ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

ELECTROMECHANICAL ENGINEERING

PROYECTO FIN DE GRADO

IMPLEMENTATION OF NON-INTRUSIVE PARAMETER ESTIMATION ALGORITHM FOR SINGLE-PHASE INDUCTION MOTORS USING TRANSIENT DATA

Autor: Pilar Serrano Ojeda

Director: Paul Scott Carney

(University of Illinois at Urbana-Champaign)

Madrid

Junio 2015

Proyecto realizado por el alumno/a:

Pilar Serrano Ojeda

Fdo.: …………………… Fecha: ……/ ……/ ……

Autorizada la entrega del proyecto cuya información no es de carácter confidencial

EL DIRECTOR DEL PROYECTO

Professor Paul Scott Carney

Fdo.: [illegible] Fecha: 03/ 06/ 2015

Vº Bº del Coordinador de Proyectos

Fernando de Cuadra

Fdo.: …………………… Fecha: ……/ ……/ ……

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

ELECTROMECHANICAL ENGINEERING

PROYECTO FIN DE GRADO

IMPLEMENTATION OF NON-INTRUSIVE PARAMETER ESTIMATION ALGORITHM FOR SINGLE-PHASE INDUCTION MOTORS USING TRANSIENT DATA

Autor: Pilar Serrano Ojeda

Director: Paul Scott Carney

(University of Illinois at Urbana-Champaign)

Madrid

Junio 2015

IMPLEMENTACIÓN DEL ALGORITMO PARA LA ESTIMACIÓN NO-INTRUSIVA DE LOS PARÁMETROS DE LOS MOTORES DE INDUCCIÓN MONOFÁSICOS A PARTIR DE LAS MEDIDAS TRANSITORIAS

Autor: Serrano Ojeda, Pilar

Director de proyecto: Carney, Scott

Entidad Colaboradora: ICAI – Universidad Pontificia Comillas.

RESUMEN DE PROYECTO

1. Introducción

El uso de motores de inducción monofásicos para aplicaciones residenciales ha dado lugar a la aparición de un tipo de fenómeno denominado *Fault-Induced Delayed Voltage Recovery* (FIDVR). El análisis y mitigación de este tipo de sucesos requieren la mejora del modelo dinámico de los motores monofásicos.

Este trabajo representa una parte de un proyecto más amplio orientado al problema de la estimación de los parámetros de los motores de inducción monofásicos mediante las mediciones de la tensión y corriente transitorias en el proceso de arranque. Estas mediciones se tomarán en las terminales del motor. Este enfoque para la estimación no intrusiva se basará en la representación dinámica espacio-temporal para obtener información acerca de los transitorios del tipo específico de motor. Esto se ve implementado por el establecimiento de los parámetros de interés, que se asumen constantes durante el proceso de arranque.

2. Metodología y resultados

El objetivo a alcanzar en este trabajo es la construcción de una plataforma física para recoger las medidas requeridas y desarrollar el modelo espacio-temporal para nuestro motor de inducción monofásico. También vamos a realizar la identificación de parámetros para validar si el resultado de la estimación es correcto.

La tarea inicial es desarrollar la plataforma física con el fin de obtener la información acerca de la corriente y el voltaje que se aplica a las terminales del motor. Dado que se necesita un microcontrolador para enviar esta información al ordenador, tendremos que aplicar la transformación conveniente a esas medidas para que el microcontrolador pueda procesar los datos. Los límites de nuestro microcontrolador son 0 – 5 Voltios. Debido a esto, tanto los valores de la corriente como de la tensión deben estar dentro de este límite. Más tarde, el microcontrolador deshará estos cambios para obtener los valores reales y trabajar con ellos cuando sea preciso.

La siguiente tarea a realizar es el desarrollo del modelo espacio-temporal para el tipo específico de motor que vamos a utilizar. En este caso, el motor proporcionado se trata de un motor de inducción monofásico de arranque por condensador. Para desarrollar un modelo dinámico para un estudio exhaustivo, es necesario disponer de información sobre la distribución de los motores a lo largo de los alimentadores del elemento a tratar, así como la estructura interna y los valores de los parámetros del motor. Esta tarea se desarrolla en el Apéndice C. Una vez desarrollado el modelo específico en el que estamos interesados y habiendo proporcionado los parámetros constantes del motor, seremos capaces de obtener los resultados de la simulación del motor.

Por último, en el Apéndice B se han desarrollado los procedimientos para estimar los parámetros del motor. Con este propósito y trabajando con la estructura específica del motor de inducción monofásico de arranque por condensador, se ha desarrollado un método en base a la realización de diferentes ensayos en el estator y el rotor del motor. Estos ensayos son: ensayo de corriente continua, ensayo sin carga y ensayo de rotor bloqueado. Estos parámetros nos permitirán tener una idea general acerca de la magnitud de los mismos y serán útiles para comprobar la validez de los resultados finales.

3. Conclusiones y futuras tareas

El trabajo realizado hasta ahora nos permitirá leer el voltaje y la corriente transitoria que fluyen a través del motor de inducción monofásico y transmitir esta información a la computadora para el correspondiente análisis.

En el Apéndice B se ha llevado a cabo el procedimiento para la identificación de los parámetros del motor. De esta forma, podremos comprobar posteriormente si el resultado de la estimación final es correcto.

En el Apéndice C se ha desarrollado el modelo espacio-temporal de la máquina de inducción monofásica de arranque por condensador. Esto permitirá realizar la simulación del motor haciendo uso del código.

En el Apéndice D se muestran los diferentes códigos que dan forma a nuestro programa para la simulación del motor.

Para futuras tareas, los avances obtenidos hasta este punto se utilizarán con el propósito final de la estimación no intrusiva de los parámetros de las máquinas de inducción mediante el uso de datos transitorios. Se usará la plataforma física y el modelo del motor para realizar la estimación en tiempo real de los parámetros. Esto significa que a través del circuito se recogerán los datos, se enviarán al ordenador, y el ordenador procesará los datos de inmediato para llevar a cabo el cálculo de los parámetros de interés.

Gracias a este trabajo seremos capaces de encontrar la manera de mitigar los efectos del *Fault-Induced Delayed Voltage Recovery* (FIDVR).

IMPLEMENTATION OF NON-INTRUSIVE PARAMETER ESTIMATION ALGORITHM FOR SINGLE-PHASE INDUCTION MOTORS USING TRANSIENT DATA

PROJECT SUMMARY

1. Introduction

The use of single-phase induction motors (SPIMs) for residential applications has led to the occurrence of so-termed Fault-Induced Delayed Voltage Recovery (FIDVR) events. The analysis and mitigation of FIDVR events require improved dynamic modeling of SPIMs.

This paper represents part of a larger project oriented to the parameter estimation problem for single-phase induction motors (SPIMs) using start-up transient voltage and current measurements available at the motor terminals. This non-intrusive estimation approach will rely on a dynamic state-space representation for modeling the SPIM transients, which is augmented by the parameters of interest assumed to be constant quantities during start-up.

2. Methodology and Results

The objective to be achieved in this paper is to build a hardware platform to collect the transient measurements and develop the state-space model for our specific single phase induction motor. We will also perform parameters identification to validate if the estimation result is correct.

The initial task was to develop the hardware platform with the purpose of obtaining the transient values of the current and voltage that are applied to the motor terminals. Since a microcontroller is needed to send this information to the computer, we will need to apply the convenient transformation to those measurements so that the microcontroller can process the data. The limits of our microcontroller are 0 – 5Volts. Due to this, both the values of the current and voltage must be within this limit. Later, the microcontroller will undo this transformation to obtain the real values and work with them as it is precise.

The next thing that had to be done was to develop the model for the specific kind of motor we were going to use. In this case, the motor we were provided with was a Capacitor – Start Single Phase Induction Motor. To develop a dynamic model for a FIDVR study, it is necessary to have information regarding the distribution of SPIMs along the feeder, as well as the internal motor winding parameter values of SPIMs. This task is developed in Appendix C (State-space CSSPIM model). Once developed the specific model in which we are interested and providing the constant parameters comprising the motor, we will be able to obtain the results of the simulation of the motor.

Finally, in Appendix B has been developed the procedures to estimate the SPIM parameters. With this purpose and working with the specific structure of Capacitor-Start single phase induction motor, a method has been developed based on the performance of different tests to the stator and rotor of the motor. These tests are: DC test, no-load test and locked-rotor test. This parameters will allow us to have a general idea about the magnitude of the parameters and will be useful to check the validity of the final results.

3. Conclusions and future work

The work done so far will allow us to read the transient voltage and current that are flowing through the single phase induction motor and transmit this information to the computer for the following analysis.

In *Apendix B* we have performed the procedure for the parameter´s identification to validate if the final estimation result is correct.

In *Apendix C* the state-space CSSPIM model has been developed for the following simulation of the motor through our code.

In *Apendix D* we show the different codes that shape our program for the motor simulation.

For future work, the progress achieved up to this point will be used for the final purpose of implementation of Non-intrusive Parameter Estimation Algorithm for Single-phase Induction Motors Using Transient Data. We will want to use the hardware components and the model for the Capacitor-Start SPIM to perform real time estimation of the parameters. This means the circuit will collect the data, send it to the computer, and the computer will process the data immediately to carry out the calculation of the parameters of interest.

Thanks to all this work we will be able to find the way to mitigate the effects of Fault-Induced Delayed Voltage Recovery (FIDVR) phenomenon.

Contents

1 Introduction

1.1 Statement of Purpose

For reasons of economy, most homes, offices, and rural areas are supplied with single-phase AC, as the power requirements of individual load items are rather small. This has led to the availability of a wide variety of small-size fractional horse power induction motors (IMs), employed in fans, refrigerators, mixers, vacuum cleaners, washing machines, kitchen appliances, etc. Usually, single-phase IMs (SPIMs) are small enough to enable direct loading tests to be taken quite economically. However, for design improvement, it is necessary to calculate the performance variables.

Although SPIMs are simpler in construction as compared to their three-phase counterparts, their analysis happens to be more complex. IEEE Standards 114–1982 and 114–2001 are available for testing SPIM; however, these do not provide the method for extracting the parameters of permanent-split capacitor-start SPIM (CSSPIM).

The parameter estimation problem has been investigated for decades but most proposed methods were based on three phase induction motors. This method is applied to SPIMs, which have simpler physical structure but more complicated analysis due to non-linear dynamic equation. Furthermore, a non-intrusive method to estimate the SPIM parameters can be applied to understand the post-fault transients at the distribution feeders to mitigate the effects of Fault-Induced Delayed Voltage Recovery phenomenon.

The Fault-Induced Delayed Voltage Recovery (FIDVR) phenomenon refers to a significant depression of distribution system voltage that typically lasts for several seconds or minutes after a fault is cleared. The stalling of SPIMs that are distributed along the feeder is the main cause of FIDVR. The analysis and mitigation of FIDVR events require improved dynamic modeling of SPIMs.

