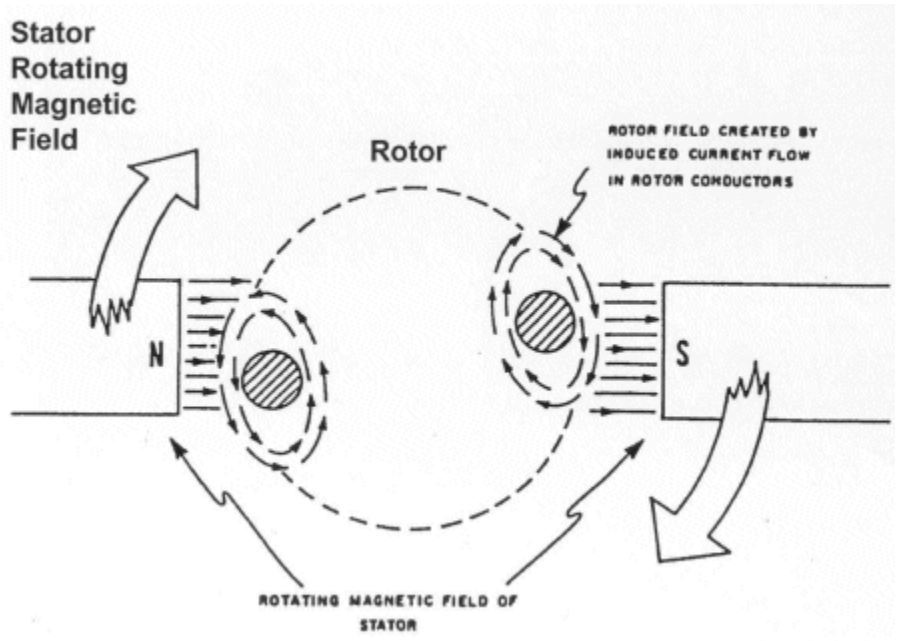

Figure 2: Example of magnetic fields and rotor rotation [10]

The rotation of an electric machine can be described mathematically using a few fundamental equations. The first equation is a special case of Faraday's Law, equation 1, which is used to calculate the Electromotive force (EMF). This law states that a voltage will be induced on any conductor if it moves with some velocity in a direction perpendicular to a magnetic field [8].

$$V = BLv$$

Equation 1: Induced voltage from a magnetic field [8]

As seen in equation 1, the voltage, V, induced on the conductor can be solved by multiplying the magnetic flux density, B, the conductor length, L, and the velocity of the conductor, v, as it moves through the magnetic field [8]. Using Faraday's law along with the Lorentz force equation, equation 2, the current induced in the conductor and strength of the magnetic field can be related to each other. The Lorentz force is solved using the cross product relationship between the current delivered to the conductor, the length of the conductor, and the magnetic field.

$$F = BLI\sin(\varphi)$$

Equation 2: Lorentz force [8]

In Equation 2, B is still the magnetic flux density in Tesla, L is the length of the conductor in m, I is the current through the conductor which is measured in Amps, and φ is the angle difference between the magnetic field and the current with units of Radians. The maximum value of the Lorentz force occurs when the phase angle between the current flowing through the conductor and magnetic field that the conductor passes through is 90 degrees.

In the case of a three-phase induction motor, where a 3-Phase AC voltage is supplied to a motor, magnetizing currents, separated by the 120 degrees each, are induced in three interconnected single phase motors. The total flux in the motor is the sum of the flux in all three phases.

There are two primary types of induction motors. Each of them utilizes a unique rotor design. The first type is typically referred to as a Squirrel cage motor, seen in Figure 3, as a result of the rotor's cage like appearance. This type of rotor is made using conducting bars that span the length of the rotor and are shorted together at the end by conductive rings [9].

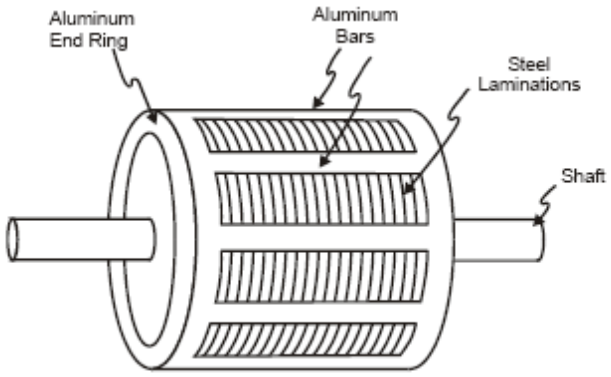

Figure 3-23. Squirrel cage induction motor rotor

Figure 3: Graphical [7] (top) and actual (bottom) squirrel cage motor [11]

The second type of induction motor is known referred to as a slip ring design, seen in Figure 4. A few differences between this type of motor and a squirrel cage motor involve the addition of winding turns on the rotor and the introduction of starting resistors. The windings are not shorted together like they are in the squirrel cage motor design; instead they are connected in series with the starting resistors, which are shorted out once the motor reaches the desired speed. The increased windings cause the induced voltage to be higher which in turn reduces the current since the overall resistance increases. This allows this type of motor to be used when reduced input power is

required. The starting resistors are variable, decreasing from a state of higher impedance to lower impedance as the motor starts up. This reduces the inrush current until full speed is achieved, at which point the slip rings are shorted in a similar fashion as they are in a squirrel cage motor. Additionally, the resistors enable a higher starting torque to be obtained since the inrush current fluctuations are lower at the start.

Figure 4: Slip ring or wound motor rotor [12]

As mentioned, one goal of this research is to develop a computer-aided model of the submersible pump system as well as the motor controller. An induction motor can be modeled using an electrical equivalent circuit. A single-phase equivalent circuit is shown in Figure 5. The air gap (distance between the stator and the rotor) between the stator and the rotor is modeled as a transformer where the mutual coupling represents the air gap. In order to simplify the model, the transformer's turns ratio is applied to the rotor's inductance and resistance in order for them to be transferred to the high side, which is the stator side of the motor. Similarly the resistance on the high side of a transformer can be shifted to the low side. The inductances can be simplified further for calculation purposes by using Equation 3. Using this equation, an inductance value (represented as L, in units of Henry) is converted into an impedance value (units of ohms), which

represents the impedance of the motor. In the equation, f is the driving frequency in hertz. The stator in Figure 5, is represented by the inductors and resistors on the left side of the circuit. In the middle of the circuit, Rc and Lm represent the motor's core copper losses. The circuit shown on the right side of the transformer represents the rotor. The value of the resistor Rrot is a function of the rotor's slip. The circuit can be simplified further if a transformer is not used as seen in Figure 6. In order to achieve this, the turns ratio of the transformer is applied to the rotor resistance and reactance using the properties of Equation 4.

$$X_L = 2\pi fL$$

Equation 3: Inductance to impedance

$$P_{in} = P_{out}$$

Equation 4: Fundamental property of transformers [8]

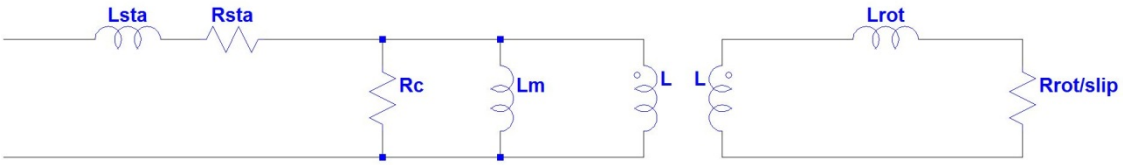

Figure 5: Single Phase equivalent circuit of a three-phase induction motor

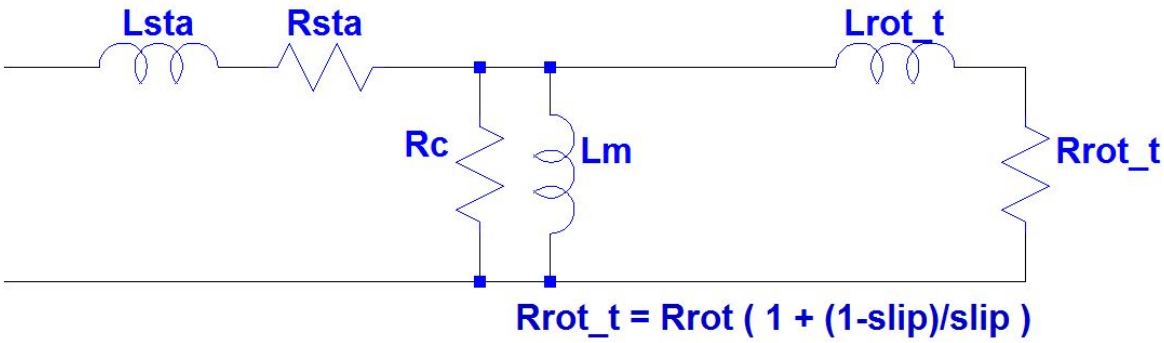

Figure 6: Simplified equivalent circuit of three-phase induction motor

One major disadvantage of induction motors is the high inrush currents they draw at startup. Inrush currents are transient in nature and have magnitudes that range anywhere from 3 to 20 times the motor's normal operating currents [6]. The currents can degrade the power system quality, trip electrical breakers, and lower the overall efficiency of the motor. Inrush currents typically last until the rotor begins to speed up to the point where the back EMF has been established. As seen in Equation 4, the inrush current can be described as a function of the rated voltage, the motor's power rating, and the rated horsepower. The motor experiencing problems at the residential subdivision has the power rating letter code J (see Table 1), has 40 horsepower, and a rated voltage of 460 VRMS. Using these values in Equation 5 the inrush current of the motor being studied should be approximately 363 A at startup. As will be shown later, this value is extremely close to those predicted by the simulation.

$$\text{Inrush Current} = \frac{(\,(\text{Code letter value}) * \text{HP} * 577\,)}{\text{Rated voltage}}$$

Equation 5: Three phase motor inrush current estimation [13]

Table 1: Inrush Current Rating Letter Code [10]

CODE LETTER	KVA/HP RANGE	APPROX. MID-RANGE VALUE	CODE LETTER	KVA/HP RANGE	APPROX. MID-RANGE VALUE
A	0.00 - 3.14	1.6	J	7.10 - 7.99	7.5
B	3.15 - 3.54	3.3	K	8.00 - 8.99	8.5
C	3.55 - 3.99	3.8	L	9.00 - 9.99	9.5
D	4.00 - 4.49	4.3	M	10.00 - 11.19	10.6
E	4.50 - 4.99	4.7	N	11.20 - 12.49	11.8
F	5.00 - 5.59	5.3	0	12.50 - 13.99	13.2
G	5.60 - 6.29	5.9	R	14.00 - 15.99	15.0
H	6.30 - 7.09	6.7			

The symmetrical nature of multi-phase induction machines does not allow them to handle a wide unbalance in the voltages and currents of the different phases. When subjected to unbalanced conditions, a motor tends to heat up rapidly as a result of the motor winding due to the unbalanced force that is extremely damaging to the internal components. In order to prevent this type of unbalanced operation, a motor controller can automatically shut down the motor upon detecting voltage or current unbalances. As will be shown later, unbalanced conditions and automatic shutdown caused the problems at the residential subdivision.

The process of derating involves reducing the input voltage or current in order to move the motor to a safe and efficient operating condition. This process can also help maintain operation during unbalanced conditions. A motor can be derated with respect to

one or several of its properties, including its voltage, current, power, and level of acceptable unbalance. Derating curves are used to adjust a motors set point in these various conditions. For example, if a 50 HP motor is operated with a 5 percent unbalance in either its voltage or current, which will require use of two separate derating, curves, a derating factor of 0.8 is recommended. What this means is that the motor can operate at 80% of its rated value or 40 HP safely [14].

Submersible Pumps:

While the background and theory provided to this point is applicable to all types of induction motors, the type of motor that is of interest here is a submersible pump. These types of pumps are used in a wide variety of applications ranging from small in home aquariums to large industrial oilrigs and water distributions systems. To better understand the functions they serve, an example of a residential pump system will be considered for simplicity. The motor is placed at the bottom of the well. As the motor rotates, large intake forces are generated which suck in the fluid. As the fluids are pulled into the intake valves they are subjected to large centrifugal forces that push the fluid up and out of pipes attached at an outlet [15]. This is shown graphically in Figure 7 and Figure 8.

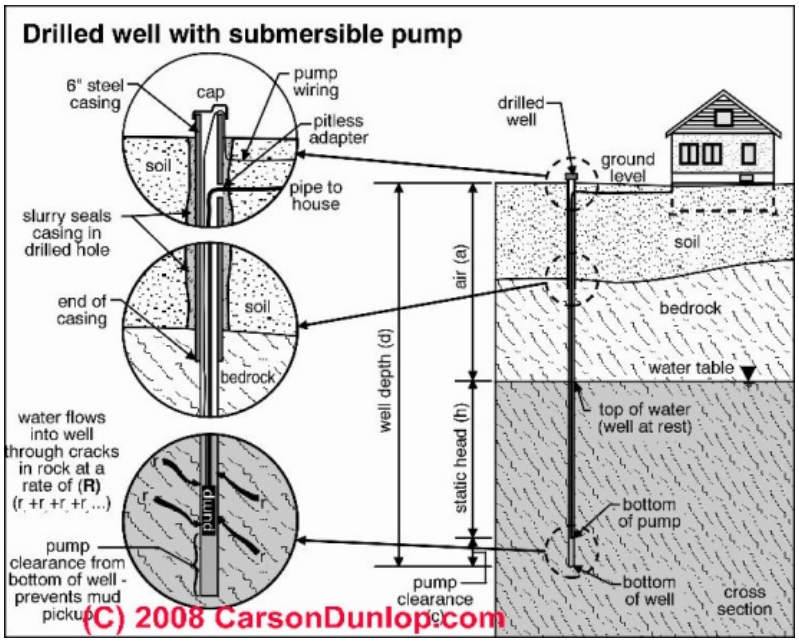

Figure 7: Overall submersible pump system [16]

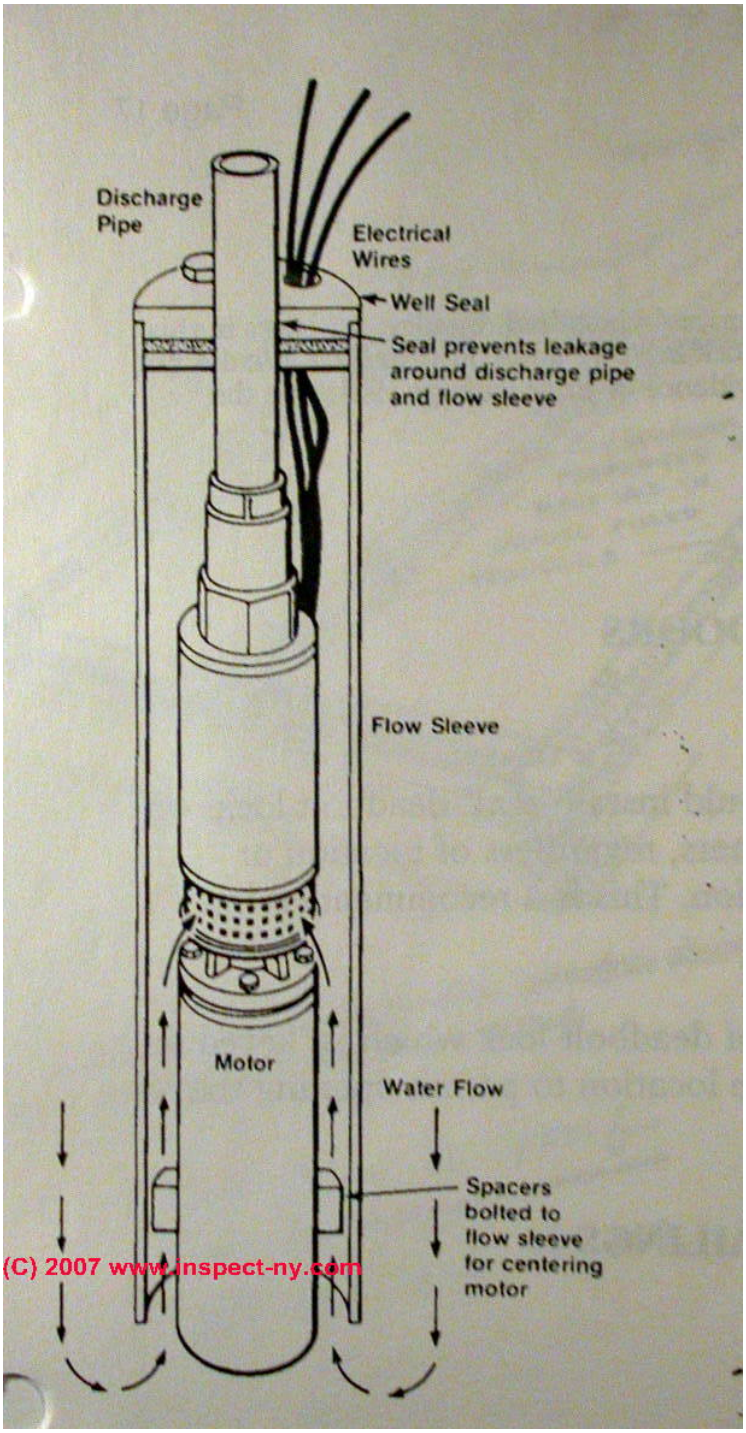

Figure 8: Detailed submersible pump [16]

AC Power Systems:

As expected, the input voltage sources that drive induction motors are usually sourced by large-scale electrical distribution systems. These types of systems date back to the 1880's and because many people were involved in the early conceptions, there is no one person that can be pointed to as the inventor of AC transmission theory. As previously mentioned, Galileo Ferraris and Nikola Tesla are credited with the rotating field theory that makes induction motor systems possible. While there was much discussion of whether alternating current (AC) or direct current (DC) electrical transmission was best, AC transmission eventually won out because of the ease at which AC voltage can be stepped up or down [8]. As the development of AC power systems grew, so did the need for the ability to regulate the line voltages and frequency. As more and more electrical loads were developed, the need for operational standards emerged. While it may not seem like much, a plot of identical single-phase systems each off by 1 Hz is shown in Figure 9. It can be seen that even small changes in frequency can have quite an impact in the creation of unbalanced forces.

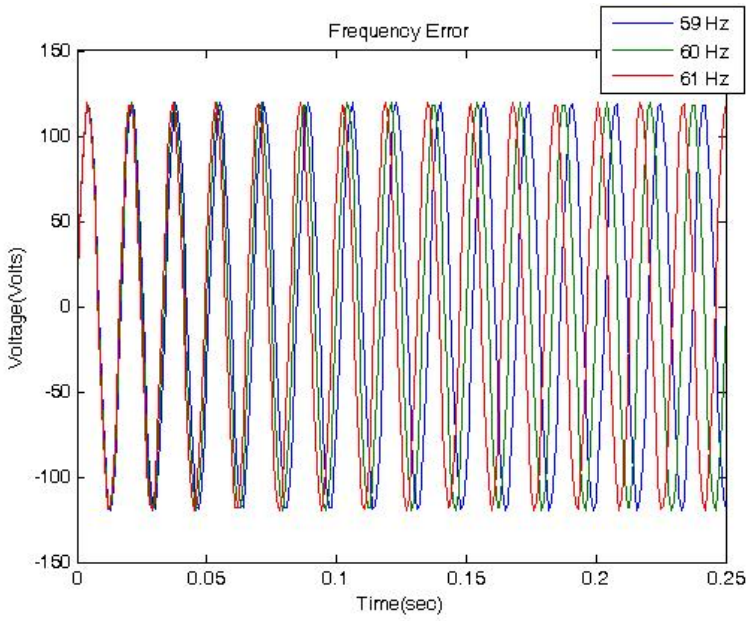

Figure 9: Frequency Error 59, 60 and 61 Hertz signals

Even today there are power generation and conversion stations in the North Eastern US that are operated at 25Hz in order to provide power to the electrical railroad systems. This low frequency allowed them to run long distance lines with the benefits of almost DC input to the electric motors [17]. Two possibilities exist for altering the frequency of an AC signal, those being rotary machines or power electronics. With the increase in the reliability and understanding of power electronics it has become much easier to regulate the frequency of the power grid. Currently the US power grid is operated at a standard of 60 Hz.

An AC induction generator is merely an induction motor operated in reverse. In the generation mode, the rotor, along with its magnetic field, is rotated by an external energy source inducing energy to the stator, which can then be supplied to a load. The external power source can come from the kinetic energy possessed by many possible working fluids. Those typically used include steam, water, compressed air, and wind for

example. Figure 10 shows a simple drawing of a hydroelectric power plant. Note that hydroelectric plants would typically be operated with synchronous AC generators, meaning rather than the induction machine just describe. The major difference between synchronous and asynchronous generators is the DC excitation current applied to the field windings.

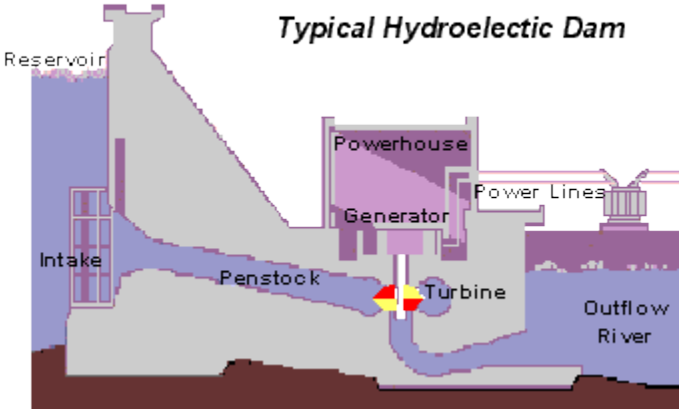

Figure 10: Hydroelectric dam [18]

In many hydroelectric power plants, the moving water is held back in a dam or retention pond [18]. During the day, water flows through a channel and through the hydro generator, seen in Figure 11. At night, when the use of energy is low, the water is pumped from the storage pond to the upper pond where the process can be repeated again the next day this operation can be seen graphically in Figure 12.

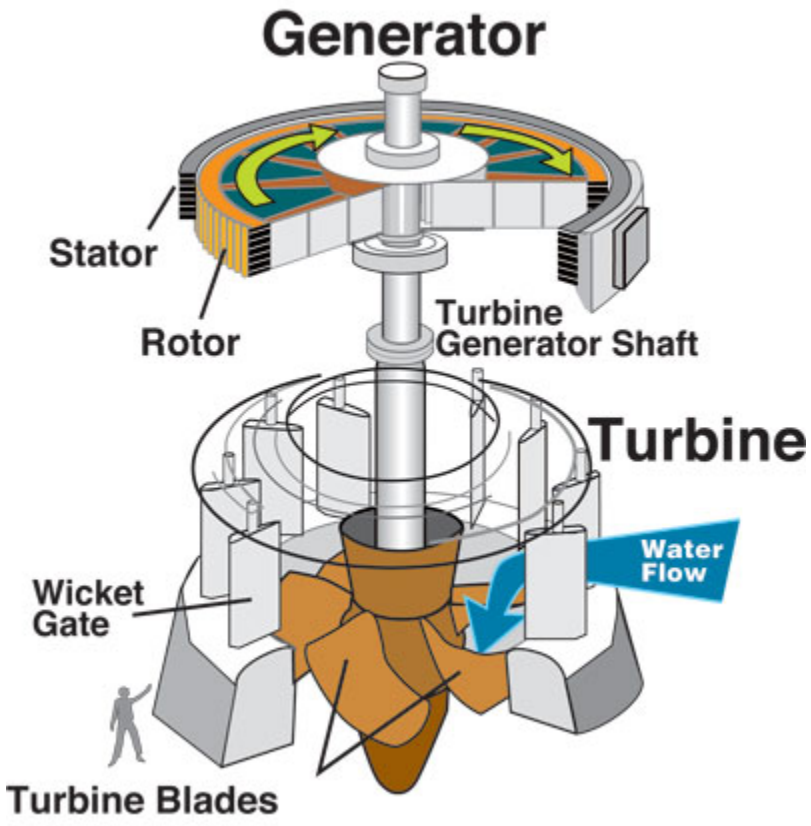

Figure 11: Hydroelectric generator turbine cutaway view [18]

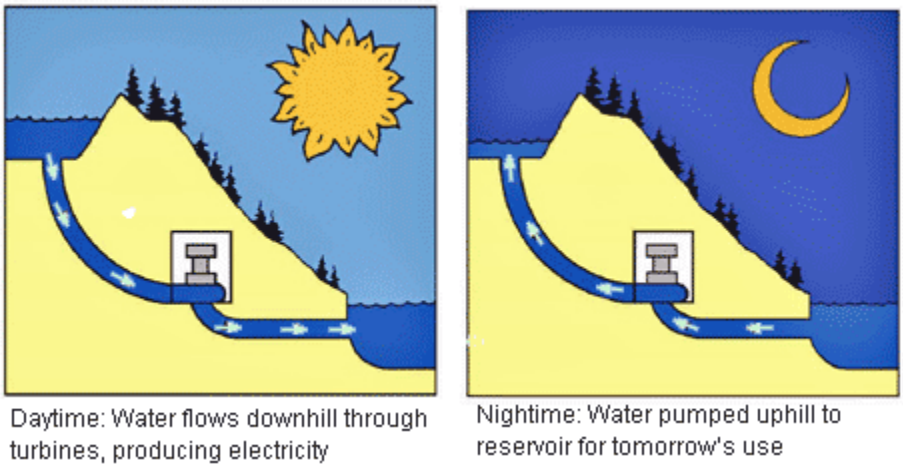

Figure 12: Retention pond reflow power generation [18]

As previously mentioned, voltage regulation is also critical. In Texas, the voltage must be maintained within +/- 5% of the line's nominal rating as dictated by ANSI standard C84.1 [19]. While this would appear to be as simple as putting some form of voltage regulator at distributed points along the line, this is not economically feasible due to the high cost of voltage regulation equipment. It is easier to strategically place these voltage regulators throughout the system and distribute the load accordingly along the lines. When an electrical utility company installs a new distribution line, such as one to a new residential subdivision, they must pay careful attention to how they load the different phases on the line. They also must consider thermal limits of the transmission line, as well as voltage drops. Some utilities may install a small tap-changing transformer to maintain voltage at the end of a line. Most utilities will intentionally set the voltage levels slightly higher than the nominal by a small percentage (<5%) in order to account for the inductive voltage drops along the transmission line and any sudden sags which may occur. This type of system operation is typically good when loads are many miles away from the generation source, but it does present a problem for sensitive loads located closer to the generation facility. In this case, installation of a tapped transformer can be helpful.

