# Online Dynamic Parameter Estimation of Synchronous Machines

A Thesis

Presented in Partial Fulfillment of the Requirements for the

Degree of Master of Science

with a

Major in Electrical Engineering

in the

College of Graduate Studies

University of Idaho

by

Michael R. West

Major Professor: Brian Johnson, Ph.D.

Committee Members: Herbert Hess, Ph.D.; Michael Santora, Ph.D.

Department Administrator: Mohsen Guizani, Ph.D.

August 2016

# Authorization to Submit Thesis

This thesis of Michael West, submitted for the degree of Master of Science with a major in Electrical Engineering and titled "Online Dynamic Parameter Estimation of Synchronous Machines," has been reviewed in final form. Permission, as indicated by the signatures and dates below, is now granted to submit final copies to the College of Graduate Studies for approval.

Major Professor: _____ Date:_____

Brian Johnson, Ph.D., P.E.

Committee Members: _____ Date:_____

Herbert Hess, Ph.D., P.E.

_____ Date: _____

Michael Santora, Ph.D.

Department Administrator: _____ Date: _____

Mohsen Guizani, Ph.D

# Abstract

Traditionally, synchronous machine parameters are determined through an offline characterization procedure. The IEEE 115 standard suggests a variety of mechanical and electrical tests to capture the parameters appropriate for representing the fundamental characteristics and behaviors of a given machine. These characteristics and behaviors can be used to develop and understand machine models that accurately reflect the machine's performance. To perform such tests the machine is required to be removed from service.

Characterizing a machine offline can result in economic losses due to down time and labor costs. Such losses may be mitigated by implementing online characterization procedures. Historically, different approaches have been taken to develop methods of calculating a machine's electrical characteristic, without removing the machine from service. Data from a machine's response to external inputs when combined with a numerical algorithm can be used to determine that machine's characteristics. This thesis explores such characterization methods and strives to compare results from the IEEE 115 standard for offline characterization with an iterative least squares approximation implemented on a 20 horsepower synchronous machine. This least squares estimation method shows encouraging results for steady-state parameters in comparison with steady-state parameters obtained through the IEEE 115 standard testing.

# Acknowledgements

To my professors, committee members, and advisors, Dr. Brian Johnson, Dr. Herb Hess, and Dr. Michael Santora, at the University of Idaho, Thank you for providing me with both extensive knowledge in the field of Electrical Engineering and for this amazing research opportunity. Special thanks to Dr. Normann Fischer and Schweitzer Engineering Laboratories for sponsoring this work as well.

# Dedication

This work is dedicated to my fiancé, Sibel Briggs, for her unwavering support and encouragement towards me. To my parents, Shawn and Leslie, for their steadfast love and positivity towards me. To my brothers, Isaac and Matt, for pushing me to my limits and being my biggest role models, and to my friends who have walked with me throughout my graduate studies. Without them, I would have never had the privilege of being where I am now.

# Table of Contents

# Figures

# Tables

# Acronyms

DLG – Double-line-to-ground fault

hp = horsepower

IEEE – Institute of Electrical and Electronics Engineers

LL – Line-to-line fault

LSE – Least Square Estimation

NERC – North American Electric Reliability Corporation

RMS – Root-Mean-Square

SLG – Single-line-to-ground fault

Std. – Standard

UI – University of Idaho

VFD – Variable Frequency Drive

3PH – Three-phase fault

# Nomenclature

## Voltages

$V_A$ – Phase A line-to-ground RMS voltage
$V_B$ – Phase B line-to-ground RMS voltage
$V_C$ – Phase C line-to-ground RMS voltage
$V_d$ – direct axis RMS voltage
$V_q$ – quadrature axis RMS voltage
$V_0$ – zero sequence RMS voltage
$V_{fd}$ – field voltage

## Currents

$I_A$ - Phase A RMS current
$I_B$ - Phase B RMS current
$I_C$ - Phase C RMS current
$I_d$ - direct axis RMS current
$I_q$ - quadrature axis RMS current
$I_0$ - zero sequence RMS current
$I_{fd}$ - field current

## Magnetic Flux

$\psi_d$ - direct axis magnetic flux
$\psi_q$ - quadrature axis magnetic flux
$\psi_0$ - zero sequence magnetic flux
$\psi_{kd}$ - direct axis mutual magnetic flux
$\psi_{kq1}$ - quadrature axis mutual magnetic flux damper winding 1
$\psi_{kq2}$ - quadrature axis mutual magnetic flux damper winding 2
$\psi_{fd}$ - field magnetic flux

**Flux Linkage**

$\lambda_d$ - direct axis magnetic flux linkage

$\lambda_q$ - quadrature axis magnetic flux linkage

$\lambda_0$ - zero sequence magnetic flux linkage

$\lambda_{kd}$ - direct axis mutual magnetic flux linkage damper winding linkage

$\lambda_{kq1}$ - quadrature axis mutual magnetic flux damper winding 1 linkage

$\lambda_{kq2}$ - quadrature axis mutual magnetic flux damper winding 2 linkage

$\lambda_{fd}$ - field magnetic flux linkage

**Inductances**

$L_{ls}$ – stator leakage inductance

$L_{md}$ - direct axis mutual inductance

$L_{mq}$ - quadrature axis mutual inductance

$L_d$ - direct axis self inductance

$L_q$ - quadrature axis self inductance

$L_{kq1}$ - quadrature axis damper winding 1 inductance

$L_{kq2}$ - quadrature axis damper winding 2 inductance

$L_{kd}$ - direct axis damper winding inductance

$L_{lfd}$ – field leakage inductance

**Resistances**

$r_s$ - stator resistance

$r_{kq1}$ – quadrature axis damper winding 1 resistance

$r_{kq2}$ - quadrature axis damper winding 2 resistance

$r_{kd}$ - direct axis damper winding resistance

$r_{fd}$ – field winding resistance

**Reactances**

$X_{ls} = \omega_{base} L_{ls}$
$X_{md} = \omega_{base} L_{md}$
$X_{mq} = \omega_{base} L_{mq}$
$X_{d} = \omega_{base} L_{d}$
$X_{q} = \omega_{base} L_{q}$
$X_{kq1} = \omega_{base} L_{kq1}$
$X_{kq2} = \omega_{base} L_{kq2}$
$X_{kd} = \omega_{base} L_{kd}$

For values and equations shown throughout this document, let superscript “r” denote rotor reference frame, and superscript prime (`) denote rotor quantity referred to the stator. For more information on this conversion, see Chapter 3.2. Also, let subscript “n” indicate nominal (base) value, and subscript “pu” indicate per-unit values. It is also important to note that, in this thesis, any errors less than 5% are considered to be acceptable.

# 1. Introduction

In the early 1900s, synchronous machines were first developed and also started to become documented. As these devices become more and more accepted, a mathematical model known as the "Park's model" was developed in 1929 [1]. This model defined one of the most practical methods of modeling a machine under ideal conditions, and gave way to much more research in this area, resulting in widely accepted models for steady-state and dynamic simulation [2-6].

## 1.1 Background and Motivation

Parameter estimation of synchronous machines is central to accurately understanding machines in the field, and in turn, understanding their dynamic response of power systems to disturbances. Generally, manufacturers provide customers with necessary machine parameters in order to accurately model the machine's behaviors. As these machines age, daily "wear and tear," and non-critical system faults cause such parameters to drift, requiring owners to remove their machine from the system and characterize and update their model. Following the IEEE 115-2009 standard, traditional methods of characterization require the machine to be taken offline, resulting in economic losses [7].

This research focuses on characterizing synchronous machines by using the dynamic response created from system faults, large changes in loading, or step changes to machine excitation. Modern technological advances have allowed for online parameter estimation to become a reasonable, viable option for power industry application. Online estimation practices are favored by system operation engineers for providing the ability to characterize synchronous machines without shutting down plant operations, and have been studied by researchers since the 1970's [8, 9].

### 1.2 Objectives

The objectives of this research are:

- Understand numerical methods for accurately determining machine model parameters based on a synchronous machine's steady-state and dynamic output data
- Create a mathematical model of a synchronous machine
- Set up a laboratory synchronous machine for testing
- Determine parameters of lab machine using IEEE std. 115-2009.
- Develop a systematic approach to apply an online machine characterization technique
- Validate parameter estimation with simulation, apply the selected numerical algorithm to the lab machine.

The contents of this thesis are broken down into 8 chapters. In Chapter 2, several historical methodologies of synchronous machine online dynamic parameterization are reviewed. This review provides the necessary conceptual background information, as well as introduces different approaches that have been used in the past. Next, Chapter 3 presents a description of the synchronous machine under investigation, as well as sets up the lab machine. Chapter 4 discusses synchronous machine modeling. In Chapter 5, results of several offline characterization tests using methods described in IEEE Std. 115-2009 are calculated to provide a baseline for comparison of online parameter estimates. In Chapter 6, an algorithm for online characterization is derived and implemented using MATLAB. Parameter estimation results are compared with a simulated machine and then tested against the lab machine. In Chapter 7, conclusions are drawn and complications are discussed. Lastly, future work needed to continue this investigation is discussed in Chapter 8.

Ultimately, this project aims to implement a system for both offline and online estimation, and compare them using a live synchronous generator. The process is shown in Figure 1.

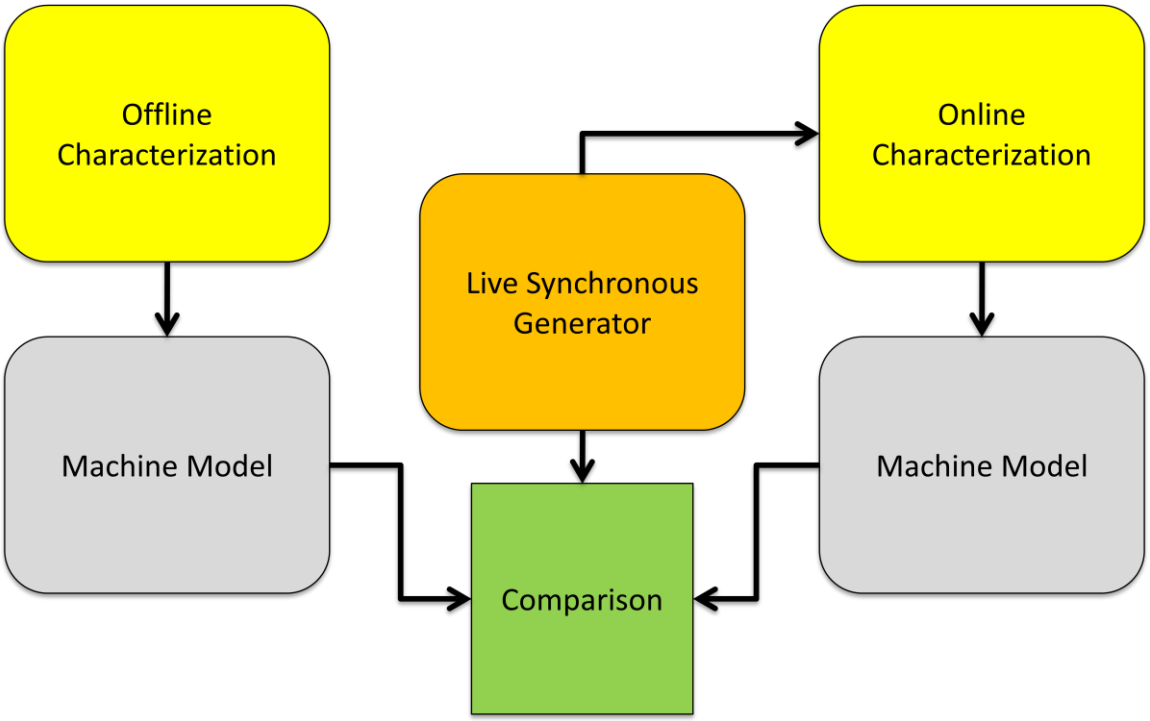


**Figure 1: High Level Representation of Combined Models**

It is important to note that this thesis focuses on several offline characterization tests, as well as an online characterization algorithm, but does not reach the point of comparing a complete validated offline model with a complete online model as shown in Figure 1, which represents one of the future goals of the project that supported this work. However, testing of the online characterization algorithm is attempted using real data captured from a synchronous machine.

# 2. Literature Review

Over time, general operation, external/non-catastrophic faults, and system disturbances tend to change a machine's characteristics. In order to satisfy the Western Electricity Coordinating Council (WECC) requirements to maintain safe and reliable operations, regularly updated machine characteristics are obtained to accurately maintain, monitor, and understand machine behaviors. Capturing these characteristics requires de-energizing the machine and conducting electrical tests that may take several days of labor. Both labor and machine downtime can result in significant economic losses. In addition, many generator owners don't have local expertise to perform offline characterization tests and instead hire consultants. Establishment of methodologies for calculating machine parameters without removing a machine from service has potential to mitigate these economic losses. These techniques are referred to as "Online Characterization Methods." The following sections capture literature on both offline and online characterization techniques and are broken down accordingly.

## 2.1 Offline Characterization

Offline characterization methodologies (also known as white-box methods) for establishing synchronous machine parameters have been well established for many years. The Institute of Electronic and Electrical Engineers (IEEE) released the first standard for such practices in the early 1940s and the methods have been built upon ever since. The most recently updated version of this standard was approved in 2009 and provides extensive documentation on reliable tests and calculations to establish both dynamic and steady-state parameter approximations [7].

The offline characterization techniques documented in this thesis follow the IEEE 115-2009 standard, and were used to develop a base line system model comparable to that of a machine owner's model, and to provide a basis for comparison to online techniques. Table 1 presents a few of the recommended tests for generator owners to perform in order to create or update a reliable, accurate model of their machine. Several of these tests are described and implemented on the University of

Idaho (UI) lab machine. For more detailed descriptions of tests associated with this standard, see [7]. Note that these tests in Table 1 are used to determine only some of the electrical characteristics. The IEEE 115-2009 standard also includes tests for mechanical properties that may be required, but are outside the scope of this thesis.

**Table 1: Offline characterization tests and parameters to be determined with each test**

| Test | Parameters To Be Found |
|---|---|
| Open Circuit Test | Open Circuit Saturation Curve |
| Short Circuit Test | Short Circuit Saturation Curve |
| Slip Test | $X_{qs}, X_{qu}, X_{ds}$ |
| Applied Negative Sequence Current | $X_2, R_2$ |
| Series Circuit | $X_0, R_0$ |
| Sudden Short Circuit | $X'_d, X''_d, \tau'_d, \tau'''_d$ |

### 2.*1.1 Open/Short Circuit Test*

The open circuit and short circuit tests are the two most common tests performed. In order to perform the open circuit test, the rotor is operating at rated speed, and the field current is incrementally increased while open-circuit terminal voltages are measured. To perform the short circuit test, the rotor is operating at rated speed, and the field current is incrementally increased while short-circuit armature current is measured.

Open and short circuit test results can be used to determine the saturation characteristics of the machine, along with the direct axis synchronous reactance. In addition, the IEEE 115 standard states that in normal machine designs, the direct axis synchronous reactance is mostly equal to the direct axis

synchronous impedance. In other words, resistances can generally be ignored. This test will be expanded on and demonstrated in Chapter 3.

### *2.1.2 Slip Test*

To perform the slip test, the rotor operates at a speed slightly different from synchronous speed with the field open-circuited. A three-phase, rated-frequency, positive-sequence power source at a voltage below the point on the open-circuit saturation curve where the curve deviates from the air-gap line is applied to the armature terminals. Measurements of armature current, armature voltage, and the voltage across the open-circuit field winding are recorded. The standard suggests using oscillography if possible. However, in the event that meters are used, a zero-center dc voltmeter should be used for field voltage measurements [7]. Figure 2 shows a SIMULINK model for simulation of the slip test. A large parallel resistance is used in this simulation to avoid numerical instability problems in the simulation, but it does not affect the test because it does not consume significant current.

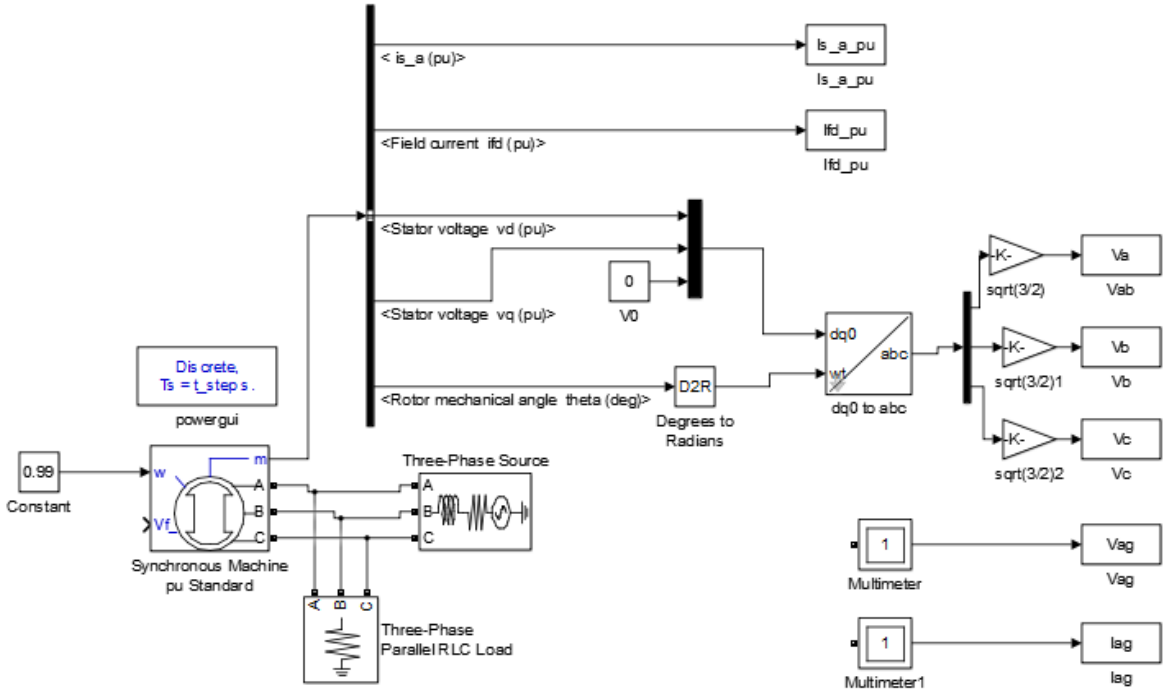


**Figure 2: Slip Test Simulation Model**

By following the steps discussed in the IEEE 115 Standard for the slip test, the waveforms in Figure 3 were plotted from a Simulink simulation and values for the quadrature axis synchronous reactance, unsaturated quadrature axis synchronous reactance, and a particular saturated direct axis synchronous reactance can be determined.

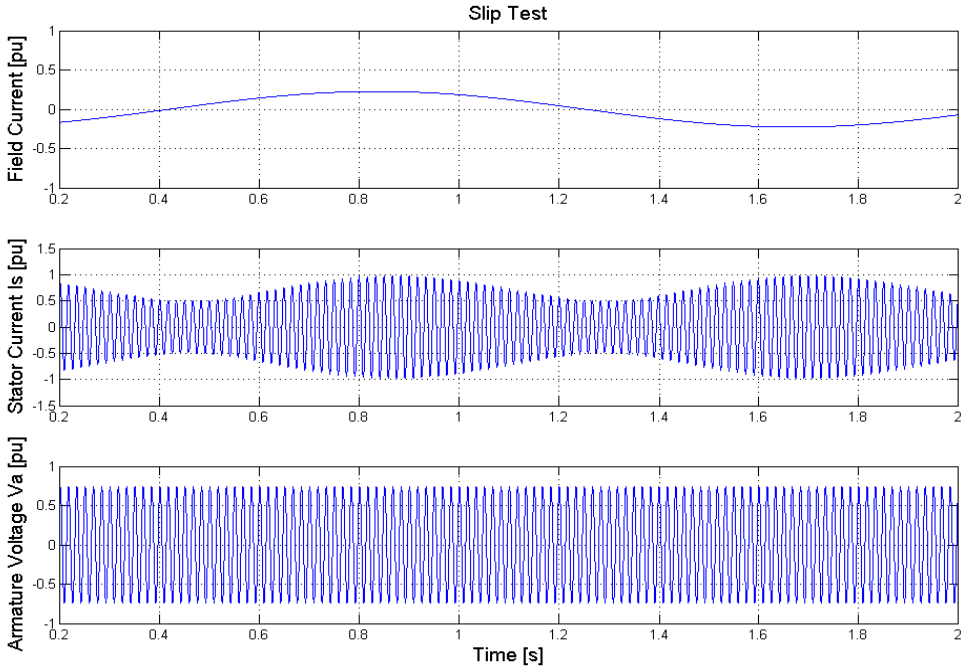


**Figure 3: Slip Test Simulation Results**

As defined by the IEEE 115-2009 standard, the following equations can be used to calculate the parameters listed in Table 1.

$$X_{qs} = \frac{E_{\min}}{I_{\max}} \tag{1}$$

$$X_{ds} = \frac{E_{\max}}{I_{\min}} \tag{2}$$

$$X_{qu} = X_{du}\frac{X_{qs}}{X_{ds}} \tag{3}$$

In eq. (1) and eq. (2), $E_{\min}$ indicates the minimum voltage peak, $E_{\max}$ is the maximum voltage peak, $I_{\min}$ is the minimum stator current peak, and $I_{\max}$ is the maximum stator current peak. The minimum voltage and maximum current occur when the field voltage is at a maximum, while the maximum voltage and minimum current occurs when the field voltage is at a minimum. The direct axis unsaturated reactance, $X_{du}$, was calculated earlier using the open and short circuit tests. Table 2 displays parameters calculated from the waveforms shown in Figure 3.

**Table 2: Calculated reactances from slip test simulation**

| | |
|---|---|
| $X_{qS}$ [pu] | 1.7493 |
| $X_{dS}$ [pu] | 1.9444 |
| $X_{qu}$ [pu] | 0.9003 |
| Error $X_{qu}$ | 7.8524% |

A discrete time-step of 1E-6 seconds was selected for the simulation. The error is calculated by using the known unsaturated reactance as entered in the machine model (set to 0.977pu in this case). This demonstrates the slip test method, although the results are dependent upon the open circuit and short circuit saturation curves. Note that the standard suggests using field voltage waveform instead of field current (as field circuit is supposed to be open). Due to limitations in the simulation platform, the field circuit cannot be truly open-circuited. However, both field current and voltage are assumed to be in phase, thus the peak and zero crossing time stamp values of the current waveform are acceptable, but may be the source of the significant errors from the simulation of the slip test.

### 2.1.2 Applied Negative Sequence Current

To perform the applied negative sequence current test, the machine is operated at rated speed with its field winding short circuited and negative sequence set of phase currents injected into the stator of the

machine. Determining negative-sequence reactance necessitates the negative sequence phase currents applied be equal to the rated current of the machine. IEEE Std. 115-2009 recommends performing two or more tests with current values above and below rated current to allow for interpolation [7].

### 2.1.3 Series Circuit

To perform the series circuit test, the windings of the three phases are connected in series and an external single-phase AC voltage is applied across the windings. Voltage and current measurements should be taken, if possible, for several values of current up to rated current or slightly higher. If the zero-sequence resistance is to be determined or if the resistance correction is to be applied, readings of power input should also be taken. Armature winding temperature will affect the zero-sequence resistance. In order to accurately measure resistance, the temperature should be measured for several higher current values [7].

### 2.1.4 Sudden Short Circuit Test

Sudden short circuit tests require more detail and equipment than the slip, negative sequence current, and series circuit tests. Care should be taken when performing such a test to ensure safety of both personnel and equipment. Oscillography of the terminal voltages and currents should be taken. To perform the sudden short circuit test, the machine is driven at rated speed, no load, and open circuited. A bolted 3 phase fault is applied to the terminals of the windings. To initiate the short circuit on two or more phases, a switch that shorts all phases approximately simultaneously is required to avoid introducing errors in the other phases of the machine [7].

An alternative to this method is to apply a sudden step change in field voltage while the machine is short circuited [10]. Doing so will provide the necessary dynamic response on the output of the machine in order to determine transient and subtransient characteristics.

### 2.2 Online Characterization

Online parameter estimation research first began as early as the 1970's as researchers began addressing shortcomings with offline measurement techniques [8]. Since then, research has been conducted and corresponding publications have significantly contributed to this concept. This section focuses on a non-comprehensive list of methods to provide insight on the history of this research and is intended to provide background information on historical techniques. Different types of numerical approaches have been made to accurately capture machine parameters from the machine's dynamic response. These methods are described as either "black-box" or "gray-box" models.

Black box models can be used to simply map the machine output data to a set of input data using either a single transfer function or cascading set of transfer functions. This method has the disadvantage that the internal parameters remain unknown as the data gets mapped, eliminating this as an option for most utility applications. Gray box models use numerical approximation methods to calculate the parameters to be entered into a given machine model.

For this thesis, the least squares estimation algorithm has been selected as the gray box model to compute the machine parameters. In 1977, an IEEE transactions article was published by C.C. Lee and T.T. Owen that was the first to use this approach [9].

#### 2.2.1 Least Squares Estimation

The least squares estimation method, first used by C.C. Lee, was performed on a small, salient-pole machine. The algorithm Lee used is visually represented and shown in Figure 4. The equations mentioned in the boxes in the figure are described below.

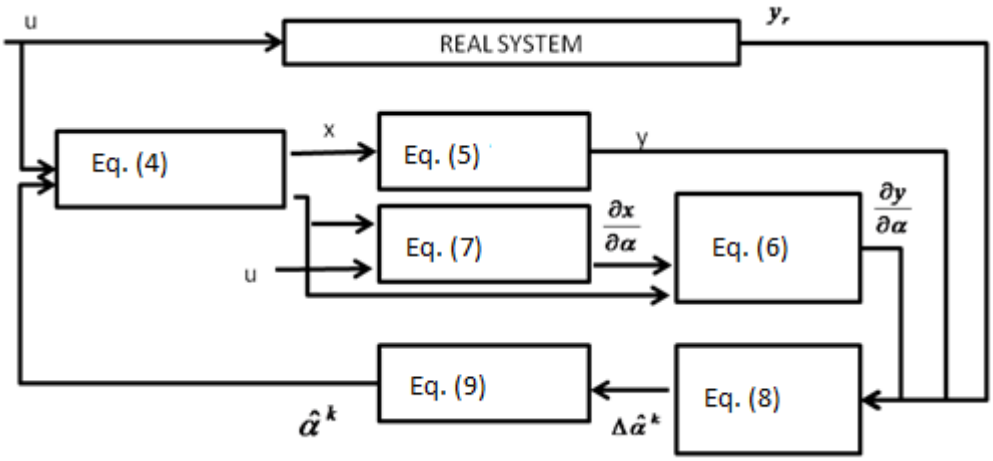


**Figure 4: Least Squares Estimator Algorithm [9]**

$$\dot{x} = A(\alpha)x + B(\alpha)u$$
$$x(t_0) = x_0 \tag{4}$$

$$y = C(\alpha)x \tag{5}$$

In the state space equation in Eq. (4), $\alpha$ is a 1 by N vector, where N is the dimension of all synchronous machine electrical parameters to be identified. These parameters can include subtransient, transient, and steady-state reactances, time constants, and resistances. Eq. (5) through Eq. (9) show the equations defined by Lee required to implement the least squares estimation algorithm.

$$\left[\frac{\partial y_{sim}(\alpha)}{\partial \alpha}\right]\Big|_{\alpha_0} = C(\alpha_0)\left[\frac{\partial x(\alpha)}{\partial \alpha}\right]\Big|_{\alpha_0} + \left[\frac{\partial C(\alpha)}{\partial \alpha}\right]\Big|_{\alpha_0} x(\alpha_0) \tag{6}$$

$$d\left(\frac{\partial x}{\partial \alpha}\right)_{\alpha_0} = C(\alpha_0)\left(\frac{\partial x}{\partial \alpha}\right)_{\alpha_0} + \left[\frac{\partial A(\alpha)}{\partial \alpha}\right]_{\alpha_0} x(\alpha_0) + \left[\frac{dB(\alpha)}{\partial \alpha}\right]_{\alpha_0} u$$
$$d\left(\frac{\partial x}{\partial \alpha}\right)_{\alpha_0}, t_0 = 0 \tag{7}$$

$$\left[\int_{t_0}^{t_f} \left(\frac{\partial y}{\partial \alpha}\right)_{a_0}{}^T R\left(\frac{\partial y}{\partial \alpha}\right)_{a_0} dt\right](\hat{\alpha} - \alpha_0) = \left[\int_{t_0}^{t_f} \left(\frac{\partial y}{\partial \alpha}\right)_{a_0}{}^T R(y_r - y(\alpha_0)dt\right] \tag{8}$$

$$\hat{\alpha}^k = \hat{\alpha}^{k-1} + G(k)\Delta\hat{\alpha}^k \tag{9}$$

By applying this method to a fifth order synchronous machine model structure, Lee was able to successfully converge to acceptable ratios of estimated to true values in less than 15 iterations, thus proving its capability [9]. This approach and its equations are selected for study in this thesis and will be expanded upon in Chapter 6.

### *2.2.2 Kalman Filtering*

In a 1981 paper, Kalman filtering was used to create a pseudo black box model estimate of a synchronous machine connected to a local power system. Figure 5 shows the flow diagram as described in [10].

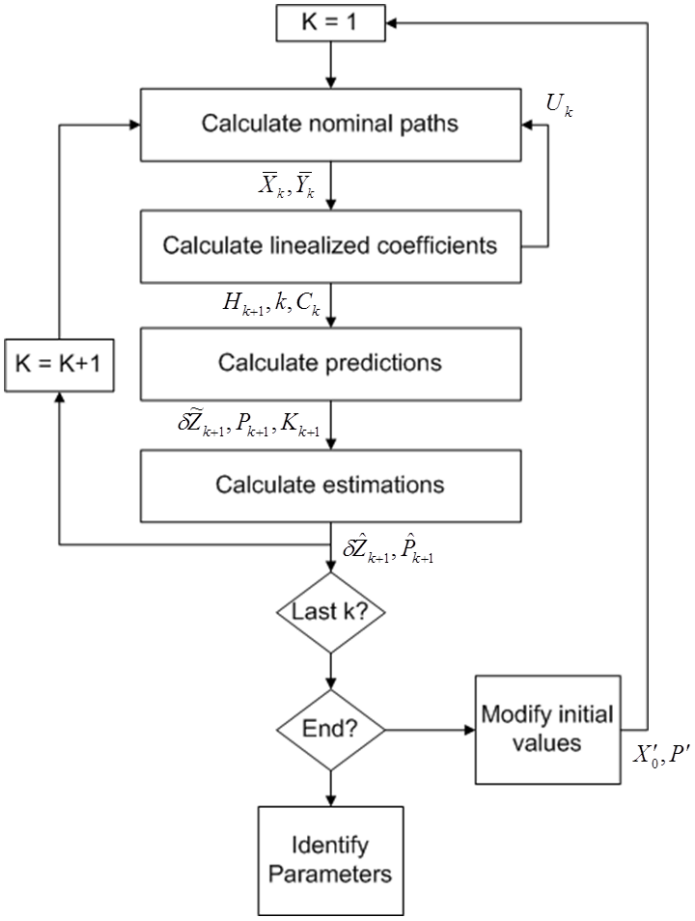


**Figure 5: Kalman filter general flow diagram [10]**

In the general flow diagram shown in Figure 5, $U_k$ represents the input vector, $\overline{X}_k$ and $\overline{Y}_k$ represent the state vector and an observation vector respectively, $H_{k+1}$ represent a state transition matrix, $k$, represents the iteration number, and $C_k$ represents an observation matrix. These are used to solve for the approximated state vector and the parameter vector $\delta\hat{Z}_{k+1}, \hat{P}_{k+1}$ respectively.

The publication used the Park's two axis model and only considered transient parameters due to limited subtransient behaviors in the machine used. Disturbances were introduced by using line switching and changes in controller settings. This method showed fairly reliable results in measured currents and output power comparisons [10].

### 2.2.3 Subset Selection

In 1999, Michael Burth discusses some solutions to ill-conditioning in machine parameter estimation calculations using the Least squares estimation for nonlinear parameters [11]. In his publication, a subset selection scheme is used to partition ill-conditioned, measured model parameters into well-conditioned parameters. Using the subset selection and reduced-order estimation algorithm, iterative estimations fixed parameters associated with ill-conditioned directions of a Hessian matrix. The Hessian Matrix $H(\theta)$ is described as a series of second partial derivatives of the least squares minimization error criterion $V(\theta)$.

$$H(\theta) = J'(\theta)J(\theta) + \sum_{n=1}^{N} r_n(\theta)\frac{\partial r_n(\theta)}{(\partial\theta)(\partial\theta')} \approx J'(\theta)J(\theta) \tag{10}$$

Where

$$r(\theta) = \hat{y}(\theta) - y \tag{11}$$

$$\hat{\theta} = \arg\min_{\theta} V(\theta) \tag{12}$$

$$V(\theta) = \frac{1}{2}\|r(\theta)\|^2 = \frac{1}{2}\sum_{n=1}^{N} r_n^2(\theta) \tag{13}$$

$$J(\theta) = \frac{\partial r(\theta)}{\partial \theta} \tag{14}$$

Residual vector $r(\theta)$ represents the difference between the N-dimension vector model predictions for the measurements ( $\hat{y}(\theta)$ ) and the N-dimension vector of measurements ( $y$ ) where ( $\theta$ ) represents the n-dimension vector of model parameters. The approximation of $H(\theta)$ is used for small residuals and is the Hessian matrix used [11].

### 2.2.4 Large Disturbance Method

Also in 1999, an online parameterization technique was published that was based on using large disturbance testing data and the extended LSE method. A dynamic response was acquired by applying a step change in the machine's excitation reference voltage. Using the extended LSE method, nonlinear parameters are approximated for various operating conditions. Sub-models are then created across the varying operating conditions and compared with large transient responses [12].

### 2.2.5 Volterra Series

In 2005, a method known as the Volterra series method was applied for synchronous generator model identification. A Volterra series generalizes Taylor series expansion and convolution integrals for a given nonlinear system. The algorithm used is described as follows:

a) Select input signal (covers all dynamics)
b) Select sampling time and experiment time
c) Apply input signal, sample input-output data
d) Select kernels (n) and terms N in the orthogonal series (lower order preferred for modeling)

e) Select orthogonal functions to be used in the following equation

$$h_n(\tau_1,...\tau_n) = \underset{N\to\infty}{Lim} \sum_{n=0}^{N}\sum_{m=0}^{N}...\sum_{k=0}^{N} a_{nm..k} \times \phi_n(\tau_1)\phi_m(\tau_2)...\phi_k(\tau_n) \quad (15)$$

f) Apply input to subsystems, solve for $u_s(t)$

$$u_s(t) = \int \phi_s(\tau_k) \times (t-\tau_k) d\tau_k \quad (16)$$

g) Form the following regression equation

$$y_t = u_t^{\ k}\theta + e_t \quad (17)$$

h) Minimize the performance index J by solving for the unknown coefficients $\theta^T$ using the least squares method.

$$J = \sum_t e_t^{\ 2} \quad (18)$$

i) Calculate the output using the coefficients identified, compare with measured values $y_t$

$$\hat{y}_t = u_t^T \hat{\theta} \quad (19)$$

Results shown in this publication used data collected from a seventh order nonlinear synchronous generator model and with small perturbation introduced in the field voltage. Walsh functions were selected for the orthogonal functions. This method produced acceptable errors for a range of machine operation scenarios [13].

### 2.2.6 Linear $H_\infty$ Infinity Method

A linear $H_\infty$ based method for machine parameter estimation was published in 2007 [14]. This approach was a black-box algorithm used with a pseudo random binary sequence signal applied to the field voltage, similar to the large disturbance method described above. By measuring power output,

terminal voltage, and field voltage, the identification method was used on a seventh order machine model. The algorithm is described as follows:

a) Select an appropriate input signal that will provide a wide range of system dynamics (publication used field voltage)

b) Select sampling time and final time

c) Apply input signal and sample both input and output data

d) Select model order (n) to be used in the following:

$$\begin{aligned} z_k &= \sum_{i=1}^{n} a_i z_{k-1} + \sum_{i=1}^{m} b_i u_{k-1} + c_0 \omega_k \\ y_k &= z_k + v_k \end{aligned} \tag{20}$$

e) Select appropriate value for the disturbance rejection factor $\gamma$ through trial and error (smallest possible value such that the following conditions are met in order to guarantee convergence):

$$\begin{aligned} & I - \tilde{g}_k \left( Q_{k+1} + \gamma^{-2} \left[ \tilde{C}_{k+1} \left( \tilde{\theta}_{k+1} \right) + \tilde{\delta}_{k+1} \right] \right) \tilde{g}_k > 0 \\ & Q > 0 \end{aligned} \tag{21}$$

f) Using Least Squares Approximation, find initial seed for model parameters $(-a_i, b_i)$ of vector $\theta_0'$ in:

$$\theta_k^t = [a_1 \ldots - a_n, b_1 \ldots - b_n, v_1 \ldots - v_n,] \tag{22}$$

g) Establish state space matrices A, B, $C_0(\theta_0)$, and D

h) Update

Simulation results demonstrate a successful model for the machine under test. The publication suggested using this model for system analysis and controller design and did not include saturation characteristics in the model, but introduced a method on doing so in the appendix of the paper [14].

### 2.2.7 Wiener-Neural Model

Also in 2007, another black-box method known as the Wiener-Neural Model was presented at the International Conference on Intelligent and Advanced Systems [15]. In that publication, a simulated synchronous machine model was connected to an infinite bus through a transmission line. Field voltage was used as the input, active output power; terminal voltage and current were selected as output signals. The step function field voltage with noise was used to provide a rich, dynamic range of output characteristics. The following steps list the algorithm used for this method

1. Select wide spectrum input signal to be applied. Magnitude should be large enough to cover nonlinearities, but small enough to be avoid machine damage
2. Select sampling time and final time (total experiment time)
3. Apply input signal to system and sample input-output data using a DAQ
4. Select the number of neural layers and Wiener model degree for best modeling
5. Use Weiner-Neural Model to calculate approximated outputs

This method showed acceptable errors for modeling a synchronous machine from online information [15].

### 2.2.8 Normalized Least Squares and Newton Raphson Iteration

In 2008, a gray-box method was proposed for a 3$^{rd}$ order model and derived [16]. This model ignored stator dynamic effects and damper windings. In this method, the normalized least squares identification method was used to define a multivariable linear transfer function in a Heffron-Philips composition. Physical parameters of the machine were then determined using Newton-Raphson iterations. This experiment was set up and compared with a motor-generator set manufacturer data

and show errors of no more than 5%. Field voltage was the only input considered in this method, outputs included the electrical power and terminal voltage [16].

### 2.2.9 Conclusion

In conclusion, there are many different numerical algorithms that have been developed for machine parameter estimation. In this thesis, the method of least squares estimation is used because many of the more accurate and complex approaches build on this algorithm. These approaches use the least squares estimation, in conjunction with other algorithms or filters, to further refine and make parameter estimation more reliable. The least squares estimation algorithm will be tested on both a 3$^{rd}$ order and a 7$^{th}$ order machine model. In other words, there are either 3 or 7 coupled state-space equations used to simulate the behaviors of a synchronous generator. It is important to define these models as 3 or 7 cross coupled differential equations, not as having a 3$^{rd}$ or 7$^{th}$ order derivative. These models are built and tested under ideal conditions and do not consider saturation, noise, or a more "realistic" external system connected to them. In addition, the models are assumed to be operating at constant, nominal speed.

The most important values for the least squares approximation method are the stator voltage outputs, the field voltage input, and the rotor speed. Using this data, the least squares approximation method can be used to iteratively calculate the machine parameters. Once these parameters are found, the algorithm will update the machine model and compare the results the measured outputs from a separate machine simulation or a real machine. From here, a constructive understanding of advantages or limitations to this method can be determined.

# 3. Laboratory Synchronous Machine Description

Maintenance on machines may require that physical components of the device be replaced entirely, causing significantly changes in the electrical characteristics. This necessitates characterization of a given machine using traditional offline techniques. Following the IEEE Std. 115-2009, a base case of parameters can be created to compare with the online techniques. A lab scale synchronous machine at the University of Idaho provides access to studying both offline and online characterizations. Nameplate ratings of the machine under investigation are shown in Table 3.

**Table 3: Synchronous Machine Nameplate Ratings**

| Synchronous Machine (100% Loading, 24 Hrs, 40 °C Rise) | |
|---|---|
| 20 H.P. | 1200 RPM |
| 220 Volts | 80% power factor |
| 46 Ampere | 6.00 Excitation Amps |
| 3 Phase | 125 Excitation Volts |
| 60 Hz | 3 Pole Pairs |

The synchronous machine provided by the University of Idaho is mechanically coupled to a 20hp induction motor and is controlled using an ABB ACS550 variable frequency drive (VFD). The coupled machines are shown in Figure 6.

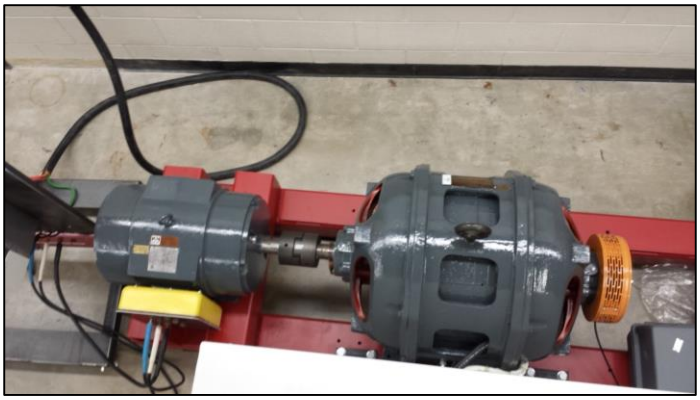

**Figure 6: Motor/Generator Set (Left: Induction Machine, Right: Synchronous Machine)**

The outputs of the synchronous machine are fed to a tap wall, which allows access to the machine's stator terminals, field terminals, and neutral point. This configuration allows for easy connection in both "Wye" (Y) and "Delta" ( $\Delta$ ) configurations, and also allows access to internal winding taps for creating internal faults. However, the work documented in this thesis focuses on response to external behavior only. The tap wall can be seen in Figure 7.