To develop a dynamic model for a FIDVR study, it is necessary to have information regarding the distribution of SPIMs along the feeder, as well as the internal motor winding parameter values of SPIMs. These are useful for constructing a composite dynamic model of both the power system and the SPIMs used in electro-magnetic transients (EMTs) simulations. This points out the importance of the parameter estimation problem, for those SPIMs that are already installed in houses. Existing approaches on this problem have mostly been developed in an off-line environment, by testing the SPIMs under various operating conditions as DC or no-load scenarios. These testing-based methods are very useful for motor design and fault diagnostics. However, they fall in short in addressing the residential SPIM parameter estimation problem for FIDVR modeling, since it is labor-intensive for power system operators to directly test every installed SPIM.

It is useful to develop a non-intrusive approach for identifying SPIM parameters. The stator current, as output to the motor dynamic model, would depend on the SPIM winding parameters and the input terminal voltage. This motivates us to identify the motor parameters by observing its start-up output current and input voltage at the motor terminals. Thanks to increasing metering deployment in distribution systems, such a measurement-based approach becomes possible and requires minimal laboring and customer cooperation. In addition to assisting power system FIDVR studies, a non-intrusive method can also help fault prognostics for SPIMs, by comparing the estimated parameters of internal components with their nominal values.

1.2 Objectives

1.2.1 Goals and benefits

- Transfer measurements automatically to computer to perform parameter estimation.

- Mitigate the effects of Fault-Induced Delayed Voltage Recovery (FIDVR) phenomenon at residential homes.
  Our purpose is to determine the exact model of the motor. This way, we will be able to set a supplement overheat protection. The objective of this protection mechanism is to shut down the motor before it increases a certain temperature.

1.2.2 Functions and features

- Build a laboratory platform including a CSSPIM (Capacitor-Start Single phase induction motor).
- Hardware to acquire transient voltage and current measurements.
- Software to interface with computer and transfer data.
- Develop the state-space CSSPIM model.
- Implementation of parameter estimation for Single-Phase Induction Motor.

2 Design

2.1 Block diagram

As shown in *Figure 1*, the electrical component of this project consists of three main blocks:

- Devices.
- Data acquisition.
- Data processing.

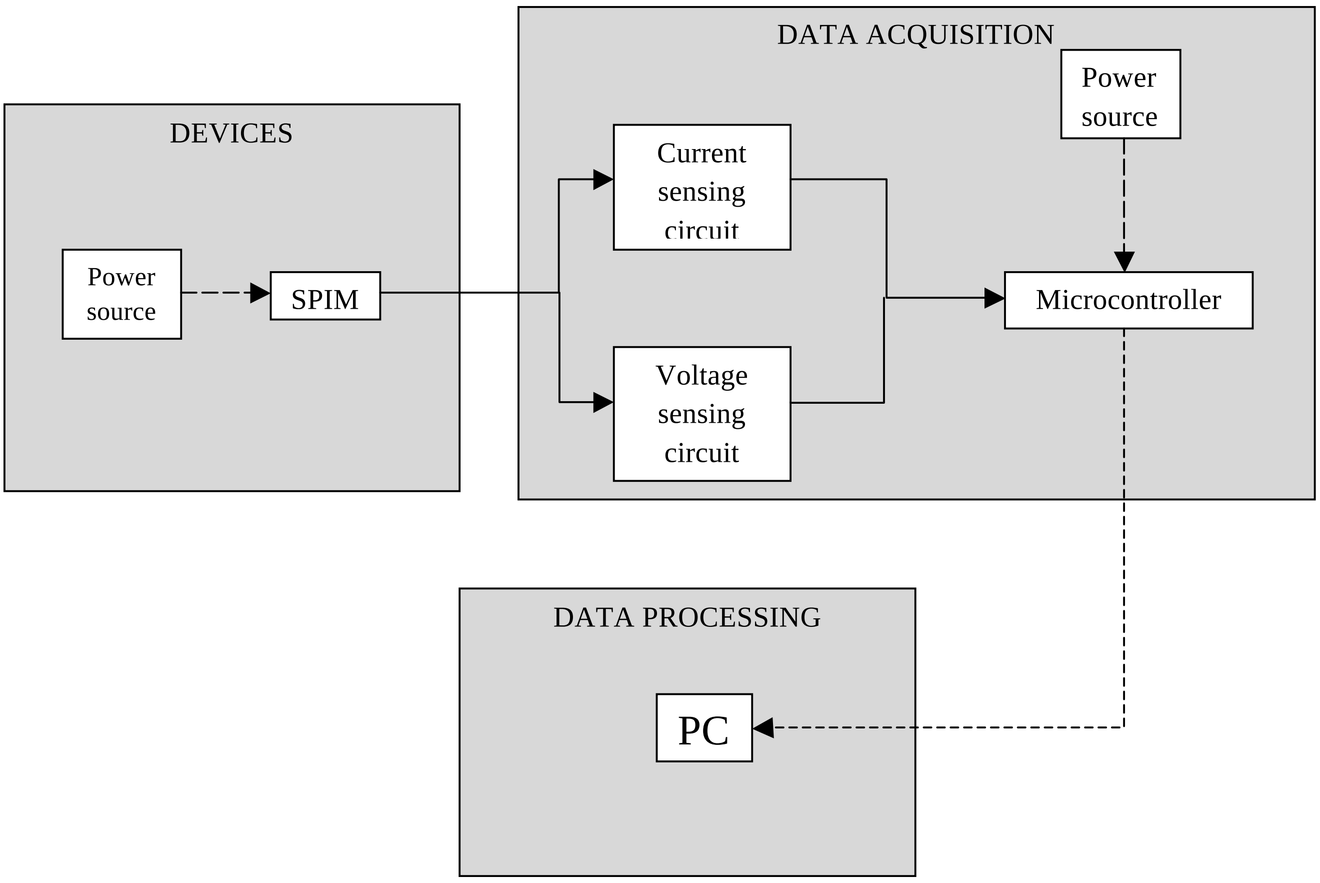


Figure 1. Block diagram of hardware design.

Power signal	– – – – – – – – –
Data signal	- - - - - - - - - - - - -

2.2 Block descriptions

The main components of the electric design are the followings:

2.2.1 Power sources

The SPIM will be supplied by a transformer in the machinery lab that will provide a maximum value of 120 V rms.

We will also need a DC supply of ±12Volts for the operational amplifiers, +5V for the Arduino and +2.5V for the offset we apply in the voltage sensing circuit.

2.2.2 SPIM (Capacitor start Single-phase Induction Motor)

This is a specific type of motor that has a capacitive circuit and a centrifugal switch connected in series to the d circuit. Once the rotor speed wr has reached a certain value (≈75%) of the rated speed, the centrifugal switch is activated to disconnect the resistor RC and the capacitor C from the d circuit. The change in the circuit reduces net electrical but still sufficient to produce enough torque for a sustained rotation.

The switching event divides the SPIM transient into two parts: start phase and run phase.

From the motor terminals we will obtain the current and voltage transient to perform parameter estimation. The estimated values are then compared with the actual motor's values to verify the effectiveness of the algorithm.

The characteristics of the Capacitor-Start SPIM will be developed in more detail in *Appendix E*.

2.2.3 Current sensing circuit

The schematic for this circuit is shown in *Figure 2*.

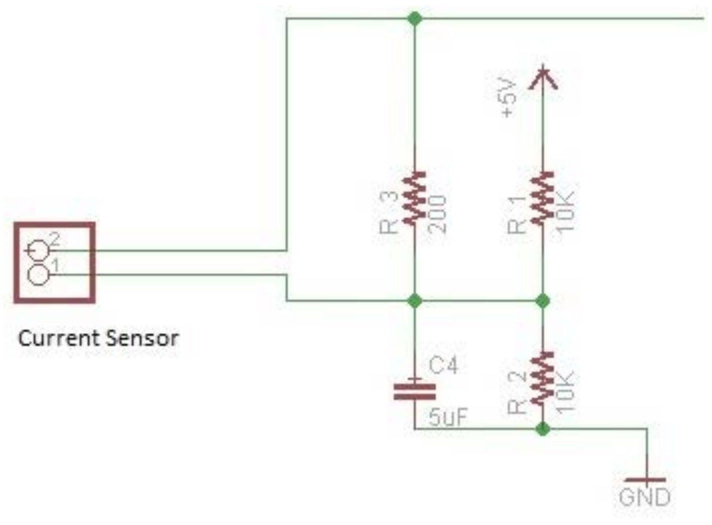


Figure 2. Current sensing circuit schematic

Its function is to measure the transient current from the induction motor terminals so we can transfer this information to the microcontroller for the corresponding operations.

We will use a Split Core Current Transformer ECS10-60 [1].

By definition, this Current Transformer can be mounted to existing panels, such as control centers or load centers, to measure or monitor wattage. These CTs can be mounted without removing existing cables for easier installation.

We must be aware about some safety issues. Before installation, safety precautions must be followed by licensed professional. Never energize primary unless short circuited terminals or burden connected to secondary.

This non-invasive current sensor can be clamped around the supply line of the electrical load to tell us how much current is passing through it.

The standard part numbers of the possible current sensors are shown in *Table 1*. We have chosen the ECS10-60 so we can measure a maximum current of 60A. We don´t know exactly what will be our values because due to the action of the switch the maximum values may change.

Table 1. Standard part numbers of current sensor.

MODEL	INPUT CURRENT	AVAILABLE OUTPUT
ECS10-15	15 A	CURRENT OUTPUTS @ 5mA, 10mA,20mA,30mA VOLTAGE OUTPUTS @ 0.25V, 0.33V, 0.5V, 1V, 2V
ECS10-30	30 A	
ECS10-50	50 A	
ECS10-60	60 A	
ECS10-75	75 A	

The schematic for the dimensions of the current sensor is shown in *Figure 3*.