As the reader is probably aware, an AC transformer is used to step the voltage of an AC bus up or down. This is accomplished using two inductors, which are coupled using a single iron core. The iron core increases the mutual coupling between the two inductors. The voltage is adjusted on the secondary by simply setting the secondary number of turns higher or lower than that of primary. The fundamental principle works off of the idea that in an ideal transformer, the power across the primary will equal the power across the secondary; this is seen in Equation 6. Michael Faraday and Joseph Henry are credited with discovering and developing the theory of electromagnetic induction, which

led to the creation of transformers [20]. By placing several different output taps on the secondary, the voltage supplied to the load can be altered without installing different transformers. These multi-tap transformers are used to make small adjustments within a power system when sensitive loads are connected.

$$V_{in}I_{in} = V_{out}\mathrm{I}_{out}$$

Equation 6: Ideal input and output relationship for a transformer [8]

In the report presented here the sensitive load is a 40 HP submersible pump. This induction motor is rated for a line-to-line voltage level of 460 Volts, which is less than the nominal three phase line-to-line voltage of 480 Volts. The reason for this lower rating is to account for the voltage drop across the line leading to the load. Since these types of pumps are often placed several hundred feet below the surface in a typical installation, the voltage drops can be quite significant. Because the depth of each pump is different, resulting in different voltage drops, a tapped transformer at the start of the feed to the motor is ideal allowing them to be adjusted easily as needed.

As implied, a transformer relies on the sinusoidal nature of an AC wave to produce a magnetic field when current is flowing. Equation 7: Relation of current and magnetic field intensity [8] relates the coupled magnetic field, H in A/m, to the induced current generated in the secondary, I_m in A, using the number of turns, N_m, and the coil length, l, in m.

$$I_m N_m = H\ell$$

Equation 7: Relation of current and magnetic field intensity [8]

Using the permeability value of the core being used, μ_r, the magnetic flux density, B in T, can be calculated using the relation shown in Equation 8: Magnetic flux density [8].

$$B = \mu H$$

Equation 8: Magnetic flux density [8]

The magnetic flux of the winding will depend on the flux density and the area of the core. If the core has any air gaps the magnetic flux of each section of the core and air gap would need to be calculated separately and then summed together due to the change of density in the air gap. The magnetic flux is easily calculated using Equation 9 where B is the flux density in T and A is the cross sectional area of the core in m^2.

$$\Phi = BA$$

Equation 9: Magnetic flux [8]

Faradays law of induction states that a voltage is induced in a coil of wire when a time varying magnetic field is located nearby and coupled into the coil. Similarly, supplying an alternating current through a wire generates a magnetic field. In Equation 10, which is a simplified version of Faraday's Law, it is shown that the voltage induced on the secondary winding is directly proportional to the number of turns. Equation 11 shows that there are several ways to use Faraday's law to evaluate the voltage induced on a transformer's windings. Along with the Lorentz force shown in Equation 2, on page 5, these equations provide the necessary links between electrical and magnetic energy in a coil.

$$V_{in} = N_m \frac{d\Phi}{dt}$$

Equation 10: Faraday's Law [8]

$$E = N_m \frac{d\Phi}{dt} = N_m \frac{d(BA)}{dt} = AN_m \frac{d(\mu H)}{dt} = \mu A N_m \frac{d}{dt}\left(\frac{N_m I_m}{\ell}\right) = \frac{\mu A\, N_m^2}{\ell} \frac{d}{dt}(I_m) = L \frac{d}{dt}(I_m)$$

Equation 11: Faraday's Law in all forms for a transformer [8]

As previously mentioned, phase unbalance is a major concern in large power systems. As the voltage and current are directly related, unbalance in one often results in the significant unbalance of the other. One of the easiest causes of phase unbalance is when a utility lineman loads one or two phases heavier than the other during an installation. There are several different ways to calculate the unbalance. The simplest is to calculate the maximum deviation from the average voltage using Equation 12: Phase voltage unbalance rating Equation 12, where V_A is voltage on phase A of the three phase system, V_B is that of phase B, V_C is the that of phase C, and Vavg is the average voltage solved using Equation 13.

$$PVUR = Max\left(\frac{V_{avg} - V_A}{V_{avg}}, \frac{V_{avg} - V_B}{V_{avg}}, \frac{V_{avg} - V_C}{V_{avg}}\right) * 100\%$$

Equation 12: Phase voltage unbalance rating [14]

$$V_{avg} = \frac{V_A + V_B + V_C}{3}$$

Equation 13: Average voltage

Equation 12: Phase voltage unbalance rating, subtracts the average voltage from each individual phase and then divides those values by the average. The output provides the percentage value of the unbalance. Another way of calculating the unbalance is to take the magnitude and angle of each phase and convert them into symmetrical components. This method provides a better understanding of the current and voltage unbalance. This is because the unbalance is shown on a single phase only and therefore only one calculation is necessary instead of multiple. Symmetrical components are made up of zero (0), negative (2) and positive sequences (1), where each one of the sequences represents the one of the directions of rotation. The positive sequence represents the natural steady rotational fields produced by a 60 Hz sine wave. The zero

sequence represents the ground or neutral path. The negative sequence represents negative rotation or any unnatural occurrences. In Equation 14, alpha represents the separation factor that is used in Equation 15 to solve for the symmetrical components.

$$\propto = e^{\frac{2}{3}\pi j}$$

Equation 14: Separation reference [14]

$$\begin{bmatrix} V_0 \\ V_1 \\ V_2 \end{bmatrix} = \left(\frac{1}{3}\right) * \begin{bmatrix} 1 & 1 & 1 \\ 1 & \propto^2 & \propto \\ 1 & \propto & \propto^2 \end{bmatrix} * \begin{bmatrix} V_A \\ V_B \\ V_C \end{bmatrix}$$

Equation 15: Phase to Symmetrical components [14]

Analysis using symmetrical components allows a motor protective system to concentrate on a single phase instead of calculating the unbalance of all three phases using their average. Using symmetrical components as seen in Equation 15, the unbalance factor becomes a simple ratio between the positive and negative sequences. This ratio shown in Equation 16, represents the negative rotation divided by the positive rotation or bad voltage divided by good voltage.

$$VUF(\%) = \frac{V_2}{V_1} * 100\%$$

Equation 16: Voltage unbalance factor (VUF) [14]

In this project, symmetrical components were not used in the analysis due to the nature of the motor protection system. The motor saver uses Equation 12, rather than the symmetrical unbalance detection method. This is partly because the high and low values of each phase voltage and current are already being measured.

Motor controller designs:

Several different variations of motor controller are used to operate a wide variety of different types of motors. They are used for driving simple single-phase motors as well as three phase motors. The most common type of controller is known as a variable frequency drive (VFD). A VFD rectifies the incoming signal and then uses power electronics to create a new single phase or polyphase, pulse width modulated (PWM) signal that can drive the motor at a desired frequency. When using a VFD the operating frequency of the motor can be lowered in order to improve efficiency [10]. By varying the frequency, the speed and torque of the motor can be adjusted. Several considerations must be taken into account when using variable frequency control. In order to maintain safe operation, the voltage must be reduced along with the frequency linearly, this is done so that the motor is not overvolted [10]. The process of lowering the frequency also lowers the motors nominal voltage, power, speed, and torque. In order to ensure this is done safely, most motors have a suggested minimum frequency value that the datasheet specifies so that the motor does not stall.

An H-Bridge or Full-Bridge rectifier is a type of switch architectures that generates the necessary PWM signal to control the motor frequency. As seen, four switches, typically MOSFETs or IGBTs, form the arms of the circuit while the motor creates the bridge. Each leg of the bridge can be individually controlled so the motor can be rotated clock wise, counter clock wise, coasted, or braked. In order to cause rotation the switches must be turned on in a specific order, this is also shown in Figure 14: Forward H-Bridge operation. If all the switches are turned off the motor is allowed to coast, however if either both high sides or both low sides are turned on the motor is shorted and the resulting reversed voltage generation causes a braking condition.

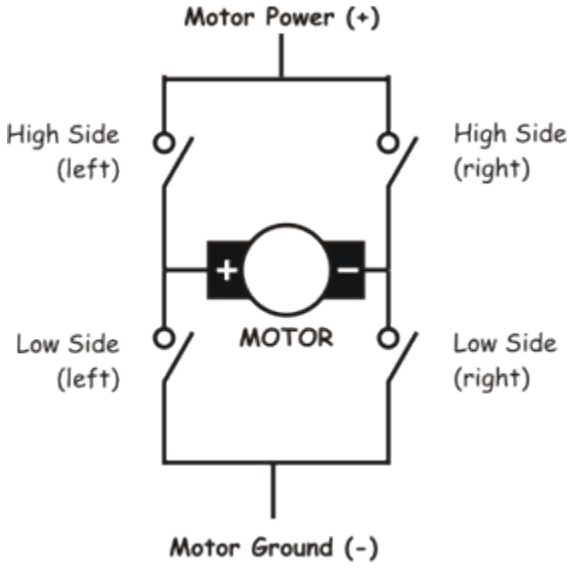

Figure 13: Basic H-Bridge [21]

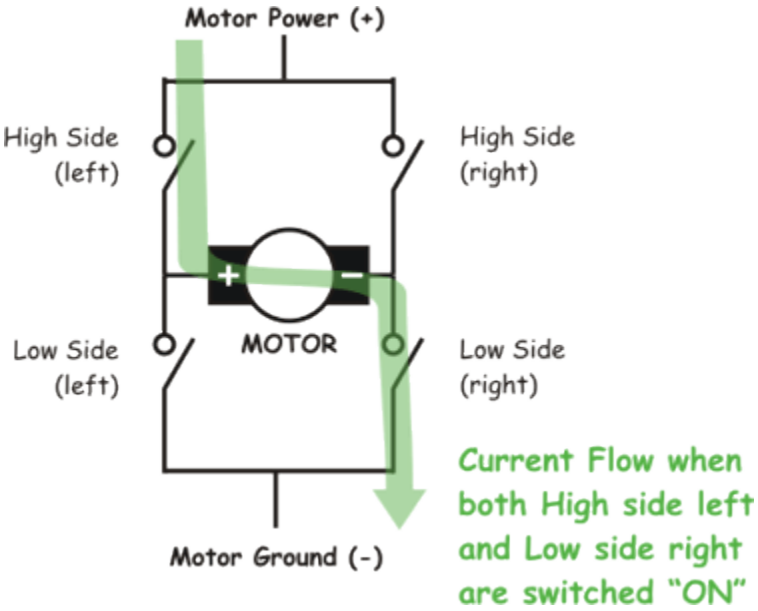

Figure 14: Forward H-Bridge operation [21]

Motor Protection Systems:

Many three phase motors are driven directly off of the utility input. This configuration is called ‘across the line starting’. In simple setups, the only protection consists of circuit breakers or fuses which fail to open in overcurrent events. The need for current overload protection stems from the damages that over current can cause internally to the motor. As the windings heat up they begin to damage the insulation of the motor, which can lead to internal shorting [22]. Manual or automatically controlled contactors or motor protection systems are also often used [13]. In this case, the motor controller is used to monitor and control the flow of energy based off of the current motor status. These systems are often elaborate microprocessor based systems, which have programmable motor characteristics specific to the motor on which it is being used. In general motor protection systems use three measurements obtained from the motor on which to perform their analysis. Those include the terminal voltages, currents, and operational frequency. The system that is being studied here at the residential subdivision uses a Symcom Motor Saver (SMS). The installed unit measures the line-to-line voltage of phases AB, BC, and CA. It evaluates each of these measurements for over/under voltage conditions and also calculates the PVUR, seen in Equation 12: Phase voltage unbalance rating, to check for unbalance. Each phase current is measured and the frequency calculated from each of the measurements is mainly used to ensure the system is within in the acceptable range suggested by the motor manufacture. While the internal workings of the Symcom Motor Saver are proprietary information, calculated guesses can be made as to how it performs its calculations and how it responds to them. In order to gather a semi-constant value for the voltage and current, the SMS likely mathematically calculates the root mean squared (rms) value for each measurement made. It likely uses these values for many of its decisions. The voltage is sampled

directly and is switched on and off via a contactor, which is a form of high current relay or switch [23]. The current is measured using current transformers.

Previous Research:

In many of the previously developed motor simulations, accurately modeling the motor's torque output is the main focus of research [24], [25], [26]. In many of the simulations that account for voltage and current unbalances, the main focuses are aimed at correctly evaluating the torque output and the motor's internal changes which result from the heat generated by the unbalance. A. R. Patel, J. K. Chauhan and D. K. Patel modeled their motor seen in Figure 15, using an unbalanced three-phase voltage source as and input and then studied the effects that the unbalance had on the stator and rotor currents, speed and torque.

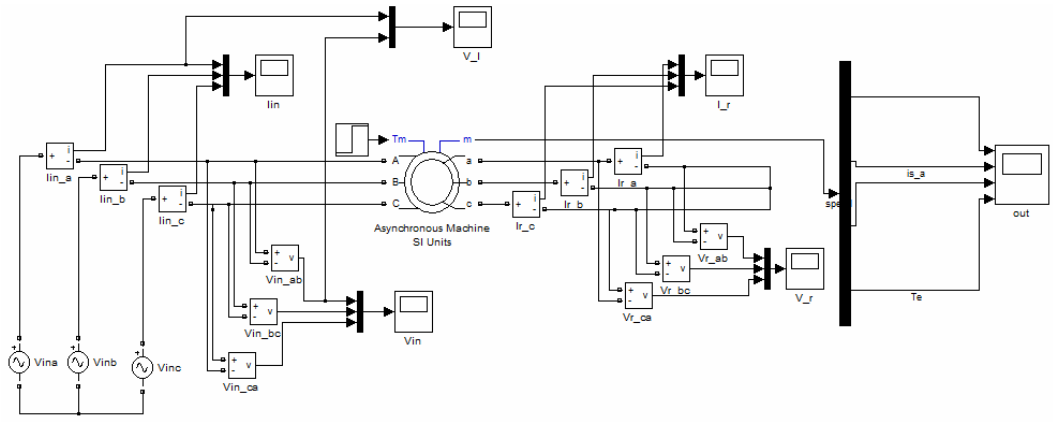

Figure 15: Model circuit from Simulation for a 3-Phase induction motor under unbalanced conditions [24]

In their analysis, Shi, Chan, Wong and Ho,use a Vabc to Vdq conversion approach to model a three phase induction motor as seen in Figure 16 [25]. This model relied on only a 3 phase voltage source to produce modeled results with respect to the motor's stator current, torque, and rotor speed.

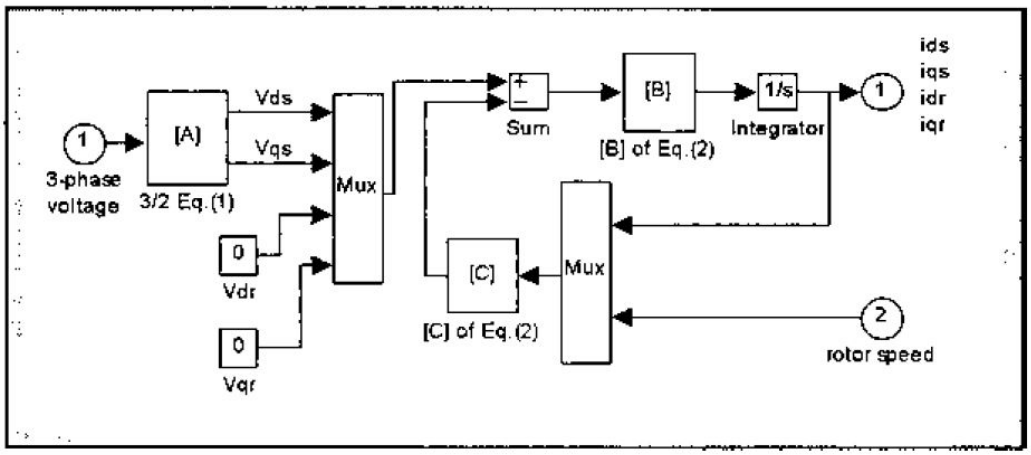

Figure 16: Model design from Modeling and simulation of the three-phase induction motor using Simulink [25]

V. Vivek, G. Uma, R. Kumudini Devi and C. Chellamuthu, developed a standard circuit model of their motor and predicted the torque and shaft speed [26]. Their results most closely match the work performed in this report and their model can be seen in Figure 17. The focus of their work was to evaluate the motor's input voltage, frequency, and the head pressures at the pump output. Their model is used to predict the needed output torque for a desired output head pressure.

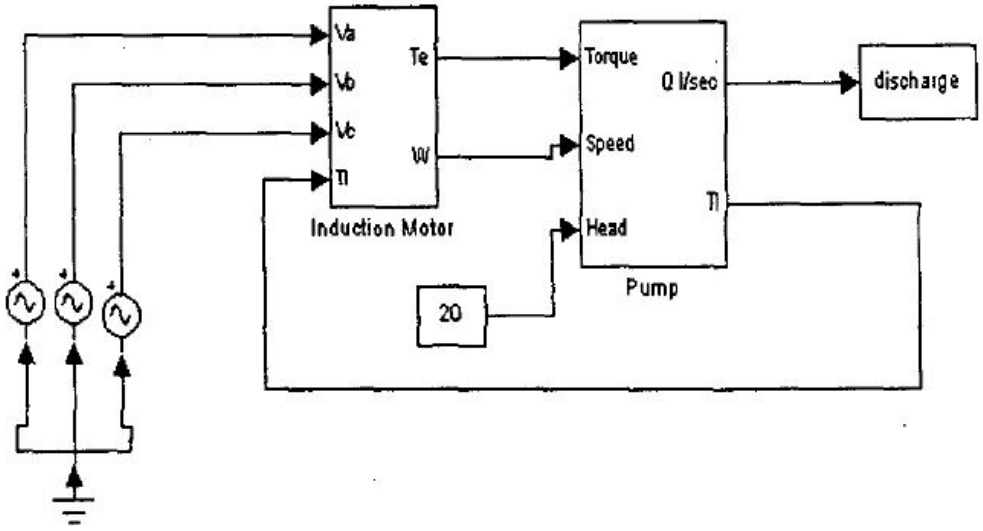

Figure 17: Black diagram of the model circuit from Performance of induction motor driven submersible pump using Matlab Simulink [26]

Chapter 2

Data analysis and modeling of a submersible pump:

When the residential subdivision first started experiencing troubles with their motor, the local utility began recording real time data to determine what events caused the trouble. The utility recorded current and voltage on each phase and saved a data point every five minutes. Every 5 minutes the recording firmware would save the minimum, average and maximum values for the current and voltage of each phase. In order to gain a more complete perspective, the local utility also contracted a local engineering firm to perform similar data collection on two minute intervals. Finally, UTA was contracted to perform data analysis on both sets of data as well as construct the computer aided model previously alluded to. Analysis of the data clearly showed that current unbalances had occurred. However, the low sampling resolution made it difficult to identify the actual occurrence and cause of the trip events. Both companies recorded data points on several occasions at the same time which provided more insight to the problem. When collecting data, the local utility located their metering equipment at the output of the transformer leading to the submersible pump system and the local engineering firm located theirs directly at the input terminals to the motor protection system. The data collected by the local utility provides a picture into when each of the three respective pump motors located at the site were operational. Evaluation of the data collected by each respective company revealed instances where the combination and timing of the individual motor startups could have caused the unbalance. The smallest of the three pumps operates at an average current of 12.5 Amps, while the two larger draw 50 Amps and 55 Amps respectively. The local utility's data seen in Table 2, lists the minimum, average and maximum currents measured during each 5 minute interval.

During the same time period, the local engineering firm also recorded data. At 6:56 PM the largest of the three motor, also the one with the motor protection system, was turned on as well as the medium pump. From the local utility data it can be seen that the medium pump successfully turned on and continued to pump while the local engineering firm data shows the largest pump shutdown due to a current unbalance.

Table 2: Local utility recording of current data for June 20th, 2011

Time	Ch. 1 Min.	Ch. 1 Avg.	Ch. 1 Max.	Ch. 2 Min	Ch. 2 Avg.	Ch. 2 Max.	Ch. 3 Min.	Ch. 3 Avg.	Ch. 3 Max.	Unbalance
18:55:00	12	12	13	10	10	12	15	15	16	21.6
19:00:00	12	55	376	11	54	383	15	65	372	12.1
19:05:00	55	81	105	52	78	102	64	95	123	12.2
19:10:00	54	55	56	53	53	55	63	64	65	11.6

Table 3: Local engineering firm recording of current data for June 20th, 2011

Time	Avg.	Ph. A	Ph. B	Ph. C	Neutral	Unbalance
6:56:00 PM	17.8	17.2	16.4	19.9	2.0	11.6%
6:58:00 PM	54.0	51.7	49.2	61.0	3.1	13.0%
7:00:00 PM	53.6	52.0	48.1	60.6	3.1	13.1%
7:02:00 PM	7.8	7.5	6.9	9.0	1.7	15.4%

Despite this observation, this did not happen at every trip event. Therefore, no conclusion could be reached, especially because of the data resolution. As mentioned earlier, the data recorders perform real time averaging of the data and save only the

minimum, average and maximum data points from the recording time interval. This makes it impossible to capture any transient events that may have caused any of the trip conditions.

It should be noted that the local utility's immediate response was to evaluate the capacitor banks closest to the site. These capacitors help maintain voltage balance, the setting of the transformer feeding the motors, the setting and operation of the voltage regulators, and finally the line leading into the residential subdivision. In case the reader is unaware, large capacitor banks are selectively placed throughout the grid to compensate for inrush currents drawn in by large inductive loads such as the motor [21] and to adjust power factor. They are not always used however, because the summer temperatures in Texas are so extreme, the local utility typically connects their capacitor banks for the duration of the summer to reduce the detrimental effects large air conditioning loads have on the power quality [21]. Since the capacitor bank had been turned on well before the tripping had started, the lack of having them installed was ruled out as a potential cause of the unbalance events.