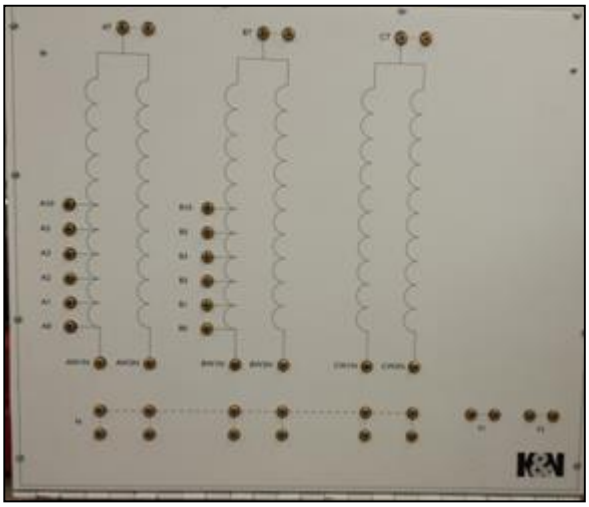

**Figure 7: Synchronous Machine Tap Wall**

The induction machine is driven by a programmable ABB ASC550 variable frequency drive. This drive comes standard with a digital front panel and analog/digital IO ports for external control. A built-in setup procedure is followed to input induction machine nameplate machine information, protection settings, acceleration/deceleration time constants, speed set-points. The drive uses the

induction machine nameplate ratings as input data for an internal machine model that is used to determine and display information such as the rotor frequency, speed, and other important parameters. Figure 8 displays an image of the VFD. For this work, the VFD is used to control the speed of the generator by using the induction machine as the prime mover.

**Figure 8: ABB ASC550 Variable Frequency Drive**

The VFD is also equipped with an internal fieldbus module. Using Modbus communications, this fieldbus may be connected to external devices which can feed data back to the VFD for control. An external fieldbus may be purchased separately if different communications protocols are required.

The synchronous machine also has an encoder attached to the rotor shaft. This encoder can output information about the rotor, including rotor frequency. With the VFD set to "Vector Speed Mode", and using the encoder's data as a feedback loop, the drive can automatically modulate the frequency to maintain constant, set-point speed with approximate accuracy of 0.1%, as described by ABB technical support. The encoder connected to the machine under test is shown in Figure 9. The encoder is manufactured by BEI Sensors, and communicates using a Synchronous Serial Interface (SSI).

**Figure 9: BEI Sensors Synchronous Machine Encoder Housing**

According to ABB, this particular single turn encoder is not supported by the ASC550 VFD, as the embedded and external fieldbus options do not support the SSI communications protocol. However, the VFD does have an internal machine model, which can closely model the rotor speed, based on the stator frequency and slip of the prime mover (in this case, the induction machine). Since the encoder is not directly supported by the drive, and the encoder is not yet fully operational to provide necessary data, it is assumed that the rotor speed operates at constant 377 rad/s (60 Hz) for the testing in Chapter 6. In reality, this is not necessarily true and will impact accuracy of test results that rely on precise rotor speed measurement. This was first noticed when testing the VFD.

By setting the VFD to operate the prime mover at a constant 60Hz, the generator output frequency was observed to be slightly less than synchronous speed. This is due to the inherent slip of the induction machine. However, the ABB ASC550 VFD has a slip compensation ratio that can be increased or decreased in order to account for this effect. The terminal voltages of the synchronous machine were connected to a SEL 411L relay and the output frequency was measured. By increasing the slip compensation ratio in the VFD, the frequency of the ABB drive was matched to the output frequency of the machine. This brute force process was used to make sure that the synchronous machine operates at precisely 60Hz during testing. Synchronous speed can also be verified using a strobe light, however, having measured rotor speed data greatly increases the accuracy of a synchronous machine model.

The generator stator output can also be fed into the University's model power system, which was originally designed to test protection systems such as relays, and which is able to simulate faults such as 3-phase and single-line-to-ground faults. The model power system includes current transformers, voltage transformers, SEL relays, and breakers. This system is designed with the capability to vary line and source parameters, and with the ability to create three phase faults, line-to-line faults, double-line-to-ground faults, and single-line-to-ground faults. Oscillography of the synchronous machine's behaviors can be captured and then used to study the synchronous machine's dynamic behaviors under external fault conditions. A one-line diagram of this system and the equipment available for use is show in Figure 10. The breaker status' shown in Figure 10 were used for testing in this thesis.

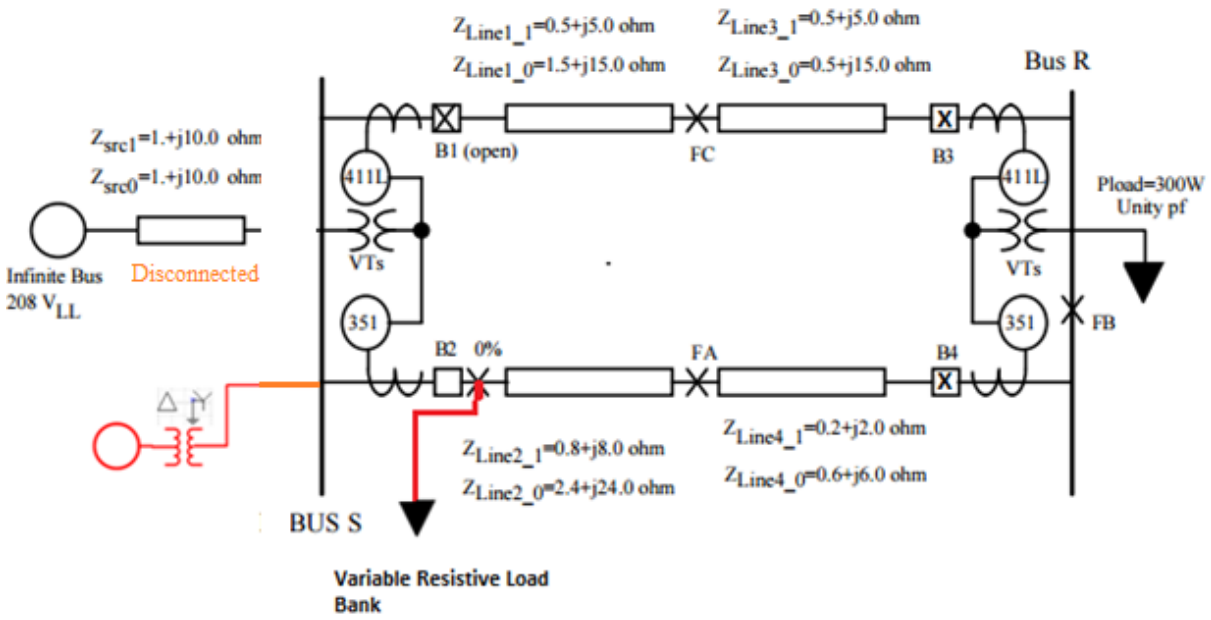


**Figure 10: University of Idaho Model Power System**

The model power system is comprised of a quasi-infinite bus from the building's power supply. Variable source impedances, and four transmission lines with variable impedance can be configured in series or in parallel. Loads may be connected to different sections of the system, and fault can be initiated at multiple locations, indicated for example by the points "0%," "FA," "FB," and "FC" in the figure. This system is rated for 208 V LL and operates at 60 Hz. Four breakers are also available and can be controlled via SEL-351S, SEL 411L or SEL-421 relays.

To mimic the transformer connection of a real generator, the synchronous machine is connected to a 208:208 delta-wye transformer. In this configuration, the terminals of the machine are connected to the delta connected transformer primary, while the terminals of the transformer secondary are connected to the model power system in a wye configuration. By connecting the synchronous machine and transformer to the model power system, and disconnecting the building power supply, fault tests can be conducted almost directly on the terminals of the machine.

The configuration shown above is only one of many different connections that can be made. For example, the synchronous machine may also be connected to “Bus R” and synchronized with the infinite bus through one or both of the available transmission lines. Doing so provides the opportunity to more realistically imitate the dynamic responses of a generator connected to a power system. A visual block diagram representation of the laboratory test setup is shown in Figure 11.

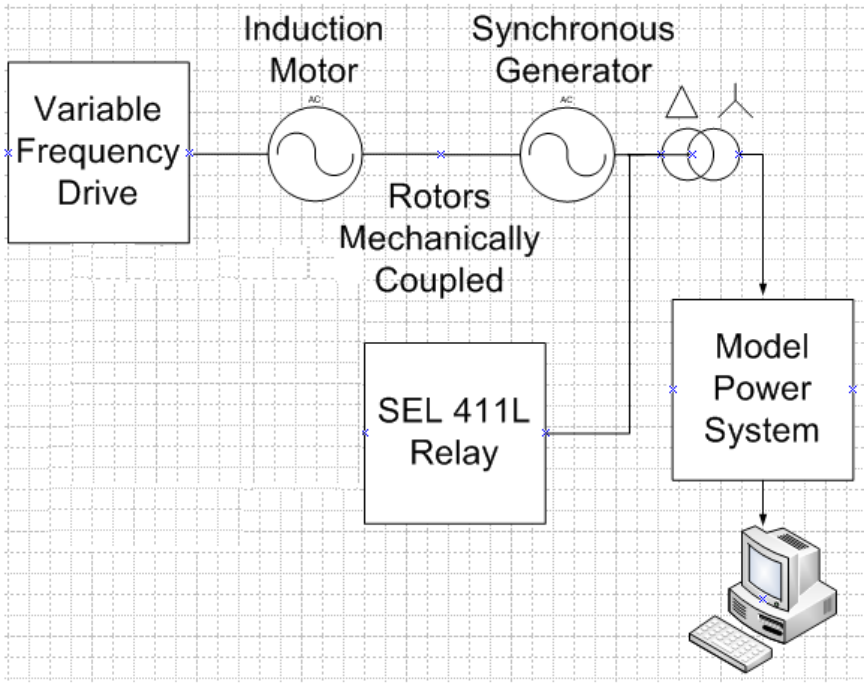


**Figure 11: University of Idaho Synchronous Machine Laboratory Setup**

# 4. State-Space Synchronous Machine Model

To model machine characteristics for parameter estimation, both 3rd and 7th order, state space model representations are used. These models are both implemented in the rotor referenced rotating dq reference frame. The 3rd order model is a simplified version of the 7th order model and does not consider the direct and quadrature axis damper windings, nor does it consider the zero sequence circuit equation. The 7th order does include the direct and quadrature axis damper windings, and the zero sequence equation. The models are built and validated using MATLAB [19], a popular tool used for computational analysis. In this chapter, the 3rd and 7th order synchronous machine models are defined.

To begin, a frame of reference must first be established in order to provide context for the state-space equations. The rotor reference frame selected to model the synchronous machine in this thesis, which is shown in Figure 12. This is based on the machine description in [18]. The particular model shown in Figure 12 has two quadrature axis damper windings, and one direct axis damper winding. The direct axis is lagging the quadrature axis by 90 degrees, and the quadrature axis is referenced to the A-phase voltage.

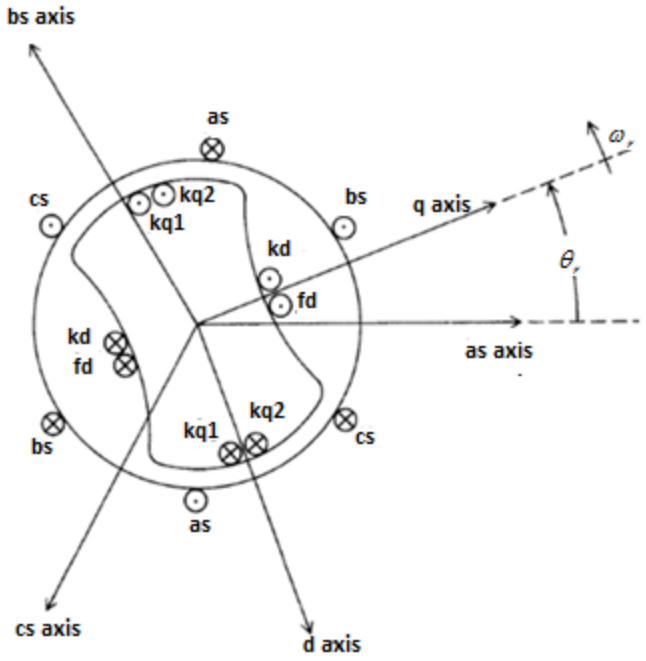


**Figure 12: Synchronous Machine DQ Reference Frame [18]**

Using this reference frame makes simulations much simpler to develop, which can be done using the Park's transformation matrix to a synchronous rotational reference frame. In turn the machine voltage, current and flux data can be viewed from the rotor. The mathematical representation of this model will be described later in this chapter.

### 4.1 7th Order Model Derivation

The circuits shown in Figures 13-15 can be used to model the synchronous machine as a generator through the Park's transformation. This generator reference is defined by the direction of the currents flowing out of the d and q axis of the machine, thus implying generation. The complete circuit diagrams in Figure 13, Figure 14, and Figure 15 are used in the 7th order model. The 3rd order model is viewed by ignoring the damper winding circuits (kq1 and kq2 in Figure 13, and kd in Figure 14), as well as the zero sequence circuit in Figure 15.

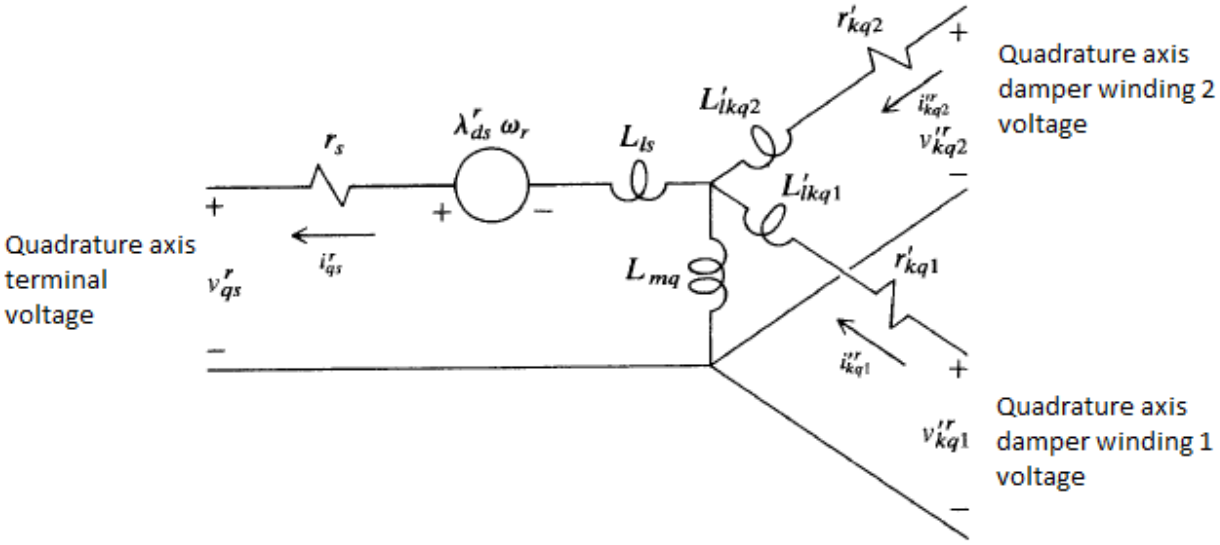


**Figure 13: Synchronous Machine Quadrature Axis Generator Model Equivalent Circuit [18]**

In Figure 13, the q-axis damper windings are shown with voltage $V`^{r}_{kq1}$ and $V`^{r}_{kq2}$. These damper bars create opposing electromagnetic forces on the machine when non-fundamental frequency or unbalanced currents are applied , resulting in damping of smaller transients or dynamic load changes that may cause the rotor of the machine to move out of synchronous speed. Note that in reality, the damper bars in a synchronous machine are shorted together, thus the terminal voltages can be assumed to be 0 at all times. In addition, not all machines have 2 sets of q-axis damper bars.

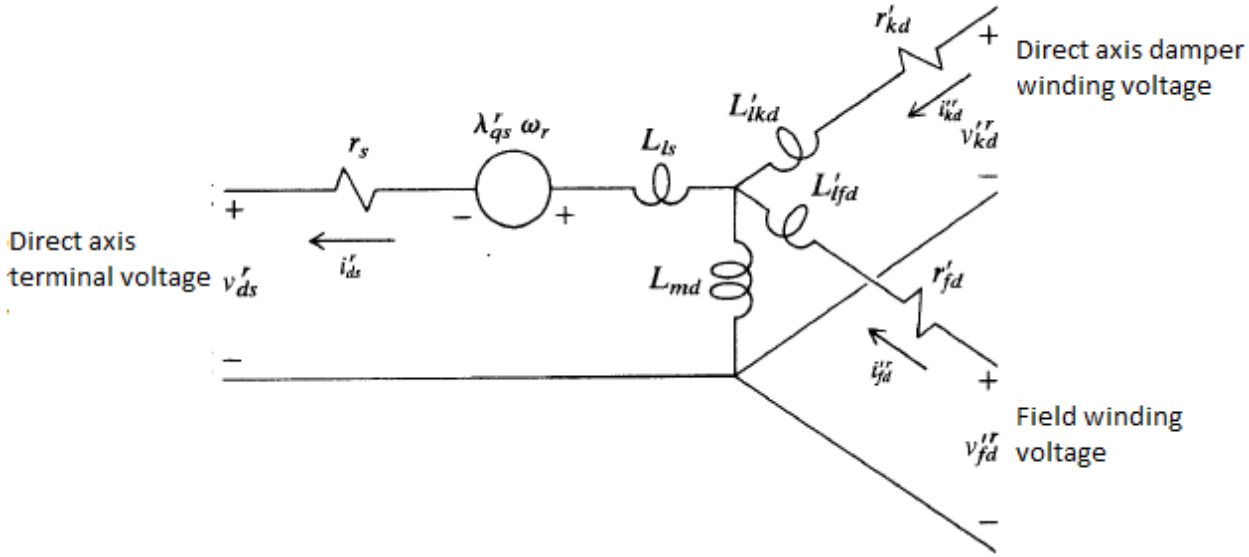


**Figure 14: Synchronous Machine Direct Axis Generator Model Equivalent Circuit [18]**

The same can be said about the d-axis modeled damper bars in Figure 14. Because the bars on the physical machine are shorted together, thus the voltage is assumed to be 0 at all times. However, the d-axis circuit also shows how the excitation system couples into the d-axis voltage.

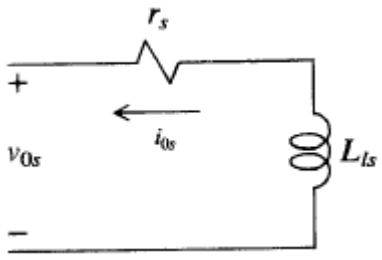

**Figure 15: Synchronous Machine Zero-Sequence Generator Model Equivalent Circuit [18]**

A cut-out cross-sectional representation of a synchronous machine is shown in Figure 16. The diagram shows the physical locations of the parameters discussed in this thesis. The "gear" shaped rotor defines the machine as a "salient pole" machine. A set of damper bars can be seen on the stator of the machine and the field windings are located on the rotor.

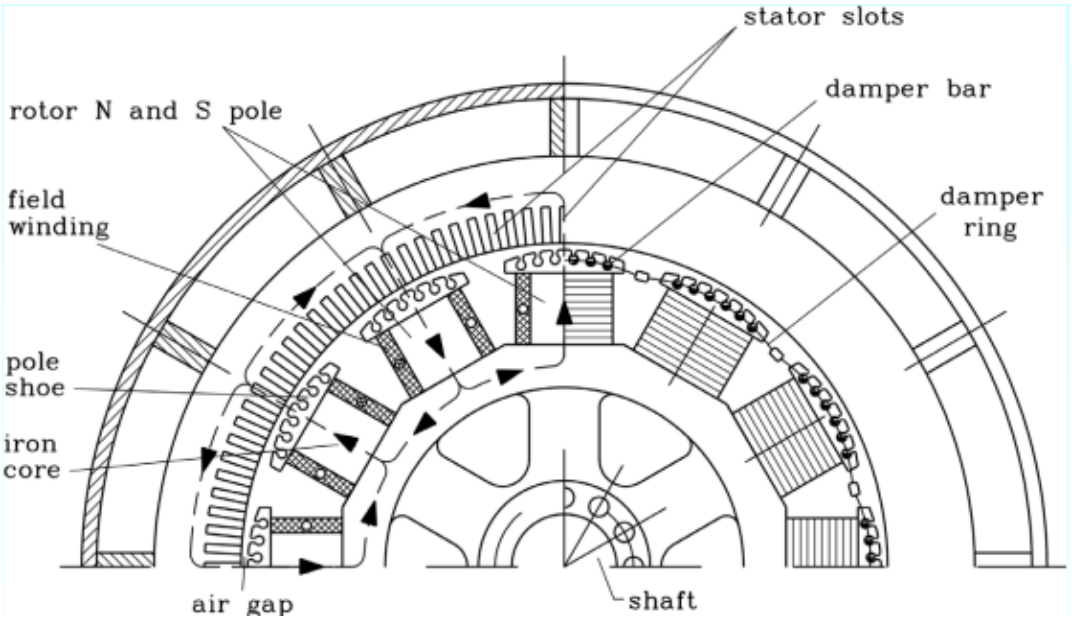


**Figure 16: Cut-Out Depiction of a Synchronous Machine [20]**

Representing machine in the Park's rotor reference frame creates the model described in the circuits in Figures 13-15. In order to convert ABC voltages and currents from the stationary ABC reference frame to the rotating dq0 reference frame, Eq. (23) – Eq. (25) are used.

$$K = \frac{2}{3} * \begin{bmatrix} \cos(\omega_r * t) & \cos(\omega_r * t - \frac{2*\pi}{3}) & \cos(\omega_r * t + \frac{2*\pi}{3}) \\ -\sin(\omega_r * t) & -\sin(\omega_r * t - \frac{2*\pi}{3}) & -\sin(\omega_r * t - \frac{2*\pi}{3}) \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \tag{23}$$

$$V_{dq0} = K * V_{ABC} \tag{24}$$

$$I_{dq0} = K * I_{ABC} \tag{25}$$

As a reminder, quantities shown in Figure 13-Figure 15 circuit drawings which have a superscript ' denote rotor quantities referred to the stator. Eq. (26) - Eq. (28) show how these values can be referred to the stator based on the turns ratio or nominal field and stator currents.

$$\frac{N_S}{N_f} = \frac{2}{3}\left(\frac{I_{fn}}{I_{Sn}}\right) \tag{26}$$

where $I_{fn}$ and $I_{Sn}$ represent the nominal field and stator currents respectively, and $\mathrm{N_f}$ and $\mathrm{N_S}$ represent the number of field and stator winding turns respectively. Using the turns ratio above, rotor quantities can be referred to the stator.

$$V'_{fd} = \frac{N_S}{N_f} V_{fd} \tag{27}$$

$$I'_{fd} = \frac{2}{3}\frac{N_f}{N_S} I_f \tag{28}$$

$$R'_{fd} = \frac{3}{2}\left(\frac{N_f}{N_S}\right)^2 R_{fd} \tag{29}$$

$$L'_{lfd} = \frac{3}{2}\left(\frac{N_f}{N_S}\right)^2 L_{lfd} \tag{30}$$

As shown, some values are multiplied by 2/3, others are multiplied by 1.5. These inverse ratios are because only two axes are while in the Parks reference frame, (d and q) but there are three stator windings. The voltage equations in equations (31)-(37) and flux linkage equations (38)-(44) are derived from the circuits shown in Figure 13-Figure 15 [18].

$$V_{qs} = -r_s i_{qs} + \omega_r \lambda_{ds} + \rho \lambda_{qs} \tag{31}$$

$$V_{ds} = -r_s i_{ds} - \omega_r \lambda_{qs} + \rho \lambda_{ds} \tag{32}$$

$$V'_{fd} = r_s i'_{fd} + \rho \lambda'_{fd} \tag{33}$$

$$V_{0s} = -r_s i_{0s} + \rho\lambda_{0s} \tag{34}$$

$$V'_{kd} = r'_{kd} i'_{kd} + \rho\lambda'_{kd} \tag{35}$$

$$V'_{kq1} = r_{kq1} i'_{kq1} + \rho\lambda'_{kq1} \tag{36}$$

$$V'_{kq2} = r_{kq2} i'_{kq2} + \rho\lambda'_{kq2} \tag{37}$$

Note that $\rho$ indicates the time derivative, $\frac{d}{dt}$. Expressions for flux linkages are defined as follows [18].

$$\lambda_{qs} = -L_{ls} i_{qs} + L_{mq}(-i_{qs} + i'_{kq1} + i'_{kq2}) \tag{38}$$

$$\lambda_{ds} = -L_{ls} i_{ds} + L_{md}(-i_{ds} + i'_{fd} + i'_{kd}) \tag{39}$$

$$\lambda'_{fd} = L'_{lfd} i'_{fd} + L_{md}(-i_{ds} + i'_{fd} + i'_{kd}) \tag{40}$$

$$\lambda_{0s} = -L_{ls} i_{0s} \tag{41}$$

$$\lambda'_{kd} = L'_{lkd} i'_{kd} + L_{md}(-i_{ds} + i'_{fd} + i'_{kd}) \tag{42}$$

$$\lambda'_{kq1} = L'_{lkq1} i'_{kq1} + L_{mq}(-i_{qs} + i'_{kq1} + i'_{kq2}) \tag{43}$$

$$\lambda'_{kq2} = L'_{lkq2} i'_{kq2} + L_{mq}(-i_{qs} + i'_{kq1} + i'_{kq2}) \tag{44}$$

Expressing equations (31)-(44) using reactances instead of inductances results in updated voltage equations (45)-(51) and (52)-(58)-

$$V_{qs} = -r_s i_{qs} + \omega_r \psi_{ds} + \rho \psi_{qs} \tag{45}$$

$$V_{ds} = -r_s i_{ds} + \omega_r \psi_{qs} + \rho \psi_{ds} \tag{46}$$

$$V'_{fd} = r_s i'_{fd} + \rho \psi'_{fd} \tag{47}$$

$$V_{0s} = -r_s i_{0s} + \rho \psi_{0s} \tag{48}$$

$$V'_{kd} = r'_{kd} i'_{kd} + \rho \psi'_{kd} \tag{49}$$

$$V'_{kq1} = r_{kq1} i'_{kq1} + \rho \psi'_{kq1} \tag{50}$$

$$V'_{kq2} = r_{kq2} i'_{kq2} + \rho \psi'_{kq2} \tag{51}$$

Using reactances instead of inductance requires re-defining the expressions for flux linkages to flux linkages per second, resulting in the following expressions.

$$\psi_{qs} = -X_{ls} i_{qs} + X_{mq}(-i_{qs} + i'_{kq1} + i'_{kq2}) \tag{52}$$

$$\psi_{ds} = -X_{ls} i_{ds} + X_{md}(-i_{ds} + i'_{fd} + i'_{kd}) \tag{53}$$

$$\psi'_{fd} = X'_{lfd} i'_{fd} + X_{md}(-i_{ds} + i'_{fd} + i'_{kd}) \tag{54}$$

$$\psi_{0s} = -X_{ls} i_{0s} \tag{55}$$

$$\psi'_{kd} = X'_{lkd} i'_{kd} + X_{md}(-i_{ds} + i'_{fd} + i'_{kd}) \tag{56}$$

$$\psi'_{kq1} = X'_{lkq1} i'_{kq1} + X_{mq}(-i_{qs} + i'_{kq1} + i'_{kq2}) \tag{57}$$

$$\psi'_{kq2} = X'_{lkq2} i'_{kq2} + X_{mq}(-i_{qs} + i'_{kq1} + i'_{kq2}) \tag{58}$$

Re-arranging the reactance based voltage equations in (45)-51), a system of state-space equations that describe the flux linkages and are dependent upon input voltages can be solved. In order to do so, several new quantities to further simplify the equations must be defined [18].

$$\psi_{mq} = X_{aq}\left(\frac{\psi_{qs}}{X_{ls}} + \frac{\psi'_{kq1}}{X'_{lkq1}} + \frac{\psi'_{kq2}}{X'_{lkq2}}\right) \tag{59}$$

$$\psi_{md} = X_{ad}\left(\frac{\psi_{ds}}{X_{ls}} + \frac{\psi'_{fd}}{X'_{lfd}} + \frac{\psi'_{kd}}{X'_{lkd}}\right) \tag{60}$$

where

$$X_{aq} = \left(\frac{1}{X_{mq}} + \frac{1}{X_{ls}} + \frac{1}{X'_{lkq1}} + \frac{1}{X'_{lkq2}}\right)^{-1} \tag{61}$$

$$X_{ad} = \left(\frac{1}{X_{md}} + \frac{1}{X_{ls}} + \frac{1}{X'_{lfd}} + \frac{1}{X'_{lkd1}}\right)^{-1} \tag{62}$$

$\psi_{mq}$ represents the q-axis mutual flux linkage, $\psi_{md}$ represents the d-axis mutual flux linkage. In addition, $X_{aq}$ and $X_{ad}$ are the respective q-axis and d-axis Thevenin equivalent reactances.

Now, the state-space integral equations for flux linkages can be introduced in Eq. (63) – Eq. (69). Eq (52) – Eq. (58) are used to solve for currents. This allows for simulation of the flux linkages and currents based solely on input voltages and rotor speed. Re-arranging the flux linkage equations to

solve for each variable associated with the time derivative through integration, the following can be stated.

$$\psi_{qs} = \int \left(V_{qs} - \omega_r \psi_{ds} - r_s i_{qs}\right) dt \tag{63}$$

$$\psi_{ds} = \int \left(V_{ds} + \omega_r \psi_{qs} - r_s i_{ds}\right) dt \tag{64}$$

$$\psi'_{fd} = \int \left( \frac{r'_{fd}}{X_{md}} e'_{xfd} + r'_{fd} i'_{fd} \right) dt \tag{65}$$

$$\psi_{0s} = \int \left(V_{0s} + r_s i_{0s}\right) dt \tag{66}$$

$$\psi'_{kd} = \int \left(V'_{kd} + r'_{kd} i'_{kd}\right) dt \tag{67}$$

$$\psi'_{kq1} = \int \left(V'_{kq1} + r_{kq1} i'_{kq1}\right) dt \tag{68}$$

$$\psi'_{kq2} = \int \left(V'_{kq2} + r_{kq2} i'_{kq2}\right) dt \tag{69}$$

The integral Eq. (63) – Eq. (69) results can then be used to calculate the branch currents in the machine circuits.

$$i_{qs} = -\frac{1}{X_{ls}} \left(\psi_{qs} - \psi_{mq}\right) \tag{70}$$

$$i_{ds} = -\frac{1}{X_{ls}} \left(\psi_{ds} - \psi_{md}\right) \tag{71}$$

$$i'_{fd} = \frac{1}{X'_{lfd}}\left(\psi'_{fd} - \psi_{md}\right) \tag{72}$$

$$i_{0s} = -\frac{1}{X'_{ls}}\left(\psi_{0s}\right) \tag{73}$$

$$i'_{kd} = \frac{1}{X'_{lkd}}\left(\psi'_{kd} - \psi_{md}\right) \tag{74}$$

$$i'_{kq1} = \frac{1}{X'_{lkq1}}\left(\psi'_{kq1} - \psi_{mq}\right) \tag{75}$$

$$i'_{kq2} = \frac{1}{X'_{lkq2}}\left(\psi'_{kq2} - \psi_{mq}\right) \tag{76}$$

**4.2 The 3rd Order Model**

Following the same procedure as shown in Section 4.2, a 3rd order model can be developed. This model is derived in the same fashion, but ignores the damper windings and zero sequence equations. Eq. (77) – Eq. (86) show the equations used for this model.

$$\psi_{qs} = \int\left(V_{qs} - \omega_r\psi_{ds} - r_s i_{qs}\right)dt \tag{77}$$

$$\psi_{ds} = \int\left(V_{ds} + \omega_r\psi_{qs} - r_s i_{ds}\right)dt \tag{78}$$

$$\psi'_{fd} = \int\left(\frac{r'_{fd}}{X_{md}}e'_{xfd} + r'_{fd}i'_{fd}\right)dt \tag{79}$$

Where

$$i_{qs} = -\frac{1}{X_{ls}}\left(\psi_{qs} - \psi_{mq}\right) \tag{80}$$

$$i_{ds} = -\frac{1}{X_{ls}}\left(\psi_{ds} - \psi_{md}\right) \tag{81}$$

$$i'_{fd} = \frac{1}{X'_{lfd}}\left(\psi'_{fd} - \psi_{md}\right) \tag{82}$$

and

$$\psi_{mq} = X_{aq}\left(\frac{\psi_{qs}}{X_{ls}}\right) \tag{83}$$

$$\psi_{md} = X_{ad}\left(\frac{\psi_{ds}}{X_{ls}} + \frac{\psi'_{fd}}{X'_{lfd}}\right) \tag{84}$$

$$X_{aq} = \left(\frac{1}{X_{mq}} + \frac{1}{X_{ls}}\right)^{-1} \tag{85}$$

$$X_{ad} = \left(\frac{1}{X_{md}} + \frac{1}{X_{ls}} + \frac{1}{X'_{lfd}}\right)^{-1} \tag{86}$$

### 4.3 MATLAB Implementation of State-Space Model

In order to use the least squares estimation method selected for online characterization in this thesis, a state-space model of the machine must first be developed. Equation (87) shows the general form of the state-space model required for the LSE method.

$$\begin{aligned} \dot{x} &= A(\alpha)x + B(\alpha)u \\ x(t_0) &= x_0 \end{aligned} \tag{87}$$

In this case, $A(\alpha)$ is an N×N matrix representing either the 3rd or 7th order coupled differential equations for synchronous machine flux linkages. The vector x represents a N×1 matrix of the current state of the flux linkages; u represents a N×1 vector of inputs to the machine. In this case, these inputs are defined as the d-axis and q-axis terminal voltages, as well as the field voltage. In Eq. (87), $B(\alpha)$ is a mapping matrix relating the inputs u. For the 7th order model, N is 7. For the 3rd order model, N is 3. The vector $\alpha$ represents the machine parameter to be estimated.

Using the equations described in Section 4.2, a program is developed within MATLAB R2013b. The MATLAB files written to generate the results shown in this section are shown in Appendix A.

Before implementing the equations in MATLAB, a generator model is built in SIMULINK to simulate a "real" machine. The parameters used in the SIMULINK machine model, and subsequently, the MATLAB model, are shown in Table 4. This is the same model that will be used to compare results from the least squares estimation, which will be discussed later.

Using these values, the SIMULINK machine outputs can be used to test the machine model built in MATLAB.

The SIMULINK model can also be used to simulate three-phase faults, as well as run steady-state conditions. In the first case, the simulation is run under steady-state, open circuit conditions. Because SIMULINK does not allow the terminals of the machine to physically be disconnected, a large resistive bank (10E7 Ohms) is connected to the terminals of the machine. In doing so, the stator currents are extremely low and can be considered to be in open circuit conditions.

**Table 4: Synchronous Machine Input Parameters**

| **Parameters referenced to stator** | | | | | | |
|---|---|---|---|---|---|---|
| **Stator:** | $L_{md}$ | $L_{mq}$ | $L_{lk}$ | | $r_s$ | |
| Value [H, Ohm] | 3.2164E-03 | 9.7153E-04 | 3.0892E-04 | | 2.9069E-03 | |
| **Field:** | $L'_{fd}$ | $r'_{fd}$ | | | | |
| Value [H, Ohm] | 3.0715E-04 | 1.9013E-03 | | | | |
| **Damper:** | $L'_{lkq1}$ | $L'_{lkq2}$ | $L'_{lkd}$ | $r'_{kq1}$ | $r'_{kq2}$ | $r'_{kd}$ |
| Value [H, Ohm] | 1.0365E-03 | 1.0365E-03 | 4.90764E-04 | 2.0081E-02 | 2.0081E-02 | 1.1900E-02 |

| **Nameplate Data** | | |
|---|---|---|
| | **Stator** | **Rotor** |
| Apparent Power [MVA] | 187 | |
| Voltage [V] | 13,800 | 226.6 |
| Current [A] | 11,000 | 1087 |
| Frequency [Hz] | 60 | |
| Pole pairs | 20 | |

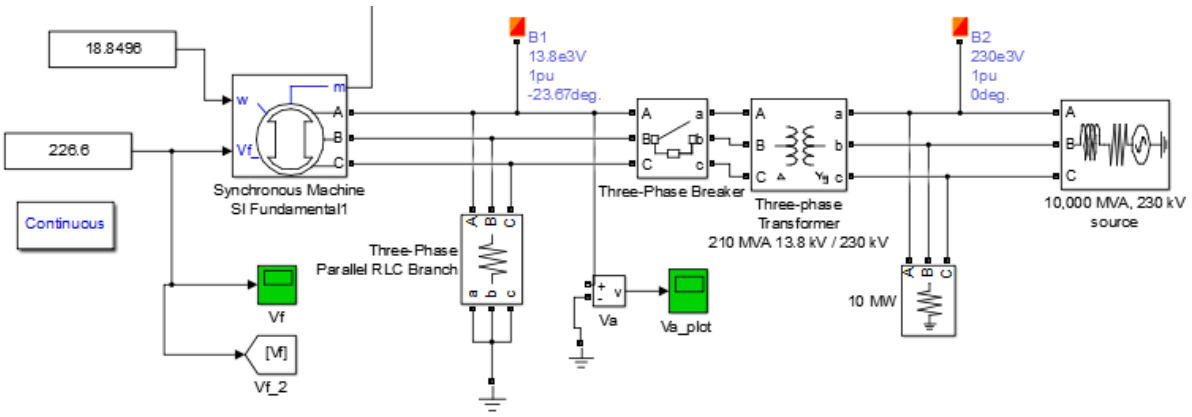


**Figure 17: SIMULINK Model of a Synchronous Machine in Open-Circuit Conditions**

Under open circuit conditions, rated stator voltages and small output currents are expected to be seen on the stator output of the machine. Translating these outputs to the DQ-domain, we expect to see nominal line-to-neutral voltage in the q-axis and no voltage in the d-axis. The output data from the SIMULINK simulation data is compared with the MATLAB model described by the equations in Chapter 4.2. The open circuit voltages from the model under test very closely match the reference SIMULINK model as shown in Figure 18. The MATLAB model finds small, but non-zero current, and therefore non-zero flux linkages, where both should be zero. Considering the large stator current rating of this machine, the error is relatively small but still worth noting the inaccuracy under this condition. Note that the calculated plots show some transient behavior early in the simulation. Note that the quadrature and direct axis voltages are not calculated. This is because these values are used as the inputs to the MATLAB model to simulate the flux linkages and currents.

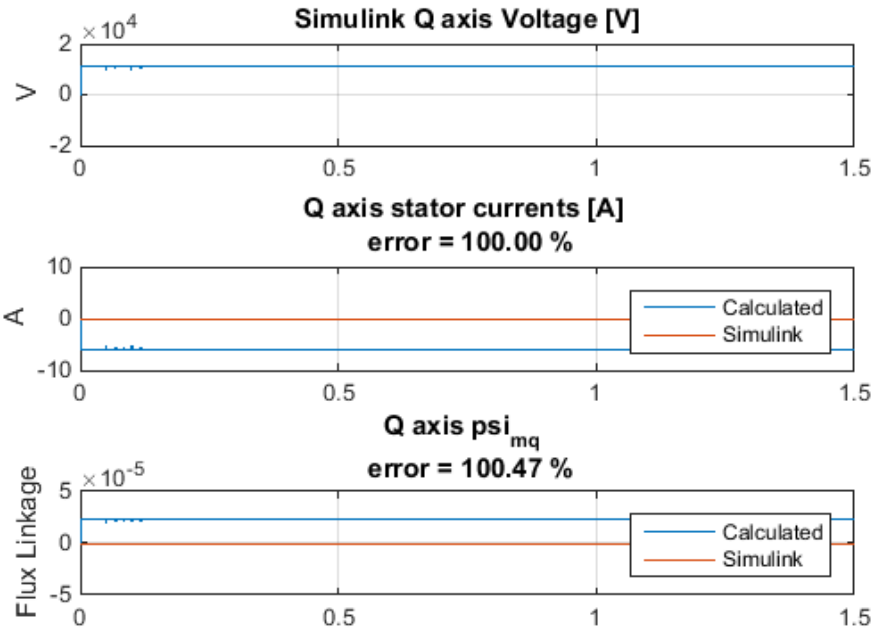


**Figure 18: Q axis voltage, current and flux linkage from SIMULINK and MATLAB simulation results - Open Circuit Conditions**

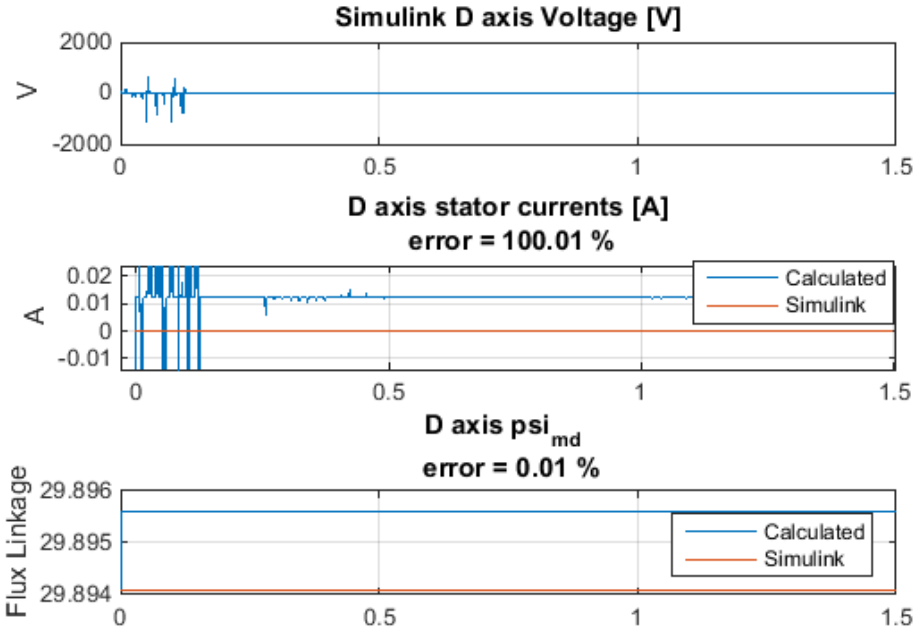


**Figure 19: D axis voltage, current and flux linkage from SIMULINK and MATLAB simulation results - Open Circuit Conditions**

Note that in both Figure 18 and Figure 19, the error appears to be extremely high. However, this is because, ideally, there would be no currents, and no mutual flux linkage in the q-axis. Slight deviations from such small values create high errors but the results still meet expectations for open circuit conditions. The mutual d-axis flux linkage is non-zero due to flux from the field voltage. This is verified by looking at Eq. (60). As this is steady state, the flux in all damper windings is zero. During open circuit conditions, non-zero field flux is present along with direct axis flux, corresponding to voltage aligned with the q-axis. Figure 20 shows, the field current in the MATLAB model matches the nominal current described in the SIMULINK model (1087A) for open circuit conditions.