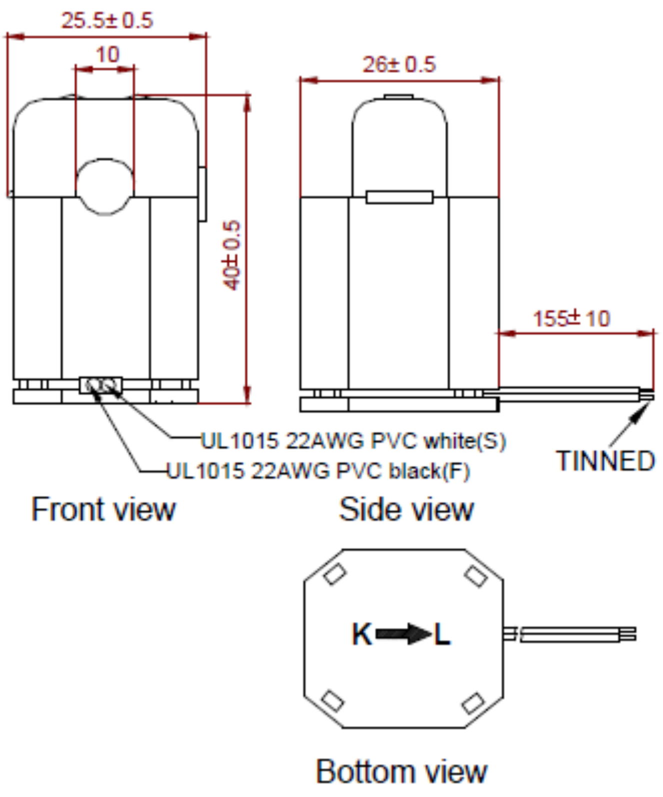
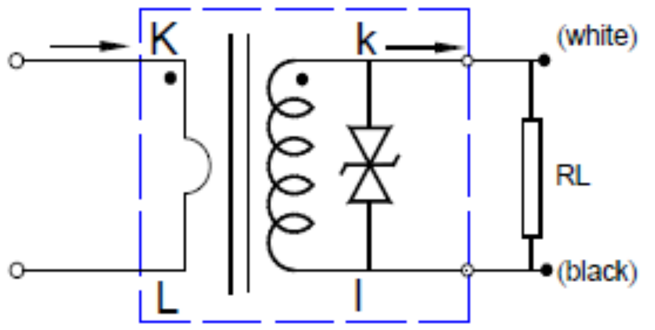


Figure 3. Current sensor dimensión (mm)

The connection diagram of the current sensor is shown in *Figure 4*:

Figure 4. Connection diagram of current



note:
1. Primary current direction from K to L
2. Secondary current directions from k to l(lead wire from white to black)
3. RL: Output load resistance

The Output Load Resistor (R13 in the schematic) has been calculated by the following equation:

$$Load\ Resistor\ (ohms) = \frac{(A_{REF} * CT_{TURNS})}{2\sqrt{2} * \max primary\ current} \qquad (1)$$

With: A_{REF} = +5V; CT_{TURNS}=3000; Max primary current= 25A.

We obtain that Load Resistor is 212.16 Ω. We choose a value under the obtained one, 200Ω.

Due to we are measuring a sinusoidal current oscillating between positive and negative values, we need to add a DC bias because the Arduino requires positive voltage between 0 and +5 Volts. We connect the end of the CT to a 2.5V level (half of the supply voltage). The signal voltage will now oscillate around 2.5V and remain positive. The resistors R11 and R12 in the circuit schematic make up a voltage divider that provides this 2.5 V level.

The capacitor C4 has a low reactance of 5 μF and provides an alternative path for the alternating current to bypass the resistor.

The input current will follow the follow transformation before we measure its value at the end of the circuit, that is, at the input of the microcontroller:

$$V_{OUT}\ (Volts) = \left(\frac{I_{IN}\ (Amps)}{3000} * 200\right) + 2.5 \qquad (2)$$

2.2.4 Voltage sensing circuit

The schematic for this circuit is shown in *Figure 5*.

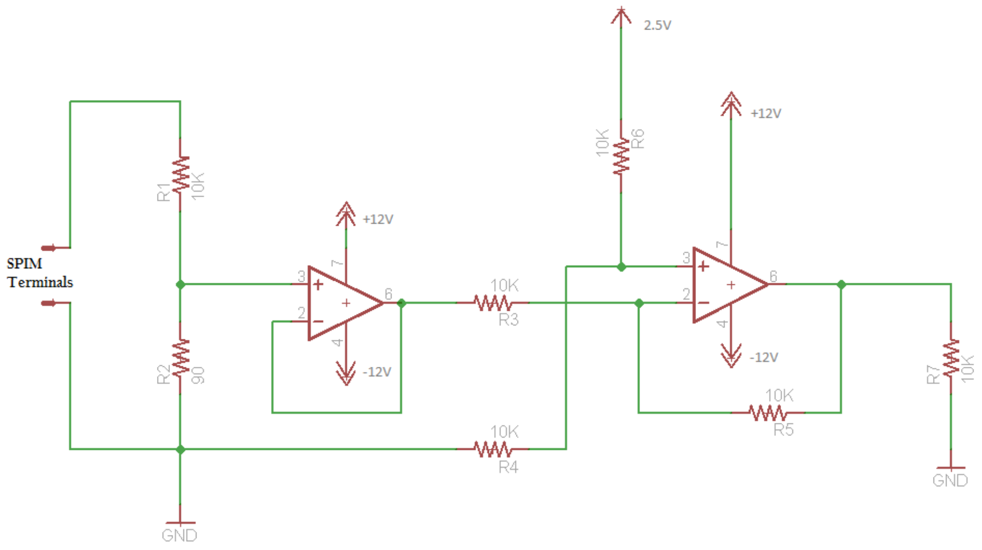


Figure 5. Voltage sensing circuit

The first part of the circuit consists of a voltage divider that will step down the voltage into a peak to peak value of less than 5 Volts. Then, the voltage would pass through a buffer amplifier [2] before getting to the operational amplifier. The operational amplifier [3] will provide a unity gain and an offset of 2.5 Volts. All of these will allow us to get at the output of the circuit a voltage in the range 0-5 Volts.

The resistances have been chosen with the purpose of obtaining the gain we need.

Table 2. Resistor values for voltage circuit

R1	10 kΩ
R2	90 Ω
R3	10 kΩ
R4	10 kΩ
R5	10 kΩ
R6	10 kΩ
R7	10 kΩ

The operational amplifiers used will have a supply of ±12 V.

The input voltage will follow the follow transformation before we measure its value at the end of the circuit:

$$V_{OUT}\ (Volts) = -\left(V_{IN}(Volts) * \frac{90}{90 + 10000}\right) + 2.5 \qquad (3)$$

2.2.5 Microcontroller

The microcontroller we will be using is Arduino Uno [4].

The sensing circuits will be connected to two analog pins of the microcontroller. This way we will receive the corresponding voltage and current transient we are applying to the SPIM. Its function is to interpret the data and to send the required information to the PC for the following analysis of the transients.

The three variables we are talking about that will be necessary are:

- Voltage transient: pin A0
- Current transient: pin A1
- Time sampling.

2.2.6 PC

The PC receives the data interpreted by the microcontroller and performs the estimation algorithm.

For the computer interface, we have designed two programs:

- Arduino: will read and storage the corresponding information of time and voltage for each data sample. It will also make the appropriate transformations to the voltages it is reading to obtain the current values of the input voltage and current.
- Matlab: We have designed a program that will save the real time, voltage and current for each sample in three different arrays. It will give us the option of plotting the result so we can see the transients that are being introduced into the motor. Before running the program we have to indicate the name of the serial port we are using and the number of samples we want to take (n).

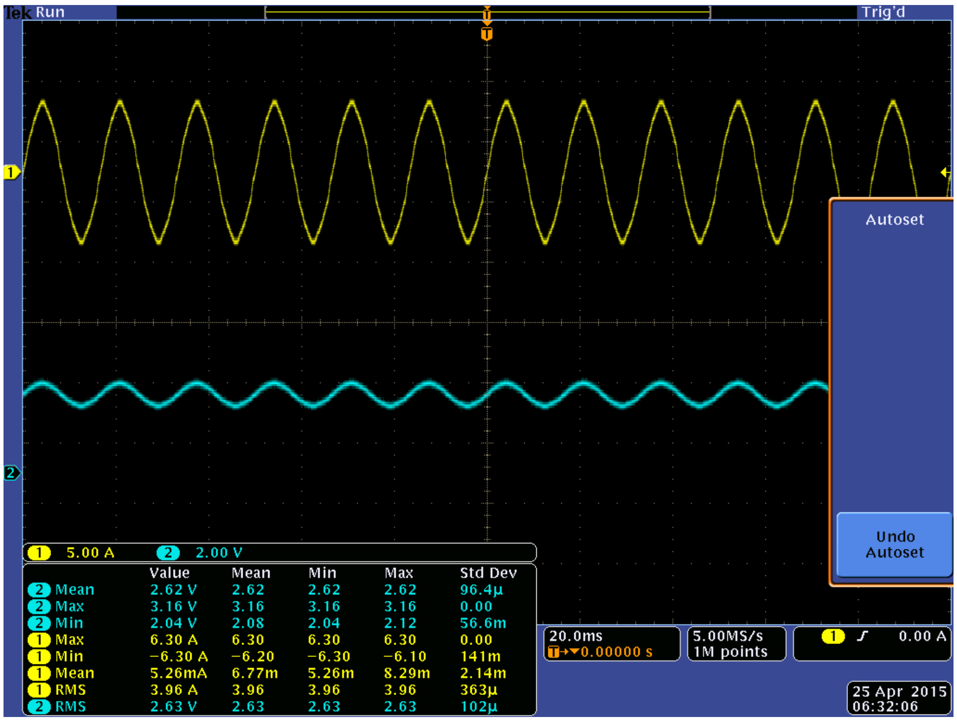
  The Matlab program that has been used is shown in the following pages.

```
function [time, voltage, current] = Matlab_Arduino(n)


close all;

clc;

y=zeros(1,1000); %array where data is saved


%initialize puerto serial

delete(instrfind({'Port'},{'COM8'}));

serial_port=serial('COM8');

serial_port.BaudRate=9600;

warning('off','MATLAB:serial:fscanf:unsuccessfulRead');

%open serial port

fopen(serial_port);


%declare counter for number of data taken

data_counter=1;


% obtaining data

while (data_counter<=n)

    dat = fscanf(serial_port, '%f');

    if(rem(data_counter,3)==1)

    time() = dat(1);

    elseif(rem(data_counter,3)==2)
    voltage() = dat(2);

    elseif(rem(data_counter,3)==0)

    current() = dat(3);

    end

    data_counter=data_counter+1;

end


%close conexion

fclose(serial_port);

delete(serial_port);

    end
```

3 Design verification

3.1 Current sensing circuit

To verify that the current sensing circuit is transforming and reducing the input current in the way we want, we will use the oscilloscope to measure the input and output signal. As we see in *Figure 6*, if we apply a current of 4 A(rms) (yellow signal), the output we obtain has an offset value of 2.5 V and the peak to peak value is less than 5 V (blue signal). This is what we expected, so this voltage can be measured by the Arduino for our purpose.