While all of the voltage regulators located up line from the residential subdivision were working correctly, one phase was set slightly lower (~5V) than the other two. This lower set point is not believed to have caused a large voltage shift because the loads and losses from the regulators to the subdivision are so heavy. While the voltage regulator was a factor in the unbalance seen by the motor it was not the only cause.

The proper setting for a transformer, which feeds a submersible pump motor, is a major topic of discussion in the power industry, due to the sensitive nature of motors. The best way to accommodate a wide variety of different motor systems is to install a transformer, which has multiple taps. This solution enables the utility to easily

compensate for any voltage adjustments needed with only minor effort. It has been very successfully implemented as means for avoiding problems with sensitive motor systems.

The computer-aided model, which will be discussed next, was developed so that data collected in this fashion could be fed into the model as the input parameters. With this ability, the model will be better enable the user to understand how the system should have responded. This is a much easier and more reliable way of accomplishing this task as simply evaluating the numbers or plots of the data does not always allow the user to see what caused the trip event. In order to develop the most flexible, robust, and reliable model that could be executed easily by the local utility, a the decision to use the Matlab/Simulink® programming language was made. Matlab's built in tools for importing Excel style data made it reliable and easy to use.

Data Analysis:

The first step of the data analysis was to sort through it and look for any obvious causes that may have tripped the motor. As previously mentioned, it was quickly determined that any transient conditions which may have caused the motor to trip could not be determined from the data due to the real time averaging that the logger implements. The data did show that there were several days where the voltages on phases A and B were lower than expected and there were times when the voltage on phase C was higher than expected. The voltage on phases A and B the voltages were 10 to 20 Volts lower than their respective average voltages this can be seen in Figure 18. The high voltages on phase C were only about 3 volts higher than the average phase C voltage. As expected, the Analysis of the data suggests that the low phase voltages on A and B cause the line-to-line voltage on AB to be much lower than desired as shown in Figure 19. This is confirmed by the voltage unbalance data calculated by the local engineering firm from their recorded data, seen in Figure 20.

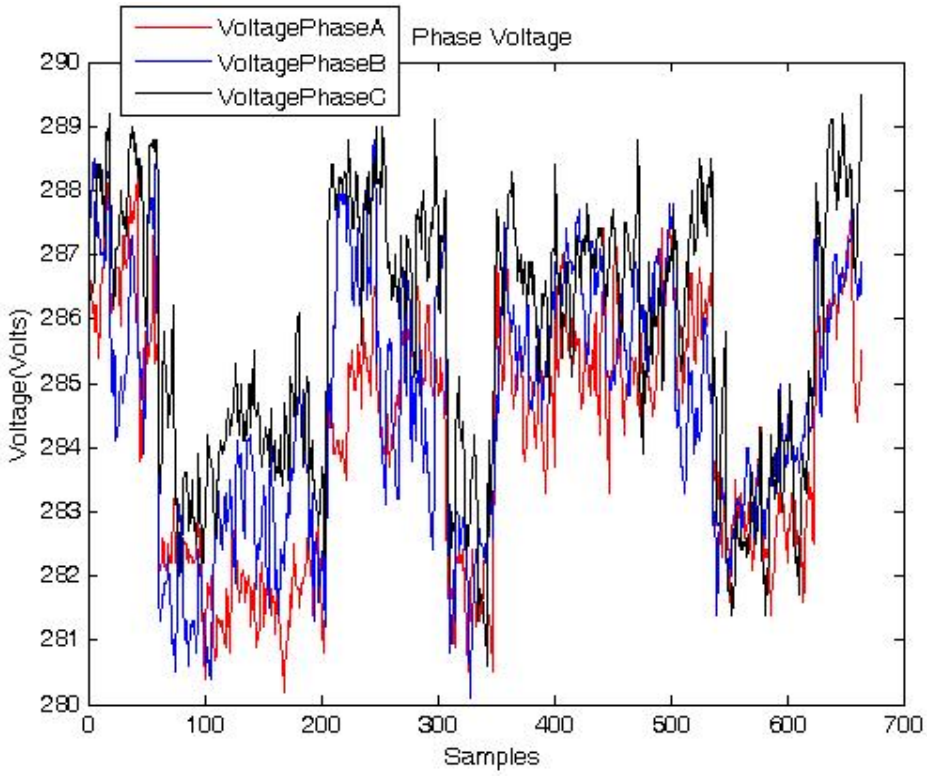

Figure 18: Phase Voltage from the local engineering firm's recording from June 24th 2011

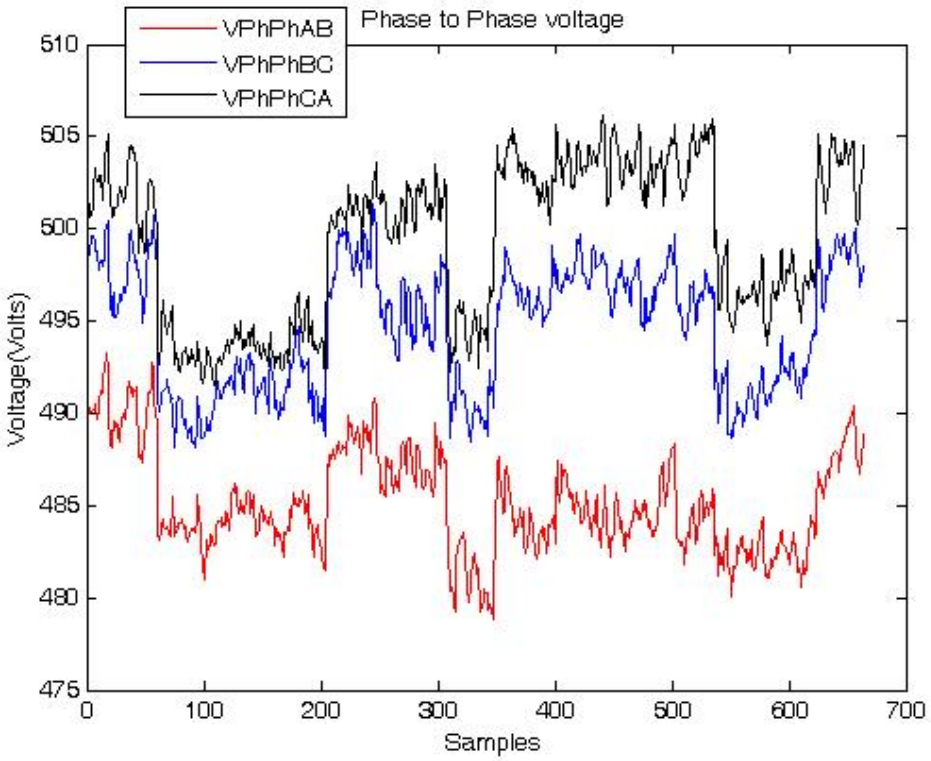

Figure 19: Phase to phase voltage from the local engineering firm's recording from June 24th 2011

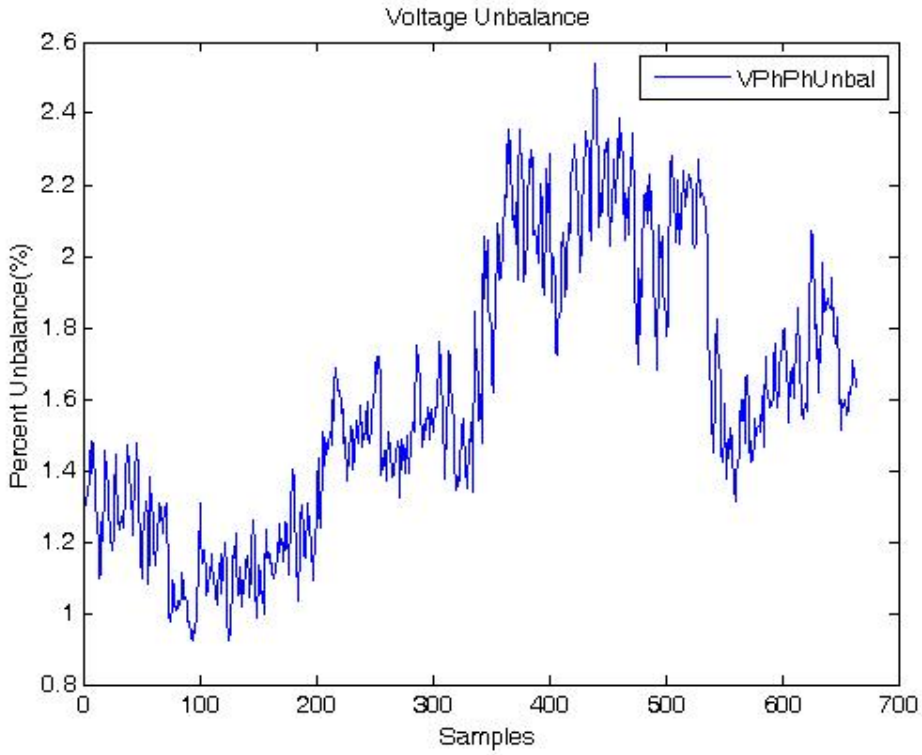

Figure 20: Voltage unbalance from the local engineering firm's recording from June 24th 2011

The motor saver primarily evaluates the phase currents and trips in the event unacceptable current imbalance is detected. In most of the cases where imbalance occurs, the currents measured on phase C were higher than its normal average when the voltage on phases A and B were low, this can be seen in Figure 21. From this data it can also be seen that there will be a significant current unbalance seen by the motor as shown in Figure 22.

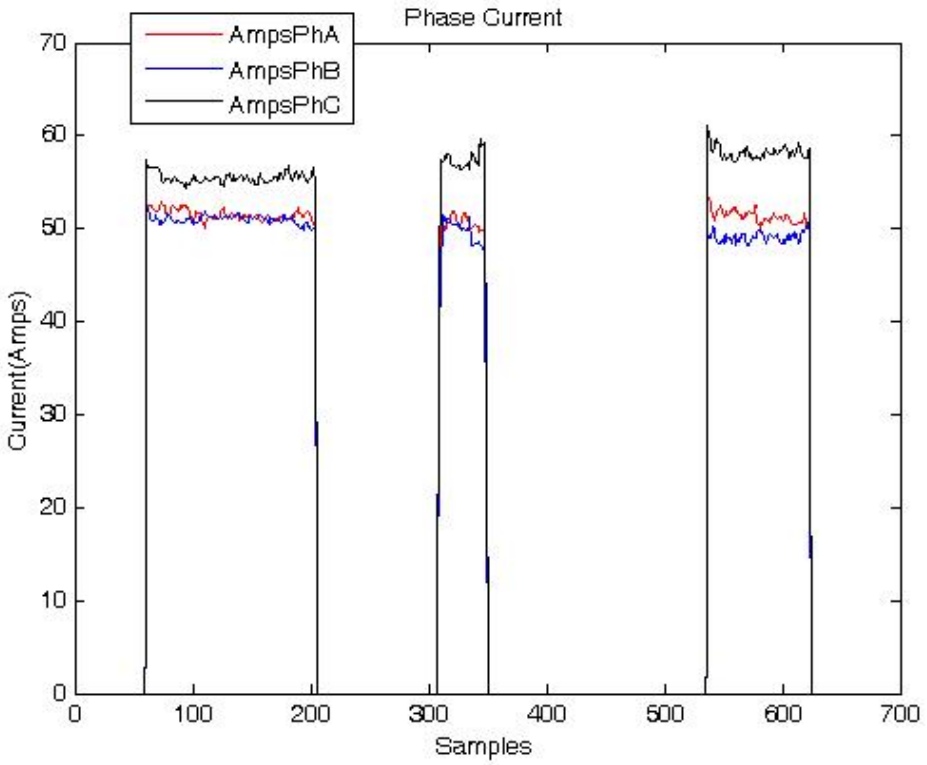

Figure 21: Phase current from the local engineering firm's recording from June 24th 2011

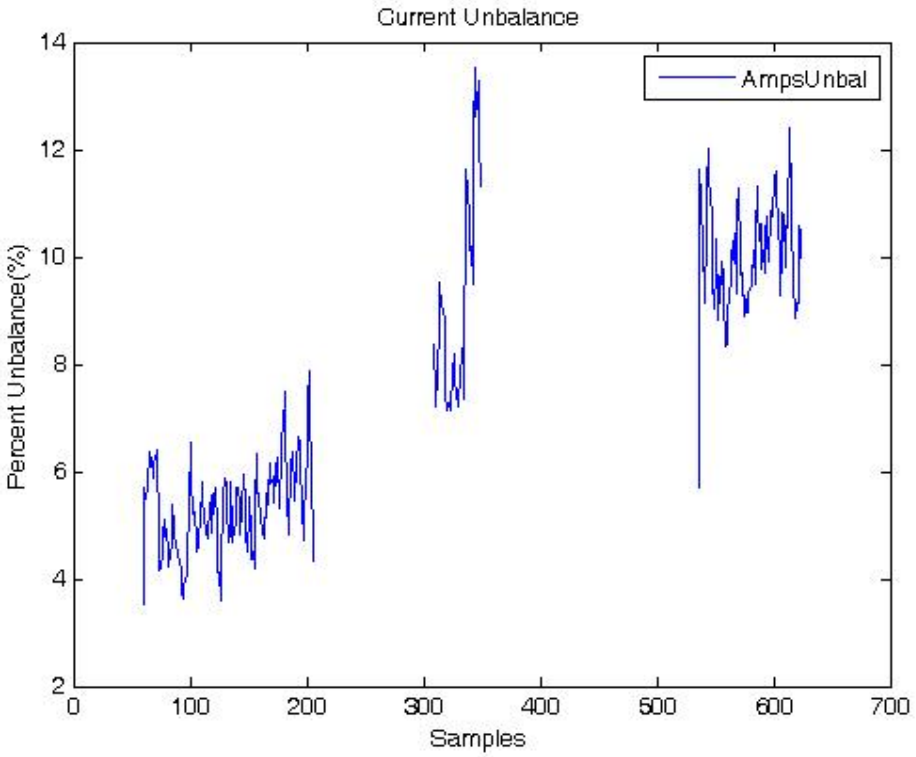

Figure 22: Current unbalance from the local engineering firm's recording from June 24th 2011

Joint analysis with the local utility of the data confirmed that the transmission line feeding the area was not properly loaded. The loading of Phases A and B were much higher than Phase C and this was likely the cause of most of the higher and lower voltages recorded. Further load studies performed by the local utility revealed that linemen were more heavily loading phase A, which is the closest line to the road and therefore the easiest to work with. This asymmetrical phase loading coupled with the voltage regulator setting error greatly contributed to the voltage unbalance at the residential subdivision.

In order to verify that the Motor Saver was properly configured, an independent data recorder was placed at the residential subdivision and it was confirmed that the

MotorSaver was not the cause of the trip events. Additionally, the electrical setup of the pumps and motors was also recorded. The overview of the entire system can be seen in Figure 23, where the panel for the motor discussed in this report is on the far left hand side. A detailed view of the electrical panel for the motor can be seen in Figure 24. The SMS motor protection system can be seen in the lower left hand side, along with contactors on the upper right hand side of Figure 24.

Figure 23: Electrical distribution panels for the residential subdivision

Figure 24: Distribution and motor protection system for the largest motor at the residential subdivision

One of its residents kept a logbook of when motor trips occurred during the period when the residential subdivision was experiencing the troubles documented here , what procedures were taken to attempt to restart the motor, and what the SMS had shown as the cause of the trip were also recorded. The logbook and correlation of the recorded data showed that there were significant problems that could have resulted from not only the voltage unbalance but also from the motors themselves. In several cases, it was documented that all three motors had turned on nearly simultaneously. Though the precise time correlation of the motors actual start times is unknown, it is possible that current unbalances could have occurred during the time when all three motors had simultaneous inrush currents. It was also mentioned by the resident that there had been previous problems at the site in which a motor had overheated, after only a few years of service, and had to be replaced with the current motor. This was intriguing since the majority of submersible pump motors are designed to have a long lifetime (on average 8-10 years) before replacement. Inquiries into the depth of the motor were made and it was believed to be approximately 300 feet deep, though both the local utility and the residential subdivision were uncertain of the exact depth. The installation contractor never responded to the queries for confirmation. Normal operating depths for submersible pumps occur about 1200 feet below the surface in this region according to the local utility. If the 300 ft. estimate is correct, it is possible that the anticipated line losses are not present and that the voltages actually applied at the motor terminals did not meet system specifications. Recall that 480 VAC is being applied to a 460 VAC motor under the assumption that line losses will bring the values down. This could explain the failure of the previous motor and describe why currents are being drawn that are higher than expected. While this shouldn't have had a major effect on the current unbalance if all three phases heat up equally, it is a recommendation to the reader to account for line

losses when installing submersible pumps. If the terminal voltage was too high, the motor at the residential subdivision should have been derated for operation at a higher operating level as was discussed earlier. These are details that cannot be accounted for in the model.

As mentioned already, Matlab/Simulink was chosen as the computer aided platform on which to develop a model of this system. This was chosen due to its relevance for solving today's complex simulation problems as well as its ease for both randomly generating operational conditions and actually accepting Excel based experimental data as inputs. The modeling toolboxes already implemented in Simulink made this a more reliable and robust solution. Matlab's computational abilities allowed both the Simulink model and Matlab data analysis tools to be coupled together into one easy to use tool with a graphical user interface.

The physical modeling toolboxes available in Simulink are already designed to simulate a wide array of physical systems including hydraulic, thermal, mechanical, and electrical systems. Within the physical modeling toolboxes are three electrical modeling toolboxes. While several libraries contain similar parts, each part is slightly different, depending on the technical use of the toolbox. For example, the transistors in the electronics toolbox differ from those in the foundational toolbox in that they allow more detailed properties adjustments.

The first step in the building of the model was development of a voltage source based that could accurately mimic the local utility power feed and digitize the recorded data as an input. For this part, a 60 Hz sine wave block with a peak of 1 is used as the base waveform and a gain block pulls in each of the different phases' magnitude from the recorded data files. When the actual recorded data is not being used, a random number generator can be implemented whose output is multiplied by the sine wave in order to

produce a simulated random voltage much like that of the power grid. This subsystem can be seen in Figure 25 and Figure 26. In order for the model to read the recorded dataset as a time based set of data it must be converted into a time series data file via a Matlab script before the simulation is ran. This code can be found in appendixes B through Y. The Matlab script that controls the model also uses the number of recorded samples along with the desired sampling time for the simulation to create the maximum run time of the simulation if this is not defined by the user. This automatically sets the simulation time to the minimum amount necessary in order to complete the simulation the user requests and save users' time.

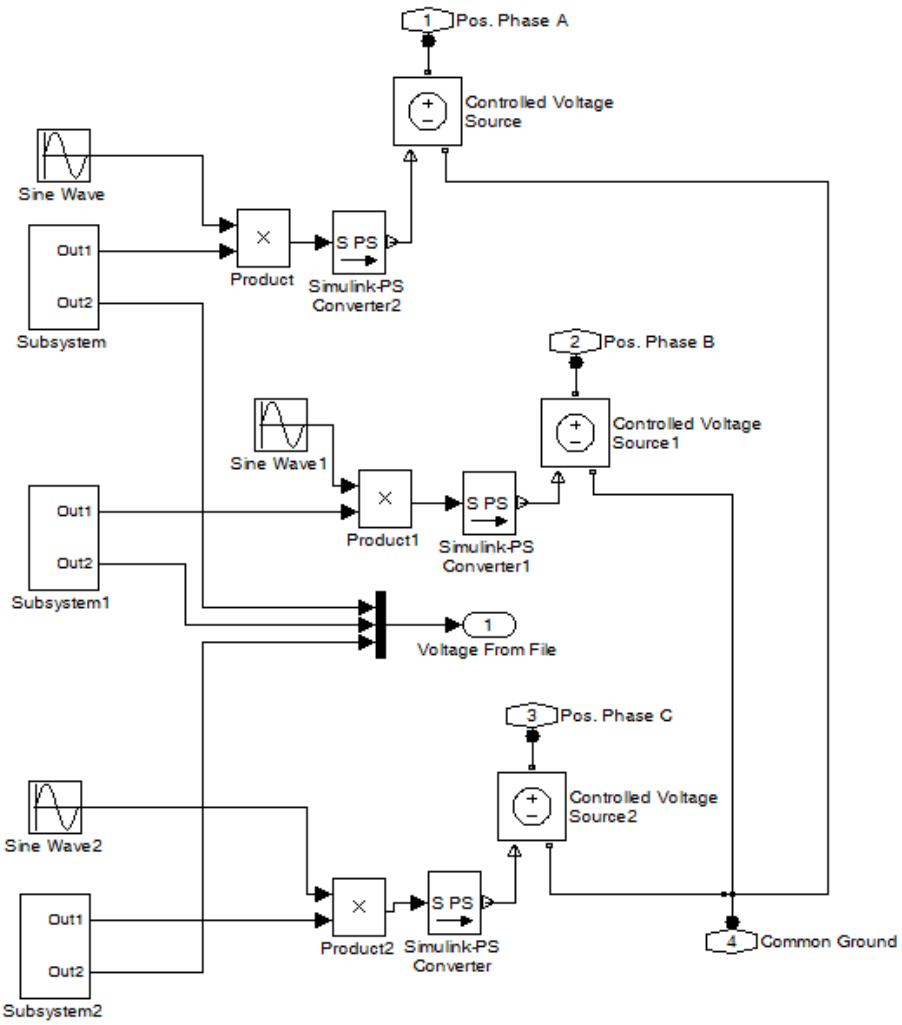

Figure 25: Model voltage source

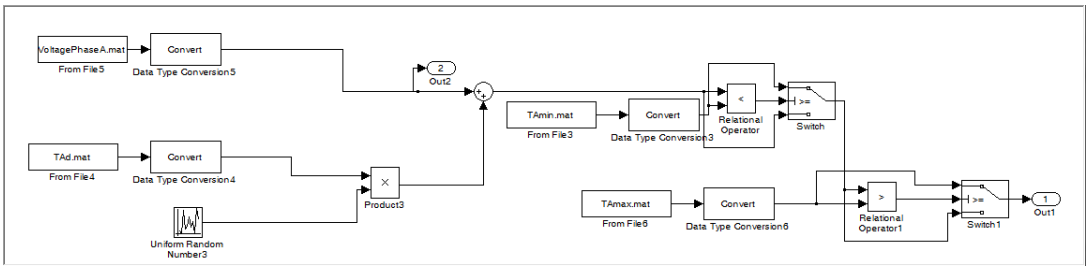

Figure 26: Model voltage source subsystem

The stop subsystem block seen in

Figure 27, is created using a file read block, a clock, and generic switch block. It uses a simple ‘greater than’ mathematical operation block to check when the clock time goes above the maximum time specified in the file. As mentioned, this block set is used in order minimize the run time of the executable program. Without this, the program could take over a day to run based on the maximum run time set during the building of the executable program.

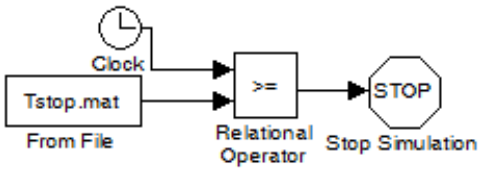

Figure 27: Model stop operation

The output of the ‘Voltage Source’ subsystem block is fed to a ‘Current Measurement’ subsystem. This subsystem, seen in Figure 28, is used as a buffer before the current is sent to the ‘Current Measurement Calculation’ subsystem. The ‘Current Measurement’ subsystem uses the pre-packaged current measurement blocks available in Simulink’s Physical Modeling Library. A view of the schematic for the ‘Current Measurement Calculation’ subsystem can be found in Figure 29. The RMS current value on each phase is calculated and used to measure the unbalance. The current measurement is controlled by the signal coming from ‘Control Input’ subsystem. This first component used in the ‘Current Measurement’ subsystem is a switch, which is used to control whether or not the current unbalance and RMS blocks see the true current or the value of zero, simply put whether the motor is on or off. As mentioned earlier, the Symcom Motor Saver (SMS) only uses measurement of the motor’s current to control the motor’s operation when the motor is running. When the motor is not running, the system evaluates the voltage to determine whether or not it is safe to operate the motor. The

simulation uses similar control system architecture as the SMS system to read in the measured current values or a fixed value of zero. If the control input is high, the RMS and current unbalance subsystem will be passed the measured current, if the control input is low then the value of zero is passed through the system.