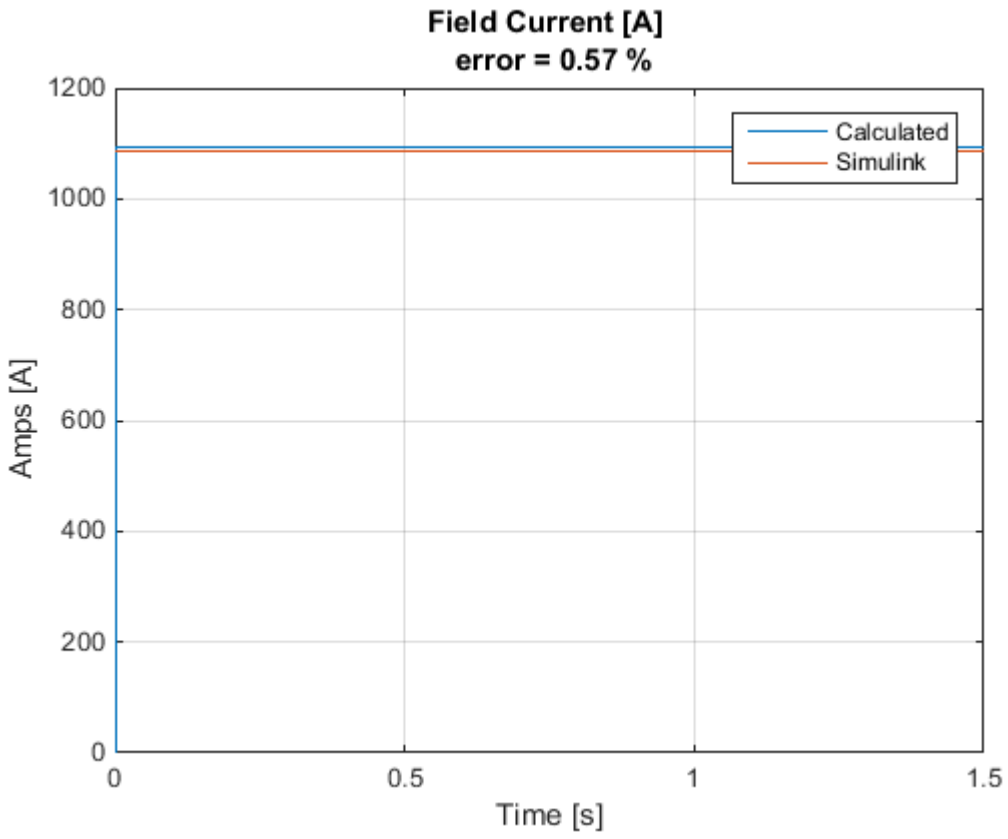


**Figure 20: SIMULINK and MATLAB Field Current - Open Circuit Conditions**

Next, the system is run under steady-state, loaded conditions. In this case, a simple parallel resistive and inductive load is used to draw power from the synchronous generator. Again, the results in

comparing the MATLAB state-space model to the reference SIMULINK model show a good match, thus verifying the validity of the mathematical model. Again, note that the quadrature and direct axis voltages are not calculated in the mathematical model. This is because these values are used as the inputs to the MATLAB model to simulate the flux linkages and currents.

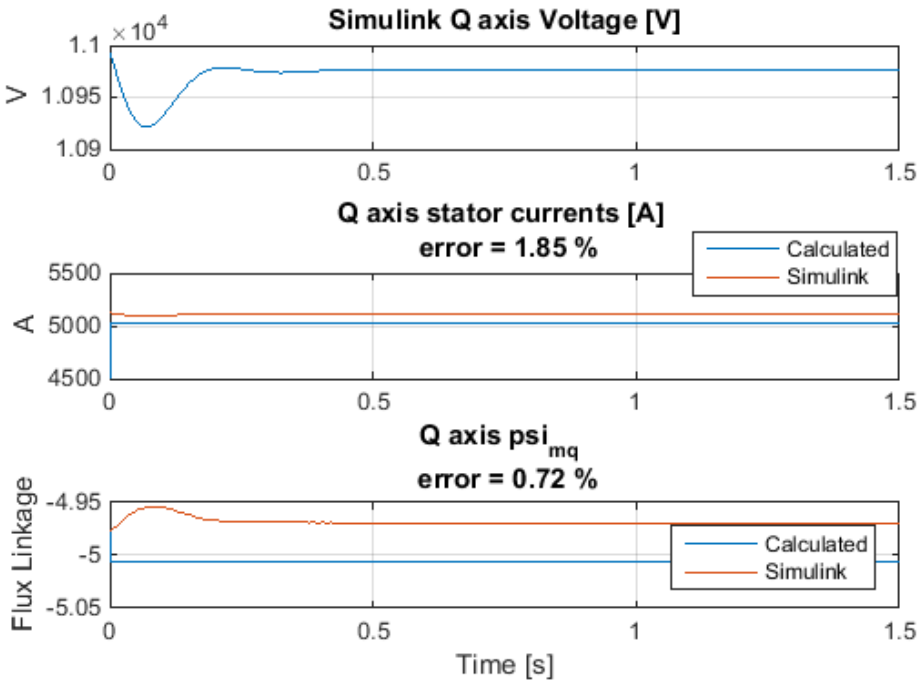


**Figure 21: Q axis voltage, current and flux linkage from SIMULINK and MATLAB simulation results - Loaded Conditions**

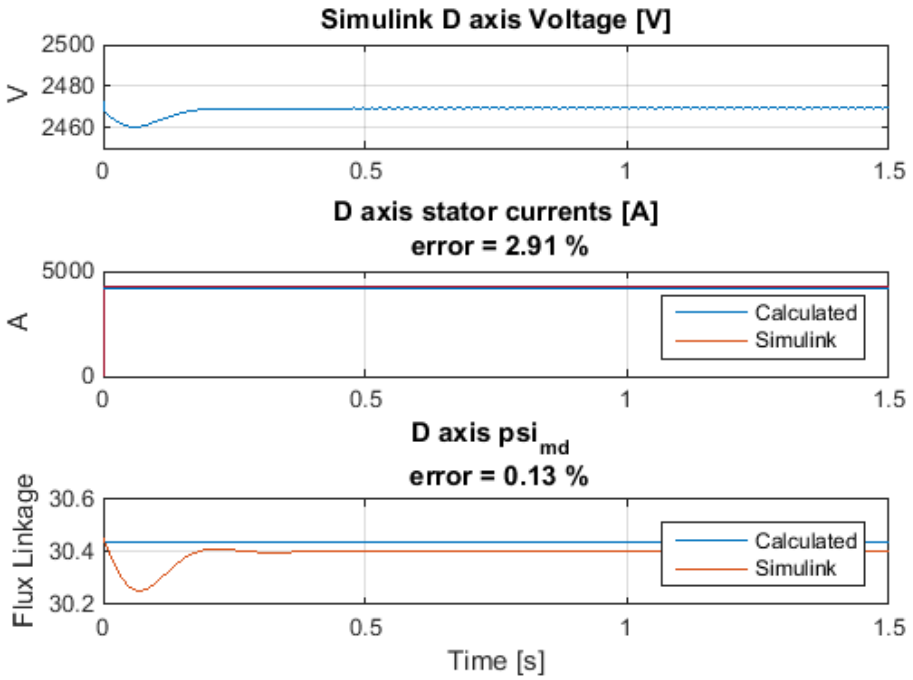


**Figure 22: D axis voltage, current and flux linkage from SIMULINK and MATLAB simulation results - Loaded Conditions**

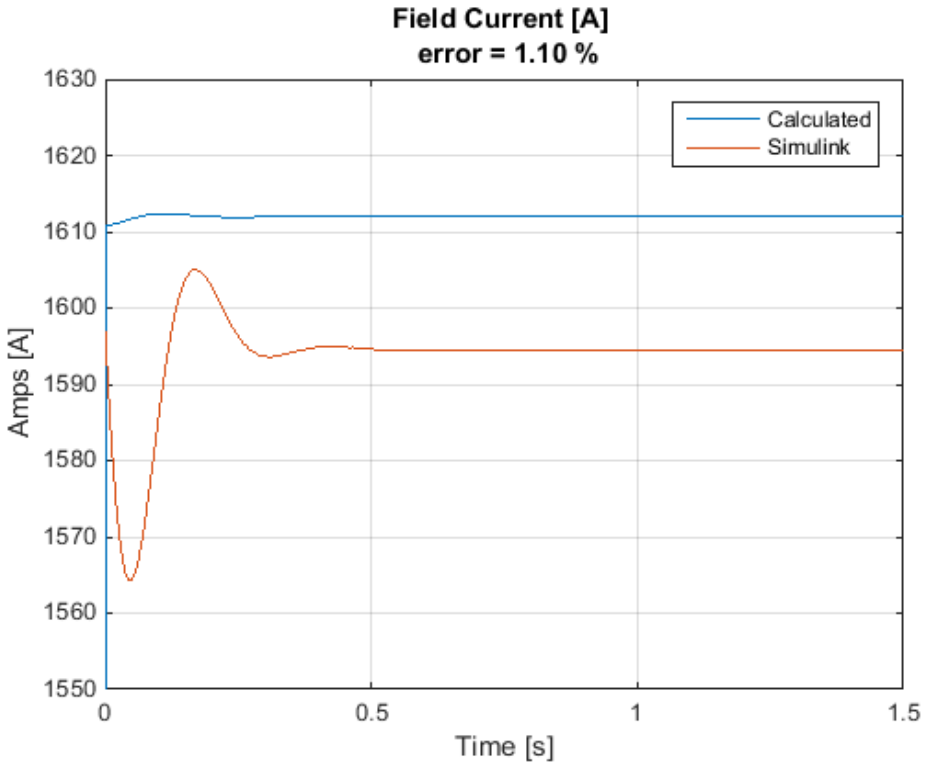


**Figure 23: SIMULINK and MATLAB Field Current - Loaded Conditions**

In this loaded case, the calculated field current is slightly higher than the SIMULINK model results. This results in a higher d-axis flux. However, the error is well within an acceptable tolerance. The initial transient, shown at the beginning of the SIMULINK results in Figures 21-23, is a start-up transient in the program. Based on the results of the comparison of the open circuit and the loaded conditions, the state-space model discussed earlier is sufficiently accurate to use for a machine modeling in the parameter estimation routine.

# 5. Offline Characterization Procedures

This chapter discusses offline characterization procedures used to determine the parameters for the 20 hp synchronous machine at the University of Idaho. These methods follow directly from the IEEE 115-2009 Standard for the purpose of developing an "experimental control" model to eventually be compared with the online method results. The machine under test is fed by an ABB Variable Frequency Drive (VFD) in order to ensure constant speed with an input signal of 60Hz.

## 5.1 Full Load Test

To ensure proper behavior during normal operating conditions, a full load test is performed by connecting the synchronous generator to a resistive load bank. The test setup for the full load test is shown in Figure 10.

To verify proper readings from the SEL 411L Relay connected to the terminals of the machine, a steady-state test was conducted. Measurements taken using multimeters and current clamps are shown in Table 5. Oscillography was also captured by the 411L relay.

**Table 5: Steady-State Test Measured Data**

| Field Voltage | Field Current | Line-Line, True RMS Terminal Voltage | True RMS Line Current | Frequency | Per Phase Resistance Y-connected |
|---|---|---|---|---|---|
| 55.2 V | 3.21A | 220 V | 3.3A | 60.01 Hz | 31 Ω |

Using the data in Table 5, a comparison can be made with the oscillography taken from the SEL 411L relay to verify that proper relay settings are entered. Oscillography captured under steady-state conditions is shown in Figure 24.

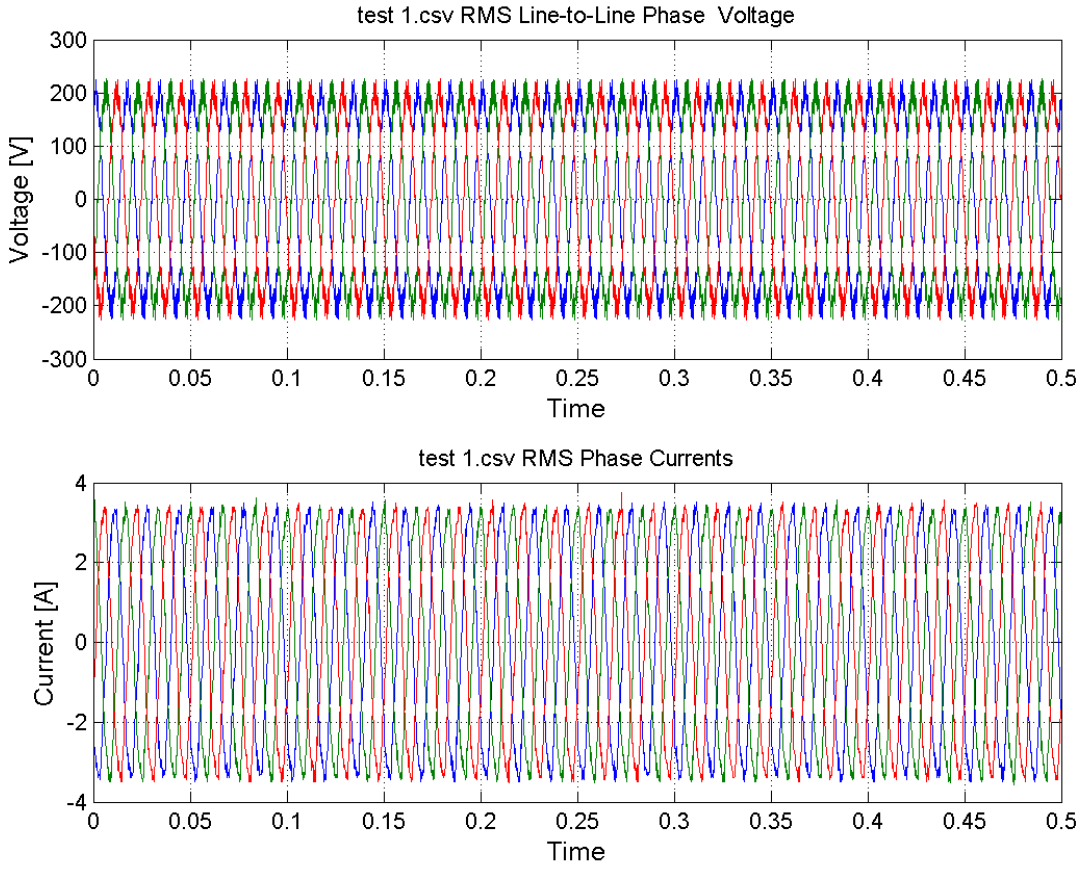


**Figure 24: Steady State Voltage and Current**

Comparison of the oscillography and the data shown in Table 5 shows the relay data is validated to accurately reflect the ohmmeter and current clamp measurements. The SEL 411L scales the voltage and currents such that the fundamental frequency sinusoidal terms have peak amplitudes equal to the RMS magnitudes. Note the harmonic content in Figure 24 as well. This is a result of slot harmonics, and will be discussed more in Chapter 4.5.

### 5.2 Open Circuit Saturation Curve

The first step in developing accurate machine parameters is to understand a machine's saturation characteristics. As described in the IEEE 115-2009 Standard, the steps performed in this test are as follows [7]:

a) Six readings should be taken below 60% of rated voltage (1 at zero excitation).
b) From 60% to 110%, readings should be taken, at a minimum, at every 5% increment in terminal voltage (minimum of 10 points). This area is a critical range, and an attempt should be made to obtain as many points as the excitation control resolution will allow.
c) Above 110%, readings should be taken, at a minimum, at two points, including one point at approximately 120% of the rated no-load field current (or at the maximum value recommended by the manufacturer).
d) At rated voltage, readings should be taken of the terminal voltage (line-to-line) of all three phases to check phase balance. These readings should be made under constant conditions of excitation and speed and with the same voltmeter.

From this data, the airgap line of the machine can be drawn by calculating the maximum slope of the curve and plotting a linear curve of the same slope, tangential to the curve [9].

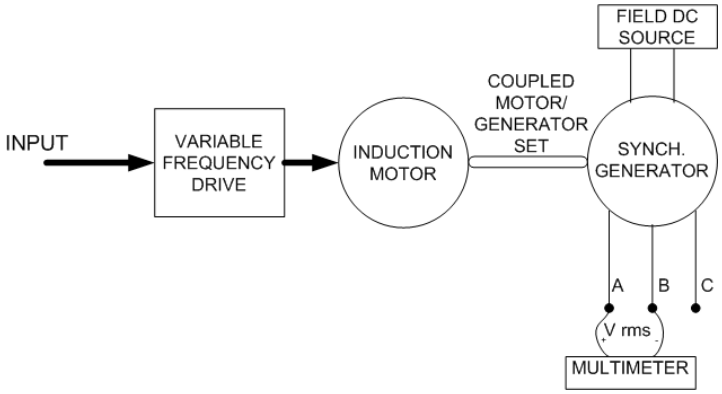


**Figure 25: Open Circuit Test Diagram**

### 5.3 Short Circuit Saturation Curve

The short circuit saturation curve complements the open circuit saturation curve when performing off-line characterization. The following procedure outlines this test as described in the IEEE 115-2009 Standard.

The machine under test is driven at rated speed. With the armature short circuited, readings of armature and field currents are recorded at armature current increments starting at 125% down to 25% of rated current. It is important to start at the higher current in order to maintain a quasi-constant temperature on the machine. If not, the changes in temperature may affect the accuracy of the test.

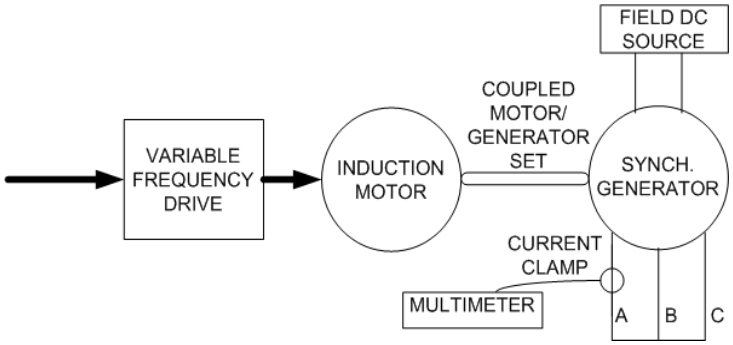


**Figure 26: Short Circuit Test Diagram**

The saturation curve test results for the laboratory synchronous machine for both the open circuit and short circuit tests are shown in Figure 27. These tests show saturation occurring just below the rated voltage of 220V line-to-line. The open circuit curve begins to deviate from the airgap line at approximately 175V.

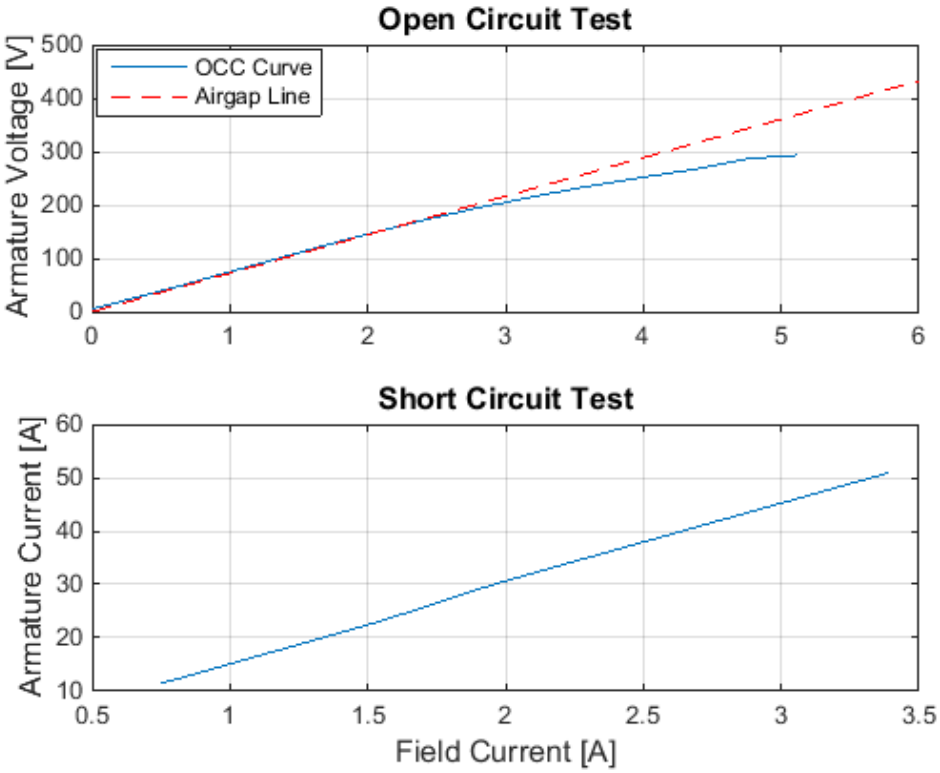


**Figure 27: Open Circuit and Short Circuit Saturation Curves**

Following the IEEE 115 Std. the unsaturated direct axis synchronous reactance can be determined from these tests:

$$X_{du} = \frac{I_{FSI}}{I_{FG}} = 1.007\text{pu} \tag{88}$$

Where $I_{FSI}$ is the field current that produces the nominal armature current on the short-circuit saturation curve, and $I_{FG}$ is the field current at base voltage on the air-gap line [7]. As a reminder, the nameplate data for the laboratory machine was described in Table 3.

### 5.4 Three-Phase Bolted Fault

According to the IEEE 115-2009 standard, a bolted fault is to be placed on the terminals of the machine during open circuit conditions. The Table 6 summarizes several three-phase bolted short circuit tests conducted at varying initial voltage conditions. Prior to the fault, the machine is run under

open –circuit conditions. Note that all tests are below rated terminal voltage. This was done to initially insure that fault currents would not exceed the rated current of the model power system.

**Table 6: Three-phase bolted fault prefault conditions measured at machine terminals**

| Fault Type: 3-phase bolted fault | | | | | | | |
|---|---|---|---|---|---|---|---|
| Test # | Field Voltage (V) | Field Current (A) | Line-to-Line Terminal Voltage (V RMS) | Phase A Current (A RMS) | Frequency | Fault Angle (deg) | Fault Duration (cycles) |
| 1 | 24.4 | 1.37 | 100 | 0.5 | 60.01 | 90 | 180 |
| 2 | 30.3 | 1.73 | 125 | 0.6 | 60.01 | 90 | 180 |
| 3 | 36.3 | 2.11 | 150 | 0.8 | 60.01 | 90 | 180 |
| 4 | 43.7 | 2.54 | 175 | 1.0 | 60.01 | 90 | 180 |

The following section describes results from each of the tests described above. Note that the relay sampled this data at 128 samples per cycle.

Any per unit calculations or conversions are selected based on nameplate rated voltage and power. For the per unit calculations, the results for base calculations are described in Table 7.

**Table 7: Per Unit Base Calculations**

| Apparent Power Base | Voltage (phase) | Current (phase) | Voltage (dq) | Current (dq) | Impedance | frequency |
|---|---|---|---|---|---|---|
| 20 h.p. | 220 V | 39.139 A | 179.63 V | 55.35 A | 3.2453 | 60Hz |

### 5.4.1 Test 1

Oscillography of the event report generated by the SEL 411L relay for the first three phase bolted fault test is shown in Figure 28. The data represents the 3 phase currents and voltages as measured at the terminals of the machine for test 1.

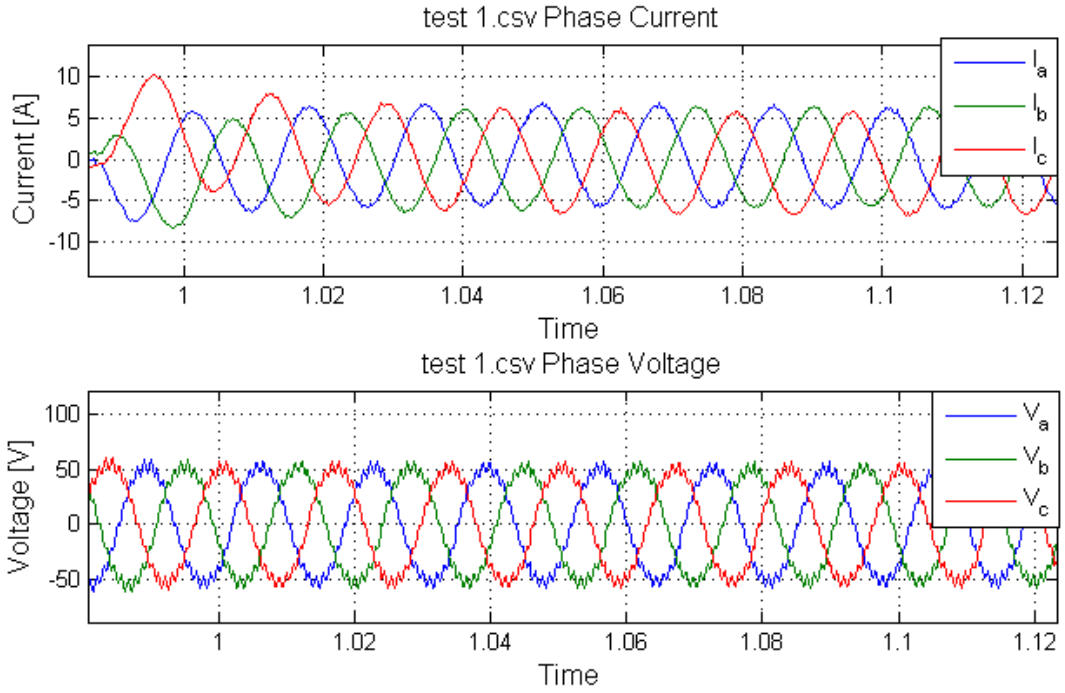


**Figure 28: Test 1 Oscillography of raw data captured**

As the voltage waveform clearly shows, a higher frequency is riding on top of the 60Hz signal. A Fast Fourier Transform (FFT) of the dataset above is shown in Figure 29. In doing so, the higher order frequencies observed at the machine terminals can be identified.

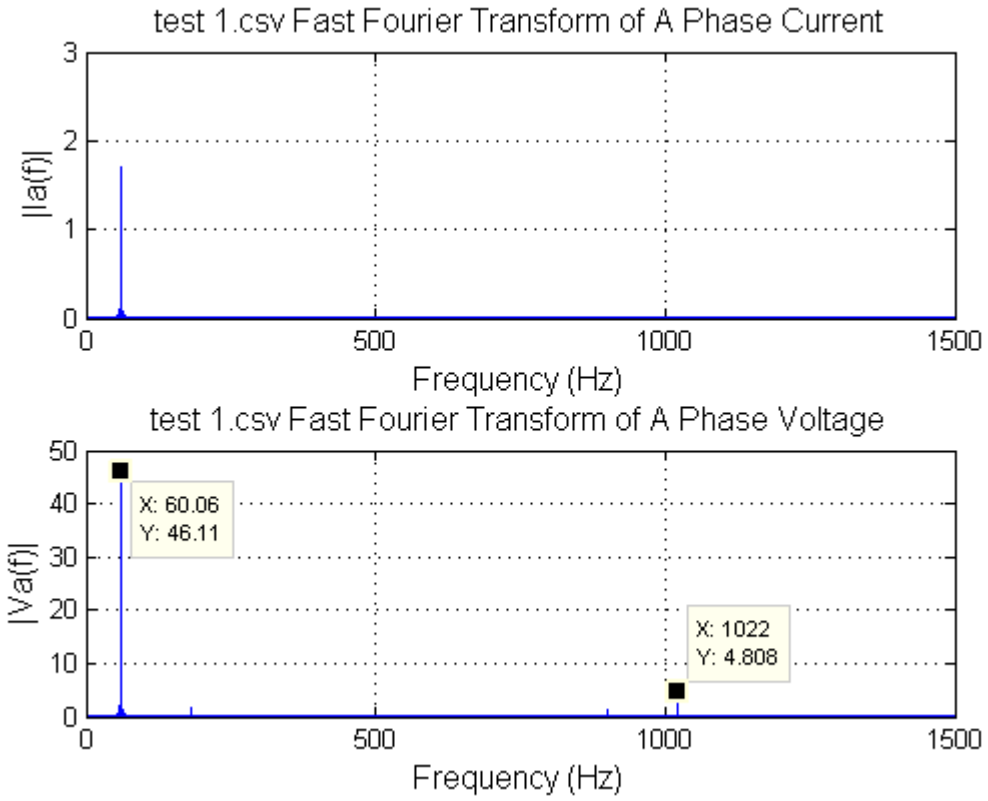


**Figure 29: Test 1 voltage and current FFT**

Both the current and voltage waveforms are dominated by the 60Hz behavior as expected and shown in Figure 29. However, higher order frequencies can easily be seen in FFT of the voltage. FFT analysis clearly shows a frequency of 1024Hz, or the 17$^{th}$ harmonic. This harmonic is a result of imperfections in slot pitch (a.k.a. slot harmonics). Using a 7$^{th}$ order Butterworth filter, with a cutoff frequency of 90Hz, the higher order frequencies can be removed from the waveform. Figure 30 shows the frequency response of the designed Butterworth filter.

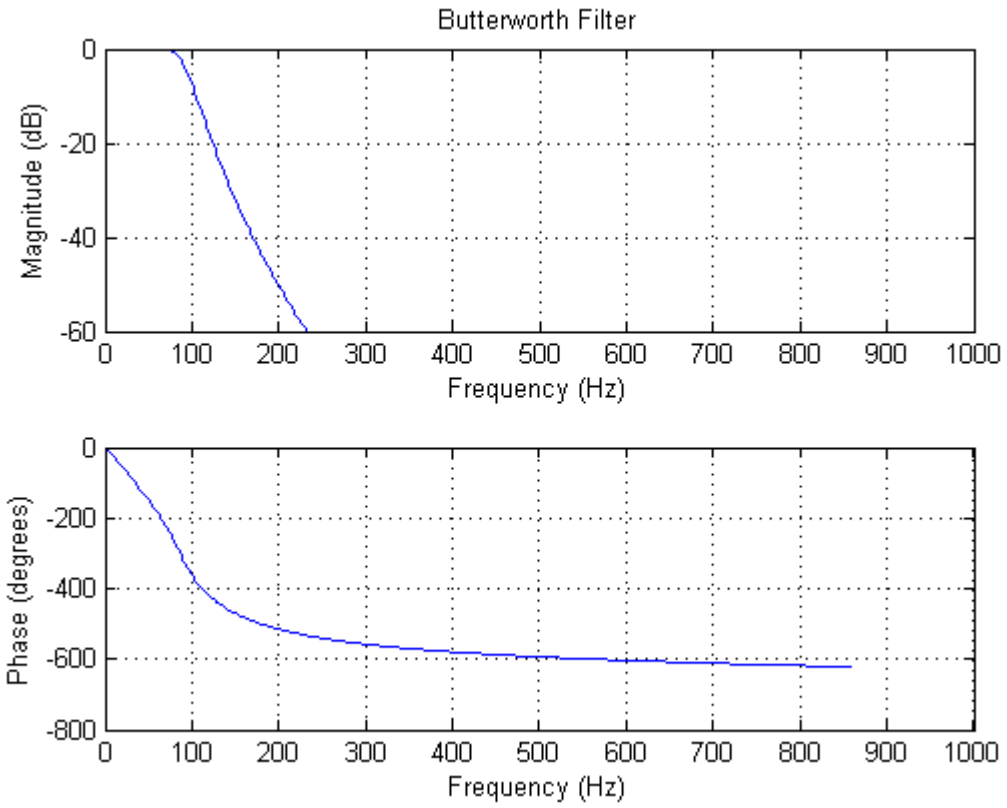


**Figure 30: Magnitude response of the lowpass butterworth filter used to remove high frequencies**

This filter is designed to remove frequencies above 90 Hz. This guarantees that any frequencies around the 60Hz range are passed with a gain of 1.0, while the higher frequencies are completely removed. This 90Hz filter point can easily be seen at the -3dB, indicating the cutoff frequency.

MATLAB's built-in Butterworth filter used with the application function "filter." The use of the Butterworth filter creates a phase shift due to the inherent nature of the Butterworth filter as shown in Figure 30. However, MATLAB has a built in routine known as "filtfilt" which applies the filter in both the forward and negative direction, creating a zero-phase distortion by canceling the phase shift. By using the "filtfilt" function, a zero phase filter is created as shown in Figure 31. To verify that no phase shift exists, the phase A voltage and currents for the raw and filtered data as shown in Figure 32. The filtered waveform clearly is in phase with the raw information.

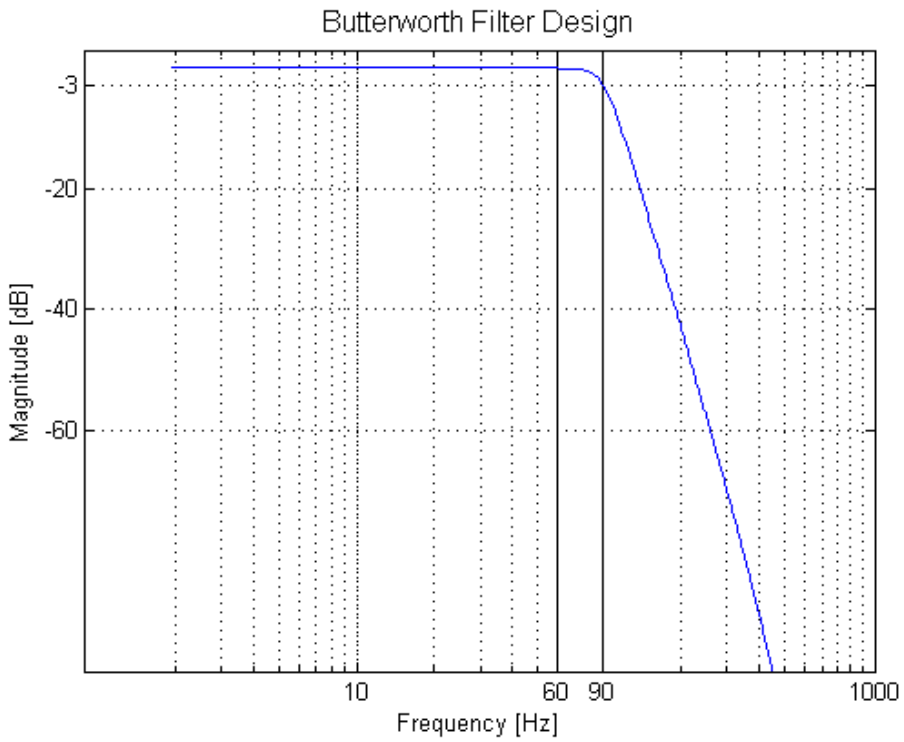


**Figure 31: Magnitude response of the lowpass butterworth filter and MATLAB "filtfilt" function**

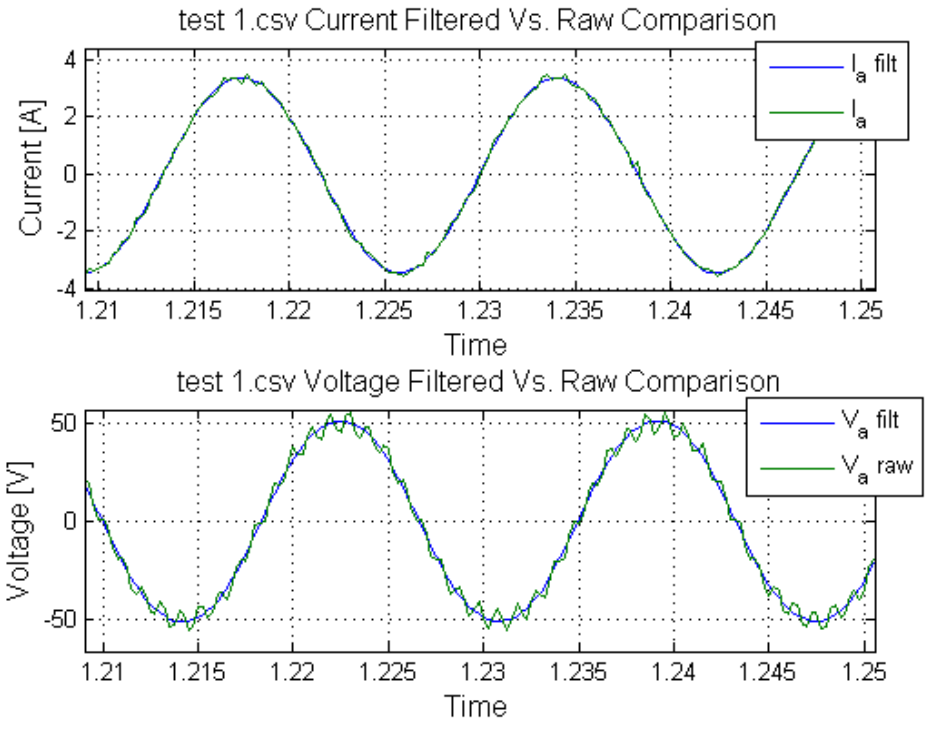


**Figure 32: Test 1 comparison of filtered and raw data**

Figure 33 shows the output with the Butterworth filter and MATLAB's filtfilt function used to filter the data collected from the test.

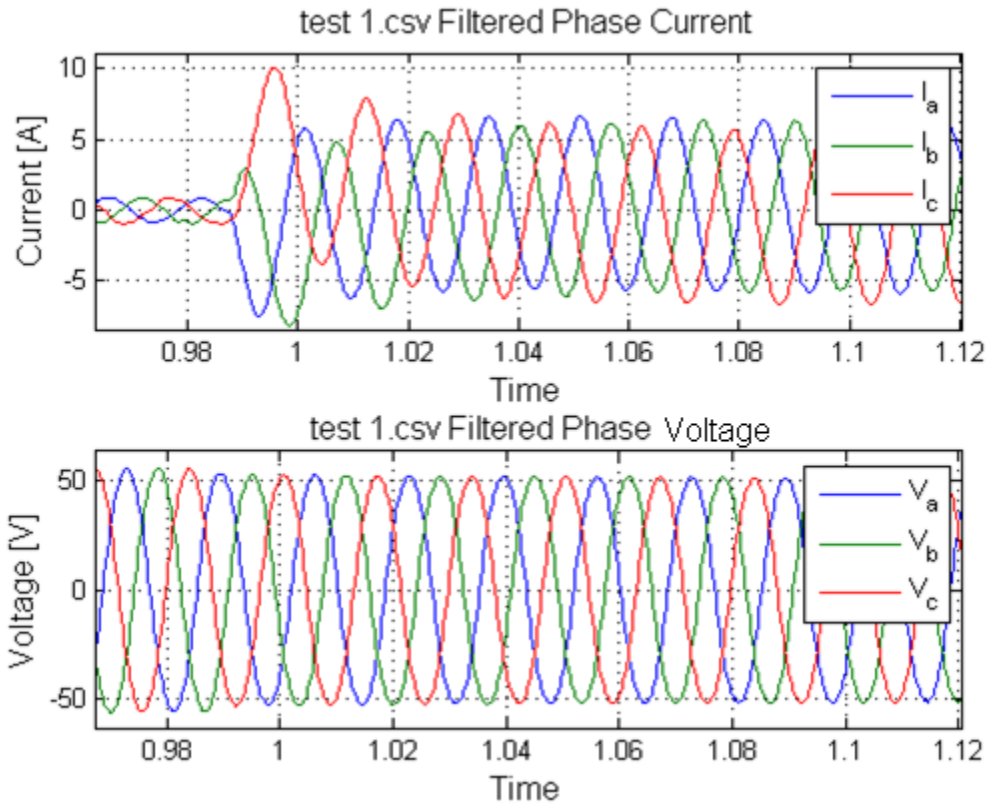


**Figure 33: Test 1 filtered output data**

The voltage waveform with the higher frequencies clearly removed is shown in Figure 33. This result has a clean, clear capture of the 60Hz signal, while the current waveform captures the low frequency and dynamic response to this three-phase bolted short circuit on the transformer terminals.

The dq0 representation of the unfiltered voltages and currents for test 1 is shown in Figure 34.