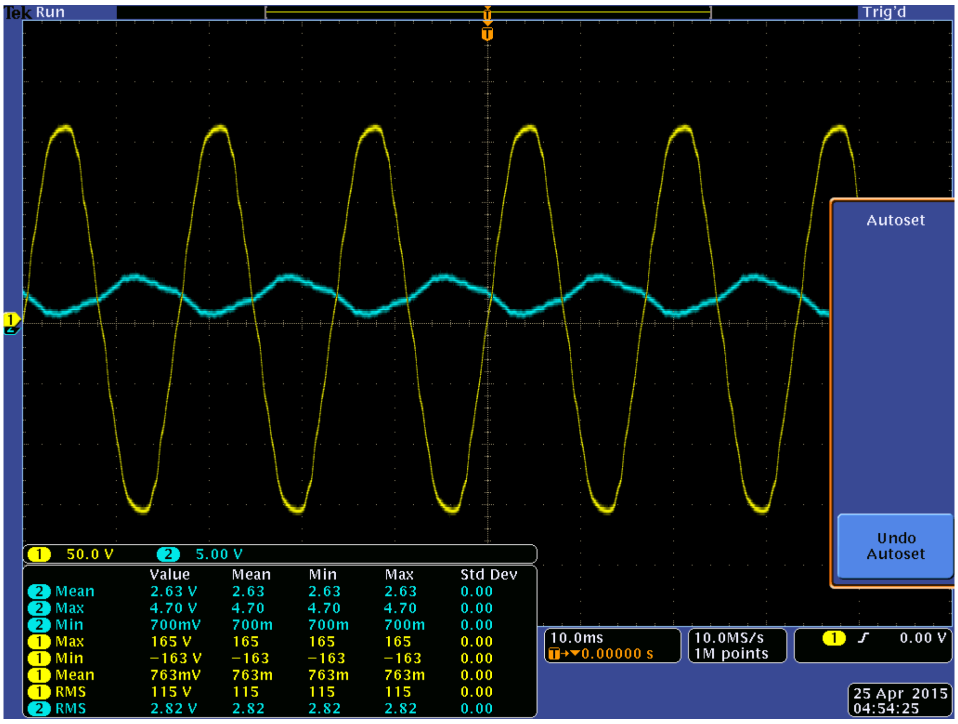


	Value	Mean	Min	Max	Std Dev
2 Mean	2.62 V	2.62	2.62	2.62	96.4µ
2 Max	3.16 V	3.16	3.16	3.16	0.00
2 Min	2.04 V	2.08	2.04	2.12	56.6m
1 Max	6.30 A	6.30	6.30	6.30	0.00
1 Min	−6.30 A	−6.20	−6.30	−6.10	141m
1 Mean	5.26mA	6.77m	5.26m	8.29m	2.14m
1 RMS	3.96 A	3.96	3.96	3.96	363µ
2 RMS	2.63 V	2.63	2.63	2.63	102µ

Figure 6. Oscilloscope measures for current sensing circuit

3.2 Voltage sensing circuit

To verify that the voltage sensing circuit is reducing the input voltage in the way we want, we will use the oscilloscope to measure the input and output signal. As we see in *Figure 7*, if we apply a voltage of 115 V (rms) (yellow signal), the output we obtain has an offset value of 2.5 V and the peak to peak value is less than 5 V (blue signal). This is what we expected, so this voltage can be measured by the Arduino for our purpose.



	Value	Mean	Min	Max	Std Dev
2 Mean	2.63 V	2.63	2.63	2.63	0.00
2 Max	4.70 V	4.70	4.70	4.70	0.00
2 Min	700mV	700m	700m	700m	0.00
1 Max	165 V	165	165	165	0.00
1 Min	−163 V	−163	−163	−163	0.00
1 Mean	763mV	763m	763m	763m	0.00
1 RMS	115 V	115	115	115	0.00
2 RMS	2.82 V	2.82	2.82	2.82	0.00

Figure 7. Oscilloscope measures for voltage sensing circuit

3.3 Microcontroller & PC

The function we want the microcontroller and PC to perform is saving the information about time, voltage and current. We would perform the parameter estimation with this information. It gives us the option to plot the results, as we see in *Figure 8*.

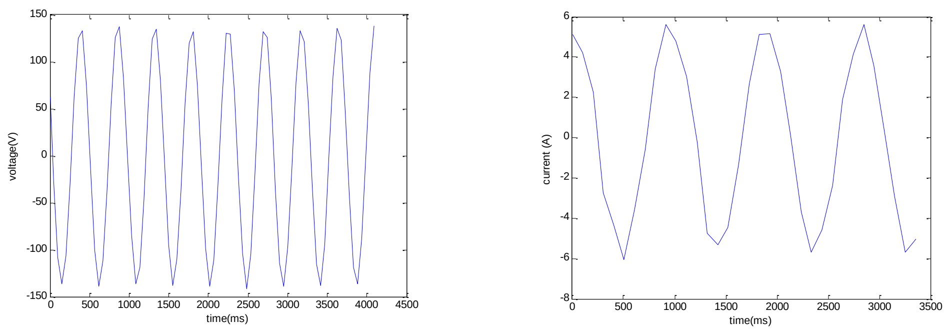


Figure 8. Matlab plots

4 Costs

In the following table (Table 3) are shown the quantities and costs of the different materials that had been purchased for the purpose of building the hardware part of the project.

Table 3. Part´s costs

Item	Quantity	Cost
Split Core Current Transformer ECS1060	1	$9.95
Resistor of 200Ω	1	$0.00475
Resistor of 90Ω	1	$0.00475
Resistor of 10 KΩ	8	8x($0.00475)
Capacitor of 5uF	1	$0.5
Operational amplifier (LM318)	1	$0.5
Operational amplifier (TL084)	1	$0.14
Microcontroller (Arduino Uno)	1	$17.67
Total		$28.8

5 Conclusion

5.1 Accomplishments

The work done so far will allow us to read the transient voltage and current that are flowing through the single phase induction motor and transmit this information to the computer for the following analysis.

In *Apendix B* we have performed the procedure for the parameter´s identification to validate if the final estimation result is correct.

In *Apendix C* the state-space CSSPIM model has been developed for the following simulation of the motor through our code.

In *Apendix D* we show the different codes that shape our program for the motor simulation.

5.2 Ethical considerations

The purpose of this project is the implementation of Non-intrusive Parameter Estimation Algorithm for Single-phase Induction Motors Using Transient Data. We assure that this will be able to do all of the details listed in the benefits and features. It will have to comply with the IEEE Code of Ethics. Of the ten that IEEE has listed, the ones that apply to this project are listed below:

- To accept responsibility in making decisions consistent with the safety, health, and welfare of the users, and to disclose promptly factors that might endanger the public or the environment;
- To be honest and realistic in stating claims or estimates based on available data;
- To improve the understanding of technology; its appropriate application, and potential consequences.

5.3 Safety statements

In this project we will be working with a single-phase induction motor and multiple electric devices so we have to take into account a few safety measures to guarantee the security of the team members and anyone who could make use of it. The main measures are:

- The terminals of a machine that has a feeding frequency converter may be live even with the machine stopped. Because of this, it is important to avoid any contact.
- Do not exceed the maximum speed permitted for the machine.
- Avoid contact with power supplies during operation.
- Do not introduce a higher voltage in the microcontroller that the maximum allowed.

5.4 Future work

For future work, the progress achieved up to this point will be used for the final purpose of implementation of Non-intrusive Parameter Estimation Algorithm for Single-phase Induction Motors Using Transient Data. We will want to use the hardware components and the model for the Capacitor-Start SPIM to perform real time estimation of the parameters. This means the circuit will collect the data, send it to the computer, and the computer will process the data immediately to carry out the calculation of the parameters of interest.

Thanks to all this work we will be able to find the way to mitigate the effects of Fault-Induced Delayed Voltage Recovery (FIDVR) phenomenon.

Appendix A: Requirements and verification table

In *Table 4* are shown the requirements that each part of the circuit must accomplish to reach our purpose. We can also see the corresponding verifications for each of the requirement. Before performing all the components we have to make sure all these verifications are fulfilled.

Table 4. Requirements and verification table

Requirement	Verification
1. Current sensing circuit	
A) The current sensor will reduce the input current from the SPIM terminals with a ratio of 60A/20mA. Placing a 200Ω resistor the transformation ratio will be 60A/4V. The tolerance accepted will be ±5%. B) We will reduce the correspondent voltage and add an offset (5V/2) value to obtain a voltage in the range (0-+5V) to introduce in the Arduino.	A) Use the voltmeter to measure the output voltage for the different current levels applied to check if it is meeting the corresponding slope. Make sure the transformation is followed with linearity. B) Use the oscilloscope to make sure the final voltage we are introducing in the microcontroller is in the required range.
2. Voltage sensing circuit	
A) The relation between the voltage in the SPIM terminals with the output voltage at the end of the operational amplifier must verify the set amplitude with a tolerance of ±5% and must be in a range of (0-+5V), so we can measure it with the Arduino	A) Use the voltmeter/oscilloscope to measure the voltage. Check the input voltage (SPIM terminals) and output voltage (after the operational amplifier) to assure this is what we expected and it meets the set gain.

B) <u>Voltage follower</u> Check that the voltage before and after the follower remains the same. C) <u>Operational-Amplifier:</u> • Resistors values within 5% tolerance • The amplitude won´t exceed the saturation limit. (+12V/-12V). • The gain of the opamp will be equal to one. It will only inverse the voltage signal. • An offset will be applied to provide just positive values to the microcontroller.	B) Obtain measures with oscilloscope C) • Measure the resistor values with a multimeter and check they meet the tolerance requirement. • Use voltmeter to ensure our voltage is within the saturation limits. • The amplitude of the signal before and after must be the same. Check with oscilloscope. • Adjust the specific offset required to meet the requirement and check in the output signal if the result is what we expected.
5.Microcontroller - Arduino Uno	
A) Make sure the microcontroller will be capturing the data at a sampling rate enough to acquire the enough values of our transient. B) The Arduino must be able to read the information that is receiving from the voltage and current sensing circuits.	A) Plot the sine wave to see if we can have a good estimation of the transient from the number of data we are measuring B) Have the Arduino print out the values that it is receiving..

Appendix B: Procedures to estimate SPIM parameters

The double-revolving-field and the cross-field theories have been widely used for analyzing the performance variables of SPIM operating in steady state. In double-revolving-field theory, magnetic field (produced by the motor) that pulsates in time but is stationary in space can be resolved into two revolving magnetic fields that are equal in magnitude but revolve synchronously in opposite directions. The induced EMFs in the rotor due to the two revolving fields are in opposition to each other.

One section of the rotor circuit is usually referred to as the forward branch, and the other is known as the backward branch. At standstill, the slip in either direction is the same and so is the rotor impedance. Therefore, starting torque developed by each revolving field is the same, and net torque developed by the motor is zero, and hence, SPIM is not self-starting. An IM can be made self-starting, if the two windings are placed in space quadrature and are connected in parallel to a single-phase source but impedances of the two windings are made unequal to produce out-of-phase currents which, in turn, set up a net unbalanced revolving field.