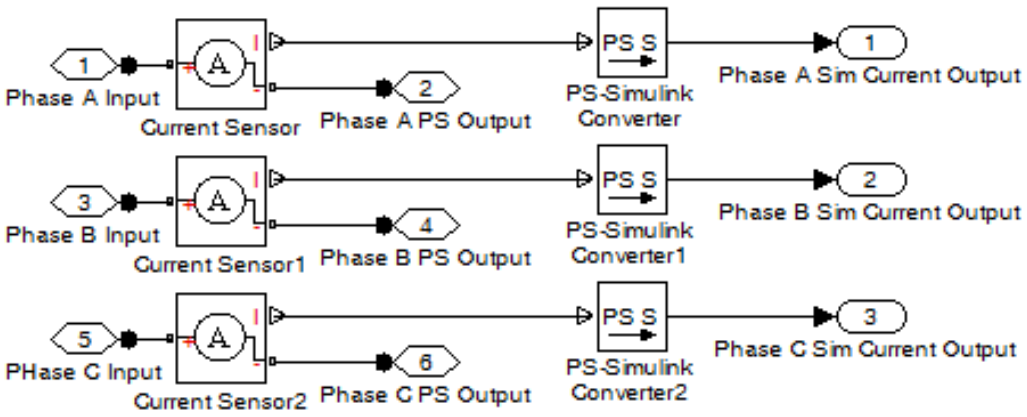

Figure 28: Model current measurement system

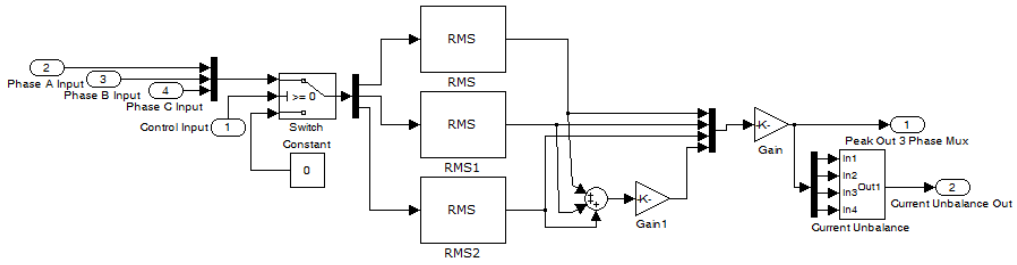

Figure 29: Model current measurement calculation system

The ‘Current Unbalance’ subsystem is implemented within the ‘Current Measurement Calculation’ subsystem. This subsystem calculates the current unbalance of the motor. This subsystem is not fed the motor current unless the control input signal is true. This is the same way the SMS protection system operates, if the control input is high then the measured current is passed through to the system. If the pump is operational, the measured current is then passed through the switch which evaluates

whether or not the RMS current is greater than 10 A. This is done to keep the calculation of a value close to zero, which will slow the simulation down due to precision of the simulation engine. In order to capture and update the maximum and minimum current values recorded, minimum and maximum measurement blocks are used respectively. These maximum and minimum values are used in the calculation of the current unbalance. The RMS current values are subtracted from the minimum and maximum currents using built in Simulink subtraction blocks and the output is then fed into an absolute value block. That output value is divided by the average current and the result is the percentage of current unbalance this unbalance is the final step in completing the unbalance equation seen in Equation 12. This value is then multiplied by a gain of 100 and then fed to the output of the subsystem block. This is all shown in Figure 30. Inside of the 'Current Unbalance' subsystem, the trip signal for the current unbalance can be found. This system evaluates the magnitude of current unbalance and compares it to a set of trip times, which were found written in the SMS manual. The signal is then sent to the trip timers, which hold the control signal low for the times proportional to the times in the manual for the SMS protection system these two subsystems can be found in Figure 31and Figure 32. In Figure 31, the trip signals are sent to an OR block and the output is sent to the motor control subsystem. A table of the trip times can be seen in table 2, these times represent the time before the motor protection system will allow the motor to attempt to restart. The trip calculation subsystem seen in Figure 32 measures the trip flag for a high condition and calculates the severity of the trip, once the trip flag has reached a time based average (Times based on the SMS datasheet table provided in Table 3) of 1 then the final trip condition is sent to the current unbalance trip subsystem.

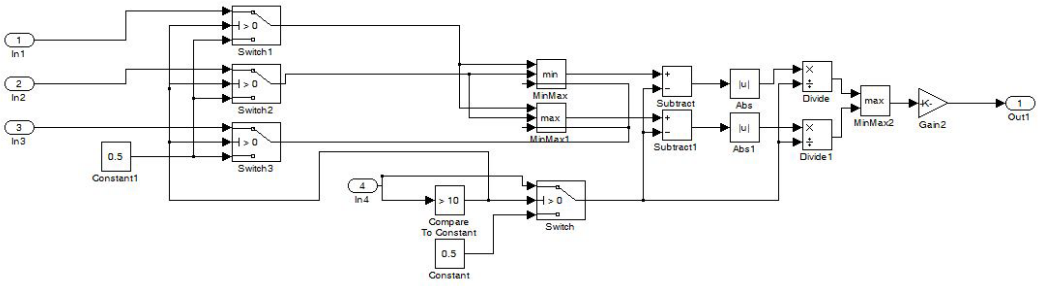

Figure 30: Unbalance Calculation Subsystem

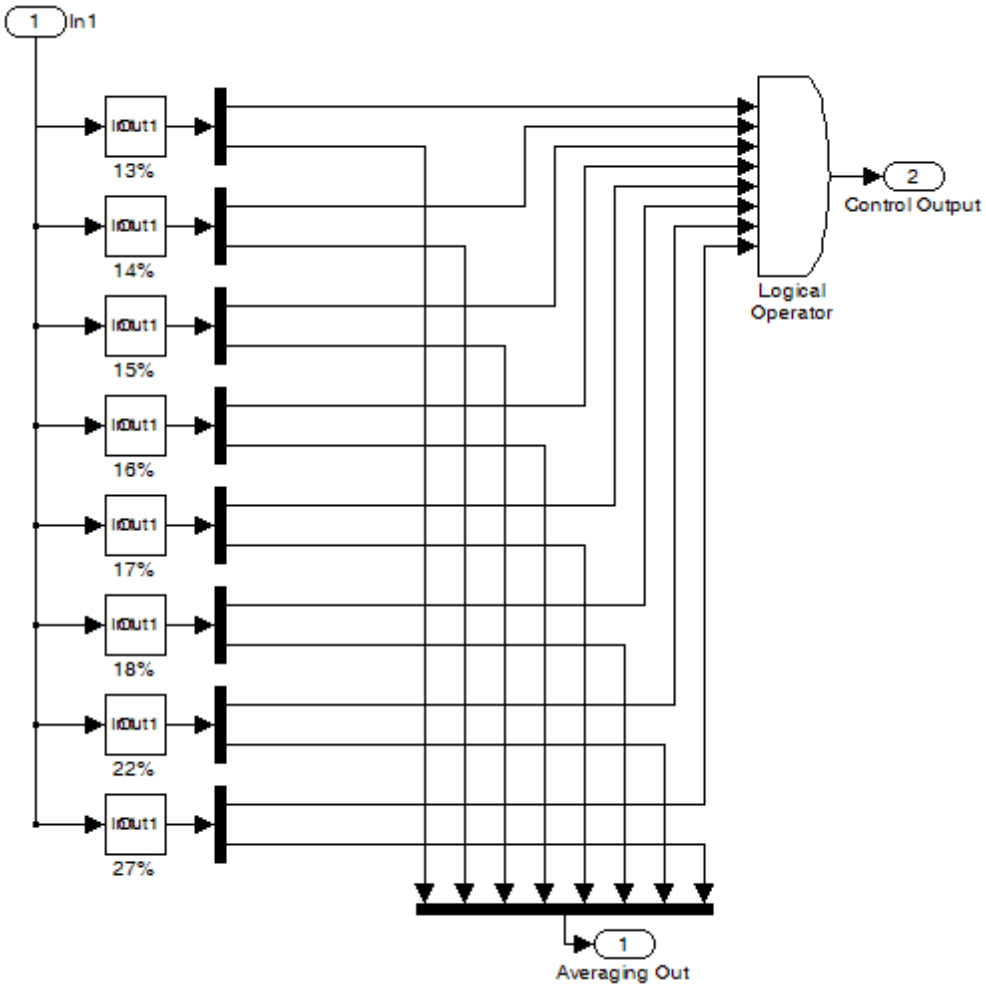

Figure 31: Model current unbalance trip system

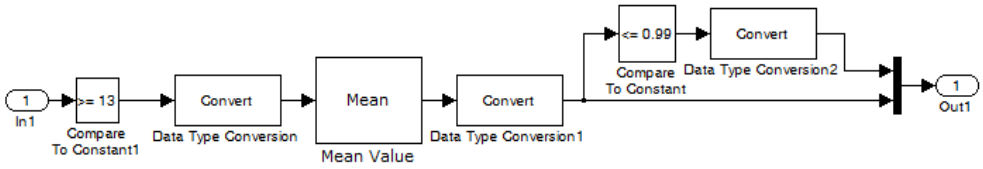

Figure 32: Model current unbalance trip calculation subsystem

The next subsystem in the model is the 'Physical Modeling' subsystem which is used to represent the three-phase circuit breaker. This block, seen in Figure 33, contains three electrical switch models, which are controlled by the 'Control Input' signal. Finally, this subsystem provides the input to the 'Motor' subsystem.

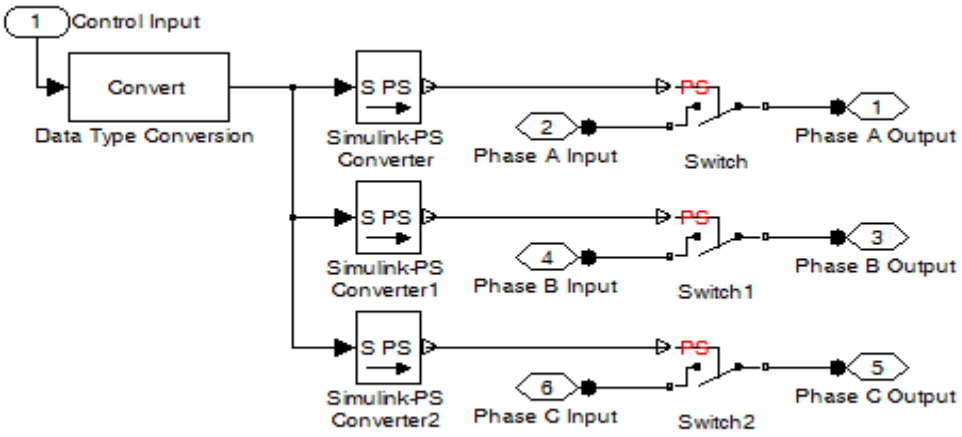

Figure 33: Model three phase breaker box

The 'Voltage Measurement' subsystem, which can be seen in Figure 34, is used to monitor the system voltages and evaluate the voltage unbalance. Once fed into the subsystem, the line-to-line voltages between each of the phases are measured and the results are multiplexed (merged into once signal line) together and then passed out of the subsystem as the voltage measurement signal. The output voltage signal is then sent to another switch, similar to the one used in the 'Current Measurement' subsystem, which is

also controlled by the ‘Control Input’ signal. When the control signal is high, meaning the switch is closed, the voltage signal is passed on to a set of RMS blocks. One RMS block is used to calculate each of the three line-to-line voltages. When the control signal is low, a zero is sent to the switch, causing it to open and nothing to be passed to the measurement blocks. The average line-to-line voltage measurements are then summed together and the resulting value is divided by the number of phases. Next, these peak values of the voltage for each of the RMS calculations are obtained by multiplying them by the square root of two. All four measurements are then fed to the ‘Voltage Unbalance’ subsystem seen in Figure 30.

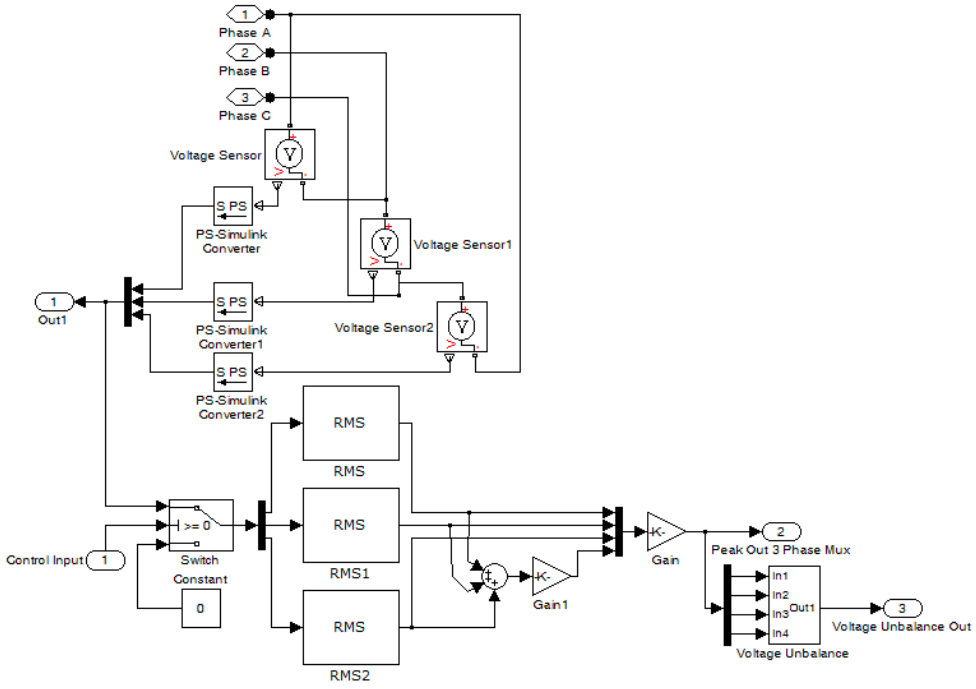

Figure 34: Model voltage measurement system

The ‘Voltage Unbalance’ block is identical to the ‘Current Unbalance’ subsystem block with the only exception being that the inputs are voltage values instead of current values. This subsystem evaluates the value of voltage unbalance within the motor itself.

The subsystem is not given the real system voltage unless the control input signal fed to it is true. Next they are fed through a switch that determines whether or not the RMS voltage is over 10 V. This is done to keep the calculation of value close to zero to a minimum, this small calculations slow the simulation down, this is done as a check If the average voltage is less than 10 V a value of 0.5 is fed into the calculation system, this is done to keep the voltage unbalance from going too large when values are close to zero. Next the minimum and maximum values of the voltage are updated similar to the way the minimum and maximum currents were saved earlier and the voltage unbalance is calculated using these values and the RMS voltages calculated earlier. The voltage unbalance is sent to the motor control subsystem, the other voltage measurements are stored in files for plotting following completion of the simulation.

Motor Model:

The motor is modeled in the classical form using a set of three single-phase series resistance and inductance (RL) circuits connected in a ‘wye’ formation. The ‘wye’ connection style was chosen based on the data sheet of the motor and recorded datasets. The stator resistance and inductance are connected in series and that is paralleled with the resistive core losses and inductive line losses. This electrical representation is then paralleled with the series RL connection which represents the rotor’s resistance and inductance. Initially, the stator resistance and inductance values were estimated to be roughly two times the value of the line-to-line impedance found within the datasheet of the motor since the line-to-line impedance of a motor is most often measured on the rotor side. It is common to take that value and double it to represent the stator as an initial guess. Using this value, along with the fixed values available to represent the rotor’s resistance and inductance, the core losses were then empirically

derived using the datasets recoded by both the local utility and a local engineering firm. After the initial simulations were run, it was determined that these estimates would have to be adjusted in order to more accurately model the motor. Using the formulas for no load and blocked rotor tests along with the recorded data, more accurate values for the motor parameters were empirically derived by the simulation using an iterative process along with the recorded data. The no load test, is where the motor has no load and the rated voltage is supplied [8]. The input current, supply voltage and power are measured. From these measurements the core resistance R_C shown in Equation 17 and reactance X_M shown in Equation 18 can be calculated [8].

$$R_C = \frac{V_{no\ load}{}^2}{P_{no\ load}}$$

Equation 17: Core loss resistance [8]

$$\mathrm{X_M} = \frac{\mathrm{V_{no\ load}}^2}{\mathrm{Q_{no\ load}}}$$

Equation 18: Magnetizing reactance [8]

The blocked rotor test starts with the rotor locked and an increasing voltage at reduced frequency (typically 15 Hz) being applied in order to achieve rated current [8]. Once this has been achieved the voltage, current and power are measured. Using the measured voltage and current the blocked circuit impedance can be calculated using Equation 19 [8]. From the measured power and current the motor resistance can be calculated using Equation 20, along with the known rotor or stator resistance to calculate the respective resistance needed to complete Equation 21 [8]. Using both the circuit impedance and the motor resistance the equivalent reactance can be found using Equation 22, and from that the stator and rotor reactance's from Equation 23.

$$\mathrm{Z_{blocked}} = \frac{\mathrm{V_{blocked}}}{\mathrm{I_{blocked}}}$$

Equation 19: Circuit impedance [8]

$$R_e = \frac{P_{blocked}}{{I_{blocked}}^2}$$

Equation 20: Equivalent motor resistance [8]

$$R_{rot} = R_e - R_{sta}$$

Equation 21: Rotor resistance [8]

$$X_e = \left(\frac{60\ \text{Hz}}{15\ \text{Hz}}\right)\left(\sqrt{{Z_{blocked}}^2 - {R_e}^2}\right)$$

Equation 22: Equivalent reactance [8]

$$X_{sta} = X_{rot} = \frac{X_e}{2}$$

Equation 23: Stator and rotor reactance [8]

The rotor resistance was based on the total resistance of each phase is multiplied by the power factor minus the stator resistance which is the value listed in the motor datasheet. The inductance value of both the stator and rotor is calculated by multiplying the total impedance of each phase by the sine of the power factor angle. The core and line lumped parameters were found using the power factor angle and the current. The resistance uses the cosine and the inductance the sine of the of the power factor angle in the calculation.

The value of the core and line losses are adjusted based on the value of the voltage unbalance, as the voltage unbalance increases the inductance and resistance for each phase is adjusted accordingly. This is done to increase the sensitivity of the motor, without any adjustments the motor achieves steady state within a few cycles unlike a real motor subjected to unbalances, as well as to account for the change in these values as the motor heats up. Because no data has been collected which measures the motor's thermal properties, it was not possible to include a thermal model of the motor in this

work. Voltage unbalance can often be a good indicator of the thermal properties of the motor so that is the reason why this was included. It is worth noting that the use of voltage unbalance measurement to model the thermal effects on the motor in this application is a novel idea, research has not found any models that combine all the systems and applications described in this report. Traditional induction motor models do not take thermal properties into account. In order for this empirically derived motor model to operate in a fashion similar to the average voltage and current unbalance values recorded by the local utility and a local engineering firm, the core and line losses of each phase of the model had to be constantly adjusted. Voltage unbalance and the resulting current unbalances cause internal heating in the motor. This in turn causes the properties of the materials to change namely the resistance. This can be seen in Equation 24.

$$R = R_0 [1 + \alpha (T - T_0)]$$

Equation 24: Resistor temperature dependence [27]

Where the T is the current temperature, T_0 is the original temperature, α is the resistor material thermal coefficient, R_0 is the starting resistance, and R is the total resistance. It can be seen that the resistance is proportionally related to the change in temperature of the material used to create the resistor. In this model the resistance and the inductances of the losses are changed proportionally to voltage unbalance in real time in order to replicate the same voltage and current unbalances as the data recorded.

In order for the model to operate without any major time delays, the motor has an additional voltage source that feeds the motor when the breakers are turned off this keeps the motor in an operating state, if the motor subsystem is not supplied a voltage the simulation enters zero unsolvable calculation loop that it is never able to exit. This

allows the speed of the model to be increased. The values of the core and line losses are pulled from the datasets during simulation.

Chapter 3

Model Control Panel and Output:

As previously mentioned, the Simulink model utilizes the mathematical computational tools embedded in Matlab. For this reason, the models input files are saved in the .mat format. Each of these input files are setup and saved to this format using Matlab scripts that were written to accompany the model. The user operation of the model as well as all data analysis are designed using a main Graphical User Interface (GUI), which is seen in Figure 35: Main program page.

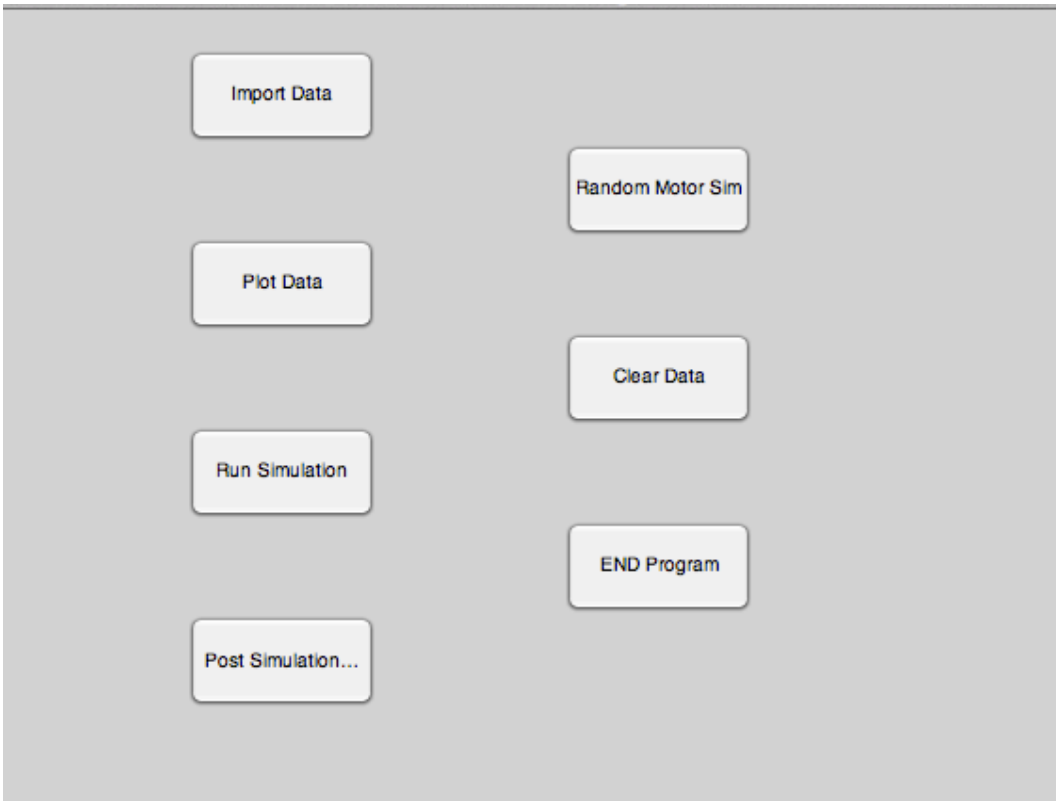

Figure 35: Main program page

The GUI provides the option to import a dataset, plot the imported data, and run a simulation using the imported data set or using the random signal generator. Once the

simulation is complete, the results can be plotted as well. Six categories are available when it comes to plotting the datasets including voltage, current, power, power factor, voltage unbalance, and current unbalance. Post simulation, the option exists to the plot the motor's control signals as well. A few figures, which demonstrate the types of plots, generated from the simulation results are found in Figure 36 and Figure 37. In Figure 36, the current unbalance can be seen spiking at the beginning and end of each run as well as a few times during the middle of each motor run. It normally operates between 6-9% current unbalance. Though there were instances where the current unbalance went above 12%, which is the cutoff for the SMS motor protection system. The control signal plots seen in Figure 37, show the details surrounding when the motor trips. The first of the four plots shown is the file control signal, which tells the simulation when the motor was on based on the data recorded. The unbalance control signal is the signal of when a trip has occurred, while the third plot is the timed trip signal that is control by the magnitude of the unbalance at the time of the trip. The final plot shown is the final control signal that the motor actually sees, it is the AND logic output of all the previous plots. While this figure does not show a trip occurring, the current unbalance does show where it did go above the first threshold for a trip. With that in mind the SMS system may have tripped at these times depending on how long it stayed above the threshold.