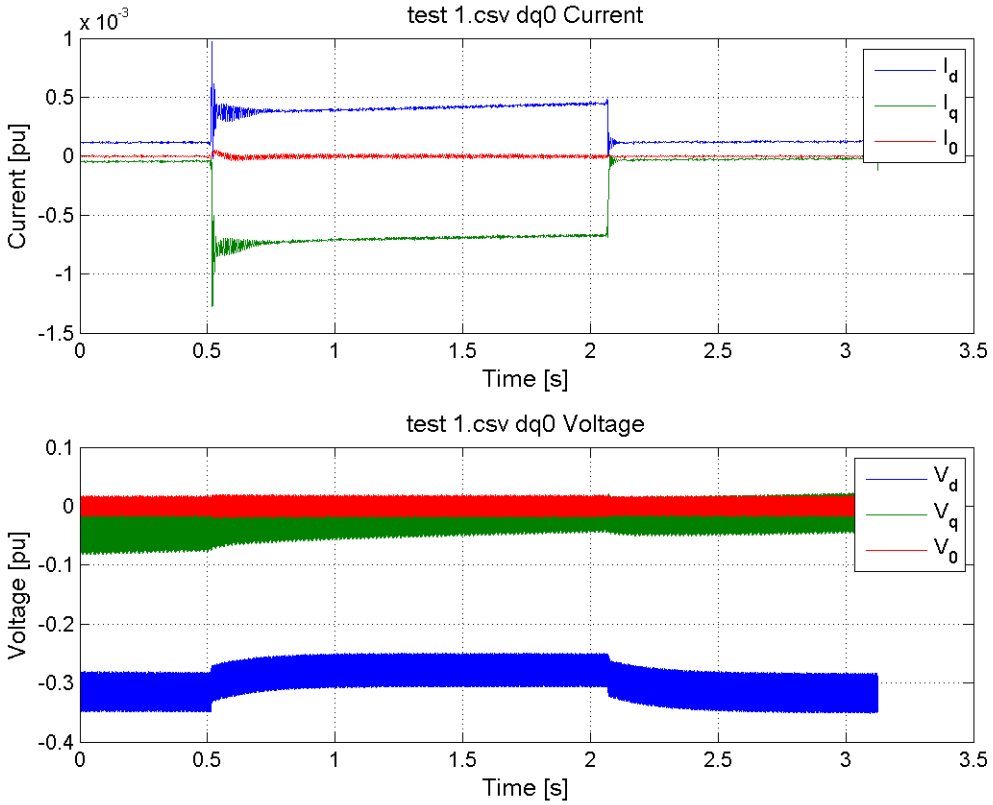


**Figure 34: Test 1 dq0 representation of phase voltages and currents (raw data)**

High order frequencies shown in the voltage waveform of Figure 34 are a result of the 17$^{th}$ harmonic. In order to see a clear representation of the direct and quadrature axis voltages, the filtered data is also converted by using the Park's transformation. The dq0 representation of the filtered voltages and currents for test 1 are shown in Figure 35.

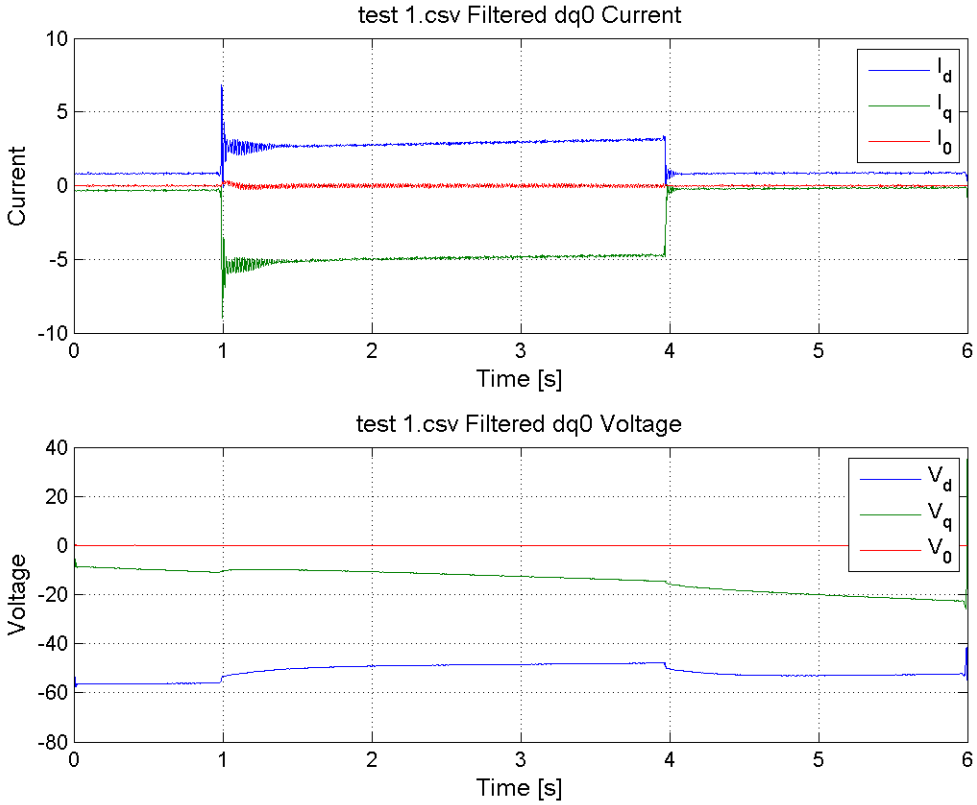


**Figure 35: Test 1 dq0 representation of filtered phase voltages and currents (filtered data)**

A much smoother signal can be seen in Figure 35. This shows the trending d and q-axis currents and voltages for test 1. Following the IEEE Std. 115, the per unit d-axis reactance can be calculated from the short circuit data. Second, an envelope of the short circuit is plotted as shown in Figure 36.

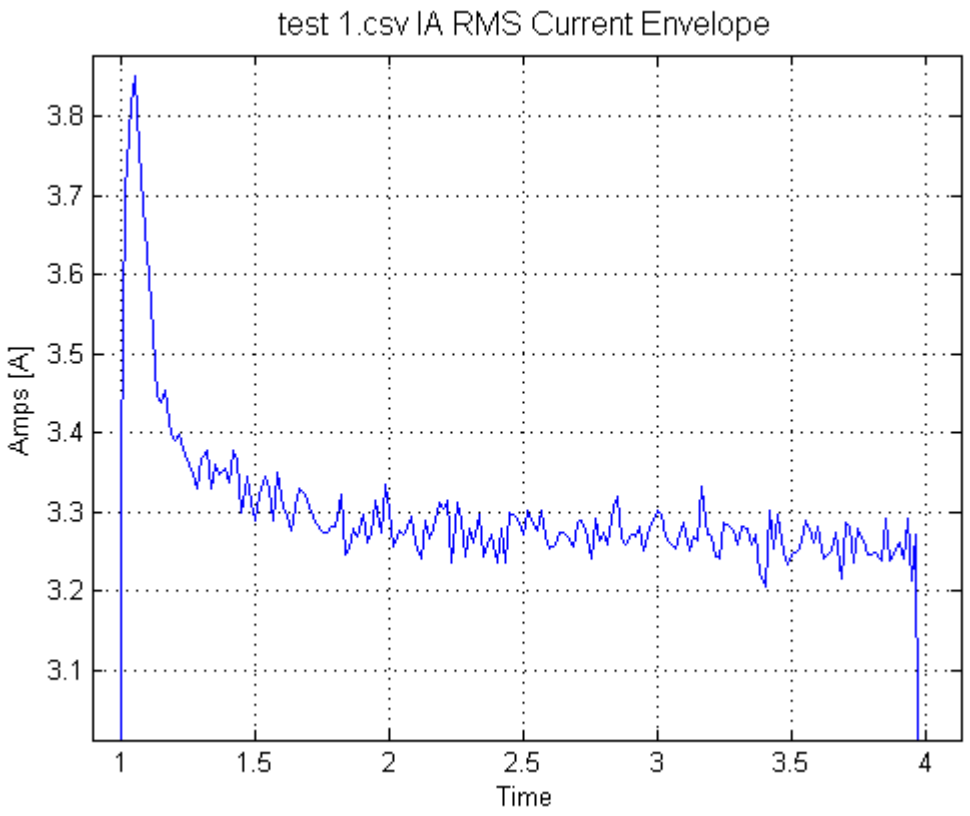


**Figure 36: Envelope of Test 1 Short Circuit Current**

By averaging the last 0.5 seconds of the fault, a steady-state, d-axis current was calculated using Eq. (89).

$$X_d = \frac{E_a}{I_d} = \frac{3.27}{57.73} = 17.66\Omega$$
$$X_{dpu} = 5.45 \tag{89}$$

Using this same envelope, the subtransient and transient reactances can be determined following Eq. (90)

$$envelope: i(t) = \sqrt{2} * \left[ \left( I_d'' - I_d' \right) * e^{-t/T_d''} + \left( I_d' - I_d \right) * e^{-t/T_d'} + \left( I_d \right) \right] \tag{90}$$

$$envelope: -I_d = i(t) = \sqrt{2} * \left[ \left( I_d'' - I_d' \right) * e^{-t/T_d''} + \left( I_d' - I_d \right) * e^{-t/T_d'} \right] \tag{91}$$

At the instance of the fault, Eq (92) can be considered to be true.

$$i(o) = \sqrt{2} * \left[ I_d'' - I_d' + I_d' - I_d \right] \tag{92}$$

$$i(o) = \sqrt{2} * \left[ I_d'' - I_d \right] \tag{93}$$

So, the derivation shown in Eq (90) - Eq (93) can be used to define the subtransient current.

$$I_d'' = \frac{i(o) + I_d}{\sqrt{2}} \tag{93}$$

Using Eq. (89), the subtransient reactance can be calculated by substituting the steady-state d-axis current for the subtransient d-axis reactance, results of this calculation are shown in (94).

$$\begin{aligned} X_d'' &= \frac{E_a}{I_d''} = \frac{57.76}{5.04} = 11.47\Omega \\ X_{dpu}'' &= 3.53 \end{aligned} \tag{94}$$

To calculate the transient reactance, the envelope needs to be plotted in a semilog scale to make the transient period look more like linear. This helps in identifying where the subtransient period ends and the transient period begins.

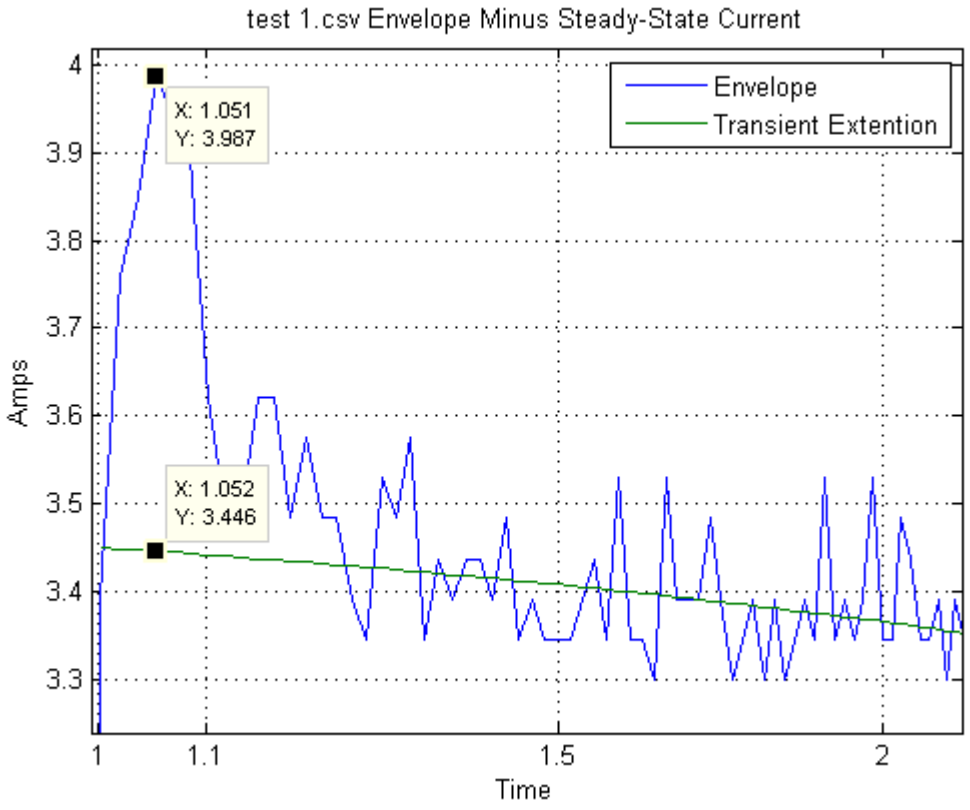


**Figure 37: Semilog x-axis plot of first cycles of the short circuit**

If the linear region is extended back to t(0), in this case, the peak, the same approach as shown in Eq. (90) – Eq. (93) can be used to determine the transient reactance. This value at t(0) was determined to be 3.5.

$$X'_d = \frac{E_a}{I'_d} = \frac{57.76}{3.44} = 12.17\Omega$$
$$X'_{dpu} = 3.75 \quad (95)$$

Now, the transient time-constant constant can be identified from the data shown in Figure 36. This can be done by using the values shown and their associated time-stamps. For transient time-constant, at t = 1.58 seconds (or 0.58 seconds into the fault), Eq. (99) is said to be true.

$$\sqrt{2} * \left[I''_d - I_d\right] * e^{\frac{-0.58}{T'_d}} = 3.35A \quad (99)$$

Solving for T`$_d$ results in a value of 1.80 seconds

The subtransient component can then be obtained by subtracting the transient component from the envelope, which can be seen in Figure 38. From this line, the approximate subtransient time-constant can be calculated using the same approach shown in Eq. (99).

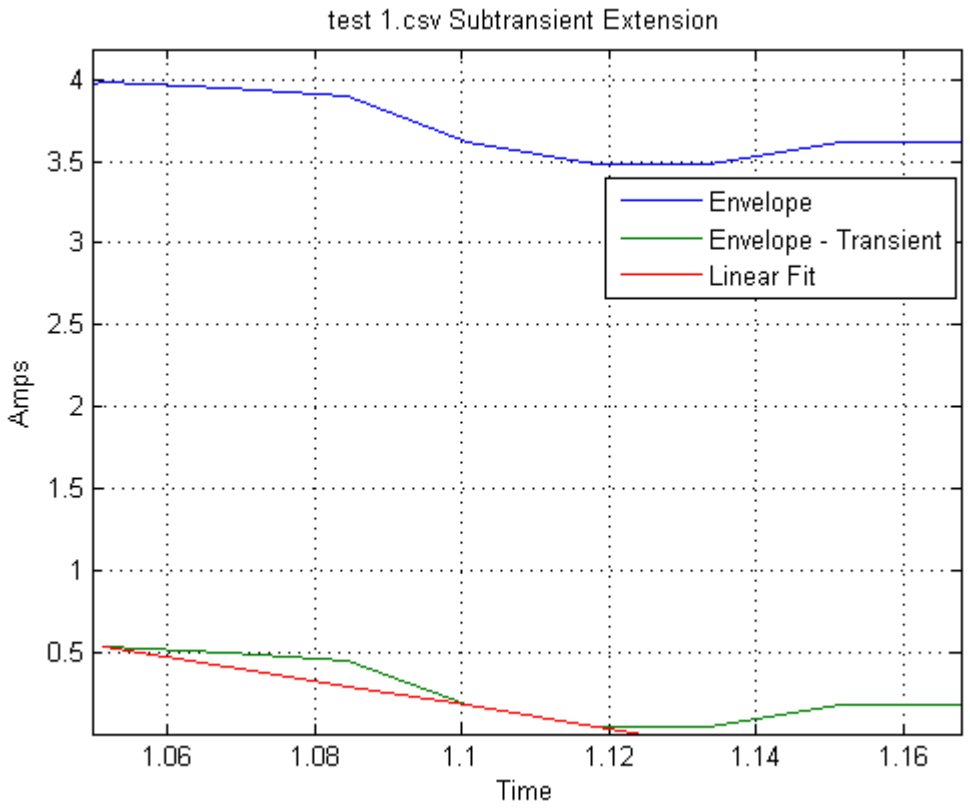


**Figure 38: Subtransient portion of Envelope**

For subtransient time-constant, at t = 1.06 seconds (or 0.06 seconds into the fault), Eq. (100) is said to be true.

$$\sqrt{2} * \left[ I_d'' - I_d' \right] * e^{\frac{-0.06}{T_d'}} = 0.51A \qquad (100)$$

Solving for T"$_d$ results in a value of 0.026 seconds. A summary of the values calculated for test 1 are shown in Table 8.

**Table 8: Summary of Test 1 Results**

| Parameter | Result: | Description: |
|---|---|---|
| $X_d$ | 5.45pu | D-axis steady-state reactance |
| $X'_d$ | 3.75pu | D-axis transient reactance |
| $X''_d$ | 3.53pu | D-axis subtransient reactance |
| $T'_d$ | 1.80 s | D-axis transient reactance |
| $T''_d$ | 0.026 s | D-axis subtransient reactance |

### 5.4.2 Test 2

Following the same procedure as test 1, test 2 was conducted at a slightly higher excitation level as described in Table 6. Oscillography of the event report generated by the SEL 411L relay for test 2 can be seen in Figure 39. The data represents the 3 phase currents and voltages as measured at the terminals of the machine in addition to the 17th harmonic term observed in test 1.

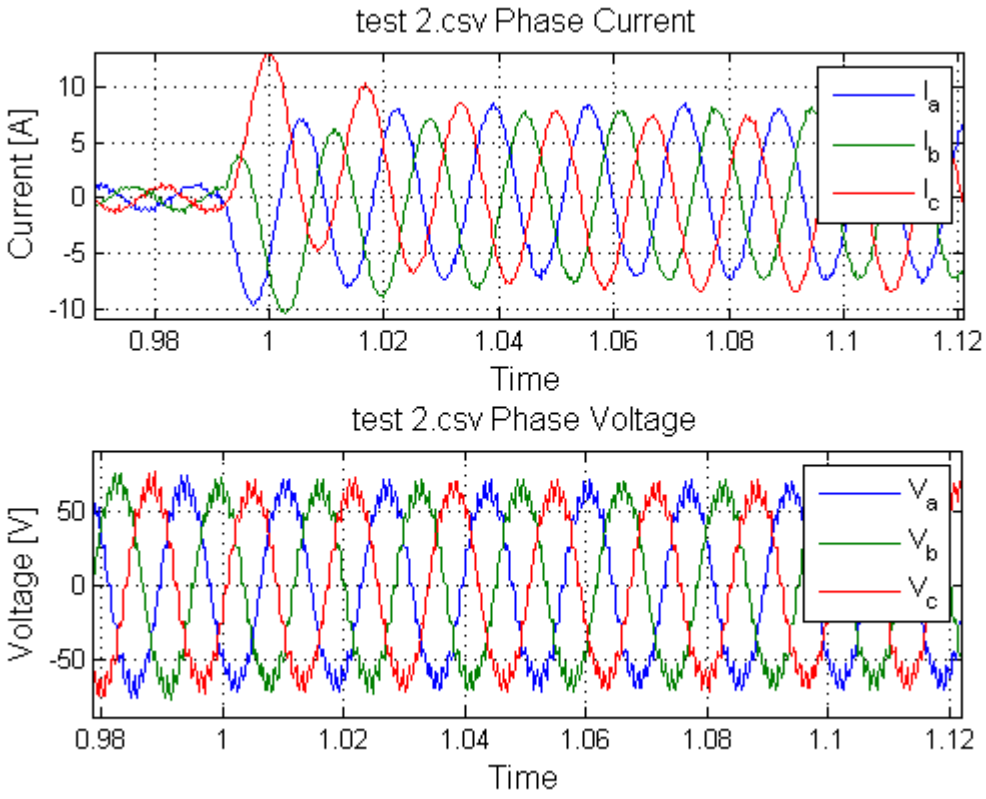


**Figure 39: Test 2 Oscillography of raw data captured**

Again, a higher frequency term is riding on top of the 60Hz signal. This higher frequency is especially prevalent in the voltage waveform. A fast Fourier transform (FFT) of the dataset above is shown in Figure 40. In doing so, the higher order frequencies observed at the machine terminals can be identified. This figure also starts to reveal a third harmonic present in the system as well.

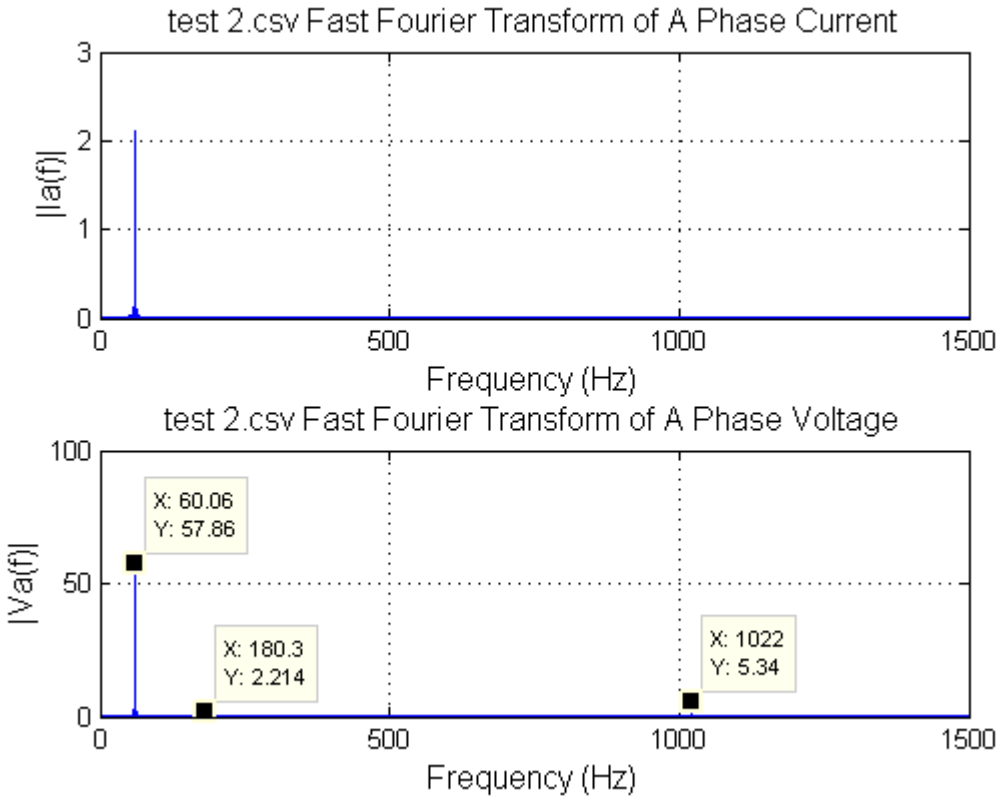


**Figure 40: Test 2 voltage and current FFT**

The Butterworth filter designed earlier and MATLAB's filtfilt function was used to filter the data collected from the test is shown in Figure 41.

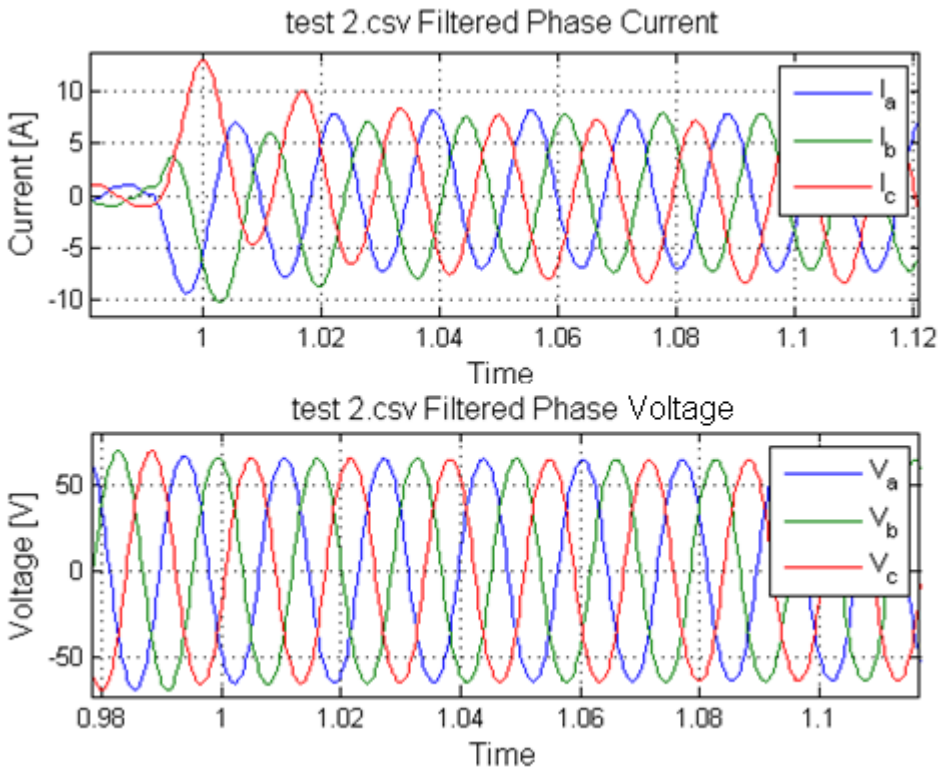


**Figure 41: Test 2 filtered output data**

The dynamics of the measured current and voltage waveforms are shown in Figure 41. Both clearly show the higher frequencies removed. The dq0 representation of the unfiltered voltages and currents for test 2 are shown in Figure 42.

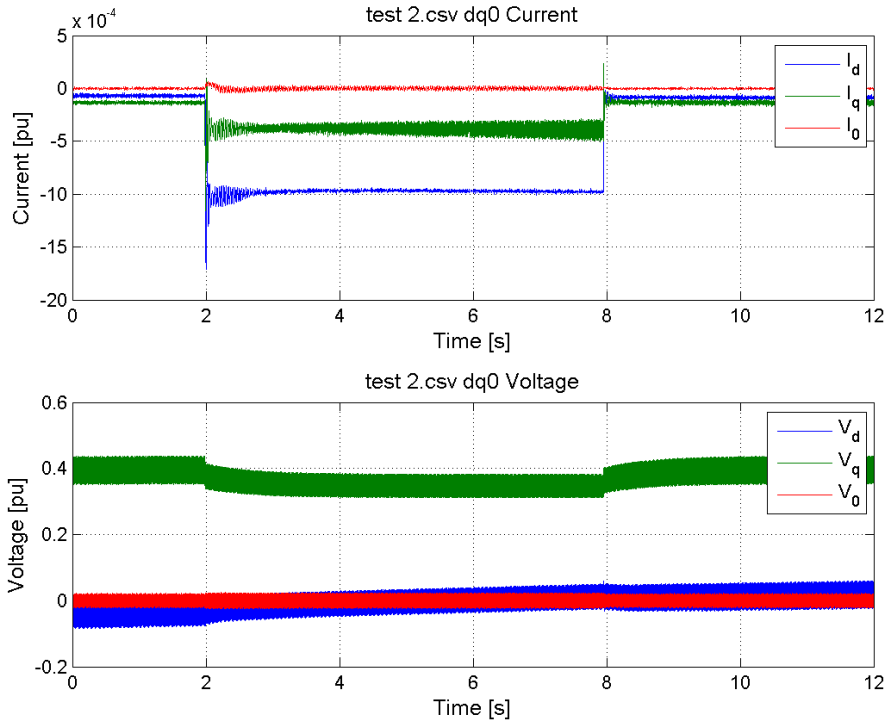


**Figure 42: Test 2 dq0 representation of phase voltages and currents (raw data)**

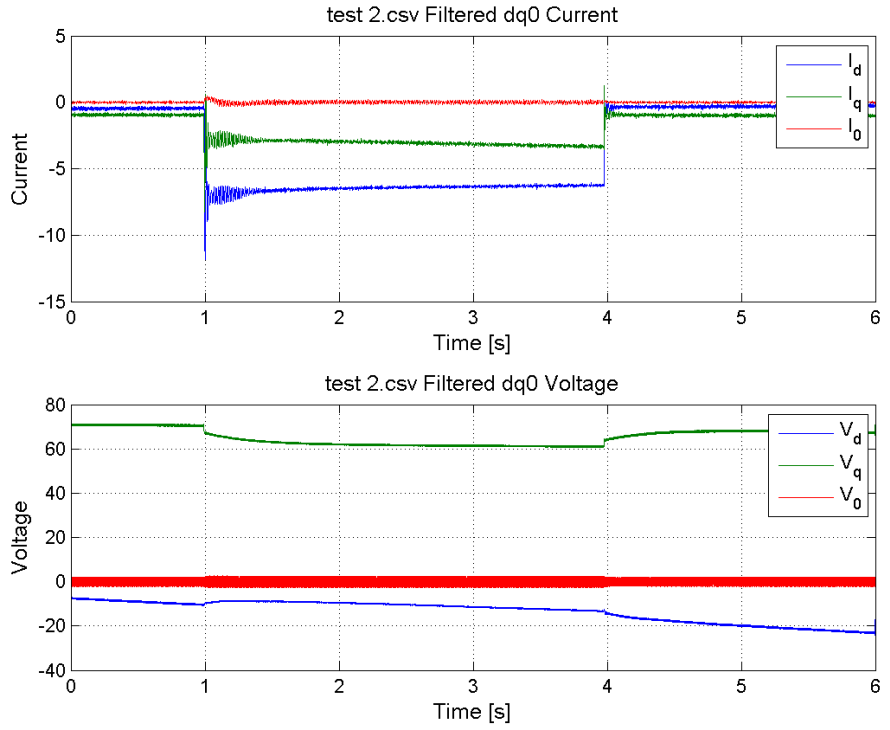


**Figure 43: Test 2 dq0 representation of filtered phase voltages and currents (filtered data)**

The filtered data shows the trending d and q-axis currents and voltages for test 2. Following the same approach for calculating reactances described in section 5.4.1, a summary of the values calculated for test 2 are shown in Table 9.

**Table 9: Summary of Test 2 Results**

| **Parameter** | **Result:** | **Description:** |
|---|---|---|
| $X_d$ | 5.34pu | D-axis steady-state reactance |
| $X'_d$ | 3.72pu | D-axis transient reactance |
| $X''_d$ | 3.48pu | D-axis subtransient reactance |
| $T'_d$ | 1.75 s | D-axis transient reactance |
| $T''_d$ | 0.078 s | D-axis subtransient reactance |

### 5.4.3 Test 3

The testing procedures were repeated with increased excitation, continuing the sequence from Table 6. The oscillography from the event report generated by the SEL 411L relay for test 3 is shown in Figure 44. The data represents the 3 phase currents and voltages as measured at the terminals of the machine.

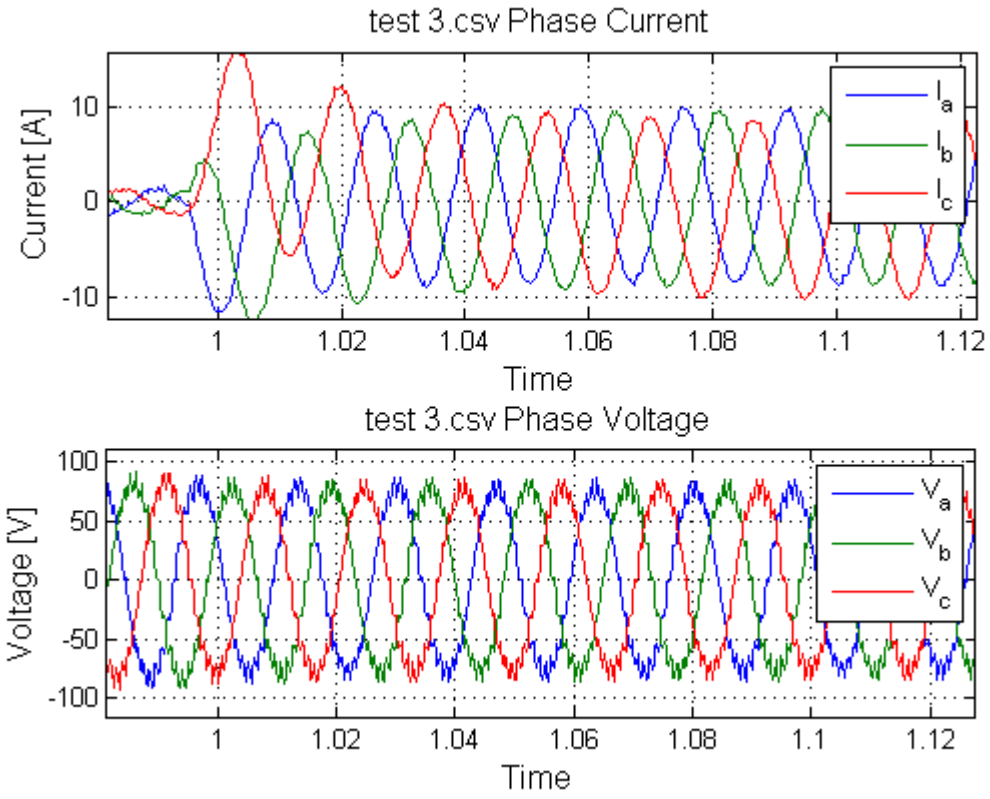


**Figure 44: Test 3 Oscillography of raw data captured**

A higher frequency is again riding on top of the 60Hz signal. This higher frequency is again especially prevalent in the voltage waveform. A FFT of the dataset above is shown in Figure 45. In doing so, the higher order frequencies observed at the machine terminals can be identified. Note that the third harmonic and 17$^{th}$ slot harmonic continue to increase as well.

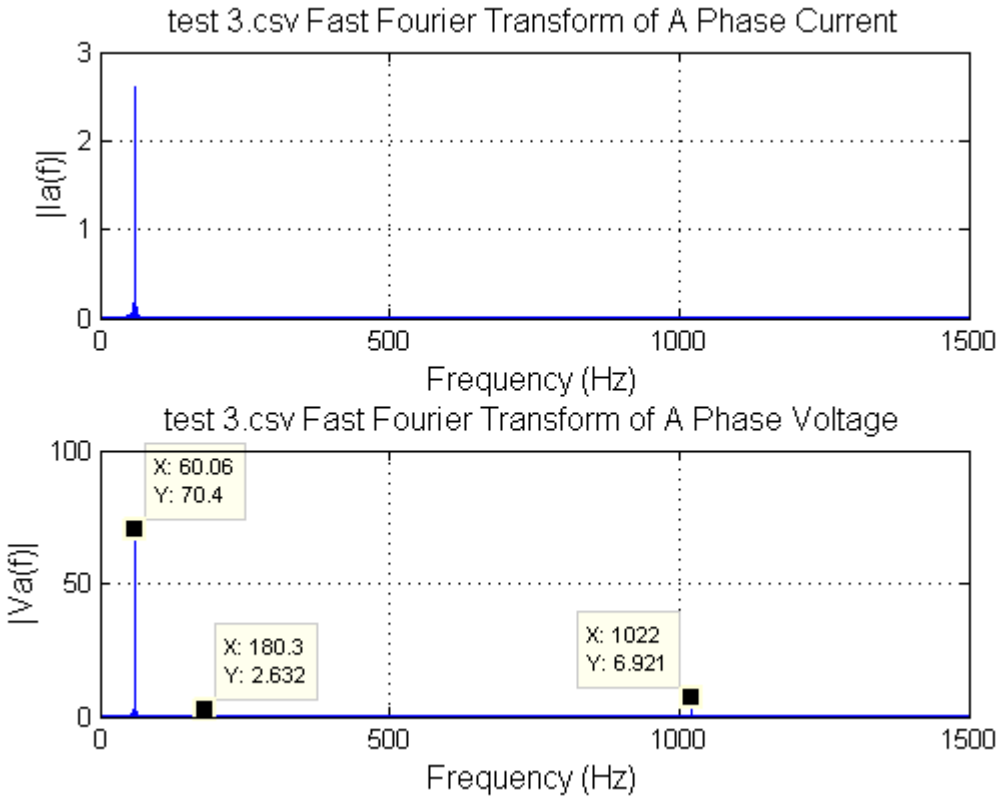


**Figure 45: Test 3 voltage and current FFT**

Results of applying the applied Butterworth filter and MATLAB's filtfilt function to filter the data collected from the test are shown in Figure 46.

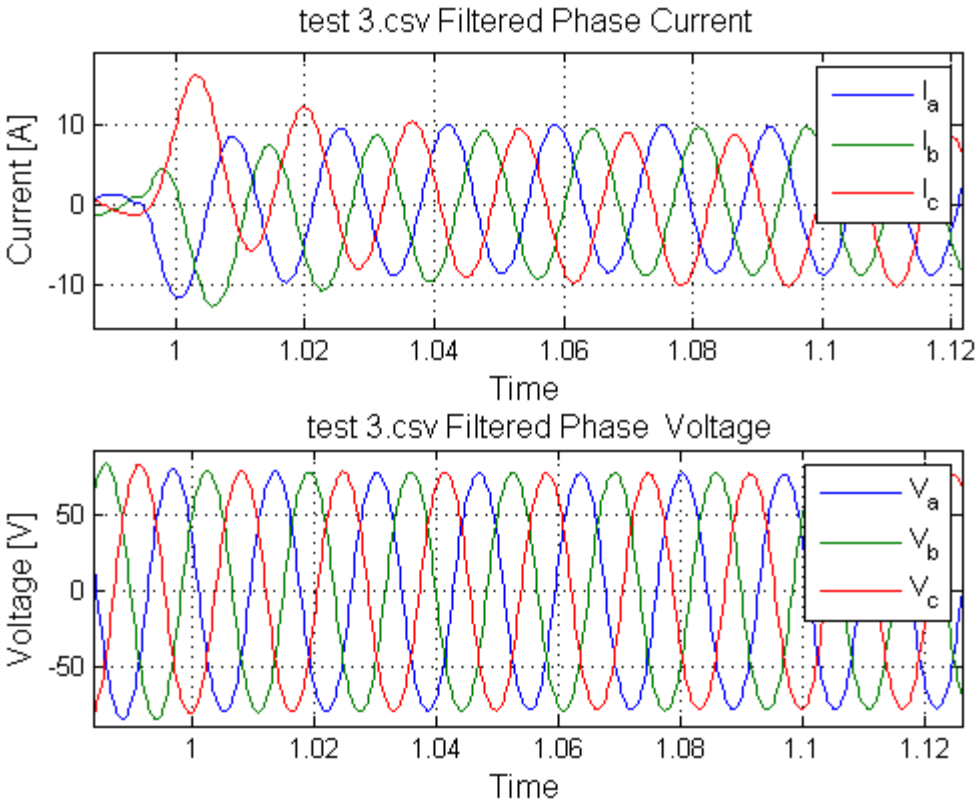


**Figure 46: Test 3 filtered output data**

The dq0 representation of the unfiltered voltages and currents for test 3 are shown in Figure 47.

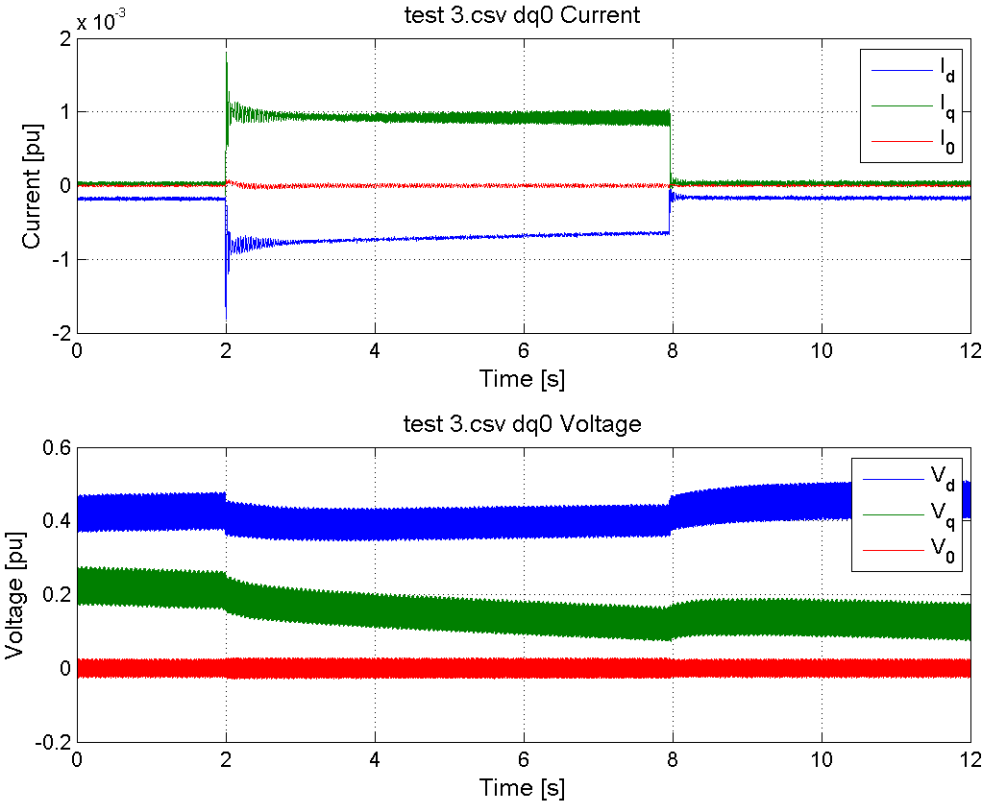


**Figure 47: Test 3 dq0 representation of phase voltages and currents (raw data)**

The dq0 representation of the filtered voltages and currents for test 3 are shown in Figure 48.