To obtain the equivalent circuit parameters, both no-load and locked-rotor tests are performed on CSSPIM (Capacitor Start – Single Phase Induction Motor). Magnetizing reactance can be obtained using the no-load test results. Stator and rotor leakage reactance and stator referred rotor winding resistance can be computed using locked-rotor test data.

Ghial Paper [5] has been used as a guide to set the method for obtaining the motor parameters.

The internal structure of the CSSPIM is shown in *Figure 8*.

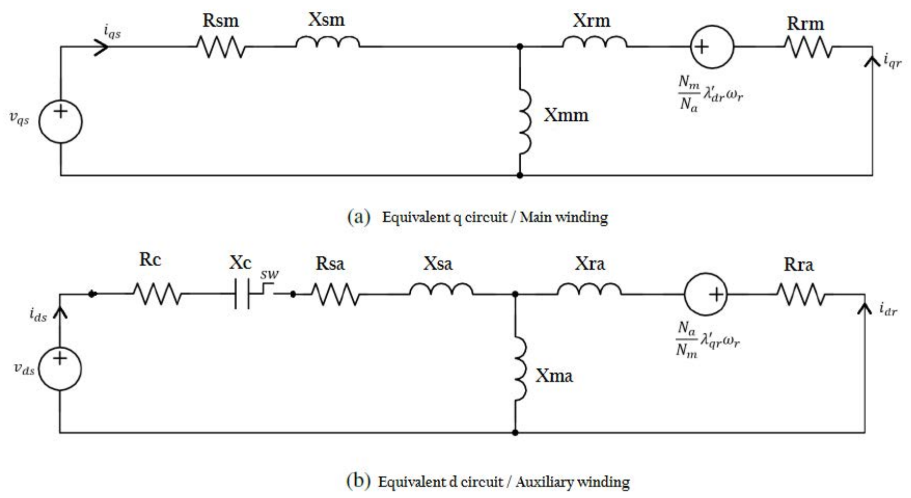

i_{qs}
Rsm
Xsm
Xrm
Rrm
i_{qr}
v_{qs}
Xmm
$\frac{N_m}{N_a}\lambda'_{dr}\omega_r$
(a) Equivalent q circuit / Main winding
Rc
Xc
sw
Rsa
Xsa
Xra
Rra
i_{ds}
v_{ds}
Xma
$\frac{N_a}{N_m}\lambda'_{qr}\omega_r$
i_{dr}
(b) Equivalent d circuit / Auxiliary winding

Figure 8. Structure of CSSPIM

A. DC Test

In this test, dc voltage is applied across the stator winding, after removing the capacitor, and the dc current is then measured. Stator winding dc resistance (R_s) thus can be computed. From this we can obtain:

- R_{sm}: stator resistance of main winding.
- R_{sa}: stator resistance of auxiliary winding.

B. No-Load Test

In **no-load test**, rated voltage V_{NL} is applied, and current I_{NL} and power P_{NL} are measured at no load. Since the no-load slip is small, the rotor resistance of forward branch is assumed to be infinite. The resistance associated with backward rotating field is small enough so that magnetizing current may be neglected, which results in the equivalent circuit of *Figure 9.*

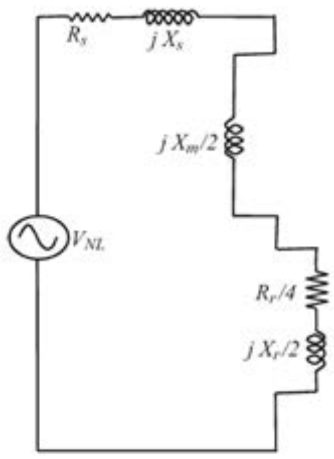


Figure 9. Equivalent circuit of SPIM at no-load condition

Method for No-load Test

The secondary of the transformer (rotor) is left open-circuited. A wattmeter is connected to the primary. An ammeter is connected in series with the primary winding (stator). A voltmeter is optional since the applied voltage is the same as the voltmeter reading. Rated voltage is applied at primary.

Since the secondary of the transformer is open, the primary draws only no-load current, which will have some copper loss. This no-load current is very small and because the copper loss in the primary is proportional to the square of this current, it is negligible. There is no copper loss in the secondary because there is no secondary current.

Current, voltage and power are measured at the primary winding.

C. Locked-rotor Test

In **locked-rotor test**, the rotor is held at standstill while exciting both the windings with locked-rotor voltage V_{LR} such that the locked-rotor line current I_{LR} equals the rated current. Since the slip at standstill is unity, the rotor circuit impedance is much smaller than the magnetizing reactance. Therefore, the magnetizing reactance may be eliminated from the equivalent circuit as in *Figure 10.*

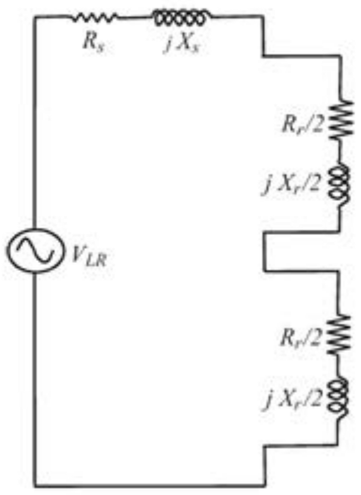


Figure 10. Equivalent circuit at locked-rotor condition

Method for Locked-rotor Test

In the locked-rotor test, the rotor is locked. A low voltage is applied on the stator terminals so that there is full load current in the stator winding, and the current, voltage and power input are measured at that point. When the rotor is stationary, the slip, $s = 1$. The test is conducted at 1/4 the rated frequency as recommended by IEEE, because the rotor's effective resistance at low frequency may differ at high frequency. The test can be repeated for different values of voltage to ensure the values obtained are consistent. As the current through the stator may exceed the rated current, the test should be conducted quickly.

Current, voltage and power are measured at the primary winding.

The locked-rotor resistance R_{LR} is computed as

$$R_{LR} = \frac{P_{LR}}{I_{LR}^2} \tag{4}$$

The locked-rotor impedance is given by

$$Z_{LR} = \frac{V_{LR}}{I_{LR}} \tag{5}$$

Locked-rotor reactance is thus computed as

$$X_{LR} = \sqrt{Z_{LR}^2 - R_{LR}^2} \tag{6}$$

Rotor winding resistance Rr is then computed as

$$R_r = R_{LR} - R_s \tag{7}$$

The stator and rotor leakage reactances are usually considered to be equal; therefore, stator leakage reactance Xs and the rotor winding reactance Xr can be obtained from:

$$X_s = X_r = \frac{X_{LR}}{2} \tag{8}$$

For the parameter estimation we will apply these three tests to each winding (main and auxiliary):

1. PARAMETER COMPUTATION OF MAIN WINDING

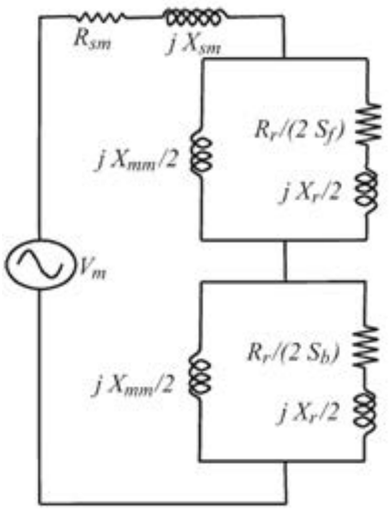


Figure11. Equivalent circuit of main winding of SPIM

The main winding resistance R_{sm} is proposed to be measured separately, by applying **dc** voltage across individual winding and measuring the dc current of each, after removing the capacitor. We must take into account that it is necessary to deduct the wire resistance the value obtained.

- Stator resistance of main winding → R_{sm}

After the **locked-rotor test** of the main winding we can obtain:

- Rotor resistance of main winding → $R_{rm} = \frac{P_{LR}}{I_{LR}^2} - R_{sm}$ (9)

The leakage reactance of each individual winding is proposed to be equal to the stator leakage reactance. Thus, the stator main winding reactance X_{sm} and rotor main winding reactance X_{rm} are given by:

$$X_{sm} = X_s = \frac{X_{LR}}{2} = \frac{\sqrt{\left(\frac{V_{LR}}{I_{LR}}\right)^2 - \left(\frac{P_{LR}}{I_{LR}^2}\right)^2}}{2} = X_{rm} \qquad (10)$$

The next step is to proceed with the values we have obtained from **<u>no-load test</u>** in the main winding.

The voltage across the main winding magnetizing reactance can be computed as:

$$V_{ab1} = \mathrm{V_{NL}} - \mathrm{I_{NL}}\left[\mathrm{X_{sm}} + \frac{\mathrm{X_r}}{2}\right] \tag{11}$$

The main winding magnetizing reactance X_{mm} is determined as:

$$\mathrm{X_{mm}} = \frac{2 * V_{ab1}}{\mathrm{I_{NL}}} = \frac{2 * \left[\mathrm{V_{NL}} - \mathrm{I_{NL}}\left(\mathrm{X_{sm}} + \frac{\mathrm{X_r}}{2}\right)\right]}{\mathrm{I_{NL}}} \tag{12}$$

2. PARAMETER COMPUTATION OF AUXILIARY WINDING

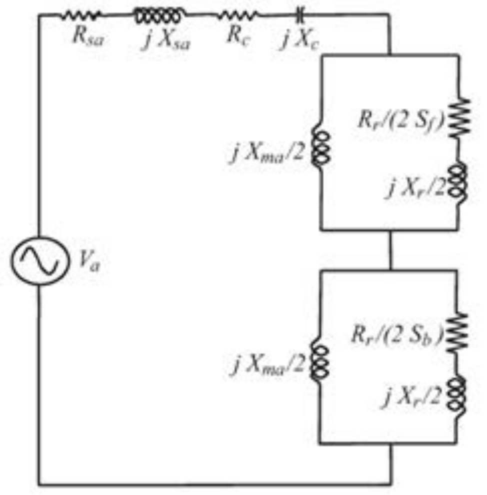


Figure 12. Equivalent circuit of auxiliary winding of SPIM

First of all, we can measure the values of X_C and R_C directly from the capacitor terminals with the multimeter:

$$X_C = \frac{1}{w * C} \tag{13}$$

Then we will follow the same process as for the main winding;

The stator resistance of auxiliary winding will be obtained from the **dc test** (deduct the wire resistance) → R_{sa}

After the **lock-rotor test** of the auxiliary winding we can obtain:

- Rotor resistance of auxiliary winding $\rightarrow R_{ra} = \frac{P_{LR}}{{I_{LR}}^2} - R_{sa}$ (14)

The leakage reactance of each individual winding is proposed to be equal to the stator leakage reactance. Thus, the stator auxiliary winding reactance X_{sa} and rotor main winding reactance X_{ra} are given by:

$$X_{sa} = X_s = \frac{X_{LR}}{2} = \frac{\sqrt{\left(\frac{V_{LR}}{I_{LR}}\right)^2 - \left(\frac{P_{LR}}{{I_{LR}}^2}\right)^2}}{2} = X_{ra} \tag{15}$$

The next step is to proceed with the values we have obtained from **no-load test** in the auxiliary winding.