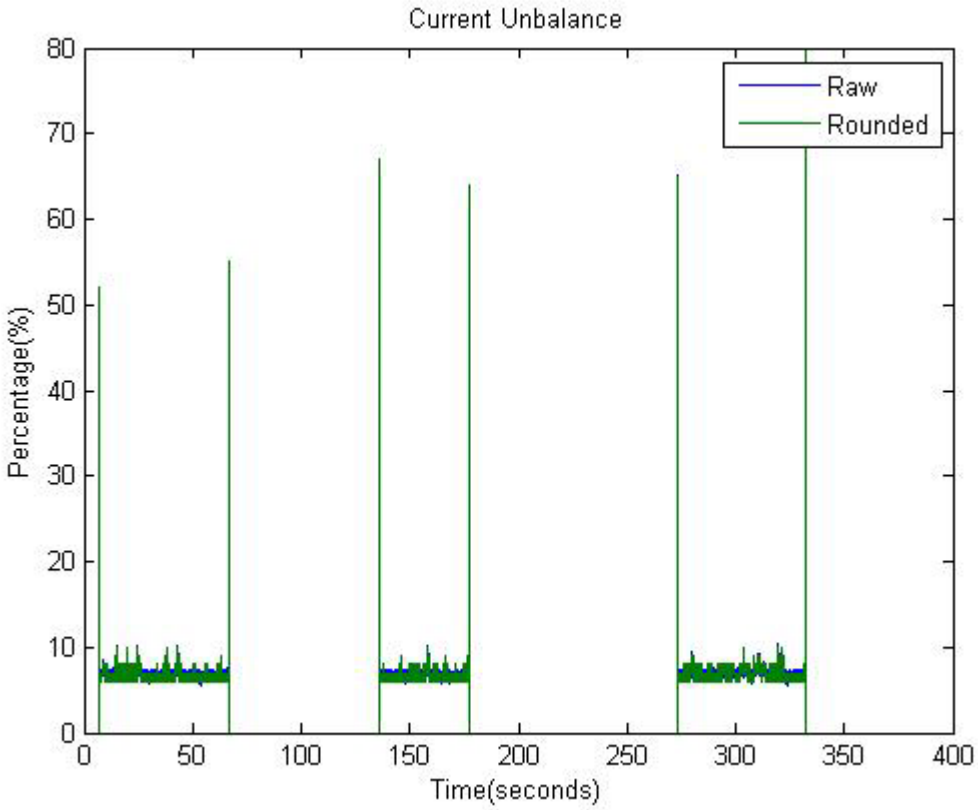

Figure 36: Current unbalance simulation results example from June 23, 2011 dataset

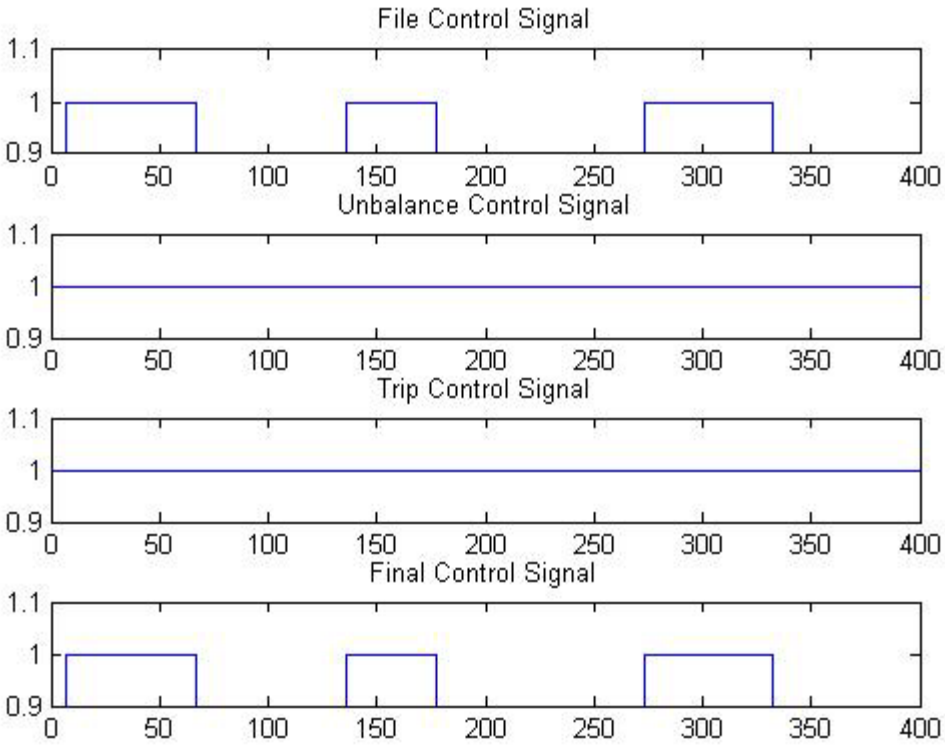

Figure 37: Control signals simulation results example from June 23, 2011 dataset

Simulation of the datasets from June 23rd and 24th, 2011, show occurrences of unbalances that likely caused the motor protection system to trip. When these trips occurred there were high and low voltage conditions that were most likely the cause. From the plots shown in Figure 18 and Figure 19, it can be seen where the voltages phase B and C, along with the line to line voltages of BC and CA where much higher that phase A or the line to line voltage of AB. These high and low voltage conditions contribute to the already small current unbalance that can occur in the motor. In this case the condition causes the system to go outside the bounds of its operational limits.

Local utility corrections:

Immediately, the local utility initiated many corrections to their distribution to a local residential subdivision. The first corrections that were made to the system involved adjusting the voltage regulators and adjusting the tap settings on the multi-feed transformers. In each case, the voltages were lowered to approximately 277 V from 283 V. This was done to prevent overvoltage conditions, which stopped the motor from turning on, especially after trip events had occurred. As previously eluded to, it is extremely important that the voltage being fed to submersible pumps be adjusted so that the source voltage is low enough by the voltage drop across the lines feeding the motor so that over voltages are not applied to the motor terminals where overheating the motor could occur. When a motor is properly installed, the installer will carefully measure the motor depth and length of line between the source and the motor so that the voltage drops can be accurately estimated. When a local residential subdivision was questioned, the line length as well as the gauge of the wire could not be verified. It is very likely possible that this is the largest problem, which caused the trip events if the voltage applied to the terminals is too high.

Chapter 4

Conclusions:

This thesis has documented the troubles encountered by a North Dallas Subdivision during the summer of 2011 when a 40 HP Franklin Electric submersible pump that they use to harvest underground water repeatedly tripped while operational. It was mentioned in the report that there are many concerns with the installation of the motor, as many of the important properties, such as the motor depth and line length, could not be accurately obtained. As indicated, several modifications to the distribution system were made by the local utility, which were all primarily aimed at lowering the voltage sent to the motor, which did prevent any tripping events from occurring after the they were made. This further suggested that the depth of the motor, and there for the line length, were shorter than they are supposed to be. Lowering the voltage helped fix this problem. As requested by the local utility a computer aided model of the motor as well as the motor controller, which monitors and controls the motor operation, was developed using the Matlab/Simulink® modeling platform. The model accurately models both aspects of the system in as closely a manner as possible. It offers the ability to simulate the system using actual data collected off the system when it is operational or random signals can be generated by the software. The model provides several plotting tools which enable the user to plot data collected on the operational system as well as the results which are generated by the simulation. The model has met all of the requirements desired by the customer, a local utility provider, and has already been delivered to their team for future use.

Appendix A

ReadMe file for Operating the Analysis and Simulation Program

Version 1.0

ReadMe file for Operating the Analysis and Simulation Program

Version 1.0

This document will provide the operating instructions for the analysis and simulation program developed by UTA for a local utility. The program is an analysis & simulation tool for a single three-phase submersible pump motor.

Introduction to research project:

Everyone uses induction motors every day even without thinking about it. As an example, some applications include drive motors for ceiling fans, washing machines, dryers, air conditioning systems, assembly lines, large and small-scale power generation, and water pumping systems. The latter application is of particular interest in the report documented here.

A North Dallas, Texas residential subdivision utilizes three submersible pumps to harvest water from underground wells and store it in above ground storage vessels. Two of the installed motors are smaller and run in a near continuous mode of operation to restore the average usage of the community. The third pump, a Franklin 40 Horsepower, with a 6-inch discharge, is larger and handles the bulk loading of the above ground storage when needed. Due to the larger size of the third motor, a commercially available motor controller, Symcom Motor Saver, is used to oversee the operation and safety of it, while the smaller motors rely mainly on conventional circuit breaker protection. In the summer of 2011, in which record heats were recorded for the majority of the summer, the motor controller on the larger unit had many unexpected trip events, which left the community without access to water for an unacceptable amount of time. Through the cooperation of the community and the local electric cooperative, an investigation into what events were causing the controller to trip was conducted. Though many immediate fixes by the local utility, which will be described later, were successful in overcoming the problems experienced, the exact cause of the trip events were not easily understood. The cause of the tripping is believed to be from current or voltage unbalance. While the majority of trips were single events, there were some days where sequential trips were recorded. A study has been commissioned by the local utility to study the causes of the unbalance, potential solutions to the problems and a model that can be used to predict and test for similar problems across their utility service grid.

Introduction to Analysis and Simulation Program:
The model has been developed in two unique versions. The first allows the user to model the system using random input signals generated by the software. This allows the user to simulate random events, which may occur naturally in the system as well as those specifically desired by the user. The second version allows the user to input real system data, which is recorded using analog to digital data acquisition devices while the system is operational. This allows the user to use real system data to model the motor and controller operation to better understand why the system behaved a certain way as the events actually occurred. The recording of this data is done using data acquisition system, the system records real time data and generates a report based on averages of the system during a given period. This model has been created using MATLAB/Simulink along with their SimPowerSystems and SimScape Simulink toolboxes.

Template description:
The template must not contain any special characters i.e. (%, #, & etc.) these will cause major errors in the final version.

This template provided has 7 header rows and 59 columns. These columns include phase to neutral voltage, line-to-line voltage, current, power, reactance, total complex power, power factor, and maximum phase to neutral voltage, minimum phase to neutral voltage, maximum line-to-line voltage, and minimum line-to-line voltage.

For the power factor if the user has the recorded power factor then it can be placed in the appropriate columns. If not the default power factor for this motor simulation is 0.707 or an angle of 45 degrees.

The power factor is used to complete all the calculations in the template file.

Template instructions:
In order to have the dataset correspond to variables correctly in the simulation model a template has been created. This template will allow the user to construct the dataset in the correct form. The first step is to copy all the data from the original dataset and paste it into the Utility Data page of this template. This can be done by right clicking the gray arrow at the top left hand corner of the spreadsheet and using the paste special command. In the paste special window select the Values and number formats condition. This will ensure proper data copying.
Once this is completed the next step is to take the data and copy only the data that starts with and includes the following line. Paste this data into the Copy Page of the spreadsheet in the same form as above in instruction one.

Table A4: Example Header

	Voltage		**V Unbalance**	**Current**		**I Unbalance**

The final data correction to be made is the power factor. If the power factor when the motor is running is not the same value as listed please change field to the correct power factor at the bottom of this page.
The data will now be completed and the data set is ready to be saved. This can be done by selecting the Final Template page of the spreadsheet and going to File > Save As > Save as type > .csv format. This will save only the data in the final formatted page. Please remember where this data is saved in order to use it in the simulation.

Program warnings and notes:
The data to be imported into the program must match the template provide in the demo folder (template.xls). It must also be saved in .csv format, due to the nature of the program. This can be done by going to the menu option File > Saveas > .csv in some Excel programs the Saveas option is under the windows button on the top left hand corner.

This program is designed to be run off a flash drive that is designated to the "G:\\" file directory. The original flash drive is set to "G:\\" directory by default but if it is not in the "G:\\" drive the directions of how to change the drive letter are linked below by operating system.

Table A5: Windows Operating Systems

Windows Operating Systems		
Windows 7	Windows Vista	Windows XP

This program also requires the Matlab compiler runtime engine. This can be found on the Matlab website
And on the flash drive in the folder MCR_Engine. It is a operating specific file please use the 32 bit or 64 bit based upon your operating system. To find which operating system you are using you can go to Start > Control Panel > System.

Main operation page:
Upon opening the program the first page that will be seen is the main operation page. On this page there are six operations that will be discussed in detail later on in this manual. The six operations are import data, plot data, run simulation, post simulation plotting, clear data, and end program. Import data is the first operation to be done. Unless data has been previously loaded and not cleared there will be no data for plotting and simulation.

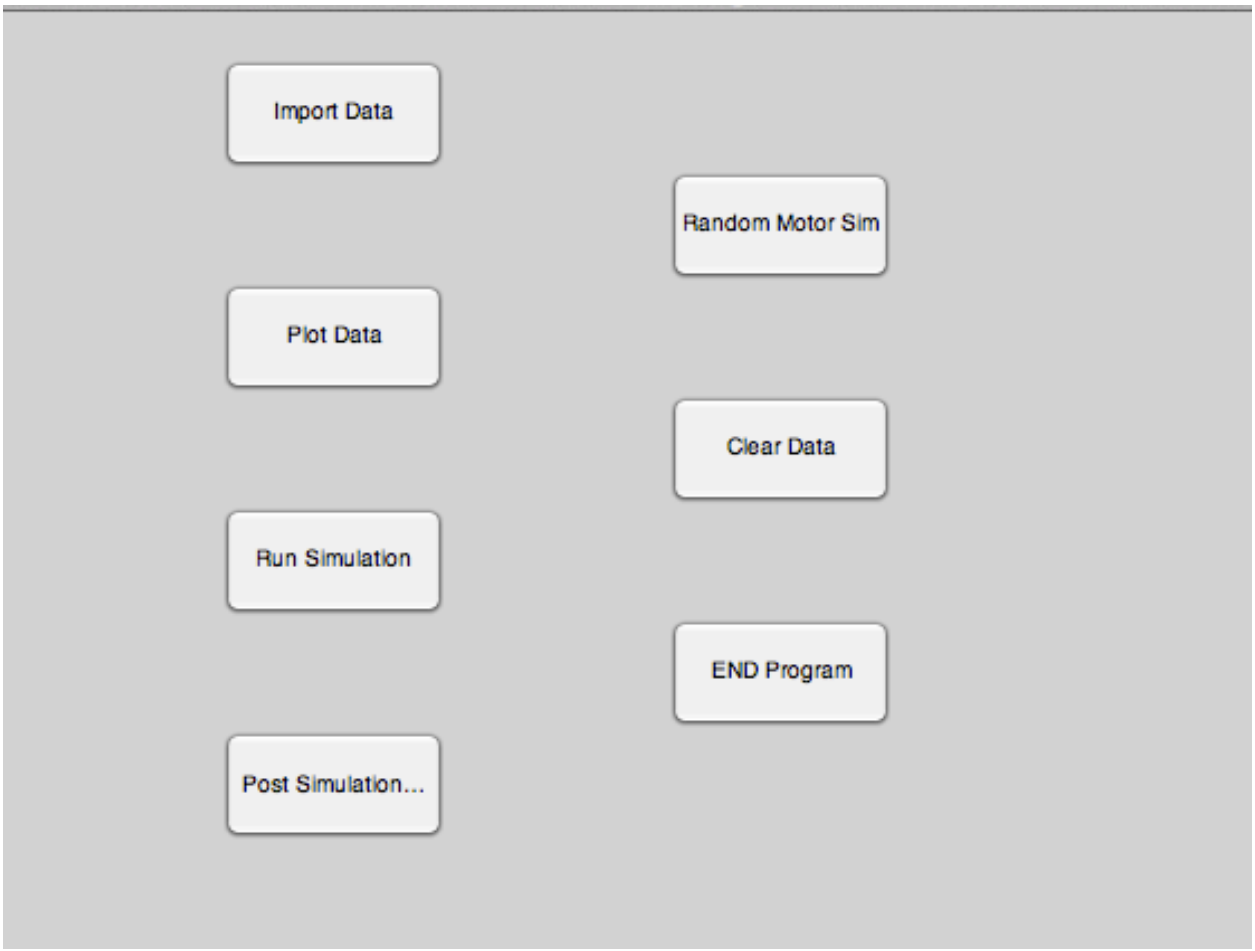

Figure A38: Main Operation Page

Main operation page import data:
The importation of data for this program is a very strict operation that relies on the data being in the provided template form for proper analysis and simulation operation. Upon clicking the import data button a warning popup will appear. This warning is just a reminder that the data must be in the proper format and contain no special characters.

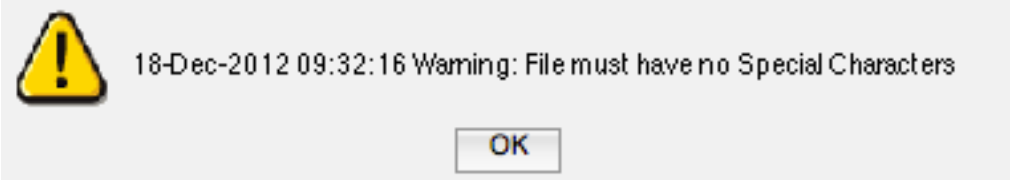

Figure A39: Program Warning
After the warning has been cleared a file operation box will open, this box will allow the user to import the data file that will be used for the remaining operations.

The program will then import this file and begin its manipulation of the file.

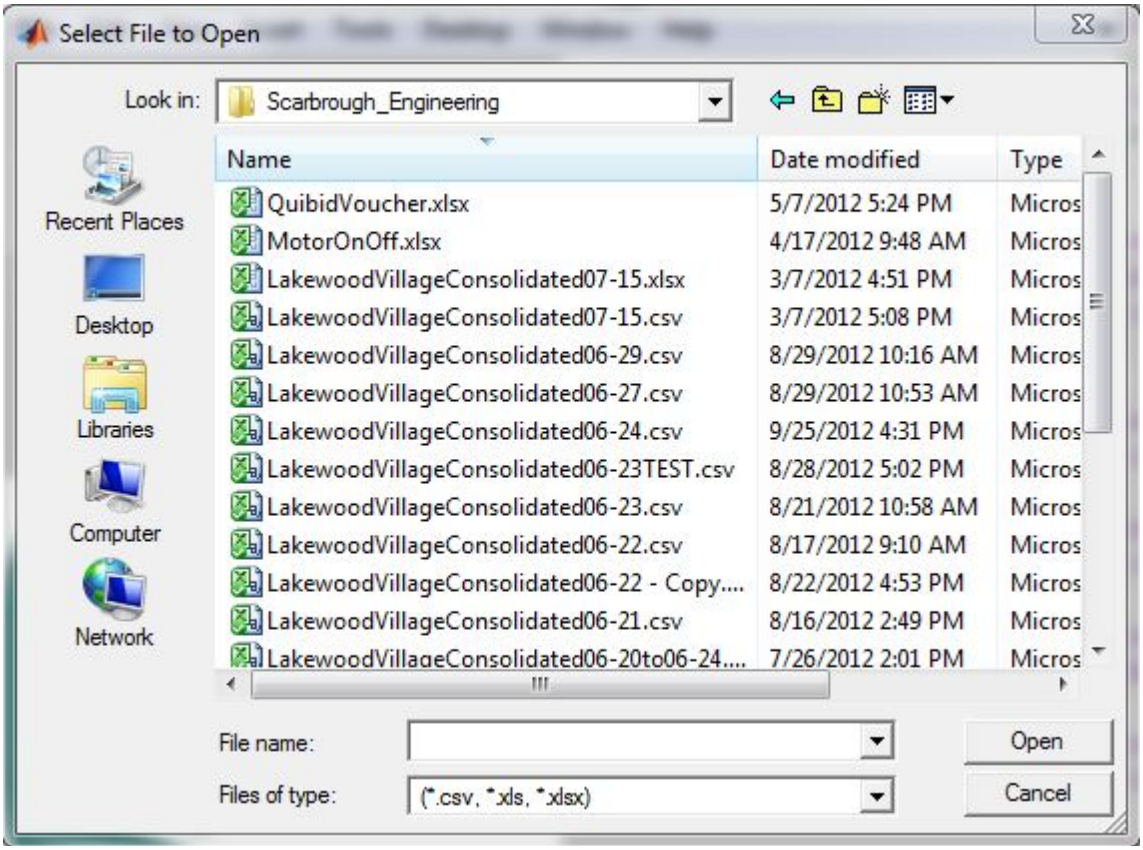

Figure A40: File Operation Window

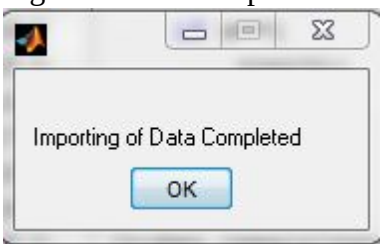

Figure A41: Importing Complete

At the conclusion of the file operation a popup window will appear letting the user know that the program has completed its operation.

Main operation page plot data:
The plot data button will open a set of popup windows that will allow the user to plot the data that was imported in the import data program. The first window will ask what data category is to be plotted phase-to-phase voltage, current, phase voltage, power, and power factor.

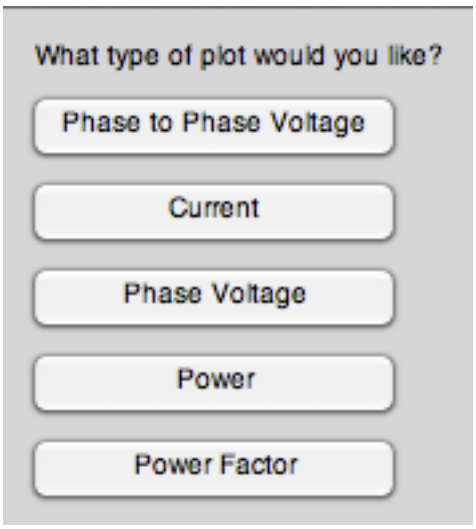

Figure A42: Data Categories

The next request is what data set within the category is to be plotted.

Figure A43: Phase-to-Phase Voltage plots available

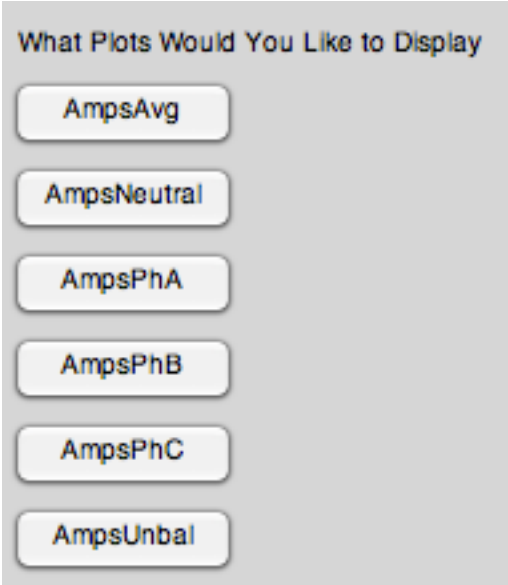

Figure A44: Current Plots available

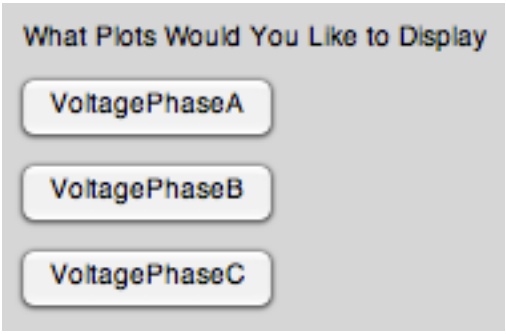

Figure A45: Phase voltage plots available

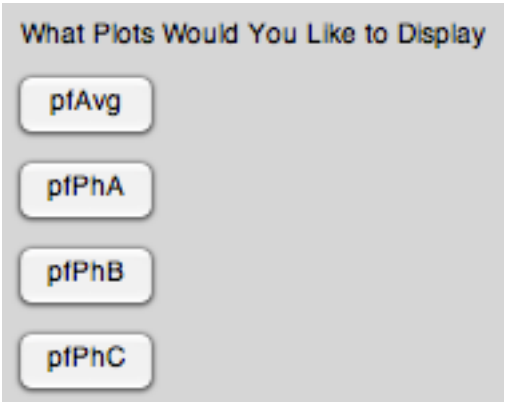

Figure A46: Power factor plots available

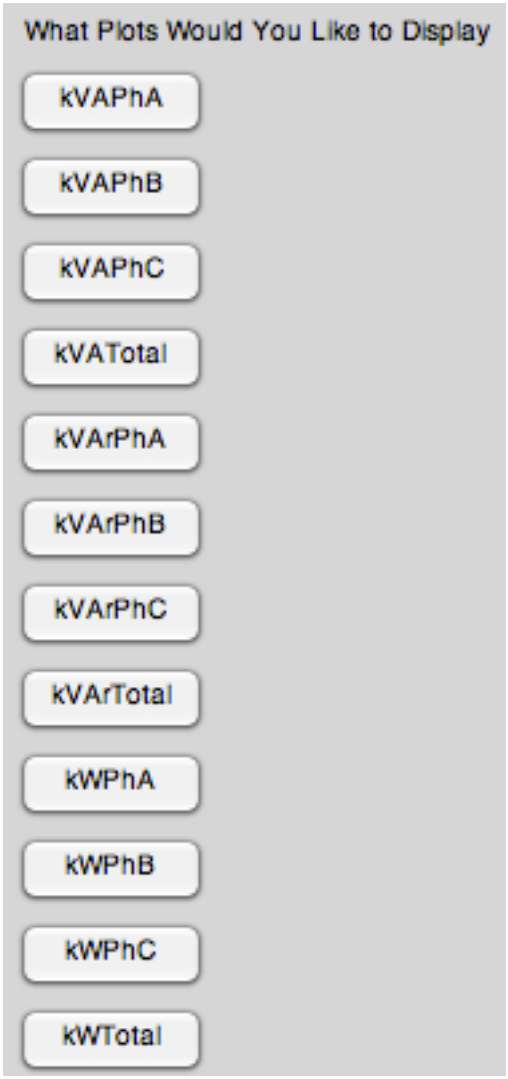

Figure A47: Power plots available

Once the data within the category is chosen the next option is the color of the plot.