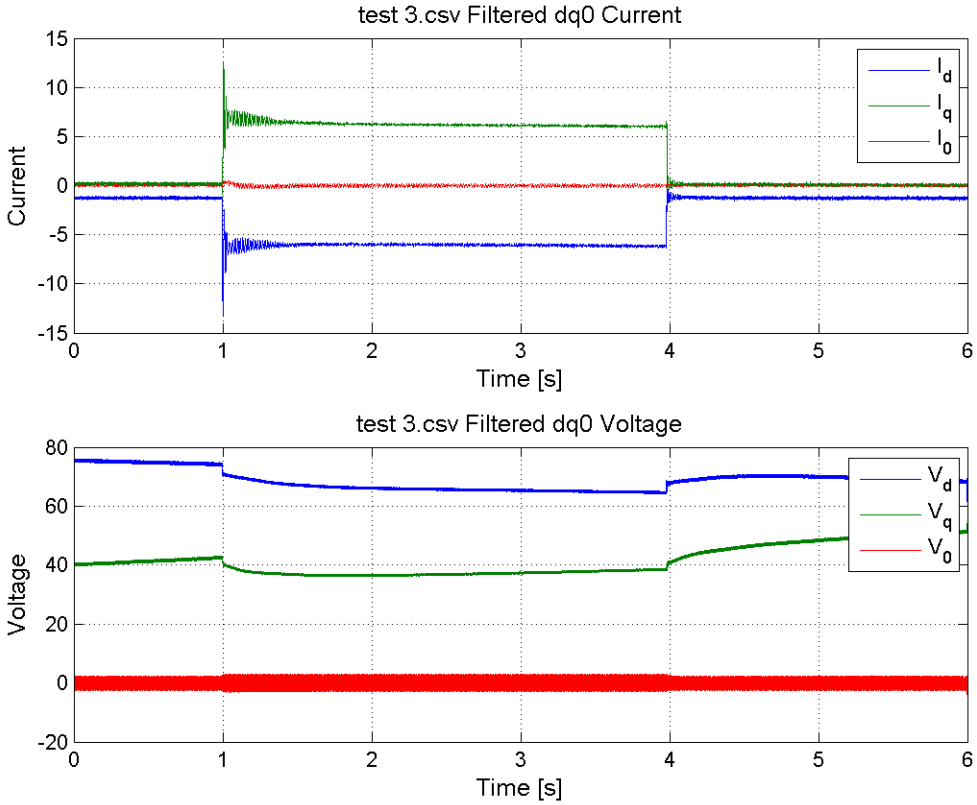


**Figure 48: Test 3 dq0 representation of filtered phase voltages and currents (filtered data)**

This filtered data shows the trending d and q-axis currents and voltages for test 3. Again, a summary of the values calculated for test 3 are shown in Table 10.

**Table 10: Summary of Test 3 Results**

| **Parameter** | **Result:** | **Description:** |
|---|---|---|
| $X_d$ | 5.31pu | D-axis steady-state reactance |
| $X'_d$ | 3.68pu | D-axis transient reactance |
| $X''_d$ | 3.46pu | D-axis subtransient reactance |
| $T'_d$ | 1.77 s | D-axis transient reactance |
| $T''_d$ | 0.034 s | D-axis subtransient reactance |

### 5.4.4 Test 4

This test is repeated with even higher excitation, in the fourth case described in Table 6. Oscillography of the event report generated by the SEL 411L relay for test 4 can be seen in Figure 49. The data represents the 3 phase currents and voltages as measured at the terminals of the machine.

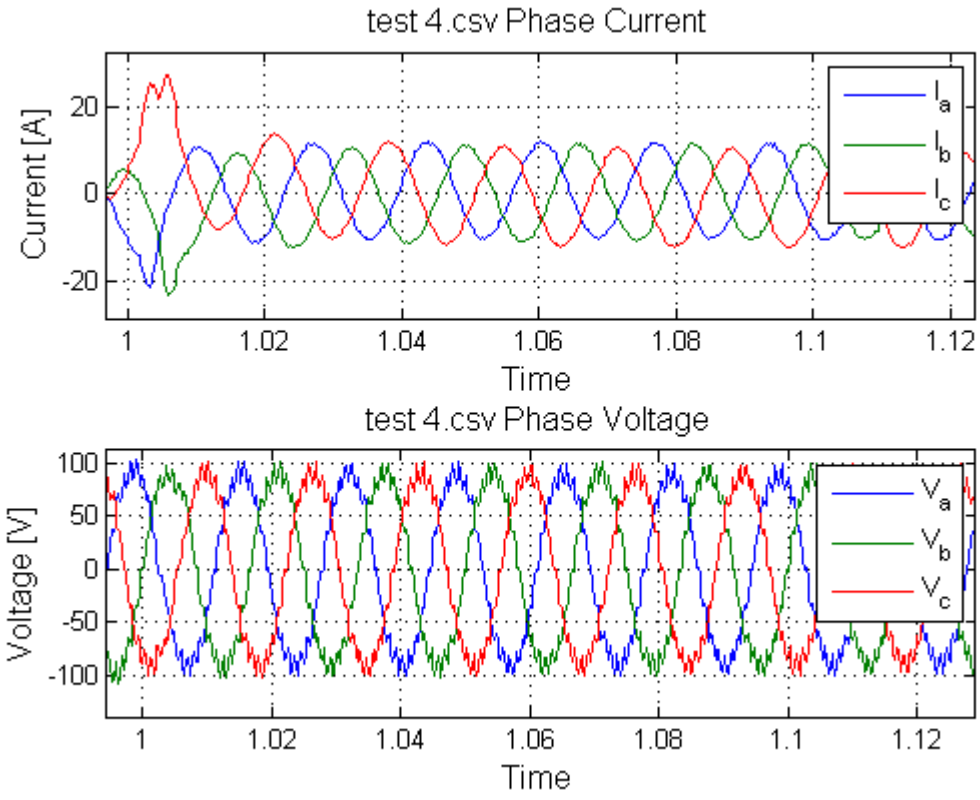


**Figure 49: Test 4 Oscillography of raw data captured**

The current and voltage waveforms are now beginning to show the effects of saturation as measured signals become more distorted. During test 4, the terminal line-to-line voltage is set to 175 V. This is right at the cusp of when saturation appears to start affecting the machine as was seen in Figure 27. A (FFT) of the dataset above can be seen in Figure 50. In doing so, the higher order frequencies observed at the machine terminals can be identified.

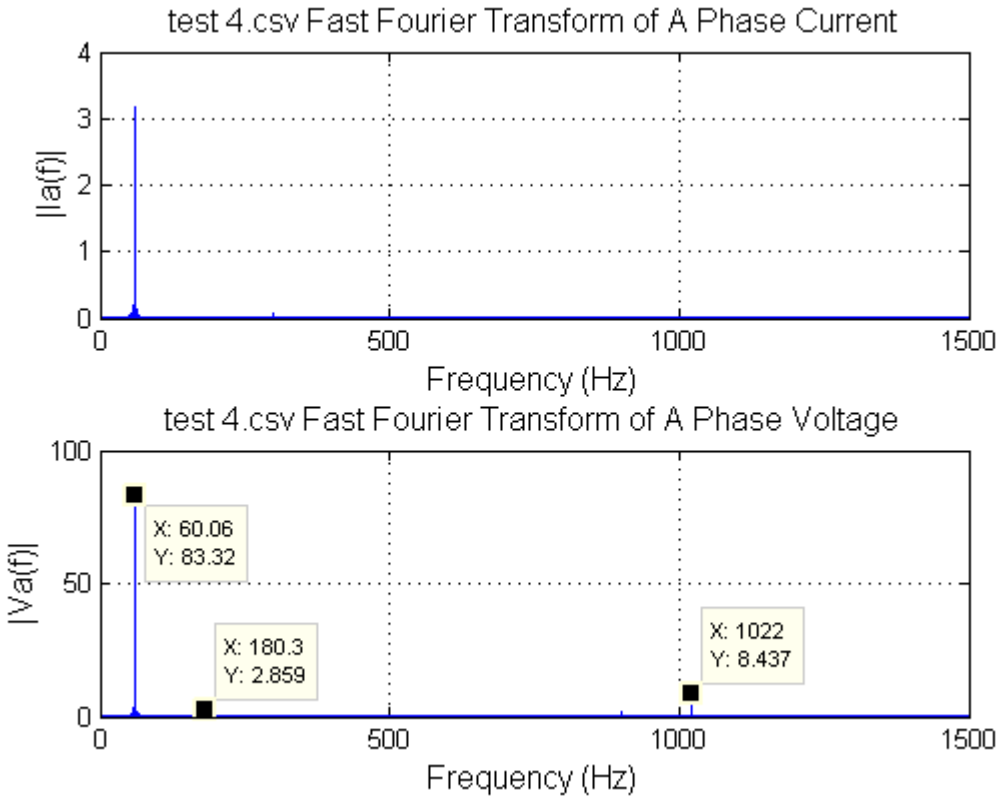


**Figure 50: Test 4 voltage and current FFT**

Filtering the data, the waveform with the higher frequencies clearly removed is shown in Figure 51. Again, this filtering provides a clear capture of the 60Hz signal and the machine's dynamic response to this three-phase bolted short circuit on the transformer terminals. A transient current peak also shows distortion as the pre-fault conditions are changed.

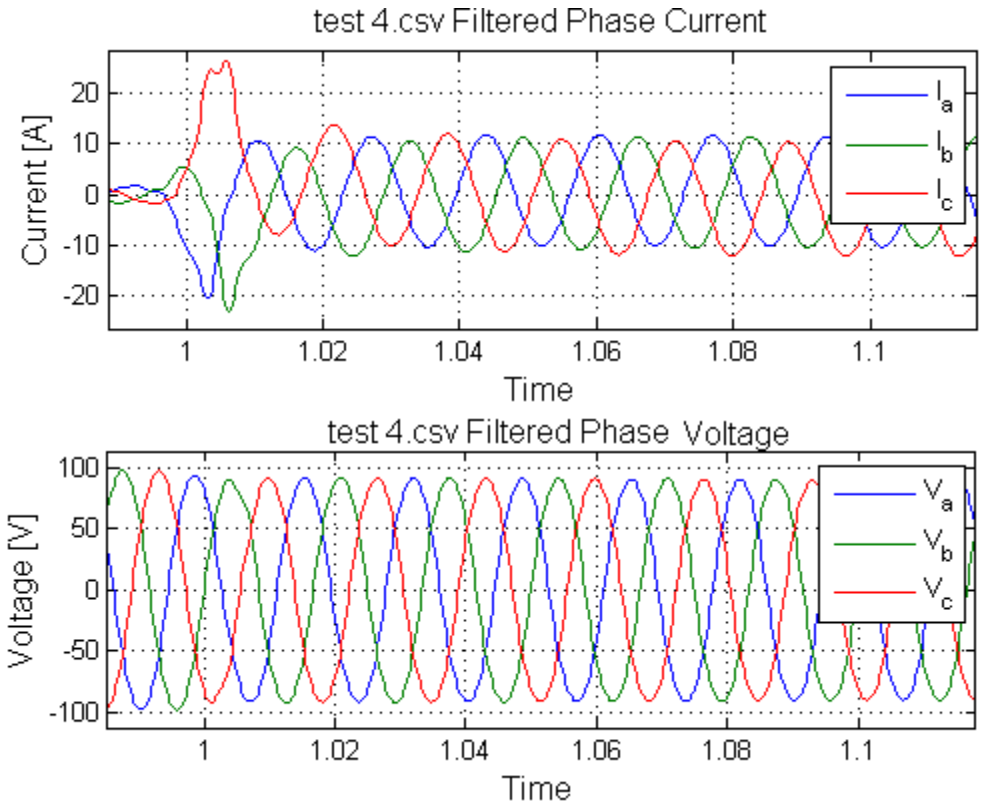


**Figure 51: Test 4 filtered output data**

The dq0 representation of the unfiltered voltages and currents for test 4 is shown in Figure 52 with the filtered dq0 representation in Figure 53.

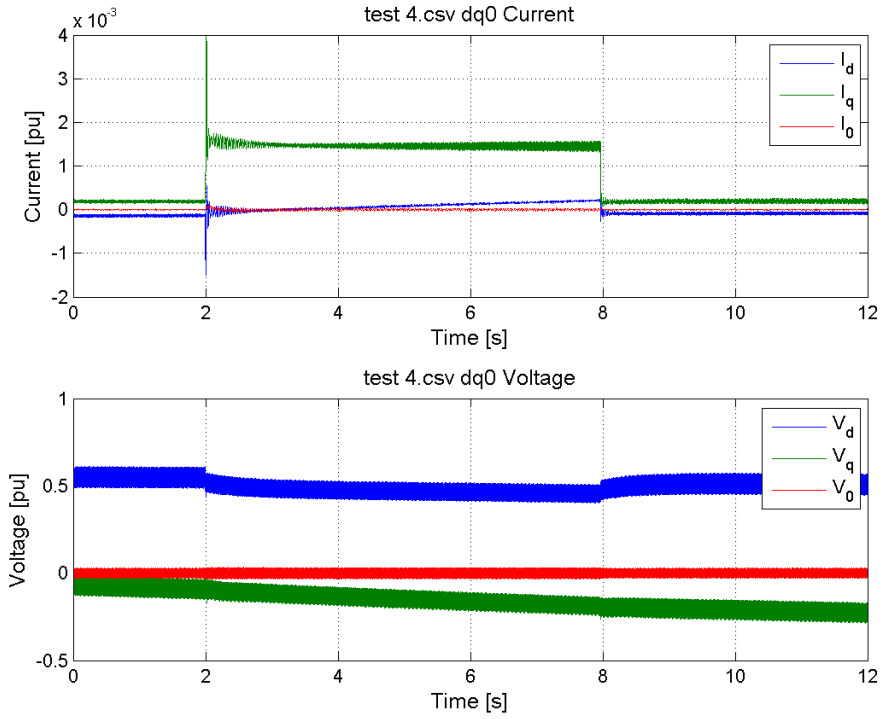


**Figure 52: Test 4 dq0 representation of phase voltages and currents (raw data)**

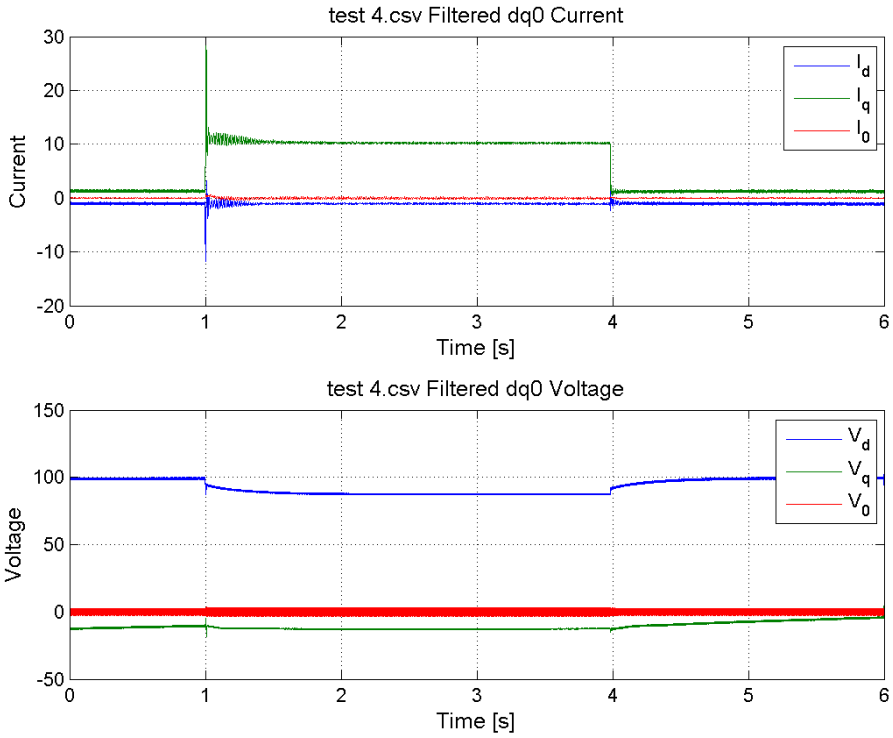


**Figure 53: Test 4 dq0 representation of filtered phase voltages and currents (filtered data)**

This filtered data shows the trending d and q-axis currents and voltages for test 4. Again, a summary of the values calculated for test 4 are shown in Table 10.

**Table 11: Summary of Test 4 Results**

| **Parameter** | **Result:** | **Description:** |
|---|---|---|
| $X_d$ | 5.21pu | D-axis steady-state reactance |
| $X'_d$ | 3.64pu | D-axis transient reactance |
| $X''_d$ | 3.40pu | D-axis subtransient reactance |
| $T'_d$ | 1.74 s | D-axis transient reactance |
| $T''_d$ | 0.086 s | D-axis subtransient reactance |

## 5.5 Conclusion

In conclusion, several offline tests were conducted including the open circuit test, the short circuit test, and several different 3-phase bolted faults. Using these tests, parameters were obtained for modeling the direct axis of the machine. These parameters include subtransient, transient, and steady-state reactances, as well as subtransient and transient time constants. While averaging across tests is not necessarily suggested in the standard, tests were conducted at different voltages to investigate how much the parameters differ at different levels of excitation. Ideally, these values would not change. Due to the slight variations in results during each test, averaging is appropriate to tie the tests together with a final value for each parameter and is shown in Table 12.

**Table 12: Summary of direct axis reactances and time constants**

| Parameter | Test 1 | Test 2 | Test 3 | Test 4 | Average |
|---|---|---|---|---|---|
| $X_d$ | 5.45pu | 5.34pu | 5.31pu | 5.21pu | 5.3275pu |
| $X'_d$ | 3.75pu | 3.72pu | 3.68pu | 3.64pu | 3.6975pu |
| $X''_d$ | 3.53pu | 3.48pu | 3.46pu | 3.40pu | 3.4675pu |
| $T'_d$ | 1.80 s | 1.75 s | 1.77 s | 1.74 s | 1.7650s |
| $T''_d$ | 0.026 s | 0.078 s | 0.034 s | 0.086 s | 0.0560s |

Table 13 shows several other measurements taken on the machine via offline testing. The stator and field resistances shown were measured by applying a current into the terminals of the machine, and measuring the resulting voltage. Using Ohm's law, the DC resistance values were calculated with the ratio of the voltage and current. The machine turns ratio (Ns/Nf) was calculated using Eq. (26).

**Table 13: Machine Parameters Determined from Offline Testing**

| Steady-State Parameters | |
|---|---|
| **Parameters** | |
| Ns/Nf | 0.0870 |
| $r_s$ | 0.1028 (Ω) |
| $r_{fd}$ | 16.9771 (Ω) |

# 6. Online Characterization

The online characterization method selected for this research is the Least Squares Approximation method. This method was selected to further study the feasibility and limitations for applying the algorithm to synchronous machines.

## 6.1 Least Squares Approximation Algorithm Derivation

The least squares approximation is an iterative approach for calculating a polynomial curve fit. The fit is determined by iterating the coefficients of a set $n^{th}$ degree polynomials, minimizing the true error between the actual data and the polynomial curve fit. True error can be calculated using vector norms or other similar measures. In this thesis the error is calculated by summing the square of the difference between each point in a variation on the Euclidean norm. This is shown in Eq. (89).

$$\eta = \sum_{k=1}^{K} \left[ y_{sys}(\alpha) - y_{sim}(\alpha) \right]^2 \tag{89}$$

Where $y_{sys}(\alpha)$ is a vector representing the discrete output of the given system (in this case, the synchronous machine), $\alpha$ is a vector of the parameters to be determined for the current iteration, and $y_{sim}(\alpha)$ is the output of the current calculated polynomial curve fit for the current iteration. Once the error has converged to an acceptable value, the polynomial is said to fit.

If needed, a weighting factor can be used to emphasize more sensitive parameters. For example, the d- and q-axis flux linkages have more of an effect on the output of the system when viewed in comparison to the mutual flux linkages of the damper windings. To use this weighting matrix, the error equation can then be re-written as shown in Eq. (90).

$$\eta = \int_0^T \left[ y_{sys}(\alpha) - y_{sim}(\alpha) \right]^T * W * \left[ y_{sys}(\alpha) - y_{sim}(\alpha) \right] dt \tag{90}$$

where *W* is a diagonal weighting matrix selected based on engineering judgment. However, in this particular implementation, the weighting matrix was selected to be an identity matrix. This is because a second weighting matrix (which will be discussed later) will be used when the required change in parameters is calculated.

In order to converge to a global minimum, the derivative of $y_{sim}(\alpha)$ with respect to each value of $\alpha$ is required and set equal to 0. For the case of this synchronous machine, the input and output equations of the state space matrix can be expanded according to their Taylor series expansion:

$$y_{sim}(\alpha) = y_{sim}(\alpha_0) + \left[\frac{\partial y_{sim}(\alpha)}{\partial \alpha}\right]\Bigg|_{\alpha_0} (\alpha - \alpha_0) + \ldots \text{higher order terms} \tag{91}$$

Where $\alpha_0$ denotes the initial guess for the machine parameters. The second derivative terms and higher order terms are ignored due to their small influence and significant increased complexity in the mathematics. By using the chain rule for derivatives, the derivative of the simulation output $y_{sim}(\alpha)$ can be calculated as follows:

$$\left[\frac{\partial y_{sim}(\alpha)}{\partial \alpha}\right]\Bigg|_{\alpha_0} = C(\alpha_0)\left[\frac{\partial x(\alpha)}{\partial \alpha}\right]\Bigg|_{\alpha_0} + \left[\frac{\partial C(\alpha)}{\partial \alpha}\right]\Bigg|_{\alpha_0} x(\alpha_0) \tag{92}$$

Substituting (92) into (91), the simulation output can be stated as:

$$y_{sim}(\alpha) = y_{sim}(\alpha_0) + \left[C(\alpha_0)\left[\frac{\partial x(\alpha)}{\partial \alpha}\right]\Bigg|_{\alpha_0} + \left[\frac{\partial C(\alpha)}{\partial \alpha}\right]\Bigg|_{\alpha_0} x(\alpha_0)\right](\alpha - \alpha_0) \tag{93}$$

where

$$\frac{\left[\frac{\partial x(\alpha)}{\partial \alpha}\right]\Big|_{\alpha_0}}{dt} = A(\alpha_0)\left[\frac{\partial x(\alpha)}{\partial \alpha}\right]\Bigg|_{\alpha_0} + \left[\frac{\partial A(\alpha)}{\partial \alpha}\right]\Bigg|_{\alpha_0} x(\alpha_0) + \left[\frac{\partial B(\alpha)}{\partial \alpha}\right]\Bigg|_{\alpha_0} u \tag{94}$$

Plugging (93) back into (90), the equation may be re-written as:

$$\eta = \int_0^T \left[ y_{sys}(\alpha) - y_{sim}(\alpha_0) - \left[ \frac{\partial y_{sim}(\alpha)}{\partial \alpha} \right]_{\alpha_0} (\alpha - \alpha_0) \right]^T * W * \left[ y_{sys}(\alpha) - y_{sim}(\alpha_0) - \left[ \frac{\partial y_{sim}(\alpha)}{\partial \alpha} \right]_{\alpha_0} (\alpha - \alpha_0) \right] dt \tag{95}$$

In order for this equation to have successfully converged, according to [9], the following must be true:

$$\frac{\partial \eta}{\partial (\alpha - \alpha_0)} = 0 \tag{96}$$

Let

$$\Delta\alpha = (\alpha - \alpha_0) \tag{97}$$

solving for $(\alpha - \alpha_0)$, then

$$\int_0^T \left[ \left[ \frac{\partial y_{sim}(\alpha)}{\partial \alpha} \right]_{\alpha_0}^T W \left( \frac{\partial y_{sim}(\alpha)}{\partial \alpha} \right) \right] dt \Delta\alpha = \int_0^T \left[ \left[ \frac{\partial y_{sim}(\alpha)}{\partial \alpha} \right]_{\alpha_0}^T W \left[ y_{sys}(\alpha) - y_{sim}(\alpha_0) \right] \right] dt \tag{98}$$

Solving for $\Delta\alpha$:

$$\Delta\alpha = \frac{\int_0^T \left[ \left[ \frac{\partial y_{sim}(\alpha)}{\partial \alpha} \right]_{\alpha_0}^T W \left[ y_{sys}(\alpha) - y_{sim}(\alpha_0) \right] \right] dt}{\int_0^T \left[ \left[ \frac{\partial y_{sim}(\alpha)}{\partial \alpha} \right]_{\alpha_0}^T W \left( \frac{\partial y_{sim}(\alpha)}{\partial \alpha} \right) \right] dt \Delta\alpha} \tag{98}$$

After $\Delta\alpha$ is calculated, the parameters are updated with equation (99), thus creating an iterative approach for solving for the machine parameters.

$$\alpha_{k+1} = \alpha_k + G\Delta\alpha_k \tag{99}$$

After the parameters are updated, the process of modeling the machine in MATLAB through equations (67)-(74), and equations (89)-(99) are repeated until convergence.

In Eq. (99), $G$ is the gain vector used to weight the change in parameters. According to [9], the optimal gain is calculated based on the following criteria.

$$\begin{aligned} G &= 1 \qquad \text{if} \left| \frac{\Delta\alpha_k}{\alpha_k} \right| < 0.25 \\ G &= 0.25 \quad \text{if} \left| \frac{\Delta\alpha_k}{\alpha_k} \right| > 0.25 \end{aligned} \tag{100}$$

When adjusting the gains around different values, this appeared to be a good starting point. However, a modified, more dynamic gain will be discussed later on. The modified dynamic gain adds additional criteria to converge on more accurate machine parameters. The LSE algorithm shown in this thesis operates on a state-space representation, thus the motivation behind development of the 3rd and 7th order models described in Chapter 4, with flux linkages selected as the states [9]. A flow diagram of the LSE algorithm can be seen in Figure 54.

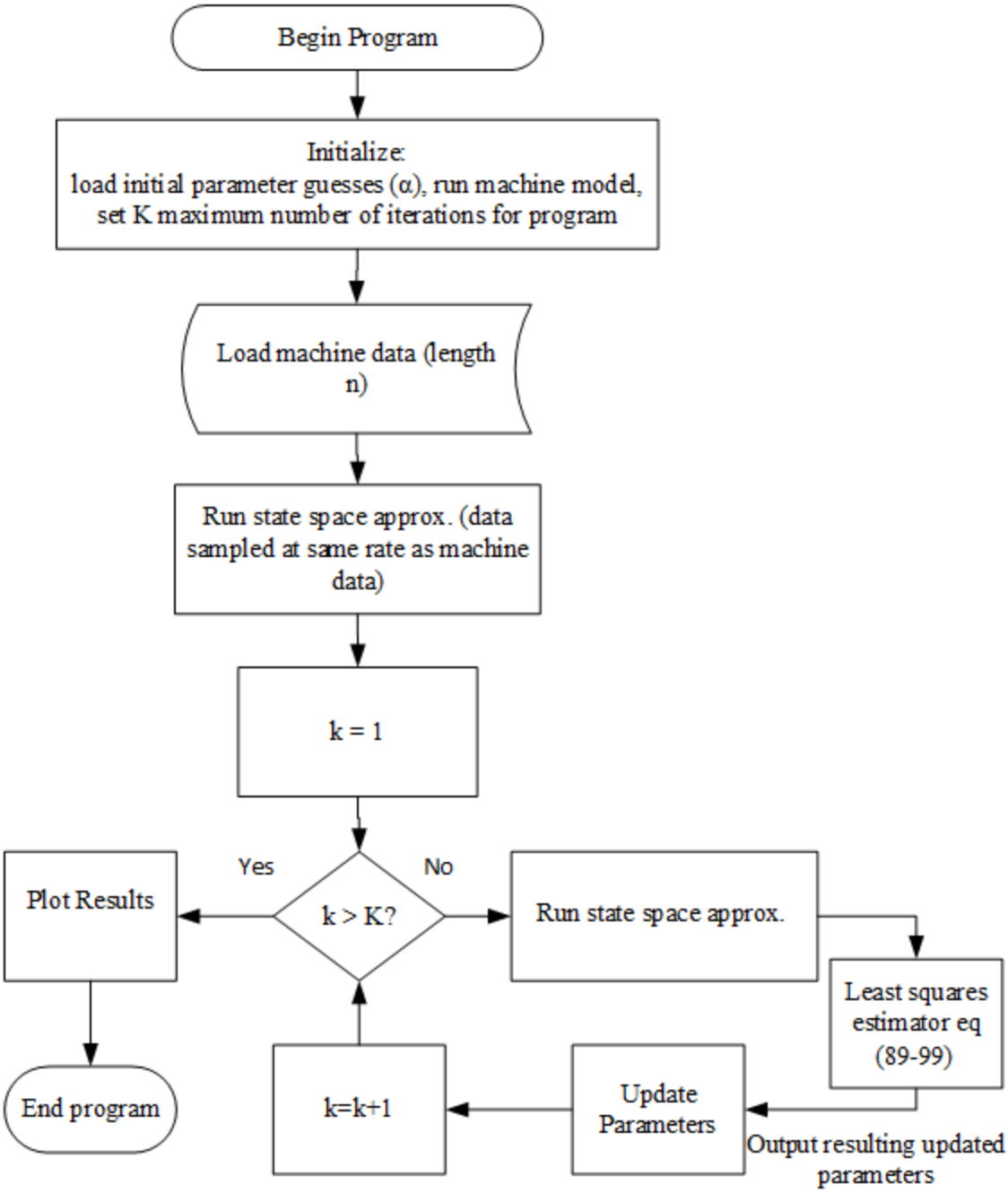


**Figure 54: Least Squares Estimation Algorithm Flow Diagram**

### 6.2 Least Squares Approximation Algorithm Results

In order to test this algorithm, the first step is to implement it on the 3rd order machine model. Using the same data collected earlier for testing the state-space model proposed in Chapter 4 against the

built-in SIMULINK machine model, the algorithm is implemented in an attempt to calculate $L_{mq}, L_{md}, \text{and} L_{lk}$.

#### 6.2.1.1 3rd Order Model

To evaluate the LSE performance on the 3rd order machine model, tests are conducted by incrementing initial guesses for parameters farther and farther from the generator parameters entered in the SIMULINK model. Before investigating the algorithm with initial guess errors, a test case was ran with all values remaining constant. In other words, the algorithm assumed itself to be converged to an appropriate solution. To begin testing the limits of the LSE, the LSE is run with all parameters set to 120% of the actual values (in this case, the "real" machine is the SIMULINK model). Results of this parameter estimation through 15 iterations can be seen in Figure 55. Table 14 shows the real values, the initial "guessed" parameters, the resulting parameters after 15 iterations, and the associated error with +20% error used on initial parameter guesses.

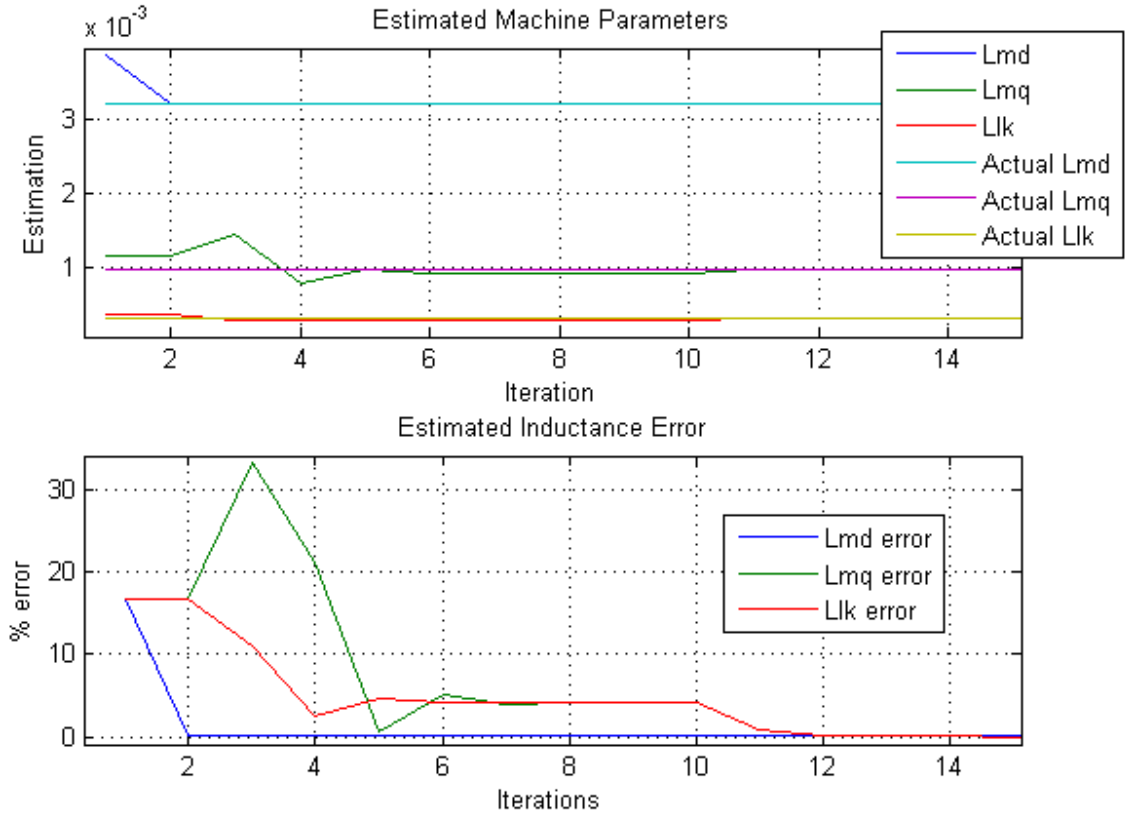


**Figure 55: Least squares estimation results with +20% initial guess error**

Lmd converges on the correct value almost immediately as shown in Figure 55. This is expected as Lmd is one of the easier parameters to identify. Note that the Lmq and Llk values both converge to parameters that are close, but not quite right when compared with the known values. This was observed in several initial test cases, and a dynamic gain was introduced into the algorithm to address this, as will be discussed. Initially, the gain is set as described in Chapter 5. If the parameters converge for more than 2 iterations, but an error of more than 5% exists between the MATLAB calculated currents and "measured" real machine currents the gains were then slightly adjusted to "kick" the algorithm out of its converged local minima to try to reach a global minimum. This causes the algorithm to continue searching for a new value for the estimated parameters.

Both Lmq and Llk converged on a set of values that are not quite correct after 7 iterations. From there, the gains are dynamically adjusted to 0.20, and the values successfully converged.

**Table 14: LSE Results starting with +20% Error in "Guessed" Parameters – 3rd order model**

| Parameter | $L_{md}$ | $L_{mq}$ | $L_{lk}$ | $L_d$ | $L_q$ |
|---|---|---|---|---|---|
| Initial Guess (α) | 3.8596E-03 | 11.6583E-04 | 3.7070E-04 | 4.2303E-03 | 1.5364E-03 |
| Actual Values (H) | 3.2164E-03 | 9.7153E-04 | 3.0892E-04 | 3.5253E-03 | 1.2804E-03 |
| Estimated Values | 3.215E-03 | 9.715E-04 | 3.0901E-04 | 3.5240E-03 | 1.2805E-03 |
| Error (%) | 0.023 | E-14 | 3.1986 | 0.037 | 0.007 |

The results show excellent convergence if the initial guesses for the parameters are 120% of the actual values. However, in not fully knowing the parameters, there is no guarantee that the initial guesses for all parameters will be within this range. Most likely, many of the parameters will be close, especially if it is a big enough machine to require tests for NERC or WECC.

To test the performance of the estimation algorithm, this process is repeated with initial guesses at 140% of the actual value. Results can be seen in Figure 56.

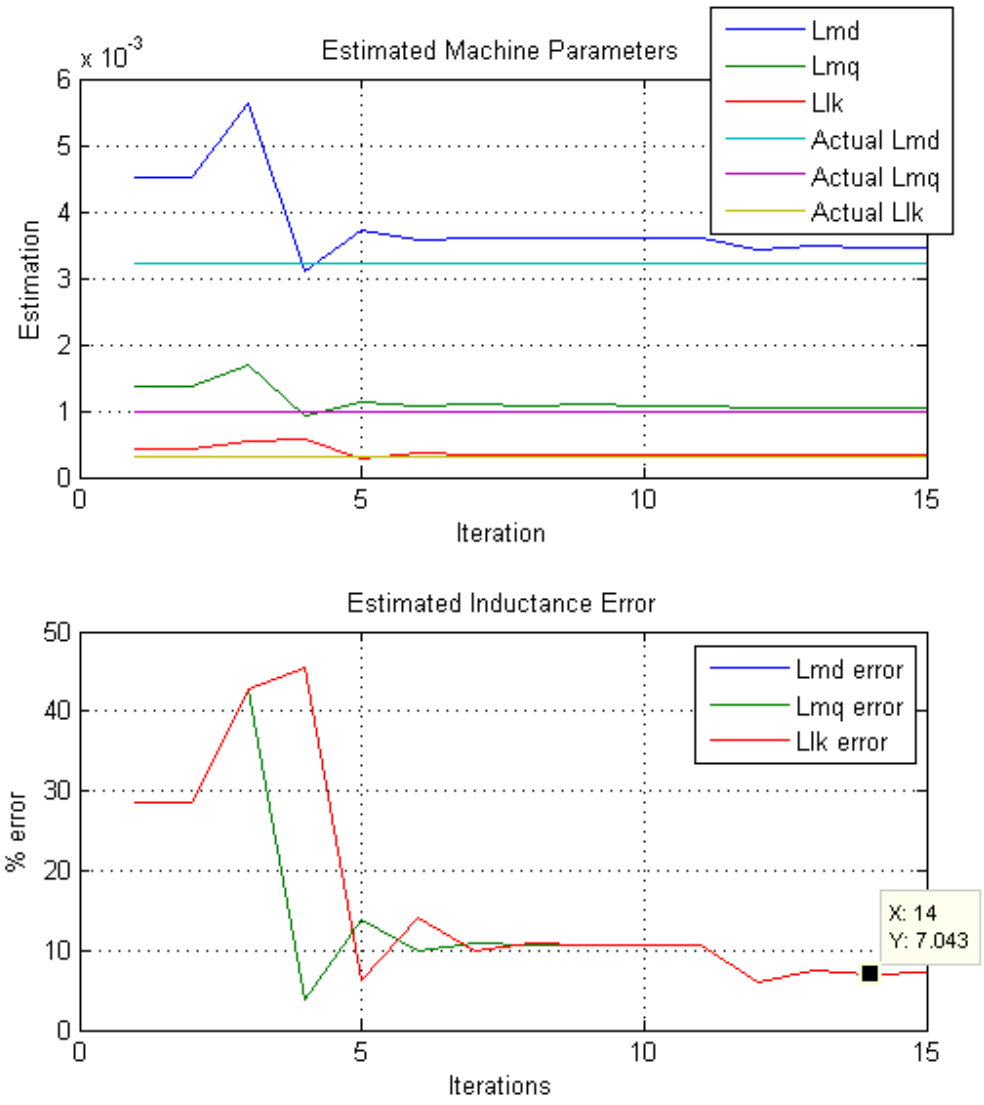


**Figure 56: Least squares estimation results with +40% initial guess error**

Comparing the results shown in Figure 56 with the results shown in Figure 55, the algorithm begins to struggle with approximating the parameters, even after adjusting the gains. This trend continues for initial guesses that move farther and farther away from the actual values.

Table 15 shows a summary of the results for testing the least squares estimation with an initial guess of 140% of the actual parameters. The results show a larger error when compared to Table 14.

**Table 15: LSE Results starting with +40% Error in "Guessed" Parameters – 3rd order model**

| Parameter | $L_{md}$ | $L_{mq}$ | $L_{lk}$ | $L_d$ | $L_q$ |
|---|---|---|---|---|---|
| Initial Guess (α) | 4.503E-03 | 13.6014E-04 | 4.3249E-04 | 4.9354E-03 | 1.7926E-03 |
| Actual Values (H) | 3.2164E-03 | 9.7153E-04 | 3.0892E-04 | 3.5253E-03 | 1.2804E-03 |
| Estimated Values | 3.464E-03 | 10.4660E-04 | 3.3279E-04 | 3.7969E-03 | 1.3794-03 |
| Error (%) | 7.15 | 7.173 | 7.176 | 7.704 | 7.732 |

Now that several cases with initial guesses above the actual parameters have been investigated, a test is conducted with initial guesses set to 80% of the actual values.

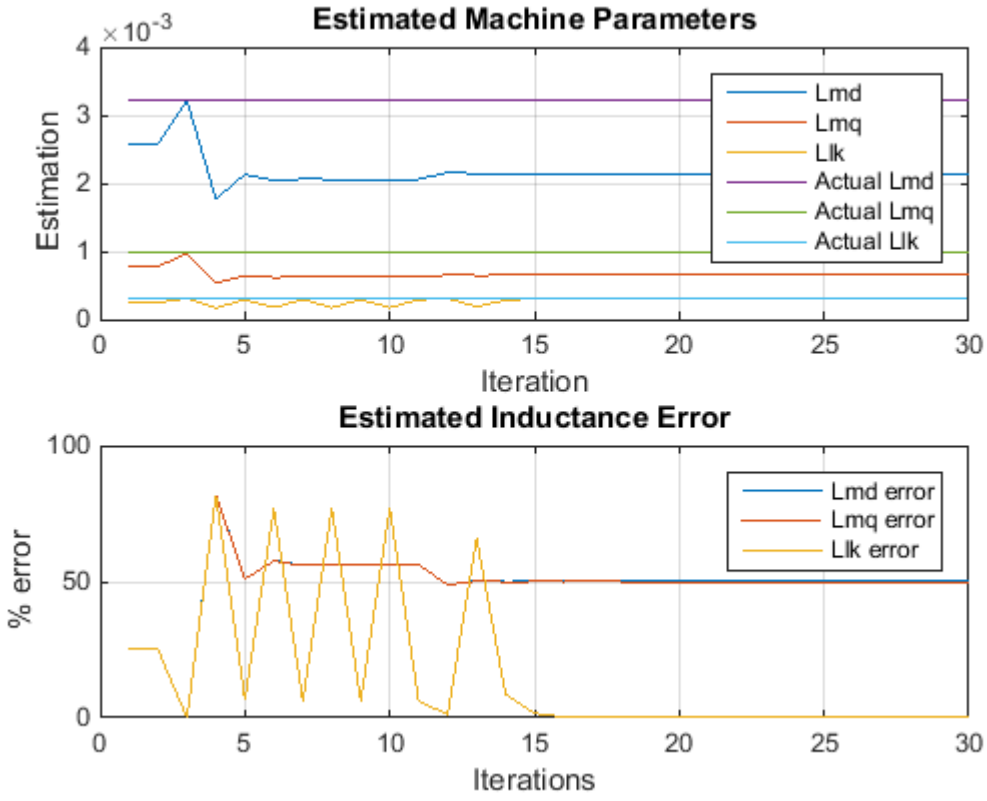


**Figure 57: Least squares estimation results with -20% initial guess error for 3rd order model**

In this particular case, Llk does converge to correct value after oscillating dramatically for 15 iterations. Extending the iterations out and letting the algorithm run longer, the algorithm converges to

a solution. However, both Lmd and Lmq converge to local minima, but not the global minimum. This results in solutions that are worse values than the initial guess. More research needs to be conducted in this algorithm to fully understand this convergence to the wrong values. This error only grows more as the initial guess moves farther and farther away from the actual value and was found to be true for all cases where the initially guessed parameters were under-estimated.