From the KVL principle we have the following equation for the circuit in no-load test:

$$V_{NL} = I_{NL}\left[\left(R_{sa} + \frac{R_{ra}}{4}\right) + j\left(X_{sa} + \frac{X_{ma}}{2} + \frac{X_{ra}}{2} - X_C\right)\right] \tag{16}$$

So we have then the corresponding equation for the module:

$$|V_{NL}| = |I_{NL}|\sqrt{\left(R_{sa} + \frac{R_{ra}}{4}\right)^2 + \left(X_{sa} - X_C + \frac{X_{ma}}{2} + \frac{X_{ra}}{2} - X_C\right)^2} \tag{17}$$

Finally, the auxiliary winding magnetizing reactance X_{ma} is determined as:

$$X_{ma} = 2\left[\sqrt{\left(\frac{V_{NL}}{I_{NL}}\right)^2 - \left(R_{sa} + \frac{R_{ra}}{4}\right)^2} - \left(X_{sa} + \frac{X_{ra}}{2} - X_C\right)\right] \tag{18}$$

Data acquisition

Once we know the procedure we have to follow, we proceed to obtaining the corresponding values from the tests in the laboratory.

In *Figure 13* we can see the bench we have been using for the performance of the tests. The single phase induction motor (2) is a Capacitor-Start Single phase induction motor (CSSPIM). We are able to set the speed and torque we want by modifying the performance of the dynamometer (3). To achieve this purpose we have used the LabView program that allows us to apply to the motor the conditions we want in each case, depending on the test we are applying. We will record the result values we read in the wattmeter (1).

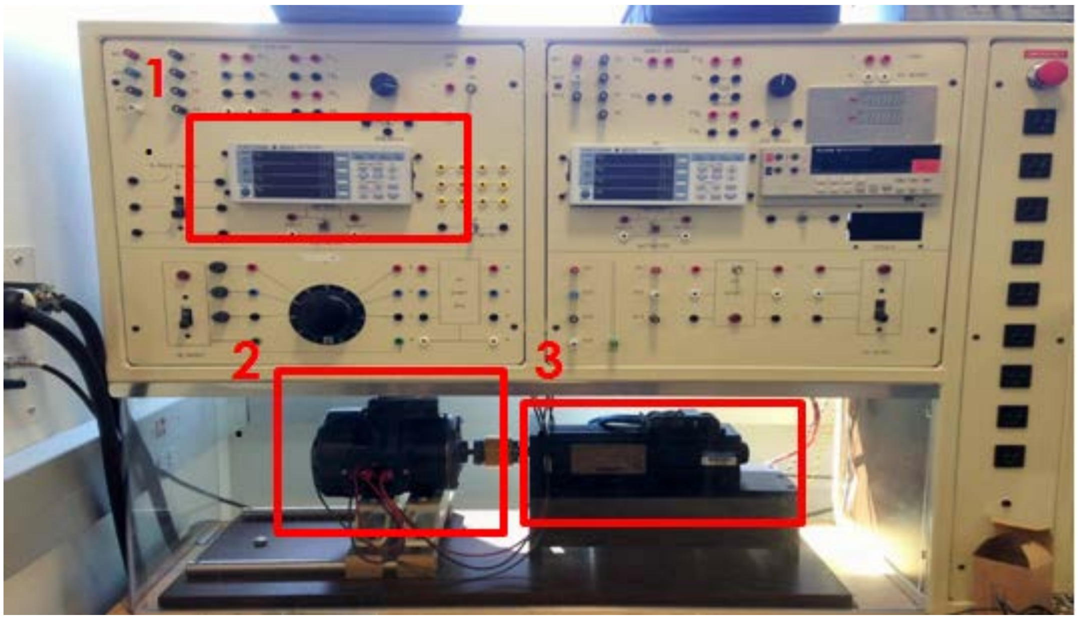
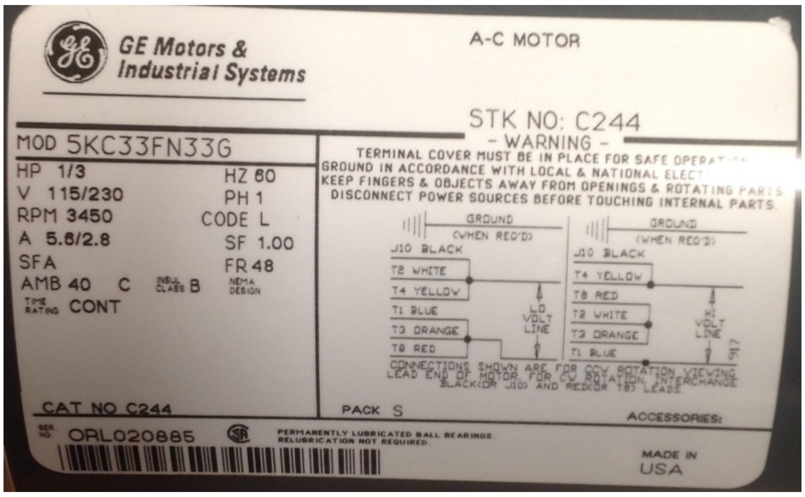


Figure 13. Lab bench

(1)	Wattmeter
(2)	Single-phase induction motor
(3)	Dynamometer

In *Figure 14* we see the sheet of characteristics of the specific CSSPIM we have been using.


GE Motors & Industrial Systems

A-C MOTOR

STK NO: C244

MOD 5KC33FN33G

HP 1/3 HZ 60
V 115/230 PH 1
RPM 3450 CODE L
A 5.6/2.8 SF 1.00
SFA FR 48
AMB 40 C INSUL CLASS B NEMA DESIGN
TIME RATING CONT

- WARNING -
TERMINAL COVER MUST BE IN PLACE FOR SAFE OPERAT
GROUND IN ACCORDANCE WITH LOCAL & NATIONAL ELECT
KEEP FINGERS & OBJECTS AWAY FROM OPENINGS & ROTATING PARTS
DISCONNECT POWER SOURCES BEFORE TOUCHING INTERNAL PARTS

CONNECTIONS SHOWN ARE FOR CCW ROTATION VIEWING LEAD END OF MOTOR. FOR CW ROTATION, INTERCHANGE BLACK(OR J10) AND RED(OR T8) LEADS.

CAT NO C244 PACK S ACCESSORIES:

SER. NO. ORL020885

PERMANENTLY LUBRICATED BALL BEARINGS. RELUBRICATION NOT REQUIRED.

MADE IN USA

Figure14. Sheet of characteristics of the CSSPIM

We will connect the terminals in the parallel configuration, so the voltage we will be applying is Low Voltage (As indicated in the sheet of characteristics).

The two first terminals (red and black) correspond to the auxiliary winding and the other four to the main winding. As we said those will be connected in parallel (orange-yellow // blue-white). So when we obtain the resistor values, we have to do the corresponding calculations.

1. **DC test**

   The values of the stator resistances of the main and auxiliary winding can be obtained by measuring its value directly from the wires of a motor that are disconnected. We cannot measure directly from the motor terminal because in that case, we will be measuring too the impedance corresponding to the capacitor and the value would be huge.

- Auxiliary winding:

$$\boldsymbol{R_{sa} = 6.099\Omega} \tag{19}$$

- Main winding:

$$R_{sm} = \mathbf{3.41\ \Omega \,||\, 3.41\Omega \ = 1.705\ \Omega} \qquad (20)$$

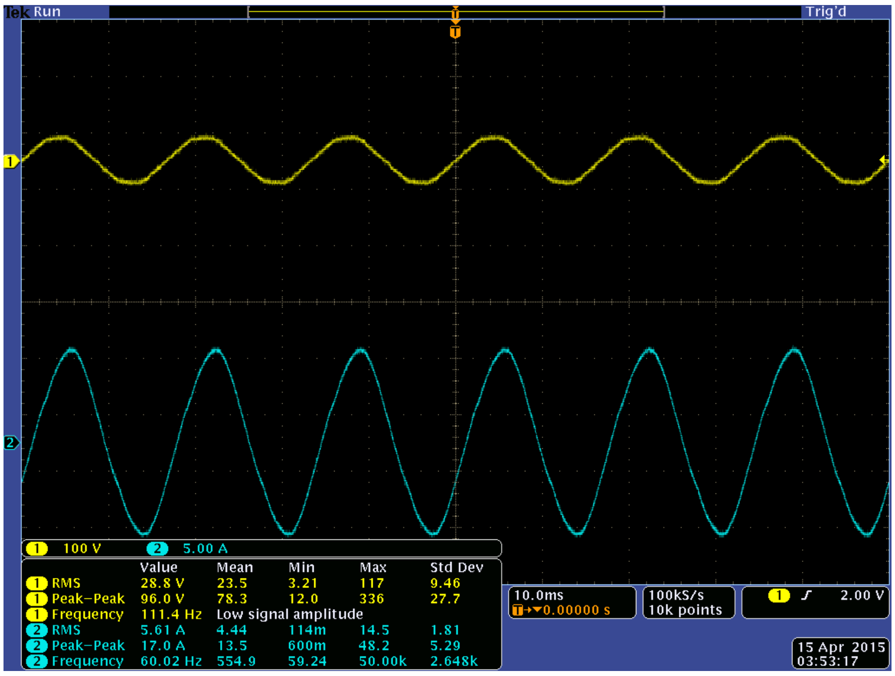
For the no-load test and locked-rotor test, we will use a dynamometer to set the motor speed in each test.

The figures in the following sections are the signals we have obtained from the oscilloscope for each of the tests applied, to the complete motor and to each of the windings.

The tables collect the values obtained for each test that we have read in the wattmeter.

2. Locked-rotor test:

- ✓ We set the speed of 0 rpm, so that the slip is 1.
- ✓ We start applying voltage until we reach the full load current. In our case (low voltage) it will be 5.6A.