Figure A48: Plot colors available

After all of these options have been chosen the program will ask the user if they are done with the plot. If not then the user can click no and the program will return to the data set window, where an additional plot from the category can be overlaid the original plot.

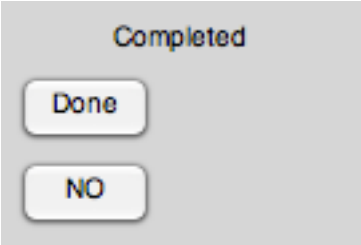

Figure A49: Plot completed

Once the user is done they can use the done button to move to the next plot or operation within the program.

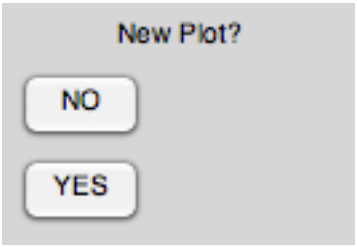

Figure A50: New plot from same category

The next program request is to know if the user would like to create an additional plot from the same category if yes then the will be returned to the initial data category page. If the answer is no the user will see the request for a new plot sequence.

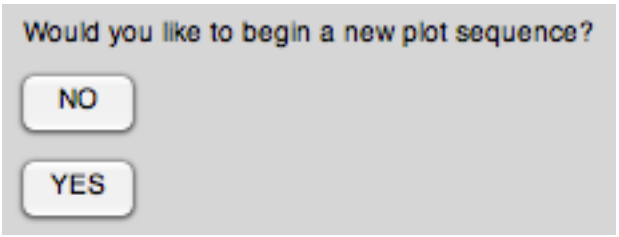

Figure A51: New plot from different category

The final request is the new plot sequence if yes is pushed the program will return the user to the data category selection page. If the user has completed their data analysis from plotting they can click the done button and be returned to the main program window.

Main operation page run simulation:
This button will initialize the parameters to run the simulation and then begin the simulation. This can take several minutes once the simulation begins. The first popup that will appear will ask the user for the simulation stop time.

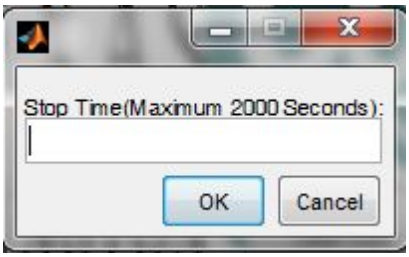

Figure A52: Run time of simulation

After the user has given the stop time the program will calculate the estimated run time of the simulation.

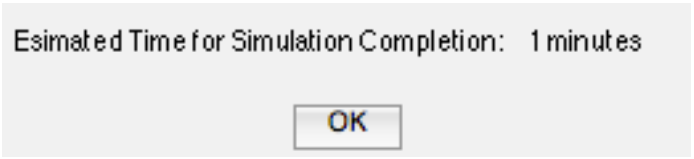

Figure A53: Estimated completion time for simulation

The program then begins the simulation; there will be no notification until it is completed.

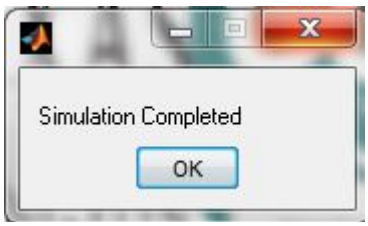

Figure A54: Simulation Completed

The program will notify the user when the simulation has been completed and then the user will then be returned to the main page.

Main operation page post simulation plotting:
This button will bring up a page with the options for the plots produce by the simulation. The plot options are voltage signals, current signals, control signals, voltage unbalance, current unbalance, and unbalance averages.

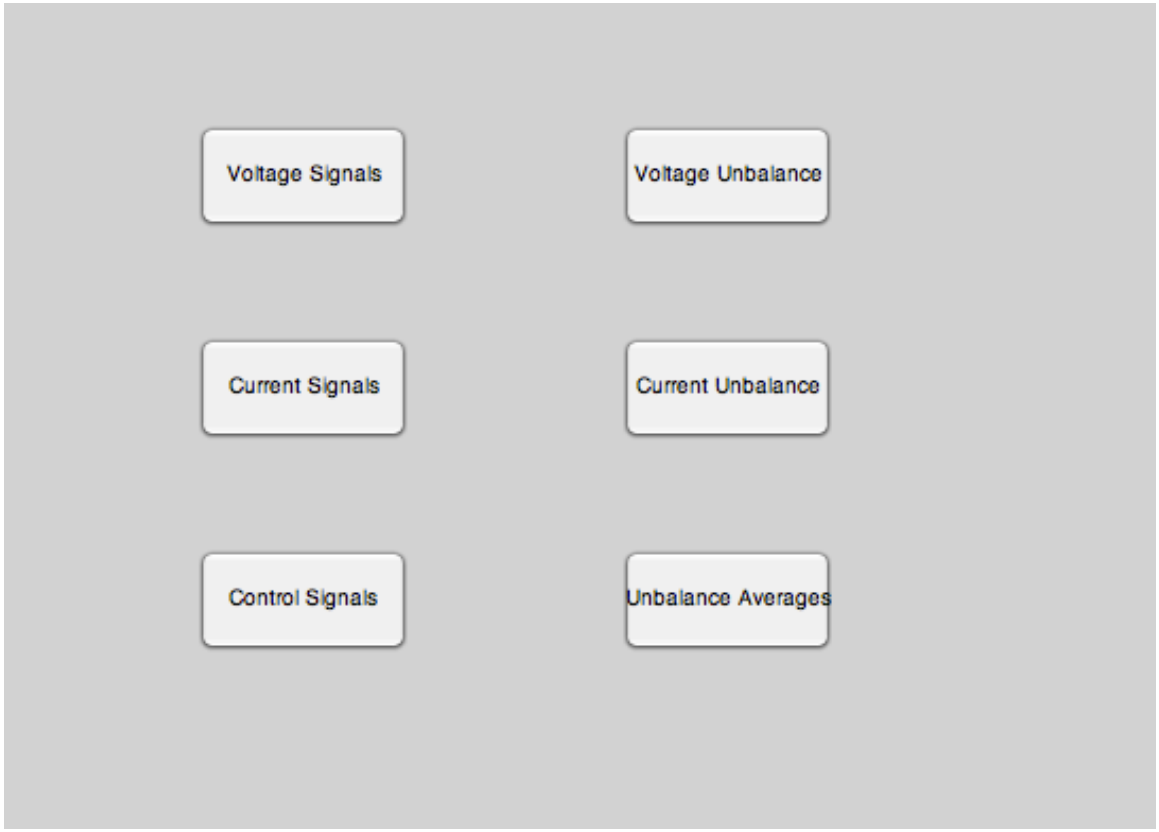

Figure A55: Post simulation plots available

The voltage signals button will open another page with buttons for the plotting of the phase to neutral peak voltage from the file, full sine wave phase to phase simulation voltage, and peak line to line simulation voltage.

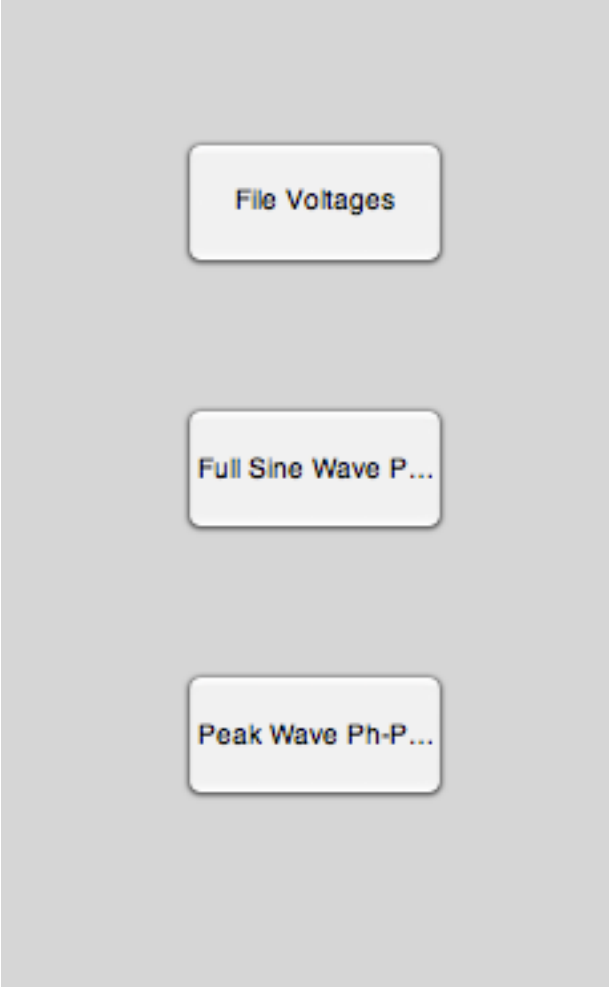

Figure A56: Post simulation voltage plots available

The five other buttons will not open additional windows. The buttons will only open the plots. The current signals that will be plotted are the three phase motor current. The control signal plot contains the control signal based on the values from the file, the control signal from current unbalance calculation, the control signal from the trip timers, and the final plot is the final control signal fed to the three phase breaker. This final control signal is the multiplication of the three previous control signals. The voltage and current unbalance plots have the raw and rounded signals of the respective unbalances. The final post simulation plot option is the unbalance averages. This plot has two windows that one with the four lowest time averages and the second with the longest averages. This is the signal that is then fed to current unbalance control timers. If the values in these averages stay at one for the given time length a trip signal is given.

Main operation page random simulation:

This button will open the preparation windows for the randomly generated voltage changes and motor turn on. This model allows the user to change the maximum and minimum voltages per phase, and the core losses of the motor per phase. This model is designed to lump line losses into the core losses, please take this into account when changing the parameters.

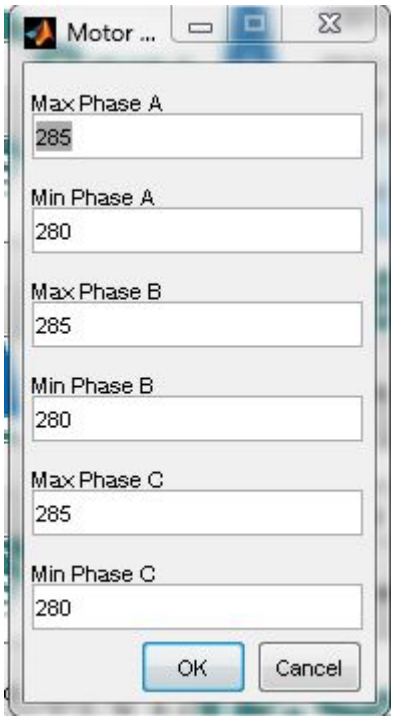

Figure A57: Motor Phase Voltage Settings

This window allows the user to set the maximum and minimum voltages per phase for the simulation.

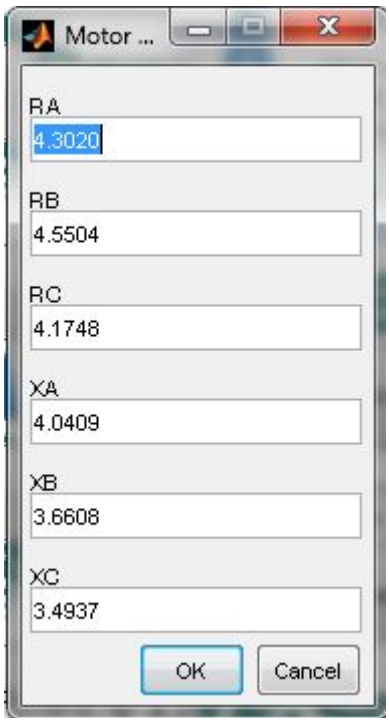

Figure A58: Motor Core Losses Settings

The parameter window allows the user to set the resistive and inductive(in ohms) core losses per phase.

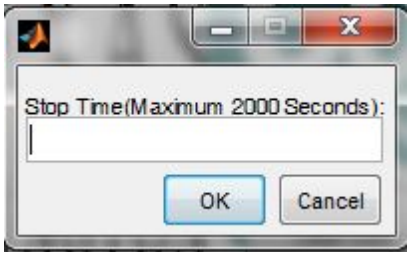

Figure A59: Run time of simulation

After the user has given the stop time the program will calculate the estimated run time of the simulation.

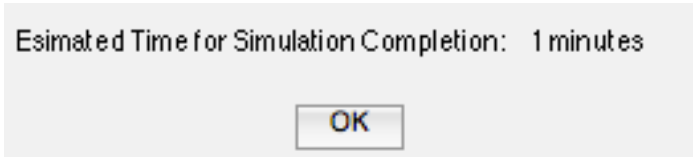

Figure A60: Estimated completion time for simulation

The program then begins the simulation; there will be no notification until it is completed.

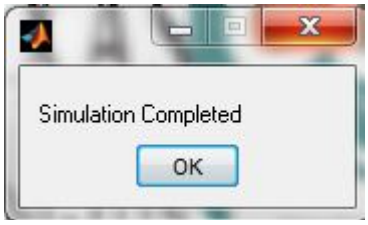

Figure A61: Simulation Completed

The program will notify the user when the simulation has been completed. When the program has completed the program will open the post simulation plot window. This window will allow the user to plot the different parameters of the model simulation.

The post simulation plot windows are the same as previously described above with the one exception of the File Voltage button. This button has been removed since the simulation does not require a voltage table from a file.

Main operation page clear data:
The clear data button will delete the internal data file produced by the program. It is recommended that this be cleared at least once every two runs. This is due to the large size of the data that is produced during long simulation runs, large amounts of disk.

Main operation page end program:
This will close the program but does not delete the data. This allows the user to reuse the data at a future date for simulation.

Figure window:

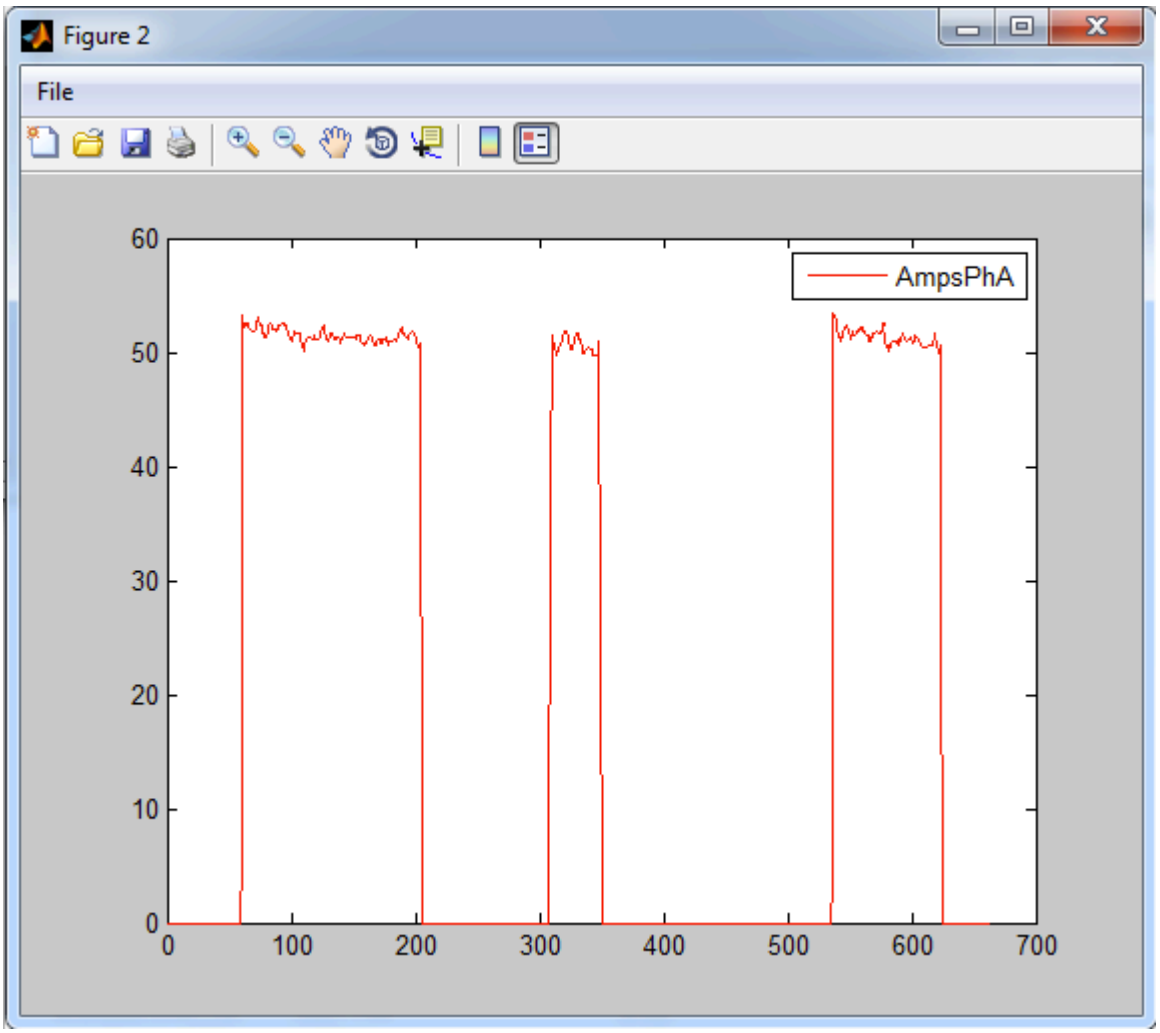

Figure A62: Example Figure Window

There are eleven options at the top of the figure window, going from left to right they are.

New Figure: For this application this should not be needed. It could be used to open a new figure window in order to open a previously saved matlab figures.

Open: This will allow the user to open previously save matlab figure.

Save: This allows the user to save a figure in multiple image formats for later use in other programs.

Print: Allows the user to print the current figure.

Zoom In: This feature will allow the user to zoom in on any portion of the figure.

Zoom Out: This feature will allow the user to zoom out on any portion of the figure.

Pan: Allows the user the ability to shift a zoomed in figure within the axis limits.

Rotate 3d: Allows 3d figures to be rotate along any of the three axis's, there are no 3d figures n this application.

Point Cursor: This allows the user to move point by point along the figure.

Color Bar: Inserts a color legend into the figure, this is not needed for this application.

Legend: Inserts a legend into the figure. Legends are already applied to all figures. This feature does allow for these legend to be edited if necessary

Contact Info:

Programmer and developer:
James Reed andrew.konamusic@gmail.com

Principal Investigators:
David Wetz wetz@uta.edu
Wei-jen Lee wlee@uta.edu

Warranty:

This program is designed to be used with the residential subdivision datasets. While it is compatible with any individual three-phase submersible pump it is not guaranteed to be accurate due to the motor designs.

Appendix B

MainGUI64Bit Code Developed for Executable File

```
% Copyright James Andrew Reed 2012 Working for a local utility & UTA
clear all; close all; clc; clear memory; format compact;
delete('G:\Project64Bit\*.mat')
delete('G:\Project64Bit\mcrCache7.17\MainGU0\MainGUI64bit\*.mat')
delete('G:\TEMP_MATLAB\*.mat')

h=figure;
% create an axes that spans the whole gui
ah = axes('unit', 'normalized', 'position', [0 0 0.25 0.25]);
% import the background image and show it on the axes
bg = imread('coserv-logo.jpg'); imagesc(bg);
% prevent plotting over the background and turn the axis off
set(ah,'handlevisibility','off','visible','off')
% making sure the background is behind all the other uicontrols
uistack(ah, 'bottom');

ah = axes('unit', 'normalized', 'position', [0.25 0.25 0.25 0.25]);
bg = imread('coserv-logo.jpg'); imagesc(bg);
set(ah,'handlevisibility','off','visible','off')
uistack(ah, 'bottom');

ah = axes('unit', 'normalized', 'position', [0.5 0.5 0.25 0.25]);
bg = imread('coserv-logo.jpg'); imagesc(bg);
set(ah,'handlevisibility','off','visible','off')
uistack(ah, 'bottom');
```

```
ah = axes('unit', 'normalized', 'position', [0.75 0.75 0.25 0.25]);
bg = imread('coserv-logo.jpg'); imagesc(bg);
set(ah,'handlevisibility','off','visible','off')
uistack(ah, 'bottom');

ah = axes('unit', 'normalized', 'position', [0 0.5 0.25 0.25]);
bg = imread('coserv-logo.jpg'); imagesc(bg);
set(ah,'handlevisibility','off','visible','off')
uistack(ah, 'bottom');

ah = axes('unit', 'normalized', 'position', [0.25 0.75 0.25 0.25]);
bg = imread('coserv-logo.jpg'); imagesc(bg);
set(ah,'handlevisibility','off','visible','off')
uistack(ah, 'bottom');

ah = axes('unit', 'normalized', 'position', [0.5 0 0.25 0.25]);
bg = imread('coserv-logo.jpg'); imagesc(bg);
set(ah,'handlevisibility','off','visible','off')
uistack(ah, 'bottom');

ah = axes('unit', 'normalized', 'position', [0.75 0.25 0.25 0.25]);
bg = imread('coserv-logo.jpg'); imagesc(bg);
set(ah,'handlevisibility','off','visible','off')
uistack(ah, 'bottom');

ah = axes('unit', 'normalized', 'position', [0.25 0 0.25 0.25]);
```

```
bg = imread('UTA_1H.jpg'); imagesc(bg);
set(ah,'handlevisibility','off','visible','off')
uistack(ah, 'bottom');

ah = axes('unit', 'normalized', 'position', [0.5 0.25 0.25 0.25]);
bg = imread('UTA_1H.jpg'); imagesc(bg);
set(ah,'handlevisibility','off','visible','off')
uistack(ah, 'bottom');

ah = axes('unit', 'normalized', 'position', [0.75 0.5 0.25 0.25]);
bg = imread('UTA_1H.jpg'); imagesc(bg);
set(ah,'handlevisibility','off','visible','off')
uistack(ah, 'bottom');

ah = axes('unit', 'normalized', 'position', [0.75 0 0.25 0.25]);
bg = imread('UTA_1H.jpg'); imagesc(bg);
set(ah,'handlevisibility','off','visible','off')
uistack(ah, 'bottom');

ah = axes('unit', 'normalized', 'position', [0 0.25 0.25 0.25]);
bg = imread('UTA_1H.jpg'); imagesc(bg);
set(ah,'handlevisibility','off','visible','off')
uistack(ah, 'bottom');

ah = axes('unit', 'normalized', 'position', [0.25 0.5 0.25 0.25]);
bg = imread('UTA_1H.jpg'); imagesc(bg);
```

```
set(ah,'handlevisibility','off','visible','off')
uistack(ah, 'bottom');

ah = axes('unit', 'normalized', 'position', [0.5 0.75 0.25 0.25]);
bg = imread('UTA_1H.jpg'); imagesc(bg);
set(ah,'handlevisibility','off','visible','off')
uistack(ah, 'bottom');

ah = axes('unit', 'normalized', 'position', [0 0.75 0.25 0.25]);
bg = imread('UTA_1H.jpg'); imagesc(bg);
set(ah,'handlevisibility','off','visible','off')
uistack(ah, 'bottom');

ha=uicontrol('style','pushbutton','string','Import Data','position',[100,350,100,50]);
set(ha,'Callback','[FullSet,rowi,x]=ImportDataMain32bit;')

hc=uicontrol('style','pushbutton','string','Plot Data','position',[100,250,100,50]);
set(hc,'Callback','PlotingMain32bit(x);')

he=uicontrol('style','pushbutton','string','Run Simulation','position',[100,150,100,50]);
set(he,'Callback','RunMotor32bit(x);')

hf=uicontrol('style','pushbutton','string','Post Simulation
Plotting','position',[100,50,100,50]);
set(hf,'Callback','PostSimPlotGUI32bit')
```

```
hd=uicontrol('style','pushbutton','string','Clear Data','position',[300,200,100,50]);
set(hd,'Callback','ClearData32bit')

hd=uicontrol('style','pushbutton','string','Random Motor Sim','position',[300,300,100,50]);
set(hd,'Callback','MotorSimRGVversion')

hb=uicontrol('style','pushbutton','string','END Program','position',[300,100,100,50]);
set(hb,'Callback','close(h);clear all;clc')
```