However, some parameters are known through other testing prior to running the parameter estimation, degrees of freedom are removed from the least squares estimation method. For example, if a user would like to estimate 5 parameters, the LSE has the freedom to change all 5 parameters as much as needed until the state-space model matches the real case. In other words, the LSE can compensate for error in one parameter, by adjusting another parameter. However, if some parameters are known to a relatively small error while others have larger errors, then the algorithm can quickly reach convergence for the more accurately known parameters to a minimal error, while continuing to search for convergence on the remaining values.

This poses challenges with a nonlinear system. Parameters such as field resistance and armature resistance can easily be measured. Running a machine in open circuit conditions also allows for easy calculation of the D-axis mutual reactance. During open circuit rated conditions, we know the following to be true as described in Chapter 3.2.

$$V_d = 0, V_q = V_{Line-Neutral}, \mathrm{i}_{\mathrm{ds}} = i_{qs} = 0 \text{, and } \omega_r = \omega_e$$

Because of this, the d-axis mutual inductance can be determined using Eq. (36) and Eq. (43), which results in Eq. (101).

$$L_{md} = \left(\frac{1}{\omega_r}\right)\frac{V_{qs}}{i'_{fd}} = \left(\frac{1}{\omega_r}\right)\frac{3}{2}\frac{N_s}{N_f}\frac{V_{qs}}{i_{fd}} \tag{101}$$

Using Eq. (101) and the open circuit data collected from running the SIMULINK model, Figure 58 shows the accuracy of this calculation, with a final error of 0.02%. The value of Lmd is treated as known in the LSE estimate plot of Figure 59.

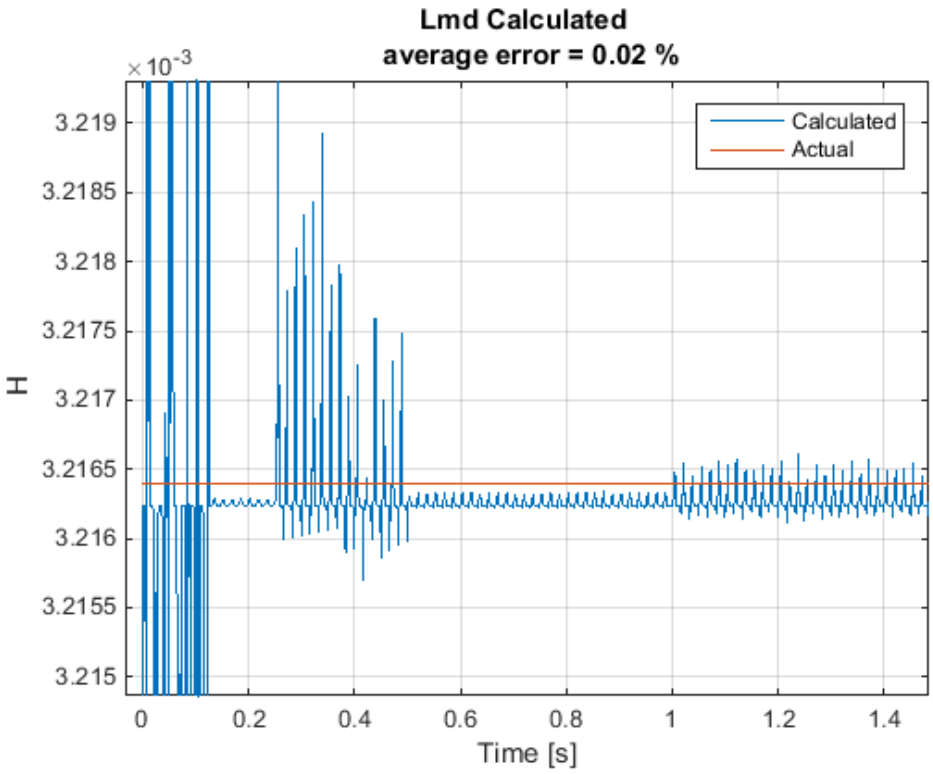


**Figure 58: Lmd Calculated Based on Rated Open-Circuit Conditions**

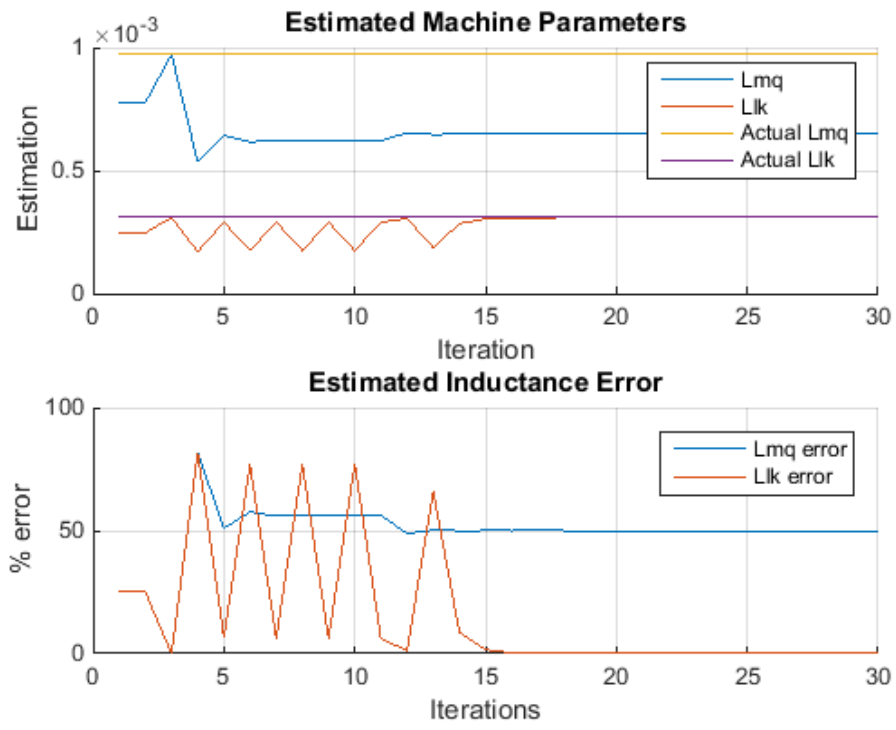


**Figure 59: Least squares estimation results with -20% initial guess error on Lmq and Llk and Lmd known**

The results shown in Figure 59 are not as expected. Reducing the degrees of freedom in this case, does not appear to make any significant differences in the estimations. The algorithm still struggles with identifying Lmq. It does still find Llk, but does not converge any faster than it did when treating Lmd as unknown.

#### 6.2.1.2 Implementing 3rd Order Model on Laboratory Machine

Applying this algorithm under steady-state loaded conditions on the actual laboratory machine, the results of the algorithm are shown in Figure 60. The values never seem to converge. The leakage inductance value does reach a steady-state, but the mutual d and q axis inductances oscillate, never reaching a steady state.

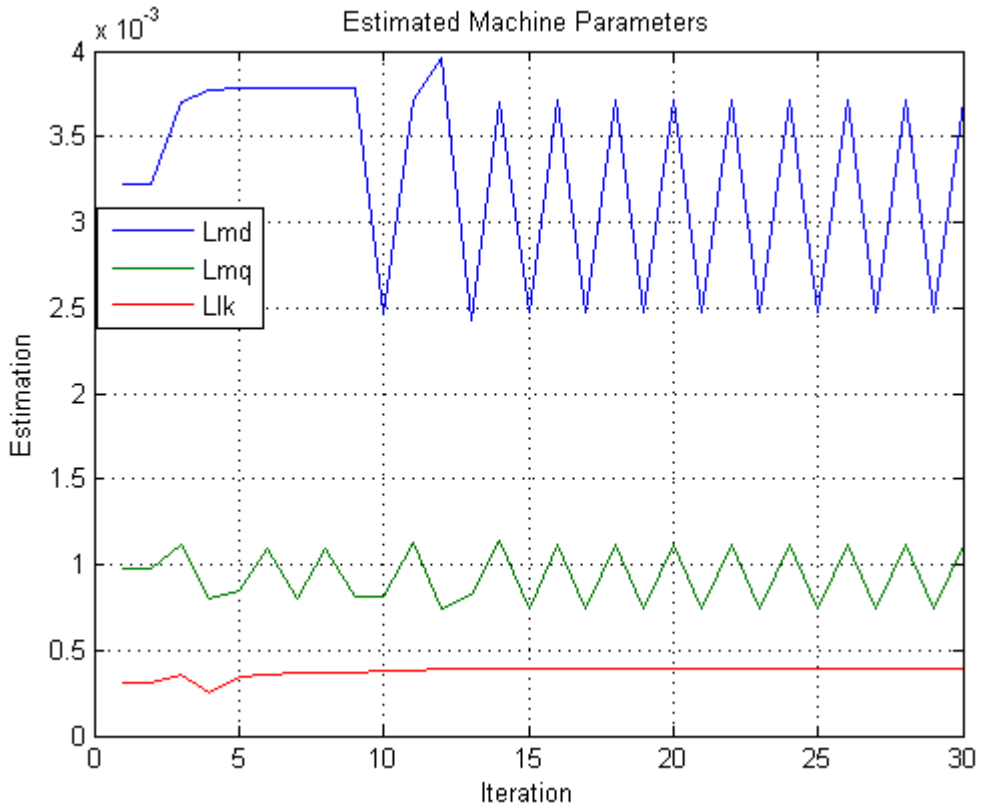


**Figure 60: LSE test on laboratory machine under steady-state, balanced conditions**

The sharp, triangular wave indicates that this is an oscillation between two discrete values. However, after adjusting the gains several times, the algorithm did not converge. It is clear that this algorithm is not fully developed, and will need more work to refine for accuracies.

### 6.2.2 7th Order Model

The same process described in Chapter 6.2.1 is used here to implement the algorithm using the 7th order model. It is expected that the models don't show much difference in behaviors because the estimation is performed under steady-state, balanced conditions. Therefore, the damper windings currents and voltage should be zero due to steady-state, and no zero-sequence voltage or current should exist under balanced conditions. The same parameters that were considered to be unknown in

the 3rd order model are used in the 7th order model, all other parameters were treated as known. Again, the initial guesses are 20% above the actual values.

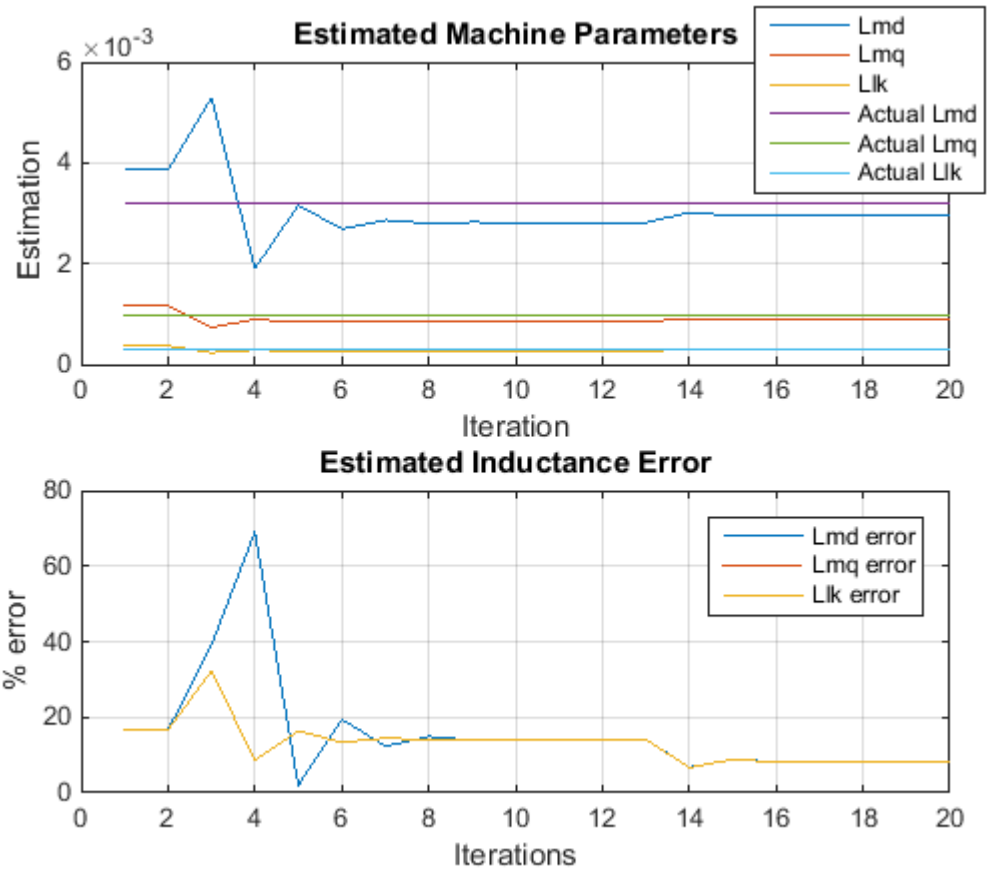


**Figure 61: Least squares estimation results with +20% initial guess error - 7th order model**

Interestingly, comparing this particular result with the results found in Figure 55, the 7th order model does not converge as well as the 3rd order model, due to having a larger number of local optima. This additional error might be attributed to the higher complexity of the machine equations. More equations give room for more degrees of freedom within the least squares estimation. It is important to also note that the gain had to be adjusted to 0.35 from 0.25 in order to obtain stable results. Table 16 shows summary results for this test.

**Table 16: LSE Results starting with +20% Error in "Guessed" Parameters – 7th order model**

| Parameter | $L_{md}$ | $L_{mq}$ | $L_{lk}$ | $L_d$ | $L_q$ |
|---|---|---|---|---|---|
| Initial Guess (α) | 3.8596E-03 | 11.6583E-04 | 3.7070E-04 | 4.2303E-03 | 1.5364E-03 |
| Actual Values (H) | 3.2164E-03 | 9.7153E-04 | 3.0892E-04 | 3.5253E-03 | 1.2804E-03 |
| Estimated Values | 2.9684E-03 | 8.9681E-04 | 2.857E-04 | 3.30E-03 | 1.182E-03 |
| Error (%) | 8.35 | 8.33 | 8.13 | 6.827 | 8.32 |

This process is repeated with initially guessed parameters set to 140% of the actual values. Results can be seen in Figure 62.

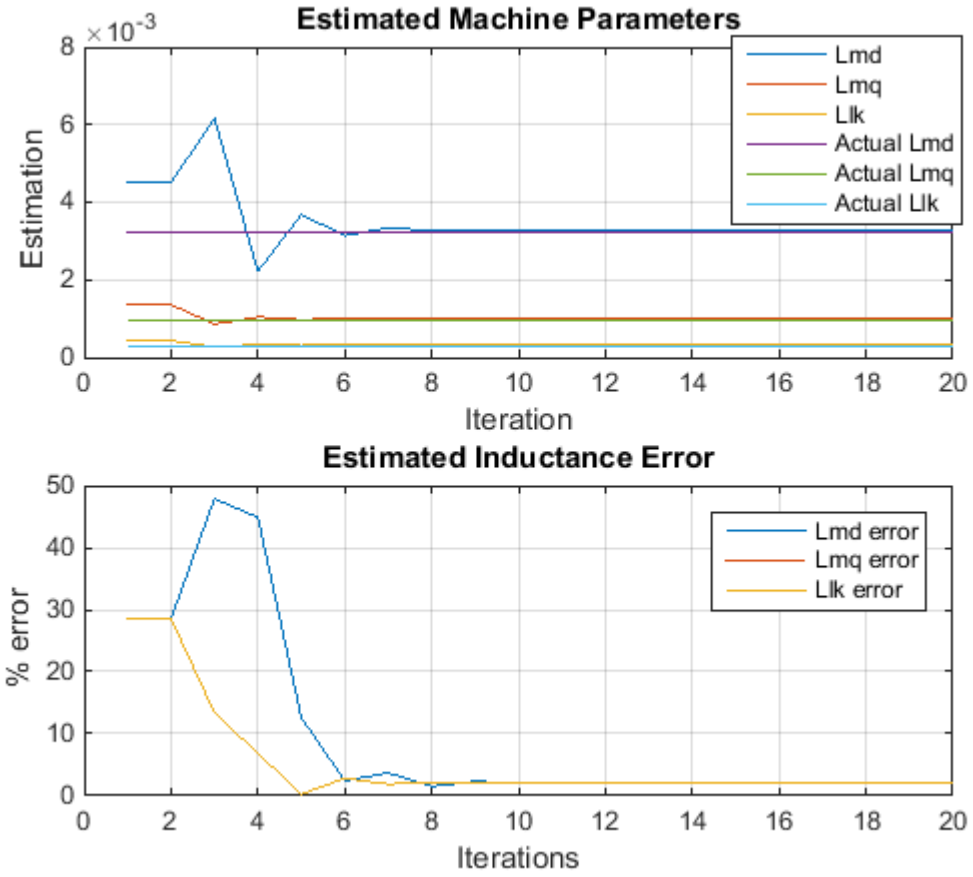


**Figure 62: Least squares estimation results with +40% initial guess error - 7th order model**

Again, a summary table of results for this test can be seen in Table 17.

**Table 17: LSE Results starting with +40% Error in "Guessed" Parameters – 7th order model**

| Parameter | $L_{md}$ | $L_{mq}$ | $L_{lk}$ | $L_d$ | $L_q$ |
|---|---|---|---|---|---|
| Initial Guess (α) | 3.8596E-03 | 11.6583E-04 | 3.7070E-04 | 4.2303E-03 | 1.5364E-03 |
| Actual Values (H) | 3.2164E-03 | 9.7153E-04 | 3.0892E-04 | 3.5253E-03 | 1.2804E-03 |
| Estimated Values | 2.9684E-03 | 8.9681E-04 | 2.857E-04 | 3.30E-03 | 1.182E-03 |
| Error (%) | 8.35 | 8.33 | 8.13 | 6.827 | 8.32 |

The results show a much better estimation in comparison to the +20% error results. This shows that the 7th order model is able to accurately estimate parameters when the initial guess is far away in this case, but appears to be too sensitive when the initial guess is much closer to the actual values. Knowing this, it is desirable to test the limits of the algorithm used in the 7th order model. After several additional tests, it was realized that convergence may continue to be possible as long as the gain is adjusted accordingly.

As an example, Table 18 demonstrates that the algorithm fails to converge with the gain set to 0.35, and the initial guesses off by +60%.

**Table 18: LSE Results starting with +60% Error in "Guessed" Parameters, gain set to 0.35 – 7th order model**

| Parameter | $L_{md}$ | $L_{mq}$ | $L_{lk}$ | $L_d$ | $L_q$ |
|---|---|---|---|---|---|
| Initial Guess (α) | 5.1452E-03 | 15.544E-04 | 4.9427E-04 | 5.6395E-03 | 2.0487E-03 |
| Actual Values (H) | 3.2164E-03 | 9.7153E-04 | 3.0892E-04 | 3.5253E-03 | 1.2804E-03 |
| Estimated Values | 3.811E-03 | 2.391E-03 | 3.6613E-04 | 4.2E-03 | 2.8E-03 |
| Error (%) | 15.61 | 59.37 | 15.63 | 19.14 | 115 |

However, if the gain is adjusted again from 0.35 to 0.55, the algorithm converges on an acceptable solution as shown in Table 19.

**Table 19: LSE Results starting with +60% Error in "Guessed" Parameters, gain set to 0.55 – 7th order model**

| Parameter | $L_{md}$ | $L_{mq}$ | $L_{lk}$ | $L_d$ | $L_q$ |
|---|---|---|---|---|---|
| Initial Guess (α) | 5.1452E-03 | 15.544E-04 | 4.9427E-04 | 5.6395E-03 | 2.0487E-03 |
| Actual Values (H) | 3.2164E-03 | 9.7153E-04 | 3.0892E-04 | 3.5253E-03 | 1.2804E-03 |
| Estimated Values | 3.319E-03 | 10.029E-04 | 3.1889E-04 | 3.638E-03 | 1.322E-03 |
| Error (%) | 3.10 | 3.13 | 3.13 | 3.10 | 3.15 |

Once the gain was adjusted, the 7th order model did converge to acceptable values. So, this means that a gain can be selected based on the level of confidence in an initial guess. The following table shows suggested gains provided for given error confidence levels. These values were determined through trial and error by setting the percent error in guessed parameters, then adjusting the gain accordingly until the algorithm successfully converged. The values listed in Table 20 are the minimum gains required for the algorithm to converge. If the initial guess is within the following confidence percentages, the algorithm and 7th order model should converge for the given parameters.

**Table 20: LSE Results starting with +60% Error in "Guessed" Parameters, gain set to 0.55 – 7th order model**

| Confidence percent (%) | α*120% | α*140% | α*160% | α*180% | α*200% |
|---|---|---|---|---|---|
| Gain (g) | 0.35 | 0.35 | 0.55 | 0.65 | 0.75 |

Note that these gains are only given for cases where the initial guesses are greater than the actual parameters. When the initial guesses are lower than the known parameters, the algorithm currently does not converge on a proper value. Also, it is important to note that the number of iterations increases as the error in the initial guess increases, and also depends the gain. A better understanding of the Least Squares Estimation theory and numerical methods is needed to understand this relationship and optimize the gain adjustments. Also note that in practice, the actual values are not known, so setting initial guesses larger than the known parameters is not reasonable.

To demonstrate the results described above, values are set to be +200% of their actual values, and the gain was set to 0.75. Results shown in Figure 63, demonstrate that the algorithm converges to acceptable values within 20 iterations.

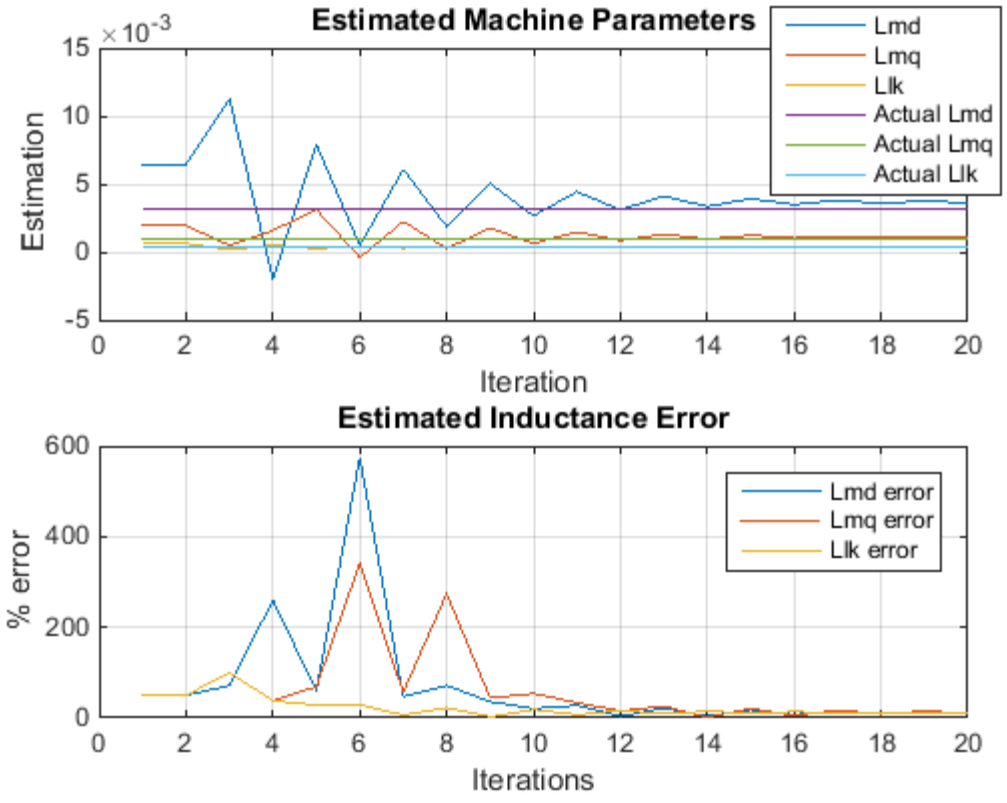


**Figure 63: Least squares estimation results with +200% initial guess error and a gain of 0.75 - 7th order model**

# 7 Conclusion

Traditional offline tests used for characterizing a synchronous machine are described in IEEE Std. 115-2009. However, removal of the machine from service is required in order to accurately conduct these tests. The purpose of this thesis was to take initial steps toward avoid this difficulty through the implementation of online characterization techniques.

In this thesis, offline and online characterization techniques were reviewed, revealing only a handful of the many tests that can be conducted to understand both mathematical and physical characteristics of a given machine. Offline tests require that a machine be physically removed from the system for tests to be conducted. Online tests imply that the machine does not have to be taken down. Both gray box and black box approaches to online characterization can yield significantly different, yet accurate results. Gray box models use input and output data combined with a numerical algorithm to calculate the machine's electrical parameters. Black box models map input data to output data through a series of cascading transfer functions. In either case, offline or online tests must be conducted throughout the duration of a machine's life, in order to maintain accurate records of the machine's behaviors. However, because the black box model does not actually determine machine parameters that would be used in standard power system dynamic simulations, a gray-box algorithm was selected for the online characterization.

This thesis did a preliminary implementation of one specific online characterization technique. The method used in this thesis has proven to be computationally heavier than expected. More work is needed to successfully operate the least squares approximation in both the $3^{rd}$ order and $7^{th}$ order model.

This thesis focused on implementing the algorithm on only 3 parameters, but the approach can be altered changed to determine more parameters. However, as stated in Chapter 2, one of the difficulties with gray box models is a lack of rich data. If only steady-state parameters are desired, it is possible to

calculate these parameters from steady-state data. In using this technique, a confidence level must be established in order to accurately converge on the actual parameters. The least squares approximation method is prone to errors simply because the data generally does not have the details needed to easily calculate the machine parameters. In other words, there are many different combinations of coefficients that can be used which may create an accurate state-estimation for a particular case, but may not necessarily represent a generic model for the machine. The difficulty lies in knowing which values are the correct values.

A high level image of progress towards the online characterization method focused on in this thesis is shown in Figure 64. Real-time data can be fed into the online characterization algorithm. This data includes the output voltages and currents, field voltage and current, rotor frequency, at a minumum.

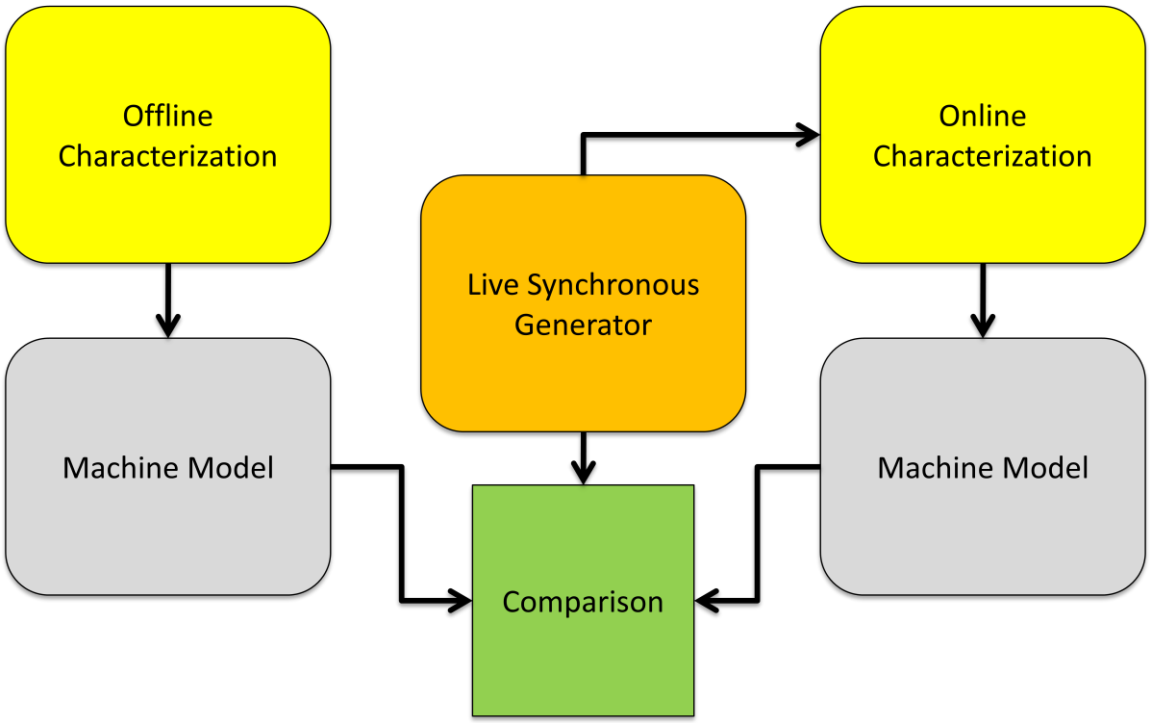


**Figure 64: High Level Representation of Combined Models**

A comparison is also made between implementing the LSE on the 3rd order model as well as the 7th order model. Because all parameters estimated in this thesis were under steady-state conditions,

comparisons between the 7th order model and 3rd order model show small differences simply because the damper windings in the 7th order model don't affect the system under steady-state conditions. Because the model is operating in balanced steady-state, there is no zero sequence response either. Therefore, the models show very similar results.

Gray box models are difficult to implement due to the lack of rich data obtained from the machine. Many degrees of freedom are allowed for the least squares estimation to adjust parameters in wrong directions. In other words, the algorithm can compensate for errors in one parameter by changing another parameter. However, some terms such as the d-axis mutual inductance can be obtained during machine start-up. It is advantageous to use as many known parameters as possible in the mathematical model in order to limit the number of degrees of freedom available for variation in the solution process, and improve estimation of the remaining parameters.

# 8 Future Work

If this method online parameter estimation were to be continued, additional research is needed to better understand optimization theory, numerical limits, and efficiency of the least squares approximation. This thesis focuses mainly on the derivation and development of a computer based model that can use live machine data to determine its parameters and validation of the LSE algorithm.

The numerical algorithm presented in this thesis needs continued development in order to improve the performance, accuracy, reproducibility, and efficiency of the calculations. An example of this is better understanding the relationship between changing the gains and the number of iterations required to successfully converge on proper values. Once fully developed, the approximation algorithm can be implemented in real, larger scale machines and compared with offline characterization results. This will provide detailed information on real-life measurement conditions and allow researchers to further develop its capabilities toward practical application. In addition, a real-time machine model can be built and the parameters can be updated whenever appropriate machine responses occur.

In addition, the higher order machine model will need to consider both saturation and noise. As of now, these are not included in the model, other than the noise in the laboratory machine which did cause problems. In other words, the results shown are, for the most part, ideal scenarios. In practice, there will be sources of error in approximations. If a machine is driven even partially into saturation, the model used will inaccurately reflect the behaviors because it does not account for saturation. For more information on saturation behavior of synchronous machines, see [18].

When ready, more rigorous testing on an actual machine is needed to see if the LSE algorithm will converge for sub-transient and transient parameters, and possibly resistances. However, some parameters, such as field resistance and DC stator resistances, are more easily obtainable through measurements. These measured parameters can be used to help reduce the number of degrees of freedom in the optimization calculation.

The models and tests shown in this thesis assumed constant, nominal rotor speed as well. In reality, this is not always the case and must be considered if using dynamic response data from the machine. As disturbances occur on the system, the speed of the machine may slowly change, thus changing results of the state-space equations. As a result the algorithm will need to incorporate frequency tracking.

Other online characterization approaches can also be considered to determine the best method to apply for the task at hand. Currently, a second approach of using Kalman filtering is under development and is also showing much more encouraging results.

# References

[1] Park, R. H.; “Two-reaction theory of synchronous machines – generalized methods of analysis – Part I,” *AIEE Transactions*, vol. 48, pp. 716-727, July 1929.

[2] Dinely, J.L., Morris, A. J.; “Synchronous generator transient control – Pt I: Theory and evaluation of alternative mathematical models,” *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-92, pp. 417-422, Apr. 1973.

[3] Dandeno, P. L., Kundur, P., Schulz, R. P.; “Recent trends and progress in synchronous machine modeling in the electric utility industry,” *Proceedings of the IEEE*, vol. 62, pp. 941-950, July 1974.

[4] Rotating Electrical Machines. Part 10: Conventions for Description of Synchronous Machines, *IEC Standard 34-10-1975*, International Electrotechnical Commission, Geneva, 1975.

[5] Schulz, R. P.; “Synchronous machine modeling,” *IEEE Symposium on Adequacy and Philosophy of Modeling: Dynamic System Performance*, IEEE Publications, 75CH0970-4-PWR, pp. 24-28.

[6] Park, R. H.; “Two-reaction theory of synchronous machines, Part II,” *AIEE Transactions*, vol. 52, pp. 352-355, June 1933.

[7] *IEEE Guide for Test Procedures for Synchronous Machines Part I Acceptance and Performance Testing Part II Test Procedures and Parameter Determination for Dynamic Analysis, IEEE Std 115-2009 (Revision of IEEE Std 115-1995)* , vol., no., pp.1,219, May 7 2010 doi: 10.1109/IEEESTD.2010.5464495

[8] Zhao, Z., Zheng, F, Gao, J., Xu L.; "A dynamic on-line parameter identification and full-scale system experimental verification for large synchronous machines," *Energy Conversion, IEEE Transactions on* , vol.10, no.3, pp.392,398, Sep 1995 doi: 10.1109/60.464859

[9] Lee CC, Owen TT, “A Weighted-Least Squares Parameters Estimator for Synchronous Machines,” *IEEE Transactions on PAS 96*, No.1, pp97-101,1977

[10] Mainba M, “Identification of Parameter for System Stability Analysis using Kalman Filter,” *IEEE Trans. On PAS*, Vol. PAS-100, No.7, 1981

[11] Burth, M.; Verghese, George C.; Velez-Reyes, M., "Subset selection for improved parameter estimation in on-line identification of a synchronous generator," *Power Systems, IEEE Transactions on* , vol.14, no.1, pp.218,225, Feb 1999

[12] Chen, X., et al; "On-line identification of synchronous generator parameter from large disturbance testing data," *Power System Technology, 1998. Proceedings. POWERCON '98. 1998 International Conference on* , vol.2, no., pp.1034,1039 vol.2, 18-21 Aug 1998

[13] Fard, R.D.; Karrari, M.; Malik, O.P., "Synchronous generator model identification for control application using volterra series," *Energy Conversion, IEEE Transactions on* , vol.20, no.4, pp.852,858, Dec. 2005

[14] Deghani, M.; Karrari, M.; Malik, O.P.; “Synchronous Generator Model Identification Using Linear $H_\infty$ Identification Method,” *IFAC 2007*, ICPS’07, July 09-11, Cluj-Napoca, Romania

[15] Sarem, Yazdan, N.; Poshtan, Javad; Poshtan, Majid; “Synchronous Generator Black Box Modeling using Wiener-Neural Model,” *International Conference on Intelligent and Advanced Systems,* 2007

[16] Perez-Londono, S.; Perez-Londono, A.; Romero-Mora, Y., "On-line identification of the physical parameters in a synchronous generator," *Transmission and Distribution Conference and Exposition: Latin America, 2008 IEEE/PES* , vol., no., pp.1,6, 13-15 Aug. 2008

[17] Chapman, S. J., *Electric Machinery Fundamentals*, Fifth Ed. McGraw-Hill, 2012, ISBN: 978-0-07-352954-7

[18] Krause, P. C., Wasynczuk, O., Sudhoff, Scott D.; *Analysis of Electric Machinery and Drive Systems*; Second Ed.Wiley-Interscience, 2002. ISBN: 0-4717-14326-X

[19] MATLAB and SIMULINK Toolbox Release 2013b, The Mathworks, Inc., Natick, Massachusetts, United States (http://www.mathworks.com/)

[20] Binder, A; *Electrically excited synchronous machines*, Darmstadt University of Technology Report, 04/2016 (http://www.ew.tu-darmstadt.de/media/ew/vortrge/greenenergyconversion/gec_4.pdf)