Appendix C

ImportDataMain32bit Code Developed for Executable File

```
function [ FullSet,rowi,x ] = ImportDataMain()
%IMPORTDATAMAIN Summary of this function goes here
%   Detailed explanation goes here

str=datestr(clock);
warn=msgbox([str,' Warning: File must have no Special Characters'],'Start of
Program','warn');
pause(5)
[Filename,path]=uigetfile('*.csv;*.xls;*.xlsx');
FullSet=importdata([path,Filename],',',15);

for i=1:15
   token=strtok(FullSet.textdata(i,1),',');
   x1=strcmp(token,'Time');x2=strcmp(token,'Date');x=or(x1,x2);
   if x==1
      rowi=i;
      break
   end
end

[token,remain]=strtok(FullSet.textdata(1,1),',');
x=strcmp(char(token),'RMS Voltage & Current Interval Report');

if x==1
   CoSERVdata32bit(FullSet,rowi)
end
```

```
if x==0
    ScarData32bit(FullSet,rowi)
    ScarCurrentUnbal32bit
end

msgbox('Importing of Data Completed')

end
```

Appendix D

CoSERVdata32bit Code Developed for Executable File

```
function [] = CoSERVdata(FullSet,rowi)
%COSERVDATA Summary of this function goes here
%   Detailed explanation goes here
[r,c]=size(FullSet.data);
    str2=char(FullSet.textdata(rowi,1));
    str1=char(FullSet.textdata(rowi-1,1));
    str=char(FullSet.textdata(rowi-2,1));
    compflag=0;compflag1=0;compflag2=0;
    count1=0;count2=0;count3=0;
    [token2,remain2]=strtok(str2,',');
    [token2,remain2]=strtok(remain2,',');
    while(compflag<2)
    if compflag1==0 & count1==0
        [token,remain]=strtok(str,',');
        else
        [token,remain]=strtok(remain,',');
    end
    while(compflag1<3)
        if compflag1==0 & count2==0
        [token1,remain1]=strtok(str1,',');
        else
        [token1,remain1]=strtok(remain1,',');
        end
    while(compflag2<3)
        clear strspace strfile strtest;
        if compflag2==0 & count3==0
```

```
        [token2,remain2]=strtok(remain2,',');
        else
        [token2,remain2]=strtok(remain2,',');
        end
    strspace=char([token,token1,token2]);
    x=length(char([token,token1,token2]));
    charcount=1;test=isstrprop(strspace,'alphanum');
    for i=1:x
        if test(i)==1
            strfile(charcount)=strspace(i);
            charcount=charcount+1;
        end
    end
    disp(strfile)
    strtest=strcat(strfile,'=FullSet.data(:,',num2str(count3+1),')');
    evalc(strtest);
    strtest=strcat(strfile,'.mat');
    save(strtest,strfile);

copyfile('G:\Project64Bit\*.mat','G:\Project64Bit\mcrCache7.17\MainGU0\MainGUI64bit')
    if count3==8
        count3=count3+1;
        [token2,remain2]=strtok(remain2,',');
        disp([token,token2])
        strspace=char([token,token2]);
    x=length(char([token,token2]));
```

```
            charcount=1;test=isstrprop(strspace,'alphanum');
            for i=1:x
                if test(i)==1
                    strfile(charcount)=strspace(i);
                    charcount=charcount+1;
                end
            end
            disp(strfile)
            strtest=strcat(strfile,'=FullSet.data(:,',num2str(count3+1),')');
            evalc(strtest);
            strtest=strcat(strfile,'.mat');
            save(strtest,strfile);

copyfile('G:\Project64Bit\*.mat','G:\Project64Bit\mcrCache7.17\MainGU0\MainGUI64bit')
            end
            if count3==18
                count3=count3+1;
                [token2,remain2]=strtok(remain2,',');
                disp([token,token2])
                    strspace=char([token,token2]);
            x=length(char([token,token2]));
            charcount=1;test=isstrprop(strspace,'alphanum');
            for i=1:x
                if test(i)==1
                    strfile(charcount)=strspace(i);
                    charcount=charcount+1;
```

```
            end
        end
        disp(strfile)
        strtest=strcat(strfile,'=FullSet.data(:,',num2str(count3+1),')');
        evalc(strtest);
        strtest=strcat(strfile,'.mat');
        save(strtest,strfile);

copyfile('G:\Project64Bit\*.mat','G:\Project64Bit\mcrCache7.17\MainGU0\MainGUI64bit')
        end
        compflag2=compflag2+1;
        count3=count3+1;
        end
        compflag1=compflag1+1;
        compflag2=0;
        count2=count2+1;
        end
        compflag=compflag+1;
        compflag1=0;
        count1=count1+1;
        end
        %delete('C:\Users\jar7046\Desktop\Dropbox\Submersible Pump
Working\EXEWorkingModels\*.mat')

copyfile('G:\Project64Bit\*.mat','G:\Project64Bit\mcrCache7.17\MainGU0\MainGUI64bit')
    end
```

Appendix D

ScarData32bit Code Developed for Executable File

```
function [] = ScarData( FullSet, rowi )
%UNTITLED Summary of this function goes here
%   Detailed explanation goes here

    [r,c]=size(FullSet.data);
    str=FullSet.textdata(rowi,1);
    [token,remain]=strtok(str,',');

    for i=1:c
        x=isnan(FullSet.data(:,i));
        if min(x)==0
            clearvars test r c strtest strf str;
            strf=[];
            [token,remain]=strtok(remain,',');
            test=isstrprop(char(token),'alphanum');
            [r,c]=size(test);
            k=0;
            strtest=char(token);
            for j=1:c
                if test(j)==0
                    k=k+1;
                end
                j+k;
                if(test(j)==1 & (j)<=c)
                    strf(j-k)=strtest(j);
                end
```

```
                end
                strf=char(strf);
                str=strcat(strf,'=FullSet.data(:,',num2str(i),')');
                evalc(str);
                strtest=strcat(strf,'.mat');
                save(strtest,strf);
            end
        end

copyfile('G:\Project64Bit\*.mat','G:\Project64Bit\mcrCache7.17\MainGU0\MainGUI64bit')
    end
```

Appendix E

ScarCurrentUnbal32bit Code Developed for Executable File

```
function [ output_args ] = ScarCurrentUnbal( input_args )
%SCARCURRENTUNBAL Summary of this function goes here
%   Detailed explanation goes here


copyfile('G:\Project64Bit\*.mat','G:\Project64Bit\mcrCache7.17\MainGU0\MainGUI64bit')
    list=dir('G:\Project64Bit\*.mat');len=length(list);
    for i=1:len
        load(list(i).name);
    end

    A=length(AmpsPhA);

    for i=1:A
        Avg(i)=(AmpsPhA(i)+AmpsPhB(i)+AmpsPhC(i))/3;

        u1=abs(AmpsPhA(i)-Avg(i));
        u2=abs(AmpsPhB(i)-Avg(i));
        u3=abs(AmpsPhC(i)-Avg(i));

        AmpsUnbal(i)=(max(max(u1,u2),max(u2,u3))/Avg(i))*100;
    end

    save('AmpsUnbal','AmpsUnbal')

copyfile('G:\Project64Bit\*.mat','G:\Project64Bit\mcrCache7.17\MainGU0\MainGUI64bit')
```

```
   list=dir('G:\Project64Bit\*VPhPh*.mat');len=length(list);
for i=1:len
    load(list(i).name);
end

A=length(VPhPhAB);

for i=1:A
    Avg(i)=(VPhPhAB(i)+VPhPhBC(i)+VPhPhCA(i))/3;

    u1=abs(VPhPhAB(i)-Avg(i));
    u2=abs(VPhPhBC(i)-Avg(i));
    u3=abs(VPhPhCA(i)-Avg(i));

    VPhPhUnbal(i)=(max(max(u1,u2),max(u2,u3))/Avg(i))*100;
end

save('VPhPhUnbal','VPhPhUnbal')

copyfile('G:\Project64Bit\*.mat','G:\Project64Bit\mcrCache7.17\MainGU0\MainGUI64bit')


     [r,c]=size(VoltagePhaseA);
for i=1:r
```

```
    test=[VoltagePhaseA(i,1),VoltagePhaseB(i,1),VoltagePhaseC(i,1)];
    MaxVPhN(i)=max(test);
    MinVPhN(i)=min(test);
    AvgVPhN(i)=mean(test);
    Num(i)=max(abs(MaxVPhN(i)-AvgVPhN(i)),abs(MinVPhN(i)-AvgVPhN(i)));
    VPhN_Unbal(i)=(Num(i)/AvgVPhN(i))*100;

    test=[VPhPhAB(i,1),VPhPhBC(i,1),VPhPhCA(i,1)];
    MaxVPhPh(i)=max(test);
    MinVPhPh(i)=min(test);
    AvgVPhPh(i)=mean(test);
    Num(i)=max(abs(MaxVPhPh(i)-AvgVPhPh(i)),abs(MinVPhPh(i)-AvgVPhPh(i)));
    VPhPh_Unbal(i)=(Num(i)/AvgVPhPh(i))*100;

    test=[AmpsPhA(i,1),AmpsPhB(i,1),AmpsPhC(i,1)];
    MaxIPhN(i)=max(test);
    MinIPhN(i)=min(test);
    AvgIPhN(i)=mean(test);
    Num(i)=max(abs(MaxIPhN(i)-AvgIPhN(i)),abs(MinIPhN(i)-AvgIPhN(i)));
    IPhN_Unbal(i)=(Num(i)/AvgIPhN(i))*100;

end

AvgRa=abs((AvgVPhPh'.^2)./(kWPhA*1e3));
 AvgRb=abs((AvgVPhPh'.^2)./(kWPhB*1e3));
```

```
AvgRc=abs((AvgVPhPh'.^2)./(kWPhC*1e3));


z=1;length(AvgRa)
for j=1:length(AvgRa)
   if AvgRa(j)>10 && AvgRa(j)<35
      ARa(z)=AvgRa(j);
      ARb(z)=AvgRb(j);
      ARc(z)=AvgRc(j);
      z=z+1;
   end
end
Rac=mean(ARa)
save Rac Rac

copyfile('G:\Project64Bit\*.mat','G:\Project64Bit\mcrCache7.17\MainGU0\MainGUI64bit')

Rbc=mean(ARb)
save Rbc Rbc

copyfile('G:\Project64Bit\*.mat','G:\Project64Bit\mcrCache7.17\MainGU0\MainGUI64bit')

Rcc=mean(ARc)
save Rcc Rcc

copyfile('G:\Project64Bit\*.mat','G:\Project64Bit\mcrCache7.17\MainGU0\MainGUI64bit')
```

```
            AvgXa=abs((AvgVPhPh'.^2)./(kVArPhA*1e3));
            AvgXb=abs((AvgVPhPh'.^2)./(kVArPhB*1e3));
            AvgXc=abs((AvgVPhPh'.^2)./(kVArPhC*1e3));

            z=1;length(AvgXa)
            for j=1:length(AvgXa)
               if AvgXa(j)>10 && AvgXa(j)<35
                  AXa(z)=AvgXa(j);
                  AXb(z)=AvgXb(j);
                  AXc(z)=AvgXc(j);
                  z=z+1;
               end
            end
            Xac=mean(AXa)/(2*pi*60)
            save Xac Xac

copyfile('G:\Project64Bit\*.mat','G:\Project64Bit\mcrCache7.17\MainGU0\MainGUI64bit')
            Xbc=mean(AXb)/(2*pi*60)
            save Xbc Xbc

copyfile('G:\Project64Bit\*.mat','G:\Project64Bit\mcrCache7.17\MainGU0\MainGUI64bit')
            Xcc=mean(AXc)/(2*pi*60)
            save Xcc Xcc
copyfile('G:\Project64Bit\*.mat','G:\Project64Bit\mcrCache7.17\MainGU0\MainGUI64bit')
         end
```

Appendix F

PlotingMain32bit Code Developed for Executable File

```
function [] = PlotingMain( x )
%PLOTINGMAIN Summary of this function goes here
%   Detailed explanation goes here

if x==1
    CoSERVPlots32bit
end

if x==0
    ScarPlotsV232bit
end

msgbox('Data Ploting Completed Completed')

end
```

Appendix G

CoSERVPlots32bit Code Developed for Executable File

```
function [] = CoSERVPlots()
%COSERVPLOTS Summary of this function goes here
%   Detailed explanation goes here


copyfile('G:\Project64Bit\*.mat','G:\Project64Bit\mcrCache7.17\MainGU0\MainGUI64bit')
    list=dir('G:\Project64Bit\*.mat');len=length(list);
    for i=1:len
        load(list(i).name);
    end

    c=who('Volt*');lenc=length(c);
    d=who('Curr*');lend=length(d);

     z=2;
    while z==2
    main=menu('What type of plot would you like?','Voltage','Current');

    if main==1
    y=2;
    while y==2
    x=2;figure;count=1;
    while x==2
    choice=menu('What Plots Would You Like to Display',c(1:lenc));
    test(count)=choice;
    cc=menu('Color of plot?','Red','Blue','Black','Green','Cyan','Magenta','Yellow');
```

```
color=['r','b','k','g','c','m','y'];
    plot(eval(char(c(choice))),color(cc));hold on
    legend(char(c(test)))
    x=menu('Completed','Done','NO')
    count=count+1;
end
y=menu('New Plot?','NO','YES')
end
clear choice  test
end

if main==2
y=2;
while y==2
x=2;figure;count=1;
while x==2
choice=menu('What Plots Would You Like to Display',d(1:lend));
test(count)=choice;
cc=menu('Color of plot?','Red','Blue','Black','Green','Cyan','Magenta','Yellow');
color=['r','b','k','g','c','m','y'];
    plot(eval(char(d(choice))),color(cc));hold on
    legend(char(d(test)))
    x=menu('Completed','Done','NO')
    count=count+1;
end
y=menu('New Plot?','NO','YES')
```

```
  end
  clear choice test
  end

   z=menu('Would you like to begin a new plot sequence?','NO','YES')

end
```

Appendix H

ScarPlotsV232bit Code Developed for Executable File

```
function [ output_args ] = ScarPlotsV2( input_args )
%SCARPLOTSV2 Summary of this function goes here
%   Detailed explanation goes here


copyfile('G:\Project64Bit\*.mat','G:\Project64Bit\mcrCache7.17\MainGU0\MainGUI64bit')
    list=dir('G:\Project64Bit\*.mat');len=length(list);
    for i=1:len
        load(list(i).name);
    end


     clear all
     list=dir('G:\Project64Bit\*.mat');len=length(list);
    for i=1:len
        load(list(i).name);
    end

    c=who('*PhPh*');lenc=length(c);
    d=who('*Amp*');lend=length(d);
    e=who('*Phase*');lene=length(e);
    f=who('*k*');lenf=length(f);
    g=who('*pf*');leng=length(g);


    z=2;
```

```
while z==2
main=menu('What type of plot would you like?','Phase to Phase
Voltage','Current','Phase Voltage','Power','Power Factor');

if main==1
y=2;
while y==2
x=2;figure;count=1;
while x==2
choice=menu('What Plots Would You Like to Display',c(1:lenc));
test(count)=choice;
cc=menu('Color of plot?','Red','Blue','Black','Green','Cyan','Magenta','Yellow');
color=['r','b','k','g','c','m','y'];
    plot(eval(char(c(choice))),color(cc));hold on
    legend(char(c(test)))
    x=menu('Completed','Done','NO')
    count=count+1;
end
y=menu('New Plot?','NO','YES')
end
clear choice  test
end

if main==2
y=2;
while y==2
```

```
x=2;figure;count=1;
while x==2
choice=menu('What Plots Would You Like to Display',d(1:lend));
test(count)=choice;
cc=menu('Color of plot?','Red','Blue','Black','Green','Cyan','Magenta','Yellow');
color=['r','b','k','g','c','m','y'];
    plot(eval(char(d(choice))),color(cc));hold on
    legend(char(d(test)))
    x=menu('Completed','Done','NO')
    count=count+1;
end
y=menu('New Plot?','NO','YES')
end
clear choice test
end

if main==3
y=2;
while y==2
x=2;figure;count=1;
while x==2
choice=menu('What Plots Would You Like to Display',e(1:lene));
test(count)=choice;
cc=menu('Color of plot?','Red','Blue','Black','Green','Cyan','Magenta','Yellow');
color=['r','b','k','g','c','m','y'];
    plot(eval(char(e(choice))),color(cc));hold on
```

```
    legend(char(e(test)))
    x=menu('Completed','Done','NO')
    count=count+1;
end
y=menu('New Plot?','NO','YES')
end
clear choice test
end

if main==4
y=2;
while y==2
x=2;figure;count=1;
while x==2
choice=menu('What Plots Would You Like to Display',f(1:lenf));
test(count)=choice;
cc=menu('Color of plot?','Red','Blue','Black','Green','Cyan','Magenta','Yellow');
color=['r','b','k','g','c','m','y'];
    plot(eval(char(f(choice))),color(cc));hold on
    legend(char(f(test)))
    x=menu('Completed','Done','NO')
    count=count+1;
end
y=menu('New Plot?','NO','YES')
end
clear choice test
```

```
    end

    if main==5
       y=2;
    while y==2
    x=2;figure;count=1;
    while x==2
    choice=menu('What Plots Would You Like to Display',g(1:leng));
    test(count)=choice;
    cc=menu('Color of plot?','Red','Blue','Black','Green','Cyan','Magenta','Yellow');
    color=['r','b','k','g','c','m','y'];
        plot(eval(char(g(choice))),color(cc));hold on
        legend(char(g(test)))
        x=menu('Completed','Done','NO')
        count=count+1;
    end
    y=menu('New Plot?','NO','YES')
    end
    clear choice test
    end

    z=menu('Would you like to begin a new plot sequence?','NO','YES')

end
```

Appendix I

RunMotor32bit Code Developed for Executable File

```
function [] = RunMotor(x)
%RUNMOTOR Summary of this function goes here
%   Detailed explanation goes here

if x==1
    abc=msgbox('This data set cannot be used for simulation.  Please See Readme file for assistance','Warning','error')
end

if x==0
    ScarMotorControl32bit
    ScarTestScript32bit
end

msgbox('Simulation Completed')

end
```

Appendix J

ScarMotorControl32bit Code Developed for Executable File

```
load('G:\Project64Bit\Rac');load('G:\Project64Bit\Rbc');load('G:\Project64Bit\Rcc');
load('G:\Project64Bit\Xac');load('G:\Project64Bit\Xbc');load('G:\Project64Bit\Xcc');

Va=load('G:\Project64Bit\VoltagePhaseA');Vb=load('G:\Project64Bit\VoltagePhaseB');Vc=load('G:\Project64Bit\VoltagePhaseC');
Ia=load('G:\Project64Bit\AmpsPhA');Ib=load('G:\Project64Bit\AmpsPhB');Ic=load('G:\Project64Bit\AmpsPhC');
Pa=load('G:\Project64Bit\kWPhA');Pb=load('G:\Project64Bit\kWPhB');Pc=load('G:\Project64Bit\kWPhC');

Ia=Ia.AmpsPhA;Ib=Ib.AmpsPhB;Ic=Ic.AmpsPhC;
Va=Va.VoltagePhaseA;Vb=Vb.VoltagePhaseB;Vc=Vc.VoltagePhaseC;
Pa=Pa.kWPhA;Pb=Pb.kWPhB;Pc=Pc.kWPhC;

AmpsPhA=Ia;VoltagePhaseA=Va;VoltagePhaseB=Vb;VoltagePhaseC=Vc;
phia=real(acosd(abs(Pa*1e3)./(Va.*Ia)));  phib=real(acosd(abs(Pb*1e3)./(Vb.*Ib)));  phic=real(acosd(abs(Pc*1e3)./(Vc.*Ic)));
Zsca=Va./Ia; Zscb=Vb./Ib; Zscc=Vc./Ic;

Xsra=0.5.*Zsca.*abs(sind(phia))*1e3/(2*pi*60);
Xsrb=0.5.*Zscb.*abs(sind(phib))*1e3/(2*pi*60);
Xsrc=0.5.*Zscc.*abs(sind(phic))*1e3/(2*pi*60);

Ima=Ia.*sind(phia);  Imb=Ib.*sind(phib);  Imc=Ic.*sind(phic);
Ica=Ia.*cosd(phia);  Icb=Ib.*cosd(phib);  Icc=Ic.*cosd(phic);
```

```
Rs=0.68
Rsa=Zsca.*cosd(phia)-Rs;
Rsb=Zscb.*cosd(phib)-Rs;
Rsc=Zscc.*cosd(phic)-Rs;

Rca=Va./Ica; Rcb=Vb./Icb; Rcc=Vc./Icc;
Xma=Va./Ima; Xmb=Vb./Imb; Xmc=Vc./Imc;

Zma=(j.*Rca.*Xma)./(Rca+j.*Xma);RA=real(Zma);XA=imag(Zma);
Zmb=(j.*Rcb.*Xmb)./(Rcb+j.*Xmb);RB=real(Zmb);XB=imag(Zmb);
Zmc=(j.*Rcc.*Xmc)./(Rcc+j.*Xmc);RC=real(Zmc);XC=imag(Zmc);

zfra=1;zfrb=1;zfrc=1;zfxa=1;zfxb=1;zfxc=1;
for i=1:length(RA)
   if isfinite(RA(i))
     ARA(zfra,1)=RA(i);
     zfra=zfra+1;
   end
   if isfinite(RB(i))
     ARB(zfrb,1)=RB(i);
     zfrb=zfrb+1;
   end
   if isfinite(RC(i))
     ARC(zfrc,1)=RC(i);
     zfrc=zfrc+1;
   end
```

```
    if isfinite(XA(i))
      AXA(zfxa,1)=XA(i);
      zfxa=zfxa+1;
    end
    if isfinite(XB(i))
      AXB(zfxb,1)=XB(i);
      zfxb=zfxb+1;
    end
    if isfinite(XC(i))
      AXC(zfxc,1)=XC(i);
      zfxc=zfxc+1;
    end
end

ARA=mean(ARA);ARB=mean(ARB);ARC=mean(ARC);
AXA=mean(AXA);AXB=mean(AXB);AXC=mean(AXC);

for i=1:length(RA)
    if ~isfinite(RA(i))
        RA(i)=ARA;
    end
    if ~isfinite(RB(i))
        RB(i)=ARB;
    end
    if ~isfinite(RC(i))
        RC(i)=ARC;
```

```
    end
    if ~isfinite(XA(i))
        XA(i)=AXA;
    end
    if ~isfinite(XB(i))
        XB(i)=AXB;
    end
    if ~isfinite(XC(i))
        XC(i)=AXC;
    end
end

RA=timeseries(RA,0:0.5:(length(RA)-1)/2)
save RA -v7.3 RA

RB=timeseries(RB,0:0.5:(length(RB)-1)/2)
save RB -v7.3 RB

RC=timeseries(RC,0:0.5:(length(RC)-1)/2)
save RC -v7.3 RC

XA=timeseries(XA,0:0.5:(length(XA)-1)/2)
save XA -v7.3 XA

XB=timeseries(XB,0:0.5:(length(XB)-1)/2)
save XB -v7.3 XB
```

```
XC=timeseries(XC,0:0.5:(length(XC)-1)/2)
save XC -v7.3 XC

AmpsPhA=timeseries(AmpsPhA,0:0.5:(length(AmpsPhA)-1)/2)
save AmpsPhA -v7.3 AmpsPhA

TAmax=max(VoltagePhaseA);TAmin=min(VoltagePhaseA);TAd=TAmax-TAmin;
TAmax=timeseries(TAmax)
save TAmax -v7.3 TAmax
TAmin=timeseries(TAmin)
save TAmin -v7.3 TAmin
TAd=timeseries(TAd)
save TAd -v7.3 TAd

VoltagePhaseA=timeseries(VoltagePhaseA,0:0.5:(length(VoltagePhaseA)-1)/2)
save VoltagePhaseA -v7.3 VoltagePhaseA

TBmax=max(VoltagePhaseB);TBmin=min(VoltagePhaseB);TBd=TBmax-TBmin;
TBmax=timeseries(TBmax)
save TBmax -v7.3 TBmax
TBmin=timeseries(TBmin)
save TBmin -v7.3 TBmin
TBd=timeseries(TBd)
save TBd -v7.3 TBd
```

```
VoltagePhaseB=timeseries(VoltagePhaseB,0:0.5:(length(VoltagePhaseB)-1)/2)
save VoltagePhaseB -v7.3 VoltagePhaseB

TCmax=max(VoltagePhaseC);TCmin=min(VoltagePhaseC);TCd=TCmax-TCmin;
TCmax=timeseries(TCmax)
save TCmax -v7.3 TCmax
TCmin=timeseries(TCmin)
save TCmin -v7.3 TCmin
TCd=timeseries(TCd)
save TCd -v7.3 TCd

VoltagePhaseC=timeseries(VoltagePhaseC,0:0.5:(length(VoltagePhaseC)-1)/2)
save VoltagePhaseC -v7.3 VoltagePhaseC

copyfile('G:\Project64Bit\*.mat','G:\Project64Bit\mcrCache7.17\MainGU0\MainGUI64bit')
clc
```

Appendix K

ScarTestScript32bit Code Developed for Executable File

```
a=tic;
fprintf('Simulation Start\n\n')
Tstop=str2num(char(inputdlg('Stop Time(Maximum 2000 Seconds): ')));

    p1 =    1.191;
    p2 =    2.623;
ftime =ceil((p1*Tstop + p2 + 15)*1.5/60);
tstr=sprintf('Esimated Time for Simulation Completion: %4.0f minutes\n',ftime)
z=msgbox(tstr);


Tstop=timeseries(Tstop)
save Tstop -v7.3 Tstop
copyfile('G:\Project64Bit\*.mat','G:\Project64Bit\mcrCache7.17\MainGU0\MainGUI64bit')

delete('G:\TEMP_MATLAB\*.mat');
copyfile('G:\Project64Bit\*.mat','G:\TEMP_MATLAB\');

tt=tic;
fprintf('Start\n')
!G:\TEMP_MATLAB\ScarMotorV4.exe
fprintf('Stop\n')
toc(tt)

copyfile('G:\TEMP_MATLAB\*.mat','G:\Project64Bit\mcrCache7.17\MainGU0\MainGUI6
4bit');
```

```
copyfile('G:\TEMP_MATLAB\*.mat','G:\Project64Bit\');


copyfile('G:\Project64Bit\*.mat','G:\Project64Bit\mcrCache7.17\MainGU0\MainGUI64bit')
toc(a)
```

Appendix L

PostSimPlotGUI32bit Code Developed for Executable File

```
function [] = PostSimPlotGUI()
%POSTSIMPLOTGUI Summary of this function goes here
%   Detailed explanation goes here
%
% load Vsignals.mat
% load Isignals.mat
% load IUnBal.mat
% load UnbalControl.mat
% load ControlSignals.mat
% load VPhPhUnBal.mat

f=figure;
fa=uicontrol('style','pushbutton','string','Voltage Signals','position',[100,300,100,50]);
set(fa,'Callback','PSPguiVoltSig32bit')
fb=uicontrol('style','pushbutton','string','Current Signals','position',[100,200,100,50]);
set(fb,'Callback','PSPguiCurrSig32bit')
fc=uicontrol('style','pushbutton','string','Unbalance
Averages','position',[300,100,100,50]);
set(fc,'Callback','PSPguiUnbalAvg32bit')
fd=uicontrol('style','pushbutton','string','Current Unbalance','position',[300,200,100,50]);
set(fd,'Callback','PSPguiCurrUnbal32bit')
fe=uicontrol('style','pushbutton','string','Voltage Unbalance','position',[300,300,100,50]);
set(fe,'Callback','PSPguiVoltUnbal32bit')
ff=uicontrol('style','pushbutton','string','Control Signals','position',[100,100,100,50]);
set(ff,'Callback','PSPguiConSig32bit')
end
```

Appendix M

PSPguiVoltSig32bit Code Developed for Executable File

```
function [] = PSPguiVoltSig()
%PSPGUIVOLTSIG Summary of this function goes here
%   Detailed explanation goes here

v=figure;
va=uicontrol('style','pushbutton','string','File Voltages','position',[100,300,100,50]);
set(va,'Callback','PlotFileVoltage32bit;')
vb=uicontrol('style','pushbutton','string','Full Sine Wave Ph-Ph
Volt','position',[100,200,100,50]);
set(vb,'Callback','PlotFullSineVoltage32bit;')
vc=uicontrol('style','pushbutton','string','Peak Wave Ph-Ph
Volt','position',[100,100,100,50]);
set(vc,'Callback','PlotPeakPhPhVoltage32bit;')

end
```

Appendix N

PlotFileVoltage32bit Code Developed for Executable File

```
function [] = PlotFileVoltage( SimVoltage )
%PLOTFILEVOLTAGE Summary of this function goes here
%   Detailed explanation goes here

load('G:\Project64Bit\Vsignals.mat')
figure
plot(SimVoltage(1,:),SimVoltage(2,:),SimVoltage(1,:),SimVoltage(3,:),SimVoltage(1,:),Si
mVoltage(4,:))
legend('Phase A','Phase B','Phase C');
title('File Voltages')
xlabel('Time(Seconds)')
ylabel('Volts')
clear all

end
```

Appendix O

PlotFullSineVoltage32bit Code Developed for Executable File

```
function [ ] = PlotFullSineVoltage( )
%PLOTFULLSINEVOLTAGE Summary of this function goes here
%   Detailed explanation goes here

load('G:\Project64Bit\Vsignals.mat')
figure
plot(SimVoltage(1,:),SimVoltage(5,:),SimVoltage(1,:),SimVoltage(6,:),SimVoltage(1,:),Si
mVoltage(7,:))
legend('Phase AB','Phase BC','Phase CA');
title('Full Sine Wave Phase-Phase Voltages')
xlabel('Time(Seconds)')
ylabel('Volts')
clear all

end
```

Appendix P

PlotPeakPhPhVoltage32bit Code Developed for Executable File

```
function [ ] = PlotPeakPhPhVoltage(  )
%PLOTPEAKPHPHVOLTAGE Summary of this function goes here
%   Detailed explanation goes here

load('G:\Project64Bit\Vsignals.mat')
figure
plot(SimVoltage(1,:),SimVoltage(8,:),SimVoltage(1,:),SimVoltage(9,:),SimVoltage(1,:),Si
mVoltage(10,:),SimVoltage(1,:),SimVoltage(11,:))
legend('Phase AB','Phase BC','Phase CA','Average Voltage');
title('Peak Phase-Phase Voltages')
xlabel('Time(Seconds)')
ylabel('Volts')
clear all

end
```

Appendix Q

PSPguiCurrSig32bit Code Developed for Executable File

```
function [ ] = PSPguiCurrSig( )
%PSPGUICURRSIG Summary of this function goes here
%   Detailed explanation goes here

load('G:\Project64Bit\Isignals.mat')
figure
plot(SimCurrent(1,:),SimCurrent(2,:),SimCurrent(1,:),SimCurrent(3,:),SimCurrent(1,:),Si
mCurrent(4,:),SimCurrent(1,:),SimCurrent(5,:))
legend('Phase A','Phase B','Phase C','Average Current')
title('System Peak Current')
xlabel('Time(Seconds)')
ylabel('Current(Amps)')
clear all

end
```

Appendix R

PSPguiUnbalAvg32bit Code Developed for Executable File

```
function [ ] = PSPguiUnbalAvg( )
%PSPGUIUNBALAVG Summary of this function goes here
%   Detailed explanation goes here

load('G:\Project64Bit\UnbalControl.mat')
puc=figure;
plot(SimIUnbalMean(1,:),SimIUnbalMean(2,:),SimIUnbalMean(1,:),SimIUnbalMean(3,:),
SimIUnbalMean(1,:),SimIUnbalMean(4,:),SimIUnbalMean(1,:),SimIUnbalMean(5,:))
legend('13%','14%','15%','16%');title('1 of 2 Control Signal Averages for Current
Unbalance');xlabel('Time(seconds)');ylabel('Control Signal');
pud=figure;
plot(SimIUnbalMean(1,:),SimIUnbalMean(6,:),SimIUnbalMean(1,:),SimIUnbalMean(7,:),
SimIUnbalMean(1,:),SimIUnbalMean(8,:),SimIUnbalMean(1,:),SimIUnbalMean(9,:))
legend('17%','18%','22%','27%');title('2 of 2 Control Signal Averages for Current
Unbalance');xlabel('Time(seconds)');ylabel('Control Signal');

end
```

Appendix S

PSPguiCurrUnbal32bit Code Developed for Executable File

```
function [  ] = PSPguiCurrUnbal(  )
%PSPGUICURRUNBAL Summary of this function goes here
%   Detailed explanation goes here

load('G:\Project64Bit\IUnBal.mat')
iu=figure;
plot(IUnBal(1,:),IUnBal(2,:),IUnBal(1,:),IUnBal(3,:));
legend('Raw','Rounded');title('Current Unbalance');
xlabel('Time(seconds)');ylabel('Percentage(%)');

end
```

Appendix T

PSPguiVoltUnbal32bit Code Developed for Executable File

```
function [ ] = PSPguiVoltUnbal(  )
%PSPGUIVOLTUNBAL Summary of this function goes here
%   Detailed explanation goes here

load('G:\Project64Bit\VPhPhUnBal.mat')
vu=figure;
plot(VUnBal(1,:),VUnBal(2,:),VUnBal(1,:),VUnBal(3,:));
legend('Raw','Rounded');title('Voltage Unbalance');
xlabel('Time(seconds)');ylabel('Percentage(%)');

end
```

Appendix U

PSPguiConSig32bit Code Developed for Executable File

```
function [  ] = PSPguiConSig( )
%PSPGUICONSIG Summary of this function goes here
%   Detailed explanation goes here

load('G:\Project64Bit\ControlSignals.mat')
cs=figure
x=round(max(SimControl(1,:)))
subplot(4,1,1)
plot(SimControl(1,:),SimControl(2,:));title('File Control Signal')
axis([0 x 0.9 1.1])
subplot(4,1,2)
plot(SimControl(1,:),SimControl(3,:));title('Unbalance Control Signal')
axis([0 x 0.9 1.1])
subplot(413)
plot(SimControl(1,:),SimControl(4,:));title('Trip Control Signal')
axis([0 x 0.9 1.1])
subplot(414)
plot(SimControl(1,:),SimControl(5,:));title('Final Control Signal')
axis([0 x 0.9 1.1])

end
```

Appendix V

ClearData32bit Code Developed for Executable File

```
function [] = ClearData()

%CLEARDATA Summary of this function goes here

%   Detailed explanation goes here


delete('G:\Project64Bit\*.mat')

delete('G:\Project64Bit\mcrCache7.17\MainGU0\MainGUI64bit\*.mat')

delete('G:\TEMP_MATLAB\*.mat')



end
```

Appendix W

MotorSimRGVversion Code Developed for Executable File

```
% Random Generated Voltage Simulation

delete('G:\Project64Bit\*.mat')
prompt={'Max Phase A','Min Phase A','Max Phase B','Min Phase B','Max Phase C','Min
Phase C'};
dlg_title='Motor Voltages';
num_lines=1;
def={'285','280','285','280','285','280'};
answer=inputdlg(prompt,dlg_title,num_lines,def);

TAmax=str2double(answer(1));
TAmin=str2double(answer(2));
TBmax=str2double(answer(3));
TBmin=str2double(answer(4));
TCmax=str2double(answer(5));
TCmin=str2double(answer(6));

TAmax=timeseries(TAmax);
TAmin=timeseries(TAmin);
TBmax=timeseries(TBmax);
TBmin=timeseries(TBmin);
TCmax=timeseries(TCmax);
TCmin=timeseries(TCmin);

save TAmax -v7.3 TAmax;
save TAmin -v7.3 TAmin;
```

```
save TBmax -v7.3 TBmax;

save TBmin -v7.3 TBmin;

save TCmax -v7.3 TCmax;

save TCmin -v7.3 TCmin;


prompt={'RA','RB','RC','XA','XB','XC'};

dlg_title='Motor Parameters';

num_lines=1;

def={'4.3020','4.5504','4.1748','4.0409','3.6608','3.4937'};

answer=inputdlg(prompt,dlg_title,num_lines,def);


RA=str2double(answer(1));

RB=str2double(answer(2));

RC=str2double(answer(3));

XA=str2double(answer(4));

XB=str2double(answer(5));

XC=str2double(answer(6));


Tstop=str2double(char(inputdlg('Stop Time(Maximum 2000 Seconds): ')));


    p1 =    1.191;

    p2 =    2.623;

ftime =ceil((p1*Tstop + p2 + 15)*1.5/60);

tstr=sprintf('Esimated Time for Simulation Completion: %4.0f minutes\n',ftime);

z=msgbox(tstr);
```

```
Tstop=timeseries(Tstop);
save Tstop -v7.3 Tstop;

RA=timeseries(RA);
RB=timeseries(RB);
RC=timeseries(RC);
XA=timeseries(XA);
XB=timeseries(XB);
XC=timeseries(XC);

save RA -v7.3 RA;
save RB -v7.3 RB;
save RC -v7.3 RC;
save XA -v7.3 XA;
save XB -v7.3 XB;
save XC -v7.3 XC;

delete('G:\TEMP_MATLAB\*.mat');
copyfile('G:\Project64Bit\*.mat','G:\TEMP_MATLAB\');

tt=tic;
fprintf('Start\n')
!G:\TEMP_MATLAB\ScarMotorV4RGV.exe
fprintf('Stop\n')
toc(tt)
```

```
        copyfile('G:\TEMP_MATLAB\*.mat','G:\Project64Bit\mcrCache7.17\MainGU0\MainGUI6
4bit');
        copyfile('G:\TEMP_MATLAB\*.mat','G:\Project64Bit\');

        PostSimPlotGUI32bitRGV
```

Appendix X

PostSimPlotGUI32bitRGV Code Developed for Executable File

```
function [] = PostSimPlotGUIRGV()
%POSTSIMPLOTGUI Summary of this function goes here
%   Detailed explanation goes here
%
% load Vsignals.mat
% load Isignals.mat
% load IUnBal.mat
% load UnbalControl.mat
% load ControlSignals.mat
% load VPhPhUnBal.mat

f=figure;
fa=uicontrol('style','pushbutton','string','Voltage Signals','position',[100,300,100,50]);
set(fa,'Callback','PSPguiVoltSig32bitRGV')
fb=uicontrol('style','pushbutton','string','Current Signals','position',[100,200,100,50]);
set(fb,'Callback','PSPguiCurrSig32bit')
fc=uicontrol('style','pushbutton','string','Unbalance
Averages','position',[300,100,100,50]);
set(fc,'Callback','PSPguiUnbalAvg32bit')
fd=uicontrol('style','pushbutton','string','Current Unbalance','position',[300,200,100,50]);
set(fd,'Callback','PSPguiCurrUnbal32bit')
fe=uicontrol('style','pushbutton','string','Voltage Unbalance','position',[300,300,100,50]);
set(fe,'Callback','PSPguiVoltUnbal32bit')
ff=uicontrol('style','pushbutton','string','Control Signals','position',[100,100,100,50]);
set(ff,'Callback','PSPguiConSig32bit')
end
```

Appendix Y

PSPguiVoltSig32bitRGV Code Developed for Executable File

```
function [] = PSPguiVoltSigRGV()
%PSPGUIVOLTSIG Summary of this function goes here
%   Detailed explanation goes here

v=figure;
vb=uicontrol('style','pushbutton','string','Full Sine Wave Ph-Ph
Volt','position',[100,200,100,50]);
set(vb,'Callback','PlotFullSineVoltage32bit;')
vc=uicontrol('style','pushbutton','string','Peak Wave Ph-Ph
Volt','position',[100,100,100,50]);
set(vc,'Callback','PlotPeakPhPhVoltage32bit;')

end
```

References

[1] CoSERV Electrical , "Analysis of Submersible Pumps (UTA).docx," Denton , 2011.

[2] Britannica, "Encyclopaedia Britannica," [Online]. Available: http://www.britannica.com/EBchecked/topic/31875/Francois-Arago. [Accessed 29 January 2013].

[3] Tesla Memorial Society of New York , "Tesla Memorial Society of New York," [Online]. Available: http://www.teslasociety.com/ac.htm. [Accessed 30 January 2013].

[4] Edison Tech Center, "Galileo Ferraris," [Online]. Available: http://edisontechcenter.org/GalileoFerraris.html. [Accessed 30 January 2013].

[5] P. A. R. Alger, "The history of induction motors in America," Proceedings of the IEEE, vol. 64, no. 9, pp. 1380-1383, 1976.

[6] Actel, "Motor Efficiency Depends Upon Power Factor," June 2009. [Online]. Available: http://www.actel.com/documents/Motor_PowerFactor_WP.pdf. [Accessed December 2012].

[7] D. Polka, "Chapter 3: AC and DC Motors - AC Motors: AC Induction Motor," [Online]. Available: http://beta.globalspec.com/reference/10791/179909/chapter-3-ac-and-dc-motors-ac-motors-ac-induction-motor. [Accessed September 2012].

[8] G. G. K. K. E. Holbert, in Electrical Energy Conversion and Transport: An interactive computer-based approach, Hoboken, John Wiley & Sons Inc., 2005.

[9] S. J. Chapman, in Electric Machinery Fundamentals Fifth Edition, New York, McGraw-Hill, 2012, pp. 307-403.

[10] D. Polka, "What is a Variable Frequency Drive?," Powerformers Engineers (P) Ltd., [Online]. Available: http://www.powerformers.com/product/ac-drives/variable-frequency-drive-systems-and-controls/what-is-a-variable-frequency-drive.html. [Accessed December 2012].

[11] m. jee, "Technology Latest Techs of Living...," blogspot, 31 January 2011. [Online]. Available: http://miansub.blogspot.com/2011/01/squirrel-cage-rotor-motor.html.

[12] S. Balu, "Slip Ring Induction Motors Basics," [Online]. Available: http://www.brighthubengineering.com/diy-electronics-devices/43725-slip-ring-induction-motors-basics/. [Accessed September 2012].

[13] Cowern Papers , "LOCKED ROTOR CODE LETTERS," Bus Design Co., 2013. [Online]. Available: http://www.motorsanddrives.com/cowern/motorterms6.html. [Accessed 28 January 2013].

[14] Pacific Gas and Electric Company , "Voltage unbalance and motors," October 2009. [Online]. Available: http://www.pge.com/includes/docs/pdfs/mybusiness/customerservice/energystatus/powerquality/voltage_unbalance_rev2.pdf. [Accessed September 2013].

[15] K. Bostick, "How Does a Submersiable Water Pump Work?," Demand Media Inc., 1999-2013. [Online]. Available: http://www.ehow.com/how-does_5005678_submersible-water-pump-work.html. [Accessed January 2013].

[16] InspectAPedia, "How Does a Submersible Well Pump System Work?,"

InspectAPedia.com, 2012. [Online]. Available: http://inspectapedia.com/water/Well_Pump_Operation_Submersible.htm. [Accessed January 2013].

[17] C. A. Woodworth, "Early Electrification of Buffalo: 60-Hz Replaces 25-Hz," IEEE, 2012. [Online]. Available: http://www.ieeeghn.org/wiki/index.php/Early_Electrification_of_Buffalo:_60-Hz_Replaces_25-Hz. [Accessed September 2012].

[18] H. Perlman, "Hydroelectric power: How it works," U.S. Department of the Interior, U.S. Geological Survey, 07 January 2013. [Online]. Available: http://ga.water.usgs.gov/edu/hyhowworks.html. [Accessed January 2013].

[19] Yepyep, "ANSI C84.1 ELECTRIC POWER SYSTEMS AND EQUIPMENT - VOLTAGE RANGES," Blogger, 2 April 2011. [Online]. Available: http://www.powerqualityworld.com/2011/04/ansi-c84-1-voltage-ratings-60-hertz.html. [Accessed January 2013].

[20] E. D. R. J. M.Whelan, "Engineering Hall of Fame: Joseph Henry," Edison Tech Center, 2012. [Online]. Available: http://www.edisontechcenter.org/JosephHenry.html. [Accessed January 2013].

[21] C. McManis, "H-Bridges: Theory and Practice," 23 December 2006. [Online]. Available: http://www.mcmanis.com/chuck/robotics/tutorial/h-bridge/index.html. [Accessed September 2012].

[22] E. J. Wiedenbrug, "Overheating Electric Motors," Aachen.

[23] AllAboutCircuits.com, "Contators," [Online]. Available: http://www.allaboutcircuits.com/vol_4/chpt_5/2.html. [Accessed January

2013].

[24] A. R. Patel, J. K. Chauhan and D. K. Patel, "SIMULATION FOR A 3-PHASE INDUCTION MOTOR UNDER UNBALANCED CONDITIONS," in National Conference on Recent Trends in Engineering & Technology, Gujarat, 2011.

[25] T. .. F. C. Y. K. W. a. S. L. .. H. K. L . SHI, "MODELLING AND SIMULATION OF THE THREE-PHASE INDUCTION MOTOR USING SIMULINK," International Journal of Engineering Education, vol. 35, pp. 169-172, 1999.

[26] V. Vivek, G. Uma, R. Kumudini Devi and C. Chellamuthu, "Performance of induction motor driven submersible pump using Matlab/Simulink," in Power System Technology Proceedings, PowerCon, 2002.

[27] HyperPhysics, "Resistance: Temperature Coefficient," [Online]. Available: http://hyperphysics.phy-astr.gsu.edu/hbase/electric/restmp.html. [Accessed January 2013].

[28] M. Holt, "Motor Calculations," Penton Media Inc., 2013. [Online]. Available: http://ecmweb.com/nec/motor-calculations. [Accessed September 2012].

[29] R. Nave, "HyperPhysic - Magnetic force," [Online]. Available: http://hyperphysics.phy-astr.gsu.edu/hbase/magnetic/magfor.html. [Accessed January 2013].

[30] "Electrical Fuse," Electrical4u.com - Online Electrical Power System Engineering Study Guide, 2012-2013. [Online]. Available: http://www.electrical4u.com/basic-electrical/electrical-hrc-fuse.php. [Accessed September 2012].

[31] D. Dunning, "What Is the Difference Between Resistive & Inductive Loads?,"

Demand Media Inc. , 1999-2013. [Online]. Available: http://www.ehow.com/info_12181159_difference-between-resistive-inductive-loads.html. [Accessed September 2013].

Biographical Information

James Andrew Reed, was born in Fort Worth, Texas in 1989. He received his B.S.E.E from the University of Texas Arlington, in December 2011. He is also in the University of Texas Arlington, B.S.E.E. Fast-Track Program and is expected to receive his M.S.E.E. in Mayof 2013. He works for Dr. David Wetz and is a member of the UTA MicroGrid research team. His research areas are in the areas of pulsed power, energy storage, renewable energy systems and power systems analysis. When not in classes he enjoys spending his time playing music, as well as competing in clay target shotgun sports.