# Appendix – MATLAB Code

## 7th Order Synchronous Machine Model

```
%% Author: Michael West
% Thesis: Online Dynamic Parameter Estimation of Synchronous Machines
% Purpose of Script: 7th Order Synchronous Machine Model

clc
clear all
close all


%% Machine Data/Stator Base values
V_sBASE = 13800*sqrt(2/3); %Base voltage
S_BASE = 187E6; %MVA BASE
Z_BASE = V_sBASE^2/S_BASE; %Impedance base
I_sBASE = sqrt(2)*S_BASE/(sqrt(3)*V_sBASE);

omega_BASE = 2*pi*60;
L_sBASE = Z_BASE/omega_BASE;

%% Field Base Values

load('Synch_Machine_Model_SIFundamental_SSv4.mat');
Synch_Machine_Model_OC = Synch_Machine_Model_SIFundamental;

    % Parse Data into variables
        t_OC = transpose(Synch_Machine_Model_OC(1,:)); % Timestamp
        Ifd = transpose(Synch_Machine_Model_OC(2,:)); % Field Current
        Vf = transpose(Synch_Machine_Model_OC(3,:)); % Field Voltage
        Is_a = transpose(Synch_Machine_Model_OC(4,:)); % Stator Current phase A
        Iq = transpose(Synch_Machine_Model_OC(5,:)); % Q axis Current
        Id = transpose(Synch_Machine_Model_OC(6,:)); % D axis Current
        Vd = transpose(Synch_Machine_Model_OC(7,:)); % D axis Voltage
        Vq = transpose(Synch_Machine_Model_OC(8,:)); % Q axis Voltage
        Vt = transpose(Synch_Machine_Model_OC(9,:)); % Terminal Voltage
        psimd = transpose(Synch_Machine_Model_OC(10,:)); % D axis mutual flux
        psimq = transpose(Synch_Machine_Model_OC(11,:)); % Q axis mutual flux
        Po = transpose(Synch_Machine_Model_OC(12,:)); % Output Real Power
        Qo = transpose(Synch_Machine_Model_OC(13,:)); % Output Reactive Power
        Wm = transpose(Synch_Machine_Model_OC(14,:)); % Rotor Mechanical Speed
        dWm = transpose(Synch_Machine_Model_OC(15,:)); % Change in Rotor Mechanical Speed
        theta_R = transpose(Synch_Machine_Model_OC(16,:)); % Rotor Mechanical Angle
        delta = transpose(Synch_Machine_Model_OC(17,:)); % Load Angle
        dtheta = transpose(Synch_Machine_Model_OC(18,:)); % Change in Rotor Mechanical Angle
        Ikq1 = transpose(Synch_Machine_Model_OC(19,:)); % Q axis damper winding current 1
        Ikq2 = transpose(Synch_Machine_Model_OC(20,:)); % Q axis damper winding current 2
        Ikd = transpose(Synch_Machine_Model_OC(21,:)); % D axis damper winding current
        I_abc_stator = transpose(Synch_Machine_Model_OC(22:24,:)); % ABC Phase Stator Currents

        % Number of machine poles
        n_poles = 40;
        start_point = 1;
%% Define Resistances/Inductances for SIFundamental Model
    % Stator
        Lmd = 3.2164E-03; % D-axis mutual inductance [H]
        Lmq = 9.7153E-04; % Q-axis mutual inductance [H]
        Llk = 3.0892E-04; %[H]
        r_s = 2.9069E-03; % [Ohm]

    % Field
        r_p_fd = 1.9013E-03; % field resistance [Ohm]
        L_p_lfd = 3.0712E-04; % field leakage inductance [H]

    % Dampers
        r_p_kq1 = 2.0081E-02; % Q-axis damper winding 1 resistance [Ohm]
        r_p_kq2 = r_p_kq1; % Q-axis damper winding 2 resistance
        r_p_kd = 1.1900E-02; % D-axis damper winding resistance
```

```
        L_p_lkq1 = 1.0365E-03; % Q-axis damper winding 1 leakage inductance [H]
        L_p_lkd = 4.9076E-04; % D-axis damper winding leakage inductance
        % let Q-axis damper winding 2 leakage inductance be defined as:
        L_p_lkq2 = L_p_lkq1; % [H]

     % Define Stator_winding/Field_winding transformation ratio:
        Ns_Nf = 0.07798;

% Define measured voltages
       V_r_qs = Vq(start_point:end);%ones(1,length(t))*mean(Vq);
       V_r_ds = Vd(start_point:end);%ones(1,length(t))*mean(Vd);
       V_pr_fd = Vf(start_point:end)*Ns_Nf; %ones(1,length(t))*mean(Vf); % Simulink model has
       specified nominal field current, so volts are entered in DC and therefore must be
       referred to the stator
       omega_b = omega_BASE; % Base frequency
       omega_r = (n_poles/2)*Wm; % rotor speed [rad/sec] eq. 1D-8 pg 58

% Convert inductances to reactances:
    % Stator
        X_md(1) = Lmd*omega_BASE;
        X_mq(1) = Lmq*omega_BASE;
        X_ls(1) = Llk*omega_BASE;

        X_d(1) = X_md + X_ls; % Krause eq 5.5-39
        X_q(1) = X_mq + X_ls; % Krause eq 5.5-40

    % Field
        X_p_lfd = L_p_lfd*omega_BASE;
        X_p_fd = X_p_lfd + X_md; % Krause eq 5.5-40

    % Dampers
        X_p_lkq1 = L_p_lkq1*omega_BASE;
        X_p_lkq2 = L_p_lkq2*omega_BASE;
        X_p_lkd = L_p_lkd*omega_BASE;

        X_p_kd = X_p_lkd + X_md; % Krause eq 5.5-40
        X_p_kq1 = X_p_lkq1 + X_mq; % Krause eq 5.5-40
        X_p_kq2 = X_p_lkq2 + X_mq; % Krause eq 5.5-40



%% Initialize flux linkages based on first measurements
       psi_r_mq(1) = psimq(start_point);
       psi_r_md(1) = psimd(start_point);
       psi_r_qs(1) = psi_r_mq(1) - Iq(start_point)*X_ls(1);
       psi_r_ds(1) = psi_r_md(1) - Id(start_point)*X_ls(1);
       psi_pr_fd(1) = psi_r_md(1) + Ifd(start_point)*(2/3)*Ns_Nf*X_ls(1);

       % Output Initial Conditions to MATLAB Command Window
       fprintf('Initial Conditions \n psi_mq = %.2f \n psi_md = %.2f \n psi_r_qs = %.2f \n
psi_r_ds = %.2f \n psi_pr_fd = %.2f \n',[psi_r_mq psi_r_md psi_r_qs psi_r_ds psi_pr_fd])

       % Initialize Xaq and Xad
       X_aq(1) = ((1/X_mq(1)) + (1/X_ls(1))+(1/X_p_lkq1) + (1/X_p_lkq2))^-1;
       X_ad(1) = ((1/X_md(1)) + (1/X_ls(1)) + (1/X_p_lfd(1))+(1/X_p_lkd))^-1;

       % Initial flux linkage states
       x_int(:,1) = [ psi_r_mq(1); psi_r_md(1); psi_r_qs(1); psi_r_ds(1); psi_pr_fd(1); 0; 0;
0; 0];

       % Put measured voltages into matrix u
       e_pr_xfd = V_pr_fd*(X_md(1)/r_p_fd); % Krause 5.5-36
       u(:,1) = [V_r_qs(1); V_r_ds(1); e_pr_xfd(1); 0; 0; 0; 0; 0; 0];


%% State Space Integral Equations
       % For trapezoidal integration
       dt = 1/8000;
       t(1) = 0;
       k=1;
```

```
    for n = 2:length(t_OC(start_point:end))
    A(:,:,k) = [0, 0, X_aq(1)/X_ls(1), 0, 0, 0, 0, X_aq(1)/X_p_kq1(1), X_aq(1)/X_p_kq2(1);...
%psimq
             0, 0, 0, X_ad(1)/X_ls(1), X_ad(1)/X_p_lfd(1), 0, X_ad(1)/X_p_lkd(1), 0, 0;...
%psimd
             r_s/X_ls(1), 0, -r_s/X_ls(1), -omega_r(n)/omega_b, 0, 0, 0, 0, 0;... %psiqs
             0, r_s/X_ls(1), omega_r(n)/omega_b, -r_s/X_ls(1), r_s/X_ls(1), 0, 0, 0, 0;...
%psids
             0, r_p_fd/X_p_lfd(1), 0, 0, -r_p_fd/X_p_lfd(1), 0, 0, 0, 0;... %psifd
             0, 0, 0, 0, 0, -r_s/X_ls(1), 0, 0, 0;... %psi0s
             0, r_p_kd/X_p_lkd(1), 0, 0, 0, 0, -r_p_kd/X_p_lkd(1), 0, 0;... % psikd
             r_p_kq1/X_p_lkq1(1), 0, 0, 0, 0, 0, 0, -r_p_kq1/X_p_lkq1(1), 0;... %psikq1
             r_p_kq2/X_p_lkq2(1), 0, 0, 0, 0, 0, 0, 0, -r_p_kq2/X_p_lkq2(1)]; %psikq2

         % B matrix for voltage measurements (note all damper bars are shorted,
         % no volage) and no zero sequence
        B(:,:,k) = [0 0 0 0 0 0 0 0 0;... % psi_mq
             0 0 0 0 0 0 0 0 0;... % psi_md
             1 0 0 0 0 0 0 0 0;... % psi_qs
             0 1 0 0 0 0 0 0 0;... % psi_ds
             0 0 r_p_fd/X_md(1) 0 0 0 0 0 0;... %psi_fd
             0 0 0 0 0 0 0 0 0;... % psi_0s
             0 0 0 0 0 0 0 0 0;...% psi_kd
             0 0 0 0 0 0 0 0 0;... %psi_kq1
             0 0 0 0 0 0 0 0 0]; %psi_kq2

        C(:,:,k) = [1/X_ls(k) 0 -1/X_ls(k) 0 0 0 0 0 0;... %i_qs
                    0 1/X_ls(k) 0 -1/X_ls(k) 0 0 0 0 0;... % i_ds
                    0 -1/X_p_lfd(k) 0 0 1/X_p_lfd(k) 0 0 0 0;... % i_fd
                    0 0 0 0 0 -1/X_ls(1) 0 0 0;... % i_0s
                    0 -1/X_p_lfd(1) 0 0 0 0 1/X_p_lfd(1) 0 0;... %i_pr_kd
                    -1/X_p_lkq1(1) 0 0 0 0 0 0 1/X_p_lkq1(1) 0;... %i_pr_kq1
                    -1/X_p_lkq2(1) 0 0 0 0 0 0 0 1/X_p_lkq2(1)]; %i_pr_kq2

            % State Space Equations X = A*x + B*u
            u(:,n) = [V_r_qs(n); V_r_ds(n); e_pr_xfd(n); 0; 0; 0; 0; 0; 0];

            % x_###(1,:) = psi_r_mq; % x_###(2,:) = psi_r_md;
            % x_###(3,:) = psi_r_qs; % x_###(4,:) = psi_r_ds;
            % x_###(5,:) = psi_r_fd; % x_###(6,:) = psi_0s;
            % x_###(7,:) = psi_pr_kd; % x_###(8,:) = psi_pr_kq1;
            % x_###(9,:) = psi_pr_kq2
            x_diff(:,n,k) =(A(:,:,k)*x_int(:,n-1) + B(:,:,k)*u(:,n-1));
            x_int(:,n,k) = (x_int(:,n-1) + 0.5*dt*(x_diff(:,n)-x_diff(:,n-1))); % Integrated
States

            % Calculate Outputs Y = Cx + v where v = 0 for now (no noise)
            i_state(:,n,k) = C(:,:,k)*x_int(:,n,k); % Outputs
            i_r_fd(1,n,k) = (3/2)*Ns_Nf^-1*i_state(3,n,k); % Field current referred back to
rotor

            % Increment Timestep
            t(n) = t(n-1)+dt;
    end

%% Plot Results
    figure(1)
        subplot(4,2,1)
        plot(t_OC(start_point:end),Vq(start_point:end))
        title('Simulink Q axis Voltage [V]')
        grid on
        ylabel('V')


        subplot(4,2,2)
        plot(t_OC(start_point:end),Vd(start_point:end))
        title('Simulink D axis Voltage [V]')
        grid on
        ylabel('V')
```

```
    subplot(4,2,3)
    plot(t,i_state(1,:))
    hold all
    plot(t_OC(start_point:end),Iq(start_point:end))
           title(sprintf('Q axis stator currents [A] \n error = %.2f %%',abs((mean(Iq)-
   mean(i_state(1,:)))/mean(i_state(1,:)))*100))
    grid on
    ylabel('A')
    legend('Calculated','Simulink')

    subplot(4,2,4)
    plot(t,i_state(2,:))
    hold all
    plot(t_OC(start_point:end),Id(start_point:end))
           title(sprintf('D axis stator currents [A]\n error = %.2f %%',abs((mean(Id)-
   mean(i_state(2,:)))/mean(i_state(2,:)))*100))
    grid on
    ylabel('A')
    legend('Calculated','Simulink')

    subplot(4,2,5)
    plot(t,x_int(1,:))
    hold all
    plot(t_OC(start_point:end),psimq(start_point:end))
           title(sprintf('Q axis psi_m_q \n error = %.2f %%',abs((mean(psimq)-
   mean(x_int(1,:)))/mean(x_int(1,:)))*100))
    ylabel('Flux Linkage')
    grid on
    legend('Calculated','Simulink')

    subplot(4,2,6)
    plot(t,x_int(2,:))
    hold all
    plot(t_OC(start_point:end),psimd(start_point:end))
           title(sprintf('D axis psi_m_d \n error = %.2f %%',abs((mean(psimd)-
   mean(x_int(2,:)))/mean(x_int(2,:)))*100))
    ylabel('Flux Linkage')
    grid on
    legend('Calculated','Simulink')

    subplot(4,1,4)
    plot(t,i_r_fd)
    hold all
    plot(t_OC(start_point:end), Ifd(start_point:end));
   title(sprintf('Field Current [A] \n error = %.2f %%',abs((mean(Ifd)-
   mean(i_r_fd))/mean(i_r_fd))*100))
    xlabel('Time [s]')
    ylabel('Amps [A]')
    legend('Calculated','Simulink')
    grid on

figure(2)
    plot(t,i_state(1,:))
    hold all
    plot(t, i_state(2,:))
    plot(t, i_state(3,:))
    grid on
    title('Calculated Currents')
    legend('Iqs','Ids','Ifd')
    xlabel('Time [s]')
    ylabel('Amps')
```

## Least Squares Estimation Implemented with 7th Order Synchronous Machine Model

```
%% Author: Michael West
% Thesis: Online Dynamic Parameter Estimation of Synchronous Machines
% Purpose of Script: Implement the Least Squares Estimation Algorithm on a
% 7th Order Synchronous Machine Model and Compare Results with Simulink
% Synchronous Machine Model.

clc
clear all
close all


%% Stator Base values
    V_sBASE = 13800*sqrt(2/3); %Base voltage
    S_BASE = 187E6; %MVA BASE
    Z_BASE = V_sBASE^2/S_BASE; %Impedance base
    I_sBASE = sqrt(2)*S_BASE/(sqrt(3)*13800);

    omega_BASE = 2*pi*60;
    L_sBASE = Z_BASE/omega_BASE;

%% Field Base Values
    I_fbase = 1087; % Amps
    V_fdBASE = 226.6; % Voltages

%% Real Machine Parameters;
    %% Define Resistances/Inductances for SIFundamental Model
    % Stator
        Lmd_r = 3.2164E-03; % D-axis mutual inductance [H]
        Lmq_r = 9.7153E-04; % Q-axis mutual inductance [H]
        Llk_r = 3.0892E-04; %[H]
        r_s_r = 2.9069E-03; % [Ohm]

        L_d_r = Llk_r+Lmd_r;
        L_q_r = Llk_r+Lmq_r;
    % Field
        r_p_fd_r = 1.9013E-03; % field resistance [Ohm]
        L_p_lfd_r = 3.0712E-04; % field leakage inductance [H]

    % Dampers
        r_p_kq1_r = 2.0081E-02; % Q-axis damper winding 1 resistance [Ohm]
        r_p_kq2_r = r_p_kq1_r; % Q-axis damper winding 2 resistance
        r_p_kd_r = 1.1900E-02; % D-axis damper winding resistance

        L_p_lkq1_r = 1.0365E-03; % Q-axis damper winding 1 leakage inductance [H]
        L_p_lkd_r = 4.9076E-04; % D-axis damper winding leakage inductance
        % let Q-axis damper winding 2 leakage inductance be defined as:
        L_p_lkq2_r = L_p_lkq1_r; % [H]

% Convert inductances to reactances:
    % Stator
        X_md_r(1) = Lmd_r*omega_BASE;
        X_mq_r(1) = Lmq_r*omega_BASE;
        X_ls_r(1) = Llk_r*omega_BASE;

        X_d_r(1) = X_md_r + X_ls_r; % Krause eq 5.5-39
        X_q_r(1) = X_mq_r + X_ls_r; % Krause eq 5.5-40

    % Field
        X_p_lfd_r = L_p_lfd_r*omega_BASE;
        X_p_fd_r = X_p_lfd_r + X_md_r; % Krause eq 5.5-40

    % Dampers
        X_p_lkq1_r = L_p_lkq1_r*omega_BASE;
        X_p_lkq2_r = L_p_lkq2_r*omega_BASE;
        X_p_lkd_r = L_p_lkd_r*omega_BASE;

        X_p_kd_r = X_p_lkd_r + X_md_r; % Krause eq 5.5-40
        X_p_kq1_r = X_p_lkq1_r + X_mq_r; % Krause eq 5.5-40
        X_p_kq2_r = X_p_lkq2_r + X_mq_r; % Krause eq 5.5-40
```

```
     % Output real machine parameters to MATLAB Command Window
        fprintf('Real Machine Parameters \n \n Stator: \n Lmd= %.8f      Xmd = %.8f \n Lmq =
%.8f     Xmq = %.8f \n Llk = %.8f     Xlk = %.8f \n r_s = %.8f \n L_d=%.8f      L_q=%.8f \n\n
Field: \n r`_fd = %.8f \n L`_lfd = %.8f    X`_lfd = %.8f \n                    X_p_fd = %.4f
\n',...
            [Lmd_r X_md_r Lmq_r X_mq_r Llk_r X_ls_r r_s_r L_d_r L_q_r r_p_fd_r L_p_lfd_r
X_p_lfd_r X_p_fd_r])

     % Define Stator_winding/Field_winding transformation ratio:
        Ns_Nf = 0.07798;

     % Number of machine poles
        n_poles = 40;
        start_point = 1;
%% Other User Inputs - use open circuit data to calculate Lmd if desired
load('Synch_Machine_Model_SIFundamental_OC.mat');
Synch_Machine_Model_OC = Synch_Machine_Model_SIFundamental;

    % Parse Data into variables
        t_OC = transpose(Synch_Machine_Model_OC(1,:)); % Timestamp
        Ifd = transpose(Synch_Machine_Model_OC(2,:)); % Field Current
        Vf = transpose(Synch_Machine_Model_OC(3,:)); % Field Voltage
        Is_a = transpose(Synch_Machine_Model_OC(4,:)); % Stator Current phase A
        Iq = transpose(Synch_Machine_Model_OC(5,:)); % Q axis Current
        Id = transpose(Synch_Machine_Model_OC(6,:)); % D axis Current
        Vd = transpose(Synch_Machine_Model_OC(7,:)); % D axis Voltage
        Vq = transpose(Synch_Machine_Model_OC(8,:)); % Q axis Voltage
        Vt = transpose(Synch_Machine_Model_OC(9,:)); % Terminal Voltage
        psimd = transpose(Synch_Machine_Model_OC(10,:)); % D axis mutual flux
        psimq = transpose(Synch_Machine_Model_OC(11,:)); % Q axis mutual flux
        Po = transpose(Synch_Machine_Model_OC(12,:)); % Output Real Power
        Qo = transpose(Synch_Machine_Model_OC(13,:)); % Output Reactive Power
        Wm = transpose(Synch_Machine_Model_OC(14,:)); % Rotor Mechanical Speed
        dWm = transpose(Synch_Machine_Model_OC(15,:)); % Change in Rotor Mechanical Speed
        theta_R = transpose(Synch_Machine_Model_OC(16,:)); % Rotor Mechanical Angle
        delta = transpose(Synch_Machine_Model_OC(17,:)); % Load Angle
        dtheta = transpose(Synch_Machine_Model_OC(18,:)); % Change in Rotor Mechanical Angle
        Ikq1 = transpose(Synch_Machine_Model_OC(19,:)); % Q axis damper winding current 1
        Ikq2 = transpose(Synch_Machine_Model_OC(20,:)); % Q axis damper winding current 2
        Ikd = transpose(Synch_Machine_Model_OC(21,:)); % D axis damper winding current
        I_abc_stator = transpose(Synch_Machine_Model_OC(22:24,:)); % ABC Phase Stator Currents

 %% Calculate Lmd based on open circuit measurements
for n = 1:length(Ifd)
        Lmd_calc(n) = (1/omega_BASE)*Vq(n)/((2/3)*(1/Ns_Nf)*Ifd(n)); % Calculate Lmd from open
circuit data
        Lmd_calc_error(n) = ((Lmd_calc(n) - Lmd_r) /Lmd_r)*100; % calculate error
end
%  figure(1)
%         plot(t_OC,Lmd_calc)
%         hold all
%         plot(t_OC, ones(1,length(Lmd_calc))*Lmd)
%         title(sprintf('Lmd Calculated \n average error = %.2f
%%',abs(mean(Lmd_calc_error))))
%         xlabel('Time [s]')
%         ylabel('H')
%         legend('Calculated','Actual')
%         grid on
 %----------------------------------%
 %----------------------------------%
 %----------------------------------%
 %----------------------------------%
 %----------------------------------%
 %----------------------------------%
 %----------------------------------%
 %----------------------------------%


 %% Initial Guess Machine Parameters (FOR LSE testing)
 error = 1.2;
```

```
% Stator
        Lmd = mean(Lmd_calc)*error ; % D-axis mutual inductance [H]*error
        Lmq = Lmq_r*error ; % Q-axis mutual inductance [H]*error
        Llk = Llk_r*error ; %[H]*error
        r_s = r_s_r*1.0; % [Ohm]

        L_d = Llk+Lmd;
        L_q = Llk+Lmq;

% Field
        r_p_fd = r_p_fd_r*1.00; % field resistance [Ohm]
        L_p_lfd = L_p_lfd_r*1.00; % field leakage inductance [H]

    % Dampers
        r_p_kq1 = r_p_kq1_r*1.00; % Q-axis damper winding 1 resistance [Ohm]
        r_p_kq2 = r_p_kq2_r*1.00; % Q-axis damper winding 2 resistance
        r_p_kd = r_p_kd_r*1.00; % D-axis damper winding resistance

        L_p_lkq1 = L_p_lkq1_r*1.00; % Q-axis damper winding 1 leakage inductance [H]
        L_p_lkd =  L_p_lkd_r*1.00; % D-axis damper winding leakage inductance
        % let Q-axis damper winding 2 leakage inductance be defined as:
        L_p_lkq2 = L_p_lkq2_r*1.00; % [H]

% Convert initial guess inductances to reactances:
    % Stator
        X_md(1) = Lmd*omega_BASE;
        X_mq(1) = Lmq*omega_BASE;
        X_ls(1) = Llk*omega_BASE;

        X_d(1) = X_md + X_ls; % Krause eq 5.5-39
        X_q(1) = X_mq + X_ls; % Krause eq 5.5-40

    % Field
        X_p_lfd(1) = L_p_lfd*omega_BASE;
        X_p_fd(1) = X_p_lfd + X_md; % Krause eq 5.5-40

    % Dampers
        X_p_lkq1(1) = L_p_lkq1*omega_BASE;
        X_p_lkq2(1) = L_p_lkq2*omega_BASE;
        X_p_lkd(1) = L_p_lkd*omega_BASE;

        X_p_kd(1) = X_p_lkd + X_md; % Krause eq 5.5-40
        X_p_kq1(1) = X_p_lkq1 + X_mq; % Krause eq 5.5-40
        X_p_kq2(1) = X_p_lkq2 + X_mq; % Krause eq 5.5-40


        X_aq(1) = ((1/X_mq(1)) + (1/X_ls(1))+(1/X_p_lkq1) + (1/X_p_lkq2))^-1;
        X_ad(1) = ((1/X_md(1)) + (1/X_ls(1)) + (1/X_p_lfd(1))+(1/X_p_lkd))^-1;

 %% Load steady-state Machine Data
load('Synch_Machine_Model_SIFundamental_SSv4.mat');
Synch_Machine_Model_OC = Synch_Machine_Model_SIFundamental;

    % Parse Data into variables
        t_OC = transpose(Synch_Machine_Model_OC(1,:)); % Timestamp
        Ifd = transpose(Synch_Machine_Model_OC(2,:)); % Field Current
        Vf = transpose(Synch_Machine_Model_OC(3,:)); % Field Voltage
        Is_a = transpose(Synch_Machine_Model_OC(4,:)); % Stator Current phase A
        Iq = transpose(Synch_Machine_Model_OC(5,:)); % Q axis Current
        Id = transpose(Synch_Machine_Model_OC(6,:)); % D axis Current
        Vd = transpose(Synch_Machine_Model_OC(7,:)); % D axis Voltage
        Vq = transpose(Synch_Machine_Model_OC(8,:)); % Q axis Voltage
        Vt = transpose(Synch_Machine_Model_OC(9,:)); % Terminal Voltage
        psimd = transpose(Synch_Machine_Model_OC(10,:)); % D axis mutual flux
        psimq = transpose(Synch_Machine_Model_OC(11,:)); % Q axis mutual flux
        Po = transpose(Synch_Machine_Model_OC(12,:)); % Output Real Power
        Qo = transpose(Synch_Machine_Model_OC(13,:)); % Output Reactive Power
        Wm = transpose(Synch_Machine_Model_OC(14,:)); % Rotor Mechanical Speed
        dWm = transpose(Synch_Machine_Model_OC(15,:)); % Change in Rotor Mechanical Speed
        theta_R = transpose(Synch_Machine_Model_OC(16,:)); % Rotor Mechanical Angle
        delta = transpose(Synch_Machine_Model_OC(17,:)); % Load Angle
```

```
        dtheta = transpose(Synch_Machine_Model_OC(18,:)); % Change in Rotor Mechanical Angle
        Ikq1 = transpose(Synch_Machine_Model_OC(19,:)); % Q axis damper winding current 1
        Ikq2 = transpose(Synch_Machine_Model_OC(20,:)); % Q axis damper winding current 2
        Ikd = transpose(Synch_Machine_Model_OC(21,:)); % D axis damper winding current
        I_abc_stator = transpose(Synch_Machine_Model_OC(22:24,:)); % ABC Phase Stator Currents
 %----------------------------------%
 %----------------------------------%


% Define measured voltages
V_r_qs = Vq(start_point:end);%ones(1,length(t))*mean(Vq);
V_r_ds = Vd(start_point:end);%ones(1,length(t))*mean(Vd);
V_pr_fd = Vf(start_point:end)*Ns_Nf; %ones(1,length(t))*mean(Vf); % Simulink model has
specified nominal field current, so volts are entered in DC and therefore must be referred to
the stator
omega_b = omega_BASE; % Base frequency
omega_r = (n_poles/2)*Wm; % rotor speed [rad/sec] eq. 1D-8 pg 58
 % Initialze LSE
 % Initialze LSE
R = [1; 1; 1; 1; 1; 1; 1]; % for positive semi-definite weight



%% Initialize flux linkages based on first measurements
        psi_r_mq(1) = psimq(start_point);
        psi_r_md(1) = psimd(start_point);
        psi_r_qs(1) = psi_r_mq(1) - Iq(start_point)*X_ls(1);
        psi_r_ds(1) = psi_r_md(1) - Id(start_point)*X_ls(1);
        psi_pr_fd(1) = psi_r_md(1) + Ifd(start_point)*(2/3)*Ns_Nf*X_ls(1);

% Output Initial Conditions to MATLAB Command Window
fprintf('Initial Conditions \n psi_mq = %.2f \n psi_md = %.2f \n psi_r_qs = %.2f \n psi_r_ds =
%.2f \n psi_pr_fd = %.2f \n',[psi_r_mq psi_r_md psi_r_qs psi_r_ds psi_pr_fd])

%% Initialize the system based on initial parameters
    k = 1; % k = iteration, first iteration is the initialization
% Number of variables to be estimated
     nvar = 3;
% For trapezoidal integration
    dt = 1/8000;
    t(1) = 0;

% Initial flux linkage states
x_int(:,1) = [ psi_r_mq(1); psi_r_md(1); psi_r_qs(1); psi_r_ds(1); psi_pr_fd(1); 0; 0; 0; 0];

% Put measured voltages into matrix u
e_pr_xfd = V_pr_fd*(X_md(1)/r_p_fd); % Krause 5.5-36
u(:,1) = [V_r_qs(1); V_r_ds(1); e_pr_xfd(1); 0; 0; 0; 0; 0; 0];

%% Initialize the system based on initial parameters
    k = 1; % k = iteration, first iteration is the initialization
% Number of variables to be estimated
     nvar = 7;
% For trapezoidal integration
    dt = 1/8000;
    t(1) = 0;

fprintf('%%---Initializing Machine Model---%%\n')

for n = 2:length(t_OC(start_point:end))
    A(:,:,k) = [0, 0, X_aq(1)/X_ls(1), 0, 0, 0, 0, X_aq(1)/X_p_kq1(1), X_aq(1)/X_p_kq2(1);...
%psimq
         0, 0, 0, X_ad(1)/X_ls(1), X_ad(1)/X_p_lfd(1), 0, X_ad(1)/X_p_lkd(1), 0, 0;... %psimd
         r_s/X_ls(1), 0, -r_s/X_ls(1), -omega_r(n), 0, 0, 0, 0, 0;... %psiqs
         0, r_s/X_ls(1), omega_r(n), -r_s/X_ls(1), 0, 0, 0, 0, 0;... %psids
         0, r_p_fd/X_p_lfd(1), 0, 0, -r_p_fd/X_p_lfd(1), 0, 0, 0, 0;... %psifd
         0, 0, 0, 0, 0, -r_s/X_ls(1), 0, 0, 0;... %psi0s
         0, r_p_kd/X_p_lkd(1), 0, 0, 0, 0, -r_p_kd/X_p_lkd(1), 0, 0;... % psikd
         r_p_kq1/X_p_lkq1(1), 0, 0, 0, 0, 0, 0, -r_p_kq1/X_p_lkq1(1), 0;... %psikq1
         r_p_kq2/X_p_lkq2(1), 0, 0, 0, 0, 0, 0, 0, -r_p_kq2/X_p_lkq2(1)]; %psikq2
```

```
     % B matrix for voltage measurements (note all damper bars are shorted,
     % no volage) and no zero sequence
    B(:,:,k) = [0 0 0 0 0 0 0 0 0;... % psi_mq
         0 0 0 0 0 0 0 0 0;... % psi_md
         1 0 0 0 0 0 0 0 0;... % psi_qs
         0 1 0 0 0 0 0 0 0;... % psi_ds
         0 0 r_p_fd/X_md(1) 0 0 0 0 0 0;... %psi_fd
         0 0 0 0 0 0 0 0 0;... % psi_0s
         0 0 0 0 0 0 0 0 0;...% psi_kd
         0 0 0 0 0 0 0 0 0;... %psi_kq1
         0 0 0 0 0 0 0 0 0]; %psi_kq2

    C(:,:,k) = [1/X_ls(k) 0 -1/X_ls(k) 0 0 0 0 0 0;... %i_qs
                0 1/X_ls(k) 0 -1/X_ls(k) 0 0 0 0 0;... % i_ds
                0 -1/X_p_lfd(k) 0 0 1/X_p_lfd(k) 0 0 0 0;... % i_fd
                0 0 0 0 0 -1/X_ls(1) 0 0 0;... % i_0s
                0 -1/X_p_lfd(1) 0 0 0 0 1/X_p_lfd(1) 0 0;... %i_pr_kd
                -1/X_p_lkq1(1) 0 0 0 0 0 0 1/X_p_lkq1(1) 0;... %i_pr_kq1
                -1/X_p_lkq2(1) 0 0 0 0 0 0 0 1/X_p_lkq2(1)]; %i_pr_kq2

        % State Space Equations X = A*x + B*u
        u(:,n) = [V_r_qs(n); V_r_ds(n); e_pr_xfd(n); 0; 0; 0; 0; 0; 0];

        % x_###(1,:) = psi_r_mq; % x_###(2,:) = psi_r_md;
        % x_###(3,:) = psi_r_qs; % x_###(4,:) = psi_r_ds;
        % x_###(5,:) = psi_r_fd; % x_###(6,:) = psi_0s;
        % x_###(7,:) = psi_pr_kd; % x_###(8,:) = psi_pr_kq1;
        % x_###(9,:) = psi_pr_kq2
        x_diff(:,n,k) =(A(:,:,k)*x_int(:,n-1) + B(:,:,k)*u(:,n-1));
        x_int(:,n,k) = (x_int(:,n-1) + 0.5*dt*(x_diff(:,n)-x_diff(:,n-1))); % Integrated
States

        % Calculate Outputs Y = Cx + v where v = 0 for now (no noise)
        i_state(:,n,k) = C(:,:,k)*x_int(:,n,k); % Outputs
        i_r_fd(1,n,k) = (3/2)*Ns_Nf^-1*i_state(3,n,k); % Field current referred back to rotor

        % Increment Timestep
        t(n) = t(n-1)+dt;

        % Output Status of Initialization to Command Window
        if n == round(length(t_OC(start_point:end))/3)
           fprintf('\n33%% Initialized\n')
        elseif n == round(length(t_OC(start_point:end))*(2/3))
           fprintf('66%% Initialized\n')
        elseif n == length(t_OC)
           fprintf('99%% Initialized\n')
        end
end

 %-----------------------------------%
 %-----------------------------------%
 %-----------------------------------%
 %-----------------------------------%
 %-----------------------------------%
 %-----------------------------------%

%% State Space Integral Equations
% For trapezoidal integration
%% Least Squares Estigmation Loop
    iterations = 20;
    num_int(1:nvar,1,1) = 0;
    den_int(1:nvar,1,1) = 0;
% Initialize C.C. Lee Equation (11)
    alpha_hat(:,1) = [Lmq; Lmd; Llk; L_p_lkq1; L_p_lkq2; L_p_lkd; r_s];

    g(1:nvar,1) = 1; % Initialize gains
    fprintf('Initializing Complete \n\n%%---Beginning Estimation Algorithm---%%...\n')
    h = waitbar(0,'Approximating Machine Parameters');

% Define a matrix of "real" machine currents (C.C. Lee y_r)
     for n = 2:length(t_OC(start_point:end));
```

```
         i_r(:,n) = [Iq(n); Id(n); Ifd(n)*(2/3)*Ns_Nf; 0; Ikq1(n); Ikq2(n); Ikd(n)]; % real
machine currents
     end

% Begin LSE Loop
    delta_alpha(1:nvar,1) = 0;
    gain_const = 0.3; % initial gain constant
    g_adj = 0; % tracker for number of times the gain is adjusted (used to limit gain
adjustment to only 1

for k = 2:iterations


       % Calculate gains
    for ii = 1:length(delta_alpha(:,k-1))
        if abs(delta_alpha(ii,k-1)/alpha_hat(ii,k-1)) < gain_const
            g(ii,k) = 1;
        elseif abs(delta_alpha(ii,k-1)/alpha_hat(ii,k-1)) >= gain_const
            g(ii,k) = gain_const*alpha_hat(ii,k-1)/abs(delta_alpha(ii,k-1));

        end
    end

    %% Update Parameters (alpha) C.C. Lee Eq (11)
        alpha_hat(:,k) = alpha_hat(:,1) + g(:,k).*delta_alpha(:,k-1);

        Lmq(k) = alpha_hat(1,k);
%         Lmq(k) = Lmq_r;
        Lmd(k) = alpha_hat(2,k);
%         Lmd(k) = mean(Lmd_calc);
        Llk(k) = alpha_hat(3,k);

        X_md = Lmd(k)*omega_BASE;
        X_mq = Lmq(k)*omega_BASE;
        X_ls = Llk(k)*omega_BASE;


        X_d = X_md + X_ls; % Krause eq 5.5-39
        X_q = X_mq + X_ls; % Krause eq 5.5-40

        X_aq = ((1/X_mq) + (1/X_ls))^-1;
        X_ad = ((1/X_md) + (1/X_ls) + (1/X_p_lfd))^-1;

    % Update Initial flux linkages
        psi_r_mq(1) = psimq(start_point);
        psi_r_md(1) = psimd(start_point);
        psi_r_qs(1) = psi_r_mq(1) - Iq(start_point)*X_ls;
        psi_r_ds(1) = psi_r_md(1) - Id(start_point)*X_ls;
        psi_pr_fd(1) = psi_r_md(1) + Ifd(start_point)*(2/3)*Ns_Nf*X_ls;
        x_int(:,1,k) = [ psi_r_mq(1); psi_r_md(1); psi_r_qs(1); psi_r_ds(1); psi_pr_fd(1); 0;
0; 0; 0];


    for n = 2:length(t_OC(start_point:end))

    A(:,:,k) = [0, 0, X_aq(1)/X_ls(1), 0, 0, 0, 0, X_aq(1)/X_p_kq1(1), X_aq(1)/X_p_kq2(1);...
%psimq
         0, 0, 0, X_ad(1)/X_ls(1), X_ad(1)/X_p_lfd(1), 0, X_ad(1)/X_p_lkd(1), 0, 0;... %psimd
         r_s/X_ls(1), 0, -r_s/X_ls(1), -omega_r(n), 0, 0, 0, 0, 0;... %psiqs
         0, r_s/X_ls(1), omega_r(n), -r_s/X_ls(1), r_s/X_ls(1), 0, 0, 0, 0;... %psids
         0, r_p_fd/X_p_lfd(1), 0, 0, -r_p_fd/X_p_lfd(1), 0, 0, 0, 0;... %psifd
         0, 0, 0, 0, 0, -r_s/X_ls(1), 0, 0, 0;... %psi0s
         0, r_p_kd/X_p_lkd(1), 0, 0, 0, 0, -r_p_kd/X_p_lkd(1), 0, 0;... % psikd
         r_p_kq1/X_p_lkq1(1), 0, 0, 0, 0, 0, 0, -r_p_kq1/X_p_lkq1(1), 0;... %psikq1
         r_p_kq2/X_p_lkq2(1), 0, 0, 0, 0, 0, 0, 0, -r_p_kq2/X_p_lkq2(1)]; %psikq2

     % B matrix for voltage measurements (note all damper bars are shorted,
     % no volage) and no zero sequence
    B(:,:,k) = [0 0 0 0 0 0 0 0 0;... % psi_mq
         0 0 0 0 0 0 0 0 0;... % psi_md
         1 0 0 0 0 0 0 0 0;... % psi_qs
```

```
          0 1 0 0 0 0 0 0;... % psi_ds
          0 0 r_p_fd/X_md(1) 0 0 0 0 0;... %psi_fd
          0 0 0 0 0 0 0 0;... % psi_0s
          0 0 0 0 0 0 0 0;...% psi_kd
          0 0 0 0 0 0 0 0;... %psi_kq1
          0 0 0 0 0 0 0 0]; %psi_kq2

    C(:,:,k) = [1/X_ls(1) 0 -1/X_ls(1) 0 0 0 0 0;... %i_qs
                0 1/X_ls(1) 0 -1/X_ls(1) 0 0 0 0;... % i_ds
                0 -1/X_p_lfd(1) 0 0 1/X_p_lfd(1) 0 0 0;... % i_fd
                0 0 0 0 0 -1/X_ls(1) 0 0;... % i_0s
                0 -1/X_p_lfd(1) 0 0 0 0 1/X_p_lfd(1) 0 0;... %i_pr_kd
                -1/X_p_lkq1(1) 0 0 0 0 0 0 1/X_p_lkq1(1) 0;... %i_pr_kq1
                -1/X_p_lkq2(1) 0 0 0 0 0 0 0 1/X_p_lkq2(1)]; %i_pr_kq2

        % State Space Equations X = A*x + B*u
        u(:,n) = [V_r_qs(n); V_r_ds(n); e_pr_xfd(n); 0; 0; 0; 0; 0; 0];

        % x_###(1,:) = psi_r_mq; % x_###(2,:) = psi_r_md;
        % x_###(3,:) = psi_r_qs; % x_###(4,:) = psi_r_ds;
        % x_###(5,:) = psi_r_fd; % x_###(6,:) = psi_0s;
        % x_###(7,:) = psi_pr_kd; % x_###(8,:) = psi_pr_kq1;
        % x_###(9,:) = psi_pr_kq2
        x_diff(:,n,k) = A(:,:,k)*x_int(:,n-1)+B(:,:,k)*u(:,n-1);
        x_int(:,n,k) = x_int(:,n-1) + 0.5*dt*(x_diff(:,n)-x_diff(:,n-1)); % Integrated States

        % Calculate Outputs Y = Cx + v where v = 0 for now (no noise)
        i_state(:,n,k) = C(:,:,k)*x_int(:,n,k); % Outputs
        i_r_fd(1,n,k) = (3/2)*Ns_Nf^-1*i_state(3,n,k); % Field current referred back to rotor

        %% LSE EQUATIONS
            % Calculate change in input states, and A,B,C Matrices w.r.t
            dx_dalpha_alpha0(:,n,k) = (x_int(:,n,k) - x_int(:,n, 1))/(k-1);
            dA_dalpha_alpha0(:,:,k) = (A(:,:,k) - A(:,:,1))/(k-1);
            dB_dalpha_alpha0(:,:,k) = (B(:,:,k) - B(:,:,1))/(k-1);
            dC_dalpha_alpha0(:,:,k) = (C(:,:,k) - C(:,:,1))/(k-1);

            % C.C. Lee Equation (6);
            eq_6(:,n,k) = A(:,:,1)*dx_dalpha_alpha0(:,n,k) +
dA_dalpha_alpha0(:,:,k)*x_int(:,n,1)...
                    + dB_dalpha_alpha0(:,:,k)*u(:,n);

            % C.C. Lee Equation (5)
            dy_dalpha_alpha0(:,n,k) = C(:,:,1)*eq_6(:,n,k) +
dC_dalpha_alpha0(:,:,k)*x_int(:,n,1);
%             dy_dalpha_alpha0(:,n,k) = C(:,:,k-1)*dx_dA_alpha0(:,n,k) +
dC_dalpha_alpha0(:,:,k)*x_int(:,n,k-1);

            if n == 2;
               % Initialze C.C. Lee Eq (9) numerator and denominator integrals
                    num_int(1:nvar,1,k) = 0;
                    den_int(1:nvar,1,k) = 0;
            end
            % C.C. Lee Equation (9)
            num(:,n,k) = dy_dalpha_alpha0(:,n,k)'*R*(i_r(:,n) - i_state(:,n,k));
            num_int(:,n,k) = num_int(:,n-1,k) + 0.5*dt*(num(:,n,k)- num(:,n-1,k));


            den(:,n,k) = dy_dalpha_alpha0(:,n,k)'*R*dy_dalpha_alpha0(:,n,k);
            den_int(:,n,k) = den_int(:,n-1,k) + 0.5*dt*(den(:,n,k) - den(:,n-1,k));


            % Increment Timestep
            t(n) = t(n-1)+dt;



    end
     % Calculate delta_alpha (alpha_hat - alpha0 in C.C. Lee eq (9))
        delta_alpha(:,k) = num_int(:,end,k) ./ (den_int(:,end,k)+0.000001); % add to
denominator to avoid divide by 0
```

```
%% add some error checking, if parameters have converged but states
%  still have more than 5% error, change gain and continue
%  Also, adjust gains only once...any more may cause diversion.

%     if k > 2
%          if abs(alpha_hat(1,k)-alpha_hat(1,k-2)) < 1E-6... % Lmq converged
%          && abs(alpha_hat(2,k)-alpha_hat(2,k-2)) < 1E-6... % Lmd converged
%          && abs(alpha_hat(3,k)-alpha_hat(3,k-2)) < 1E-6... % Llk converged
%          && ((mean(i_r(1,:))-mean(i_state(1,:,k)))/mean(i_state(1,:,k)))*100 < -5 ... % q-
axis stator current error <-5%
%          && ((mean(i_r(2,:))-mean(i_state(2,:,k)))/mean(i_state(2,:,k)))*100 < -5 ... % q-
axis stator current error <5%
%          &&  g_adj < 1;
%
%
%             gain_const = gain_const - 0.07;
%             g_adj = g_adj+1;
%             fprintf(' \ngain constant changed to %.4f \n',gain_const)
%
%          elseif abs(alpha_hat(1,k)-alpha_hat(1,k-2)) < 1E-6...
%          && abs(alpha_hat(2,k)-alpha_hat(2,k-2)) < 1E-6...
%          && abs(alpha_hat(3,k)-alpha_hat(3,k-2)) < 1E-6...
%          && ((mean(i_r(1,:))-mean(i_state(1,:,k)))/mean(i_state(1,:,k)))*100 > 5 ...
%          && ((mean(i_r(2,:))-mean(i_state(2,:,k)))/mean(i_state(2,:,k)))*100 > 5 ...
%          &&  g_adj < 1;
%
%             gain_const = gain_const + 0.05;
%             g_adj = g_adj+1;
%         fprintf(' \ngain constant changed to %.4f \n',gain_const)
%          end
%     end

        %% save num_int and den_int values
    num_int_save(:,k) = num_int(:,end,k);
    den_int_save(:,k) = den_int(:,end,k);
    waitbar(k/iterations,h,sprintf('Approximating Machine Parameters \n %.3f %%
Complete',(k/iterations)*100));

end
close(h)


    figure(1)
        subplot(4,2,1)
        plot(t_OC(start_point:end),Vq(start_point:end))
        title('Simulink Q axis Voltage [V]')
        grid on
        ylabel('V')


        subplot(4,2,2)
        plot(t_OC(start_point:end),Vd(start_point:end))
        title('Simulink D axis Voltage [V]')
        grid on
        ylabel('V')


        subplot(4,2,3)
        plot(t,i_state(1,:,end),'-b')
        hold all
        plot(t_OC(start_point:end),Iq(start_point:end),'-r')
        title(sprintf('Q axis stator currents [A] \n error = %.2f %%',abs((mean(Iq)-
mean(i_state(1,:,end)))/mean(i_state(1,:,end)))*100))
        grid on
        ylabel('A')
        legend('Calculated','Simulink')

        subplot(4,2,4)
        plot(t,i_state(2,:,end),'-b')
```

```
        hold all
        plot(t_OC(start_point:end),Id(start_point:end),'-r')
        title(sprintf('D axis stator currents [A]\n error = %.2f %%',abs((mean(Id)-
mean(i_state(2,:,end)))/mean(i_state(2,:,end)))*100))
        grid on
        ylabel('A')
        legend('Calculated','Simulink')

        subplot(4,2,5)
        plot(t,x_int(1,:,end),'-b')
        hold all
        plot(t_OC(start_point:end),psimq(start_point:end),'-r')
        title(sprintf('Q axis psi_m_q \n error = %.2f %%',abs((mean(psimq)-
mean(x_int(1,:)))/mean(x_int(1,:)))*100))
        ylabel('Flux Linkage')
        grid on
        legend('Calculated','Simulink')

        subplot(4,2,6)
        plot(t,x_int(2,:,end),'-b')
        hold all
        plot(t_OC(start_point:end),psimd(start_point:end),'-r')
        title(sprintf('D axis psi_m_d \n error = %.2f %%',abs((mean(psimd)-
mean(x_int(2,:)))/mean(x_int(2,:)))*100))
        ylabel('Flux Linkage')
        grid on
        legend('Calculated','Simulink')



        subplot(4,1,4)
        plot(t,i_r_fd(1,:,end),'-b')
    %     plot(t,(-i_r_ds + i_pr_fd))
        hold all
        plot(t_OC(start_point:end), Ifd(start_point:end),'-r');
        title(sprintf('Field Current [A] \n error = %.2f %%',abs((mean(Ifd)-
mean(i_r_fd(1,:,end)))/mean(i_r_fd(1,:,end)))*100))
        xlabel('Time [s]')
        ylabel('Amps [A]')
        legend('Calculated','Simulink')
        grid on

  figure(2)
        plot(t,i_state(1,:,end))
        hold all
        plot(t, i_state(2,:,end))
        plot(t, i_state(3,:,end))
        grid on
        title('Calculated Currents')
        legend('Iqs','Ids','Ifd')
        xlabel('Time [s]')
        ylabel('Amps')

% create vectors to plot the real values
    Lmd_rplot = ones(1,k)*Lmd_r;
    Lmq_rplot = ones(1,k)*Lmq_r;
    Llk_rplot = ones(1,k)*Llk_r;
    Xd_rplot = ones(1,k)*X_d_r;
    Xq_rplot = ones(1,k)*X_q_r;

    for nn = 1:k
        Lmd_error(nn) = abs(((Lmd_rplot(1,nn) - Lmd(end,nn))/Lmd(end,nn))*100);
        Lmq_error(nn) = abs(((Lmq_rplot(1,nn) - Lmq(end,nn))/Lmq(end,nn))*100);
        Llk_error(nn) = abs(((Llk_rplot(1,nn) - Llk(end,nn))/Llk(end,nn))*100);
    end

    figure
        subplot(2,1,1)
        plot(Lmd(end,:))
        hold all
        plot(Lmq(end,:))
```

```
        plot(Llk(end,:))
        plot(Lmd_rplot)
        plot(Lmq_rplot)
        plot(Llk_rplot)
        grid on
        xlabel('Iteration')
        ylabel('Estimation')
        legend('Lmd','Lmq','Llk','Actual Lmd','Actual Lmq','Actual Llk')
        title('Estimated Machine Parameters')

        subplot(2,1,2)
        plot(Lmd_error)
        hold all
        plot(Lmq_error)
        plot(Llk_error)
        legend('Lmd error','Lmq error','Llk error')
        xlabel('Iterations')
        ylabel('% error')
        title('Estimated Inductance Error')
        grid on

    figure
        plot(Lmd(end,:)/Lmd_r)
        hold all
        plot(Lmq(end,:)/Lmq_r)
        plot(Llk(end,:)/Llk_r)
        grid on
        xlabel('Iterations')
        ylabel('Ratio')
        title('Ratio of estimated to real parameters')
        legend('Lmd','Lmq','Llk')


        L_d = Llk(end)+Lmd(end);
        L_q = Llk(end)+Lmq(end);
% Output estimated machine parameters to MATLAB Command Window
fprintf('Estimated Machine Parameters \n Lmd= %.8f \n Lmq = %.8f \n Llk = %.8f \n L_d=%.4f
L_q=%.4f\n',[Lmd(end,end) Lmq(end,end) Llk(end,end) L_d L_q])
fprintf('Estimation Complete \n')


((Lmd_r-Lmd(end))/Lmd(end))*100
((Lmq_r-Lmq(end))/Lmq(end))*100
((Llk_r-Llk(end))/Llk(end))*100
```

## 3rd Order Synchronous Machine Model

```
%% Author: Michael West
% Thesis: Online Dynamic Parameter Estimation of Synchronous Machines
% Purpose of Script: 3rd Order Synchronous Machine Model


clc
clear all
close all


%% Machine Data/Stator Base values
V_sBASE = 13800*sqrt(2/3); %Base voltage
S_BASE = 187E6; %MVA BASE
Z_BASE = V_sBASE^2/S_BASE; %Impedance base
I_sBASE = sqrt(2)*S_BASE/(sqrt(3)*V_sBASE);

omega_BASE = 2*pi*60;
L_sBASE = Z_BASE/omega_BASE;

%% Field Base Values
I_fbase = 1087; % Amps
V_fdBASE = 226.6; % Voltages

        % Number of machine poles
        n_poles = 40;
        start_point = 1;

%% Real Machine Parameters;
    % Stator
        Lmd_r = 3.2164E-03; % D-axis mutual inductance [H]
        Lmq_r = 9.7153E-04; % Q-axis mutual inductance [H]
        Llk_r = 3.0892E-04; %[H]
        r_s_r = 2.9069E-03; % [Ohm]

        L_d_r = Llk_r+Lmd_r;
        L_q_r = Llk_r+Lmq_r;
    % Field
        r_p_fd_r = 1.9013E-03; % field resistance [Ohm]
        L_p_lfd_r = 3.0712E-04; % field leakage inductance [H]

 % Convert inductances to reactances:
    % Stator
        X_md_r = Lmd_r*omega_BASE; %
        X_mq_r = Lmq_r*omega_BASE;
        X_ls_r = Llk_r*omega_BASE;

        X_d_r = X_md_r + X_ls_r; % Krause eq 5.5-39
        X_q_r = X_mq_r + X_ls_r; % Krause eq 5.5-40

    % Field
        X_p_lfd_r = L_p_lfd_r*omega_BASE;
        X_p_fd_r = X_p_lfd_r + X_md_r; % Krause eq 5.5-40

 % Output real machine parameters to MATLAB Command Window
        fprintf('Real Machine Parameters \n \n Stator: \n Lmd= %.4f      Xmd = %.4f \n Lmq =
%.4f     Xmq = %.4f \n Llk = %.4f     Xlk = %.4f \n r_s = %.4f \n L_d=%.4f      L_q=%.4f \n\n
Field: \n r`_fd = %.4f \n L`_lfd = %.4f    X`_lfd = %.4f \n                    X_p_fd = %.4f
\n',...
            [Lmd_r X_md_r Lmq_r X_mq_r Llk_r X_ls_r r_s_r L_d_r L_q_r r_p_fd_r L_p_lfd_r
X_p_lfd_r X_p_fd_r])

     % Define Stator_winding/Field_winding transformation ratio:
        Ns_Nf = 0.07798;

load('Synch_Machine_Model_SIFundamental_SSv4.mat');
Synch_Machine_Model_OC = Synch_Machine_Model_SIFundamental;

    % Parse Data into variables
        t_OC = transpose(Synch_Machine_Model_OC(1,:)); % Timestamp
```

```
        Ifd = transpose(Synch_Machine_Model_OC(2,:)); % Field Current
        Vf = transpose(Synch_Machine_Model_OC(3,:)); % Field Voltage
        Is_a = transpose(Synch_Machine_Model_OC(4,:)); % Stator Current phase A
        Iq = transpose(Synch_Machine_Model_OC(5,:)); % Q axis Current
        Id = transpose(Synch_Machine_Model_OC(6,:)); % D axis Current
        Vd = transpose(Synch_Machine_Model_OC(7,:)); % D axis Voltage
        Vq = transpose(Synch_Machine_Model_OC(8,:)); % Q axis Voltage
        Vt = transpose(Synch_Machine_Model_OC(9,:)); % Terminal Voltage
        psimd = transpose(Synch_Machine_Model_OC(10,:)); % D axis mutual flux
        psimq = transpose(Synch_Machine_Model_OC(11,:)); % Q axis mutual flux
        Po = transpose(Synch_Machine_Model_OC(12,:)); % Output Real Power
        Qo = transpose(Synch_Machine_Model_OC(13,:)); % Output Reactive Power
        Wm = transpose(Synch_Machine_Model_OC(14,:)); % Rotor Mechanical Speed
        dWm = transpose(Synch_Machine_Model_OC(15,:)); % Change in Rotor Mechanical Speed
        theta_R = transpose(Synch_Machine_Model_OC(16,:)); % Rotor Mechanical Angle
        delta = transpose(Synch_Machine_Model_OC(17,:)); % Load Angle
        dtheta = transpose(Synch_Machine_Model_OC(18,:)); % Change in Rotor Mechanical Angle
        Ikq1 = transpose(Synch_Machine_Model_OC(19,:)); % Q axis damper winding current 1
        Ikq2 = transpose(Synch_Machine_Model_OC(20,:)); % Q axis damper winding current 2
        Ikd = transpose(Synch_Machine_Model_OC(21,:)); % D axis damper winding current
        I_abc_stator = transpose(Synch_Machine_Model_OC(22:24,:)); % ABC Phase Stator Currents

% Define measured voltages
V_r_qs = Vq(start_point:end);%ones(1,length(t))*mean(Vq);
V_r_ds = Vd(start_point:end);%ones(1,length(t))*mean(Vd);
V_pr_fd = Vf(start_point:end);%*Ns_Nf; %ones(1,length(t))*mean(Vf); % Simulink model has
specified nominal field current, so volts are entered in DC and therefore must be referred to
the stator
omega_b = omega_BASE; % Base frequency
omega_r = (n_poles/2)*Wm; % rotor speed [rad/sec] eq. 1D-8 pg 58

%% Convert inductances to reactances:
    % Stator
        X_md = Lmd_r*omega_BASE; %
        X_mq = Lmq_r*omega_BASE;
        X_ls = Llk_r*omega_BASE;

        X_d = X_md + X_ls; % Krause eq 5.5-39
        X_q = X_mq + X_ls; % Krause eq 5.5-40

    % Field
        X_p_lfd = L_p_lfd_r*omega_BASE;
        X_p_fd = X_p_lfd + X_md; % Krause eq 5.5-40


% Initialize flux linkages
    psi_r_mq(1) = psimq(start_point);
    psi_r_md(1) = psimd(start_point);
    psi_r_qs(1) = psi_r_mq(1) - Iq(start_point)*X_ls(1);
    psi_r_ds(1) = psi_r_md(1) - Id(start_point)*X_ls(1);
    psi_pr_fd(1) = psi_r_md(1) + Ifd(start_point)*(2/3)*Ns_Nf*X_ls(1);
    % Output Initial Conditions to MATLAB Command Window
    fprintf('Initial Conditions \n psi_mq = %.2f      psi_md = %.2f \n psi_r_qs = %.2f
psi_r_ds = %.2f \n psi_pr_fd = %.2f \n \n',[psi_r_mq psi_r_md psi_r_qs psi_r_ds psi_pr_fd])


    X_aq(1) = ((1/X_mq(1)) + (1/X_ls(1)))^-1;
    X_ad(1) = ((1/X_md(1)) + (1/X_ls(1)) + (1/X_p_lfd(1)))^-1;

    e_pr_xfd = V_pr_fd*(X_md(1)/r_p_fd_r); % Krause 5.5-36

    x_int(:,1) = [ psi_r_mq(1); psi_r_md(1); psi_r_qs(1); psi_r_ds(1); psi_pr_fd(1)];
    u(:,1) = [V_r_qs(1); V_r_ds(1); e_pr_xfd(1)];



%% Initialize the system based on initial parameters
    k = 1; % k = iteration, first iteration is the initialization
% Number of variables to be estimated
     nvar = 3;
% For trapezoidal integration
```

```
    dt = 1/8000;
    t(1) = 0;

    for n = 2:length(t_OC(start_point:end))


        % State Space Matrices
            A(:,:,k) = [0, 0, X_aq(k)/X_ls(k), 0, 0;...
                 0, 0, 0, X_ad(k)/X_ls(k), X_ad(k)/X_p_lfd(k);...
                 r_s_r/X_ls(k), 0, -r_s_r/X_ls(k), -omega_r(n), 0;...
                 0, r_s_r/X_ls(k), omega_r(n), -r_s_r/X_ls(k), r_s_r/X_ls(k);...
                 0, r_p_fd_r/X_p_lfd(k), 0, 0, -r_p_fd_r/X_p_lfd(k)];

            B(:,:,k) = [0 0 0;...
                 0 0 0;...
                 1 0 0;...
                 0 1 0;...
                 0 0 r_p_fd_r/X_md(k)];

            C(:,:,k) = [1/X_ls(k) 0 -1/X_ls(k) 0 0;...
                        0 1/X_ls(k) 0 -1/X_ls(k) 0;...
                        0 -1/X_p_lfd(k) 0 0 1/X_p_lfd(k)];

        % State Space Equations X = A*x + B*u
              u(:,n) = [V_r_qs(n); V_r_ds(n); e_pr_xfd(n)];

              % x_###(1,:) = psi_r_mq; % x_###(2,:) = psi_r_md;
              % x_###(3,:) = psi_r_qs; % x_###(4,:) = psi_r_ds;
              % x_###(5,:) = psi_r_fd;
              x_diff(:,n,k) = A(:,:,k)*x_int(:,n-1)+B(:,:,k)*u(:,n-1);
              x_int(:,n,k) = x_int(:,n-1) + 0.5*dt*(x_diff(:,n)-x_diff(:,n-1)); % Integrated
States

              % Calculate Outputs Y = Cx + v where v = 0 for now (no noise)
              i_state(:,n,k) = C(:,:,k)*x_int(:,n,k); % Outputs
              i_r_fd(1,n,k) = (3/2)*Ns_Nf^-1*i_state(3,n,k); % Field current referred back to
rotor

              % Increment Timestep
              t(n) = t(n-1)+dt;
    end

%% Plot Results
    figure(1)
        subplot(4,2,1)
        plot(t_OC(start_point:end),Vq(start_point:end))
        title('Simulink Q axis Voltage [V]')
        grid on
        ylabel('V')


        subplot(4,2,2)
        plot(t_OC(start_point:end),Vd(start_point:end))
        title('Simulink D axis Voltage [V]')
        grid on
        ylabel('V')


        subplot(4,2,3)
        plot(t,i_state(1,:,end))
        hold all
        plot(t_OC(start_point:end),Iq(start_point:end))
        title(sprintf('Q axis stator currents [A] \n error = %.2f %%',abs((mean(Iq)-
mean(i_state(1,:,end)))/mean(i_state(1,:,end)))*100))
        grid on
        ylabel('A')
        legend('Calculated','Simulink')

        subplot(4,2,4)
        plot(t,i_state(2,:,end))
        hold all
```

```
        plot(t_OC(start_point:end),Id(start_point:end))
        title(sprintf('D axis stator currents [A]\n error = %.2f %%',abs((mean(Id)-
mean(i_state(2,:,end)))/mean(i_state(2,:,end)))*100))
        grid on
        ylabel('A')
        legend('Calculated','Simulink')

        subplot(4,2,5)
        plot(t,x_int(1,:,end))
        hold all
        plot(t_OC(start_point:end),psimq(start_point:end))
        title(sprintf('Q axis psi_m_q \n error = %.2f %%',abs((mean(psimq)-
mean(x_int(1,:)))/mean(x_int(1,:)))*100))
        ylabel('Flux Linkage')
        grid on
        legend('Calculated','Simulink')

        subplot(4,2,6)
        plot(t,x_int(2,:,end))
        hold all
        plot(t_OC(start_point:end),psimd(start_point:end))
        title(sprintf('D axis psi_m_d \n error = %.2f %%',abs((mean(psimd)-
mean(x_int(2,:)))/mean(x_int(2,:)))*100))
        ylabel('Flux Linkage')
        grid on
        legend('Calculated','Simulink')



        subplot(4,1,4)
        plot(t,i_r_fd(1,:,end))
        hold all
        plot(t_OC(start_point:end), Ifd(start_point:end));
        title(sprintf('Field Current [A] \n error = %.2f %%',abs((mean(Ifd)-
mean(i_r_fd(1,:,end)))/mean(i_r_fd(1,:,end)))*100))
        xlabel('Time [s]')
        ylabel('Amps [A]')
        legend('Calculated','Simulink')
        grid on

  figure(2)
        plot(t,i_state(1,:,end))
        hold all
        plot(t, i_state(2,:,end))
        plot(t, i_state(3,:,end))
        grid on
        title('Calculated Currents')
        legend('Iqs','Ids','Ifd')
        xlabel('Time [s]')
        ylabel('Amps')
```

## Least Squares Estimation Implemented with 3rd Order Synchronous Machine Model

```
%% Author: Michael West
% Thesis: Online Dynamic Parameter Estimation of Synchronous Machines
% Purpose of Script: Implement the Least Squares Estimation Algorithm on a
% 3rd Order Synchronous Machine Model and Compare Results with Simulink
% Synchronous Machine Model.

clc
clear all
close all
tic

%% Machine Data/Stator Base values
V_sBASE = 13800*sqrt(2/3); %Base voltage
S_BASE = 187E6; %MVA BASE
Z_BASE = V_sBASE^2/S_BASE; %Impedance base
I_sBASE = sqrt(2)*S_BASE/(sqrt(3)*V_sBASE);

omega_BASE = 2*pi*60;
L_sBASE = Z_BASE/omega_BASE;

%% Field Base Values
I_fbase = 1087; % Amps
V_fdBASE = 226.6; % Voltages

        % Number of machine poles
        n_poles = 40;
        start_point = 1;

%% Real Machine Parameters;
    % Stator
        Lmd_r = 3.2164E-03; % D-axis mutual inductance [H]
        Lmq_r = 9.7153E-04; % Q-axis mutual inductance [H]
        Llk_r = 3.0892E-04; %[H]
        r_s_r = 2.9069E-03; % [Ohm]

        L_d_r = Llk_r+Lmd_r;
        L_q_r = Llk_r+Lmq_r;
    % Field
        r_p_fd_r = 1.9013E-03; % field resistance [Ohm]
        L_p_lfd_r = 3.0712E-04; % field leakage inductance [H]

 % Convert inductances to reactances:
    % Stator
        X_md_r = Lmd_r*omega_BASE; %
        X_mq_r = Lmq_r*omega_BASE;
        X_ls_r = Llk_r*omega_BASE;

        X_d_r = X_md_r + X_ls_r; % Krause eq 5.5-39
        X_q_r = X_mq_r + X_ls_r; % Krause eq 5.5-40

    % Field
        X_p_lfd_r = L_p_lfd_r*omega_BASE;
        X_p_fd_r = X_p_lfd_r + X_md_r; % Krause eq 5.5-40

 % Output real machine parameters to MATLAB Command Window
        fprintf('Real Machine Parameters \n \n Stator: \n Lmd= %.4f      Xmd = %.4f \n Lmq =
%.4f     Xmq = %.4f \n Llk = %.4f     Xlk = %.4f \n r_s = %.4f \n L_d=%.4f      L_q=%.4f \n\n
Field: \n r`_fd = %.4f \n L`_lfd = %.4f    X`_lfd = %.4f \n                     X_p_fd = %.4f
\n',...
            [Lmd_r X_md_r Lmq_r X_mq_r Llk_r X_ls_r r_s_r L_d_r L_q_r r_p_fd_r L_p_lfd_r
X_p_lfd_r X_p_fd_r])

     % Define Stator_winding/Field_winding transformation ratio:
        Ns_Nf = 0.07798;

%% Load Open Circuit Machine Data
load('Synch_Machine_Model_SIFundamental_OC.mat');
Synch_Machine_Model_OC = Synch_Machine_Model_SIFundamental;
```

```
    % Parse Data into variables
        t_OC = transpose(Synch_Machine_Model_OC(1,:)); % Timestamp
        Ifd = transpose(Synch_Machine_Model_OC(2,:)); % Field Current
        Vf = transpose(Synch_Machine_Model_OC(3,:)); % Field Voltage
        Is_a = transpose(Synch_Machine_Model_OC(4,:)); % Stator Current phase A
        Iq = transpose(Synch_Machine_Model_OC(5,:)); % Q axis Current
        Id = transpose(Synch_Machine_Model_OC(6,:)); % D axis Current
        Vd = transpose(Synch_Machine_Model_OC(7,:)); % D axis Voltage
        Vq = transpose(Synch_Machine_Model_OC(8,:)); % Q axis Voltage
        Vt = transpose(Synch_Machine_Model_OC(9,:)); % Terminal Voltage
        psimd = transpose(Synch_Machine_Model_OC(10,:)); % D axis mutual flux
        psimq = transpose(Synch_Machine_Model_OC(11,:)); % Q axis mutual flux
        Po = transpose(Synch_Machine_Model_OC(12,:)); % Output Real Power
        Qo = transpose(Synch_Machine_Model_OC(13,:)); % Output Reactive Power
        Wm = transpose(Synch_Machine_Model_OC(14,:)); % Rotor Mechanical Speed
        dWm = transpose(Synch_Machine_Model_OC(15,:)); % Change in Rotor Mechanical Speed
        theta_R = transpose(Synch_Machine_Model_OC(16,:)); % Rotor Mechanical Angle
        delta = transpose(Synch_Machine_Model_OC(17,:)); % Load Angle
        dtheta = transpose(Synch_Machine_Model_OC(18,:)); % Change in Rotor Mechanical Angle
        Ikq1 = transpose(Synch_Machine_Model_OC(19,:)); % Q axis damper winding current 1
        Ikq2 = transpose(Synch_Machine_Model_OC(20,:)); % Q axis damper winding current 2
        Ikd = transpose(Synch_Machine_Model_OC(21,:)); % D axis damper winding current
        I_abc_stator = transpose(Synch_Machine_Model_OC(22:24,:)); % ABC Phase Stator Currents



 %% Calculate Lmd based on open circuit measurements
for n = 1:length(Ifd)
    Lmd_calc(n) = (1/omega_BASE)*Vq(n)/((2/3)*(1/Ns_Nf)*Ifd(n));

    % calculate error
    Lmd_calc_error(n) = ((Lmd_calc(n) - Lmd_r) /Lmd_r)*100;
end
 figure(1)
        plot(t_OC,Lmd_calc)
        hold all
        plot(t_OC, ones(1,length(Lmd_calc))*Lmd_r)
        title(sprintf('Lmd Calculated \n average error = %.2f %%',abs(mean(Lmd_calc_error))))
        xlabel('Time [s]')
        ylabel('H')
        legend('Calculated','Actual')
        grid on




%% Initial Guess Machine Parameters
% Stator
        Lmd = mean(Lmd_calc)*1.0; % D-axis mutual inductance [H]
        Lmq = Lmq_r*1.0; % Q-axis mutual inductance [H]
        Llk = Llk_r*1.0; %[H]
        r_s = r_s_r*1.0; % [Ohm]

        L_d = Llk+Lmd;
        L_q = Llk+Lmq;
    % Field
        r_p_fd = r_p_fd_r*1.00; % field resistance [Ohm]
        L_p_lfd = L_p_lfd_r*1.00; % field leakage inductance [H]

%% Load steady-state Machine Data
load('Synch_Machine_Model_SIFundamental_SSv4.mat');
Synch_Machine_Model_OC = Synch_Machine_Model_SIFundamental;

    % Parse Data into variables
        t_OC = transpose(Synch_Machine_Model_OC(1,:)); % Timestamp
        Ifd = transpose(Synch_Machine_Model_OC(2,:)); % Field Current
        Vf = transpose(Synch_Machine_Model_OC(3,:)); % Field Voltage
        Is_a = transpose(Synch_Machine_Model_OC(4,:)); % Stator Current phase A
        Iq = transpose(Synch_Machine_Model_OC(5,:)); % Q axis Current
        Id = transpose(Synch_Machine_Model_OC(6,:)); % D axis Current
```

```
        Vd = transpose(Synch_Machine_Model_OC(7,:)); % D axis Voltage
        Vq = transpose(Synch_Machine_Model_OC(8,:)); % Q axis Voltage
        Vt = transpose(Synch_Machine_Model_OC(9,:)); % Terminal Voltage
        psimd = transpose(Synch_Machine_Model_OC(10,:)); % D axis mutual flux
        psimq = transpose(Synch_Machine_Model_OC(11,:)); % Q axis mutual flux
        Po = transpose(Synch_Machine_Model_OC(12,:)); % Output Real Power
        Qo = transpose(Synch_Machine_Model_OC(13,:)); % Output Reactive Power
        Wm = transpose(Synch_Machine_Model_OC(14,:)); % Rotor Mechanical Speed
        dWm = transpose(Synch_Machine_Model_OC(15,:)); % Change in Rotor Mechanical Speed
        theta_R = transpose(Synch_Machine_Model_OC(16,:)); % Rotor Mechanical Angle
        delta = transpose(Synch_Machine_Model_OC(17,:)); % Load Angle
        dtheta = transpose(Synch_Machine_Model_OC(18,:)); % Change in Rotor Mechanical Angle
        Ikq1 = transpose(Synch_Machine_Model_OC(19,:)); % Q axis damper winding current 1
        Ikq2 = transpose(Synch_Machine_Model_OC(20,:)); % Q axis damper winding current 2
        Ikd = transpose(Synch_Machine_Model_OC(21,:)); % D axis damper winding current
        I_abc_stator = transpose(Synch_Machine_Model_OC(22:24,:)); % ABC Phase Stator Currents



 %----------------------------------%
 %----------------------------------%
% Define measured voltages
V_r_qs = Vq(start_point:end);%ones(1,length(t))*mean(Vq);
V_r_ds = Vd(start_point:end);%ones(1,length(t))*mean(Vd);
V_pr_fd = Vf(start_point:end);%*Ns_Nf; %ones(1,length(t))*mean(Vf); % Simulink model has
specified nominal field current, so volts are entered in DC and therefore must be referred to
the stator
omega_b = omega_BASE; % Base frequency
omega_r = (n_poles/2)*Wm; % rotor speed [rad/sec] eq. 1D-8 pg 58
 % Initialze LSE
R = [1; 1; 1]; % for positive semi-definite weight

% Convert inductances to reactances:
    % Stator
        X_md(1) = Lmd(1)*omega_BASE; %
        X_mq(1) = Lmq(1)*omega_BASE;
        X_ls(1) = Llk(1)*omega_BASE;

        X_d(1) = X_md(1) + X_ls(1); % Krause eq 5.5-39
        X_q(1) = X_mq(1) + X_ls(1); % Krause eq 5.5-40

    % Field
        X_p_lfd(1) = L_p_lfd(1)*omega_BASE;
        X_p_fd(1) = X_p_lfd(1) + X_md(1); % Krause eq 5.5-40


%% Initialize flux linkages
    psi_r_mq(1) = psimq(start_point);
    psi_r_md(1) = psimd(start_point);
    psi_r_qs(1) = psi_r_mq(1) - Iq(start_point)*X_ls(1);
    psi_r_ds(1) = psi_r_md(1) - Id(start_point)*X_ls(1);
    psi_pr_fd(1) = psi_r_md(1) + Ifd(start_point)*(2/3)*Ns_Nf*X_ls(1);
    % Output Initial Conditions to MATLAB Command Window
    fprintf('Initial Conditions \n psi_mq = %.2f      psi_md = %.2f \n psi_r_qs = %.2f
psi_r_ds = %.2f \n psi_pr_fd = %.2f \n \n',[psi_r_mq psi_r_md psi_r_qs psi_r_ds psi_pr_fd])


    X_aq(1) = ((1/X_mq(1)) + (1/X_ls(1)))^-1;
    X_ad(1) = ((1/X_md(1)) + (1/X_ls(1)) + (1/X_p_lfd(1)))^-1;

    e_pr_xfd = V_pr_fd*(X_md(1)/r_p_fd); % Krause 5.5-36

    x_int(:,1) = [ psi_r_mq(1); psi_r_md(1); psi_r_qs(1); psi_r_ds(1); psi_pr_fd(1)];
    u(:,1) = [V_r_qs(1); V_r_ds(1); e_pr_xfd(1)];



%% Initialize the system based on initial parameters
    k = 1; % k = iteration, first iteration is the initialization
% Number of variables to be estimated
     nvar = 3;
```

```
% For trapezoidal integration
    dt = 1/8000;
    t(1) = 0;

fprintf('%%---Initializing Machine Model---%%\n')
    for n = 2:length(t_OC(start_point:end))


        % State Space Matrices
            A(:,:,k) = [0, 0, X_aq(k)/X_ls(k), 0, 0;...
                 0, 0, 0, X_ad(k)/X_ls(k), X_ad(k)/X_p_lfd(k);...
                 r_s/X_ls(k), 0, -r_s/X_ls(k), -omega_r(n), 0;...
                 0, r_s/X_ls(k), omega_r(n), -r_s/X_ls(k), r_s/X_ls(k);...
                 0, r_p_fd/X_p_lfd(k), 0, 0, -r_p_fd/X_p_lfd(k)];

            B(:,:,k) = [0 0 0;...
                 0 0 0;...
                 1 0 0;...
                 0 1 0;...
                 0 0 r_p_fd/X_md(k)];

            C(:,:,k) = [1/X_ls(k) 0 -1/X_ls(k) 0 0;...
                        0 1/X_ls(k) 0 -1/X_ls(k) 0;...
                        0 -1/X_p_lfd(k) 0 0 1/X_p_lfd(k)];

        % State Space Equations X = A*x + B*u
              u(:,n) = [V_r_qs(n); V_r_ds(n); e_pr_xfd(n)];

              % x_###(1,:) = psi_r_mq; % x_###(2,:) = psi_r_md;
              % x_###(3,:) = psi_r_qs; % x_###(4,:) = psi_r_ds;
              % x_###(5,:) = psi_r_fd;
              x_diff(:,n,k) = A(:,:,k)*x_int(:,n-1)+B(:,:,k)*u(:,n-1);
              x_int(:,n,k) = x_int(:,n-1) + 0.5*dt*(x_diff(:,n)-x_diff(:,n-1)); % Integrated
States

              % Calculate Outputs Y = Cx + v where v = 0 for now (no noise)
              i_state(:,n,k) = C(:,:,k)*x_int(:,n,k); % Outputs
              i_r_fd(1,n,k) = (3/2)*Ns_Nf^-1*i_state(3,n,k); % Field current referred back to
rotor

              % Increment Timestep
              t(n) = t(n-1)+dt;

              % Output Status of Initialization to Command Window
              if n == round(length(t_OC(start_point:end))/3)
                  fprintf('\n33%% Initialized\n')
              elseif n == round(length(t_OC(start_point:end))*(2/3))
                  fprintf('66%% Initialized\n')
              elseif n == length(t_OC)
                  fprintf('99%% Initialized\n')
              end
    end

%% Least Squares Estigmation Loop
    iterations = 10;
    num_int(1:nvar,1,1) = .001;
    den_int(1:nvar,1,1) = .001;
% Initialize C.C. Lee Equation (11)
    alpha_hat(:,1) = [Lmq; Lmd; Llk;];

    g(1:nvar,1) = 1; % Initialize gains
    fprintf('Initializing Complete \n\n%%---Beginning Estimation Algorithm---%%...\n')
    h = waitbar(0,'Approximating Machine Parameters');

    % Define a matrix of "real" machine currents (C.C. Lee y_r)
     for n = 2:length(t_OC(start_point:end));

         i_r(:,n) = [Iq(n); Id(n); Ifd(n)*(2/3)*Ns_Nf]; % real machine currents
     end

% Begin LSE Loop
```

```
delta_alpha(1:nvar,1) = 0;
for k = 2:iterations


       % Calculate gains
    for ii = 1:length(delta_alpha(:,k))
        if abs(delta_alpha(ii,k-1)/alpha_hat(ii,k-1)) < 0.25
            g(ii,k) = 0.5;
        elseif abs(delta_alpha(ii,k-1)/alpha_hat(ii,k-1)) >= 0.25
            g(ii,k) = 0.1*alpha_hat(ii,k-1)/abs(delta_alpha(ii,k-1));
        end
    end
%         g(n,1:3,k) = 0.0001;
    %% Update Parameters (alpha) C.C. Lee Eq (11)
        alpha_hat(:,k) = alpha_hat(:,k-1) + g(:,k).*delta_alpha(:,k-1);

        Lmq(k) = alpha_hat(1,k);
        Lmd(k) = alpha_hat(2,k);
%         Lmd(n,k) = mean(Lmd_calc);
        Llk(k) = alpha_hat(3,k);

        X_md = Lmd(k)*omega_BASE;
        X_mq = Lmq(k)*omega_BASE;
        X_ls = Llk(k)*omega_BASE;

        X_d = X_md + X_ls; % Krause eq 5.5-39
        X_q = X_mq + X_ls; % Krause eq 5.5-40

        X_aq = ((1/X_mq) + (1/X_ls))^-1;
        X_ad = ((1/X_md) + (1/X_ls) + (1/X_p_lfd))^-1;

    % Update Initial flux linkages
        psi_r_mq(1) = psimq(start_point);
        psi_r_md(1) = psimd(start_point);
        psi_r_qs(1) = psi_r_mq(1) - Iq(start_point)*X_ls;
        psi_r_ds(1) = psi_r_md(1) - Id(start_point)*X_ls;
        psi_pr_fd(1) = psi_r_md(1) + Ifd(start_point)*(2/3)*Ns_Nf*X_ls;


    for n = 2:length(t_OC(start_point:end))
        % Update State-Space Matrices
        A(:,:,k) = [0, 0, X_aq/X_ls, 0, 0;...
             0, 0, 0, X_ad/X_ls, X_ad/X_p_lfd;...
             r_s/X_ls, 0, -r_s/X_ls, -omega_r(n), 0;...
             0, r_s/X_ls, omega_r(n), -r_s/X_ls, r_s/X_ls;...
             0, r_p_fd/X_p_lfd, 0, 0, -r_p_fd/X_p_lfd];

        B(:,:,k) = [0 0 0;...
             0 0 0;...
             1 0 0;...
             0 1 0;...
             0 0 r_p_fd/X_md];

        C(:,:,k) = [1/X_ls 0 -1/X_ls 0 0;...
                    0 1/X_ls 0 -1/X_ls 0;...
                    0 -1/X_p_lfd 0 0 1/X_p_lfd];

        %% State Space Equations X = A*x + B*u
            u(:,n) = [V_r_qs(n); V_r_ds(n); e_pr_xfd(n)];

            % x_###(1,:) = psi_r_mq; % x_###(2,:) = psi_r_md;
            % x_###(3,:) = psi_r_qs; % x_###(4,:) = psi_r_ds;
            % x_###(5,:) = psi_r_fd;
            x_diff(:,n,k) = A(:,:,k)*x_int(:,n-1)+B(:,:,k)*u(:,n-1);
            x_int(:,n,k) = x_int(:,n-1) + 0.5*dt*(x_diff(:,n)-x_diff(:,n-1)); % Integrated
States

            % Calculate Outputs Y = Cx + v where v = 0 for now (no noise)
            i_state(:,n,k) = C(:,:,k)*x_int(:,n,k); % Outputs
            i_r_fd(1,n,k) = (3/2)*Ns_Nf^-1*i_state(3,n,k); % Field current referred back to
rotor
```

```
        %% LSE EQUATIONS
            % Calculate change in input states, and A,B,C Matrices w.r.t
            dx_dA_alpha0(:,n,k) = (x_int(:,n,k) - x_int(:,n, k-1));
            dA_dalpha_alpha0(:,:,k) = A(:,:,k) - A(:,:,k-1);
            dB_dalpha_alpha0(:,:,k) = B(:,:,k) - B(:,:,k-1);
            dC_dalpha_alpha0(:,:,k) = C(:,:,k) - C(:,:,k-1);

            % C.C. Lee Equation (6);
            eq_6(:,n,k) = A(:,:,k)*dx_dA_alpha0(:,n,k) +
dA_dalpha_alpha0(:,:,k)*x_int(:,n,k)...
                    + dB_dalpha_alpha0(:,:,k)*u(:,n);

            % C.C. Lee Equation (5)
            dy_dalpha_alpha0(:,n,k) = C(:,:,k)*eq_6(:,n,k) +
dC_dalpha_alpha0(:,:,k)*x_int(:,n,k);
%             dy_dalpha_alpha0(:,n,k) = C(:,:,k-1)*dx_dA_alpha0(:,n,k) +
dC_dalpha_alpha0(:,:,k)*x_int(:,n,k-1);

            if n == 2;
               % Initialze C.C. Lee Eq (9) numerator and denominator integrals
                    num_int(1:nvar,1,k) = .001;
                    den_int(1:nvar,1,k) = .001;
            end
            % C.C. Lee Equation (9)
            num(:,n,k) = dy_dalpha_alpha0(:,n,k)'*R*(i_r(:,n) - i_state(:,n,k));
            num_int(:,n,k) = num_int(:,n-1,k) + 0.5*dt*(num(:,n,k)- num(:,n-1,k));


            den(:,n,k) = dy_dalpha_alpha0(:,n,k)'*R*dy_dalpha_alpha0(:,n,k);
            den_int(:,n,k) = den_int(:,n-1,k) + 0.5*dt*(den(:,n,k) - den(:,n-1,k));


            % Increment Timestep
            t(n) = t(n-1)+dt;


    end
     % Calculate delta_alpha (alpha_hat - alpha0 in C.C. Lee eq (9))
        delta_alpha(:,k) = num_int(:,end,k) ./ (den_int(:,end,k)+0.000001); % add to
denominator to avoid divide by 0

    % save num_int and den_int values
    num_int_save(:,k) = num_int(:,end,k);
    den_int_save(:,k) = den_int(:,end,k);
    waitbar(k/iterations,h,sprintf('Approximating Machine Parameters \n %.3f %%
Complete',(k/iterations)*100));
end
L_d = Llk(end)+Lmd(end);
        L_q = Llk(end)+Lmq(end);
% Output estimated machine parameters to MATLAB Command Window
        fprintf('Estimated Machine Parameters \n Lmd= %.4f \n Lmq = %.4f \n Llk = %.4f \n
L_d=%.4f L_q=%.4f\n',[Lmd(end,end) Lmq(end,end) Llk(end,end) L_d L_q])
    close(h)

    figure(1)
        subplot(4,2,1)
        plot(t_OC(start_point:end),Vq(start_point:end))
        title('Simulink Q axis Voltage [V]')
        grid on
        ylabel('V')


        subplot(4,2,2)
        plot(t_OC(start_point:end),Vd(start_point:end))
        title('Simulink D axis Voltage [V]')
        grid on
        ylabel('V')


        subplot(4,2,3)
```

```
        plot(t,i_state(1,:,end))
        hold all
        plot(t_OC(start_point:end),Iq(start_point:end))
        title(sprintf('Q axis stator currents [A] \n error = %.2f %%',abs((mean(Iq)-
mean(i_state(1,:,end)))/mean(i_state(1,:,end)))*100))
        grid on
        ylabel('A')
        legend('Calculated','Simulink')

        subplot(4,2,4)
        plot(t,i_state(2,:,end))
        hold all
        plot(t_OC(start_point:end),Id(start_point:end))
        title(sprintf('D axis stator currents [A]\n error = %.2f %%',abs((mean(Id)-
mean(i_state(2,:,end)))/mean(i_state(2,:,end)))*100))
        grid on
        ylabel('A')
        legend('Calculated','Simulink')

        subplot(4,2,5)
        plot(t,x_int(1,:,end))
        hold all
        plot(t_OC(start_point:end),psimq(start_point:end))
        title(sprintf('Q axis psi_m_q \n error = %.2f %%',abs((mean(psimq)-
mean(x_int(1,:)))/mean(x_int(1,:)))*100))
        ylabel('Flux Linkage')
        grid on
        legend('Calculated','Simulink')

        subplot(4,2,6)
        plot(t,x_int(2,:,end))
        hold all
        plot(t_OC(start_point:end),psimd(start_point:end))
        title(sprintf('D axis psi_m_d \n error = %.2f %%',abs((mean(psimd)-
mean(x_int(2,:)))/mean(x_int(2,:)))*100))
        ylabel('Flux Linkage')
        grid on
        legend('Calculated','Simulink')



        subplot(4,1,4)
        plot(t,i_r_fd(1,:,end))
        hold all
        plot(t_OC(start_point:end), Ifd(start_point:end));
        title(sprintf('Field Current [A] \n error = %.2f %%',abs((mean(Ifd)-
mean(i_r_fd(1,:,end)))/mean(i_r_fd(1,:,end)))*100))
        xlabel('Time [s]')
        ylabel('Amps [A]')
        legend('Calculated','Simulink')
        grid on

  figure(2)
        plot(t,i_state(1,:,end))
        hold all
        plot(t, i_state(2,:,end))
        plot(t, i_state(3,:,end))
        grid on
        title('Calculated Currents')
        legend('Iqs','Ids','Ifd')
        xlabel('Time [s]')
        ylabel('Amps')

% create vectors to plot the real values
    Lmd_rplot = ones(1,k)*Lmd_r;
    Lmq_rplot = ones(1,k)*Lmq_r;
    Llk_rplot = ones(1,k)*Llk_r;
    Xd_rplot = ones(1,k)*X_d_r;
    Xq_rplot = ones(1,k)*X_q_r;

    for nn = 1:k
```

```
        Lmd_error(nn) = abs(((Lmd_rplot(1,nn) - Lmd(end,nn))/Lmd(end,nn))*100);
        Lmq_error(nn) = abs(((Lmq_rplot(1,nn) - Lmq(end,nn))/Lmq(end,nn))*100);
        Llk_error(nn) = abs(((Llk_rplot(1,nn) - Llk(end,nn))/Llk(end,nn))*100);
    end

    figure
        subplot(2,1,1)
        plot(Lmd(end,:))
        hold all
        plot(Lmq(end,:))
        plot(Llk(end,:))
        plot(Lmd_rplot)
        plot(Lmq_rplot)
        plot(Llk_rplot)
        grid on
        xlabel('Iteration')
        ylabel('Estimation')
        legend('Lmd','Lmq','Llk','Actual Lmd','Actual Lmq','Actual Llk')
        title('Estimated Machine Parameters')

        subplot(2,1,2)
        plot(Lmd_error)
        hold all
        plot(Lmq_error)
        plot(Llk_error)
        legend('Lmd error','Lmq error','Llk error')
        xlabel('Iterations')
        ylabel('% error')
        title('Estimated Inductance Error')
        grid on

    figure
        plot(Lmd(end,:)/Lmd_r)
        hold all
        plot(Lmq(end,:)/Lmq_r)
        plot(Llk(end,:)/Llk_r)
        grid on
        xlabel('Iterations')
        ylabel('Ratio')
        title('Ratio of estimated to real parameters')
        legend('Lmd','Lmq','Llk')
fprintf('Estimation Complete \n')
toc
```