1 100 V   2 5.00 A

	Value	Mean	Min	Max	Std Dev
1 RMS	28.8 V	23.5	3.21	117	9.46
1 Peak–Peak	96.0 V	78.3	12.0	336	27.7
1 Frequency	111.4 Hz	Low signal amplitude			
2 RMS	5.61 A	4.44	114m	14.5	1.81
2 Peak–Peak	17.0 A	13.5	600m	48.2	5.29
2 Frequency	60.02 Hz	554.9	59.24	50.00k	2.648k

10.0ms
T→▼0.00000 s
100kS/s
10k points
1 ʃ 2.00 V
15 Apr 2015
03:53:17

Figure 15. Locked rotor test signal

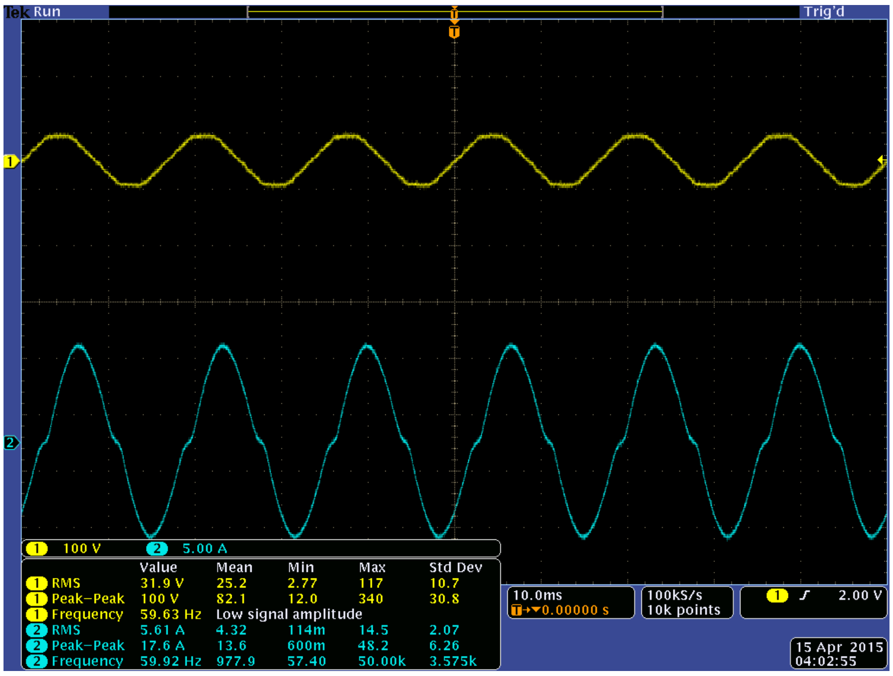
✓ We perform this test to the main winding and to the auxiliary winding separately:

❖ **Main winding:**

With the purpose of obtaining the corresponding values just for the main winding, we have to previously disconnect the auxiliary winding, so the voltage applied will affect exclusively the main winding.

Table 5. Results for locked-rotor test in main winding

V_{IN} (V)	I_{IN}(A)	P_{IN}(W)	PF
39.534	7.038	182.72	0.6592
25.164	4.133	64.15	0.6165



```
                Value      Mean    Min     Max     Std Dev
1 RMS           31.9 V     25.2    2.77    117     10.7
1 Peak–Peak     100 V      82.1    12.0    340     30.8
1 Frequency     59.63 Hz   Low signal amplitude
2 RMS           5.61 A     4.32    114m    14.5    2.07
2 Peak–Peak     17.6 A     13.6    600m    48.2    6.26
2 Frequency     59.92 Hz   977.9   57.40   50.00k  3.575k
```

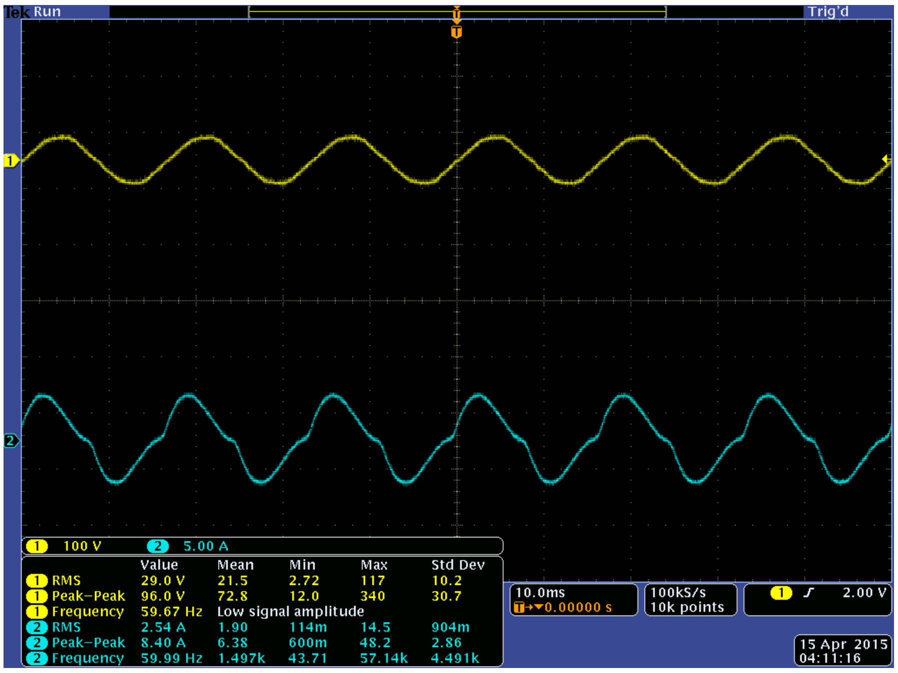
Figure 16. Locked rotor test signal for main winding

❖ **Auxiliary winding**:

As we did with the main winding, now we have to previously disconnect the main winding, so the voltage applied will affect exclusively the auxiliary winding. We set the current to half the rated current so the switch don´t goes off.

Table 6. Results for locked-rotor test in auxiliary winding

V_{IN} (V)	I_{IN}(A)	P_{IN}(W)	PF
42.130	3.4318	112.77	0.7866
24.54	2.6733	41.77	0.8185

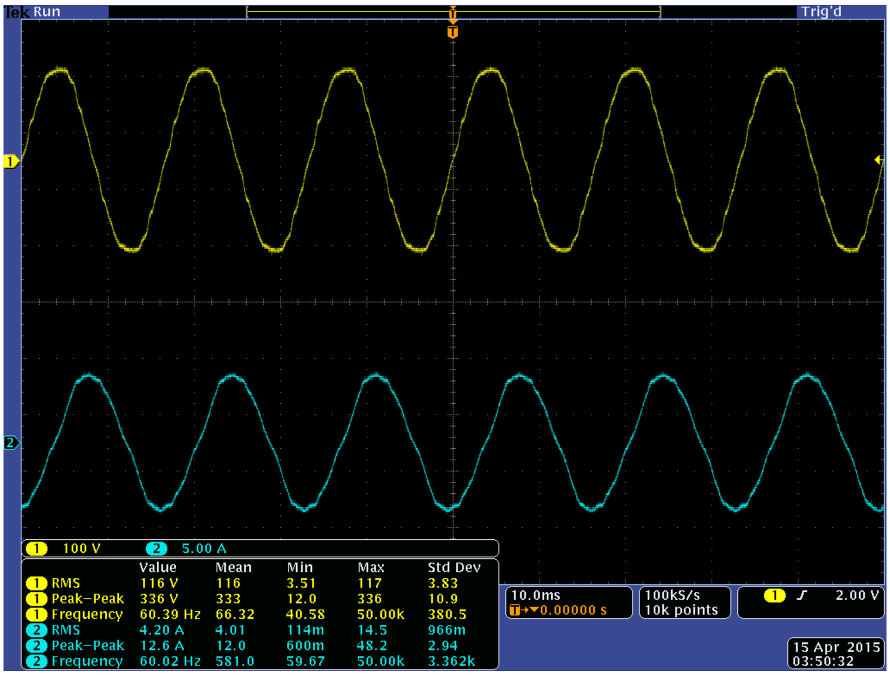


```
                Value     Mean      Min     Max     Std Dev
1 RMS           29.0 V    21.5      2.72    117     10.2
1 Peak-Peak     96.0 V    72.8      12.0    340     30.7
1 Frequency     59.67 Hz  Low signal amplitude
2 RMS           2.54 A    1.90      114m    14.5    904m
2 Peak-Peak     8.40 A    6.38      600m    48.2    2.86
2 Frequency     59.99 Hz  1.497k    43.71   57.14k  4.491k
```

Figure 17. Locked rotor test signal for auxiliary winding

3. No-load test:

- ✓ We set the speed at the one rated for the motor, in this case 3450 rpm.
- ✓ We apply the rated voltage (Low voltage = 115V).



	Value	Mean	Min	Max	Std Dev
1 RMS	116 V	116	3.51	117	3.83
1 Peak–Peak	336 V	333	12.0	336	10.9
1 Frequency	60.39 Hz	66.32	40.58	50.00k	380.5
2 RMS	4.20 A	4.01	114m	14.5	966m
2 Peak–Peak	12.6 A	12.0	600m	48.2	2.94
2 Frequency	60.02 Hz	581.0	59.67	50.00k	3.362k

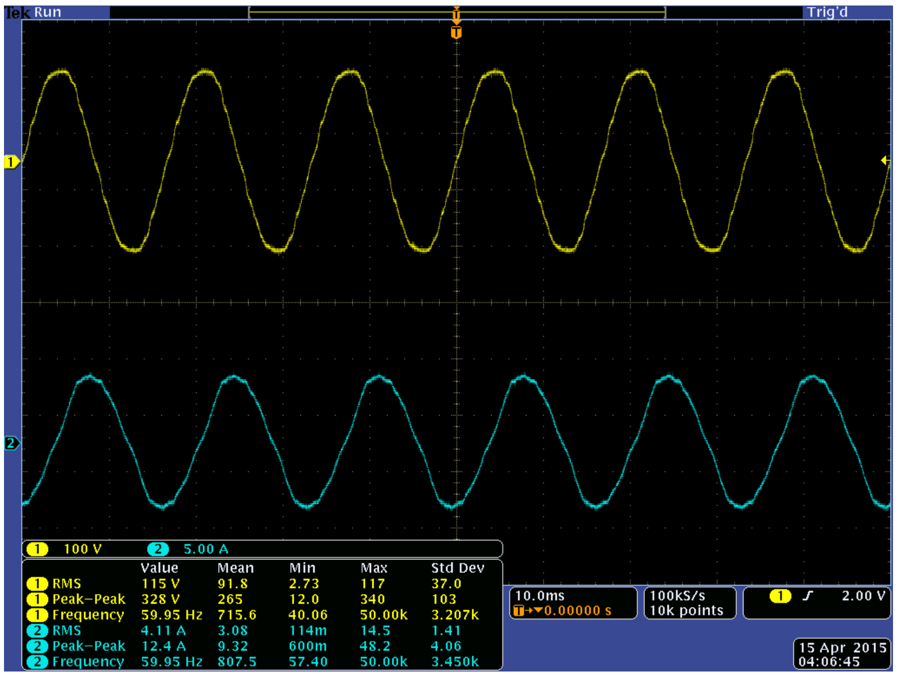
Figure 18. No-load test signal

✓ We perform this test to the main winding and to the auxiliary winding separately:

❖ Main winding:

Table 7 . Results for no-load test in main-winding

V_{IN} (V)	I_{IN}(A)	P_{IN}(W)	PF
39.931	3.9346	97.83	0.6353
25.523	2.5202	46.47	0.7182



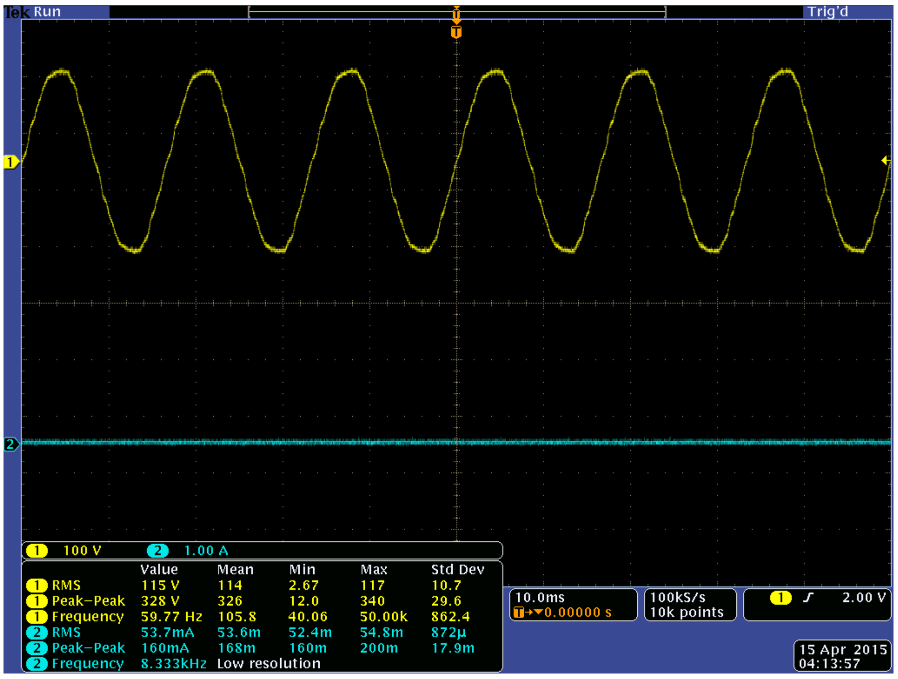
```
              Value      Mean    Min     Max     Std Dev
1 RMS         115 V      91.8    2.73    117     37.0
1 Peak–Peak   328 V      265     12.0    340     103
1 Frequency   59.95 Hz   715.6   40.06   50.00k  3.207k
2 RMS         4.11 A     3.08    114m    14.5    1.41
2 Peak–Peak   12.4 A     9.32    600m    48.2    4.06
2 Frequency   59.95 Hz   807.5   57.40   50.00k  3.450k
```

Figure 19. No-load test signal for main winding

❖ Auxiliary winding

Table 8. Results for no-load test in auxiliary winding.

V_{IN} (V)	I_{IN}(A)	P_{IN}(W)	PF
39.312	4.87521	159.45	0.8404
25.432	2.999	59.78	0.7804



```
              Value      Mean    Min     Max     Std Dev
1 RMS         115 V      114     2.67    117     10.7
1 Peak–Peak   328 V      326     12.0    340     29.6
1 Frequency   59.77 Hz   105.8   40.06   50.00k  862.4
2 RMS         53.7mA     53.6m   52.4m   54.8m   872µ
2 Peak–Peak   160mA      168m    160m    200m    17.9m
2 Frequency   8.333kHz   Low resolution
```

Figure 20. No-load test signal for auxiliary signal

The next step is to calculate the parameters value using the measurements obtained:

$$\mathrm{R_{sm}} = 1.705\ \boldsymbol{\Omega} \tag{21}$$

$$\mathrm{R_{rm}} = \frac{182.72}{7.038^2} - 1.705 = 1.985\ \boldsymbol{\Omega} \tag{22}$$

$$\mathrm{X_{sm}} = \mathrm{X_{rm}} = \frac{\mathrm{X_{LR}}}{2} = \frac{\sqrt{\left(\frac{39.534}{7.038}\right)^2 - \left(\frac{182.72}{7.038^2}\right)^2}}{2} = 2.118\ \boldsymbol{\Omega} \tag{23}$$

$$\mathrm{X_{mm}} = \frac{2\left[39.931 - 3.9346\left(\frac{3}{2} * 2.118\right)\right]}{3.9346} = 13.943\Omega \tag{24}$$

$$R_C = 0.148\Omega \tag{25}$$

$$X_C = \frac{1}{(185.1 + 10^{-6}) * (2 * 60 * \pi)} = 14.33\ \Omega \tag{26}$$

$$\mathrm{R_{sa}} = 6.099\Omega \tag{27}$$

$$\mathrm{R_{ra}} = \frac{112.77}{3.4318^2} - 6.099 = 3.47\Omega \tag{28}$$

$$X_{sa} = X_{ra} = \frac{X_{LR}}{2} = \frac{\sqrt{\left(\frac{42.130}{3.4318}\right)^2 - \left(\frac{112.77}{3.4318^2}\right)^2}}{2} = 3.84\boldsymbol{\Omega} \tag{29}$$

$$X_{ma} = 2\left[\sqrt{\left(\frac{39.312}{4.87521}\right)^2 - \left(6.099 + \frac{3.47}{4}\right)^2} - \left(3.84 + \frac{3.84}{2} - 14.33\right)\right] = \tag{30}$$

$$= 25.26\boldsymbol{\Omega}$$

Appendix C: State-space CSSPIM model

For the program that will perform the simulation of the capacitor-start SPIM, we have to develop a model by setting the corresponding equations for the two phases we will have during the SPIM performance

a) Main circuit
b) Auxiliary circuit

The state-space representation for modeling the dynamic x with input u can be written as:

$$M\dot{x} = A(x)x + Bu \tag{31}$$

where the constant B = $[1; 1; 0; 0; 0; 0; 0]^T$ relates the input voltage *u* to the first two stator winding equations. Matrices M and A depend on the motor parameters and dynamic states, including the inductance, resistance, and capacitance values, as well as current variables and rotor speed.

We have to distinguish between the start-phase and run-phase:

START-PHASE

The switch is connected. We apply KVL to the four inner circuits.

The dynamic system state variables are stacked in the vector:

$$x = [iqs, ids, iqr, idr, v_c, wr]^T \tag{32}$$

where wr is the rotor electrical speed and the other current and voltage variables are the variables that will affect to the circuit.

We have six variables, so we need to set six equations: one for each inner circuit, one for the voltage across the capacitor, and one for Newton´s Second Law applied to the rotor.

- Main winding stator:

$$v_{qs} - i_{qs} \cdot r_s - L_{ls} \cdot \frac{di_{qs}}{dt} - L_{ms}\left(\frac{di_{qs}}{dt} + \frac{di_{qr}}{dt}\right) = 0 \tag{33}$$

$$\frac{di_{qs}}{dt}(L_{ls} + L_{ms}) + \frac{di_{qr}}{dt}L_{ms} = v_{qs} - i_{qs} \cdot r_s \tag{34}$$

- Auxiliary winding stator:

$$v_{ds} - r_c \cdot C \cdot \frac{dv_c}{dt} - v_c - i_{ds} \cdot r_S - L_{lS} \cdot \frac{di_{ds}}{dt} - L_{mS}\left(\frac{di_{ds}}{dt} + \frac{di_{dr}}{dt}\right) = 0 \tag{35}$$

$$\frac{di_{ds}}{dt}(L_{lS} + L_{mS}) + \frac{di_{dr}}{dt}L_{mS} + r_c \cdot C \cdot \frac{dv_c}{dt} = -v_c - i_{ds} \cdot r_S + v_{ds} \tag{36}$$

- Main winding rotor:

$$L_{ms}\left(\frac{di_{qs}}{dt} + \frac{di_{qr}}{dt}\right) + L'_{lr} \cdot \frac{di_{qr}}{dt} - \frac{N_m}{N_a} \cdot \lambda_{dr} \cdot w_r + i_{qr} \cdot r'_r = 0 \tag{37}$$

Since

$$\lambda_{dr} = i_{dr} \cdot L'_{lR} + (i_{ds} + i_{dr}) \cdot L_{mS} = i_{ds} \cdot L_{mS} + i_{dr}(L'_{lR} + L_{mS}) \tag{38}$$

We have that

$$\frac{di_{qs}}{dt}L_{ms} + \frac{di_{qr}}{dt}(L_{ms} + L'_{lr}) =$$
$$= -i_{qr} \cdot r'_r + \frac{N_m}{N_a}w_r \cdot i_{ds} \cdot L_{mS} + \frac{N_m}{N_a}w_r \cdot i_{dr}(L'_{lR} + L_{mS}) \tag{39}$$

- Auxiliary winding rotor:

$$L_{mS}\left(\frac{di_{ds}}{dt}+\frac{di_{dr}}{dt}\right)+L'_{lR}\cdot\frac{di_{dr}}{dt}-\frac{N_a}{N_m}\cdot\lambda_{qr}\cdot w_r+i_{dr}\cdot r'_R=0 \tag{40}$$

Since

$$\lambda_{qr}=i_{qr}\cdot L'_{lr}+\left(i_{qs}+i_{qr}\right)\cdot L_{ms}=i_{qs}\cdot L_{ms}+i_{qr}(L'_{lr}+L_{ms}) \tag{41}$$

We have that

$$\begin{gathered}\frac{di_{ds}}{dt}L_{mS}+\frac{di_{dr}}{dt}(L_{mS}+L'_{lR})=\\=-i_{dr}\cdot r'_R-\frac{N_a}{N_m}w_r\cdot i_{qs}\cdot L_{ms}-\frac{N_a}{N_m}w_r\cdot i_{qr}(L'_{lr}+L_{ms})\end{gathered} \tag{42}$$

- For the voltage across the capacitor:

$$C\cdot\frac{d\dot{v}_c}{dt}=i_{ds} \tag{43}$$

The last row comes from the Newton’s second law applied to the rotor: *the product of angular acceleration and the rotor’s moment of inertia equals to the net torque on the shaft, which is the sum of electrical and mechanical torque.* Note that this equation generalizes for any p-pole capacitor-start - capacitor-run SPIMs.

The correspondent matrixes for this phase are:

$$M=\begin{bmatrix}L_{ls}+L_{ms} & 0 & L_{ms} & 0 & 0 & 0\\ 0 & L_{lS}+L_{mS} & 0 & L_{mS} & r_cC & 0\\ L_{ms} & 0 & L_{ms}+L'_{lr} & 0 & 0 & 0\\ 0 & L_{mS} & 0 & L_{mS}+L'_{lR} & 0 & 0\\ 0 & 0 & 0 & 0 & C & 0\\ 0 & 0 & 0 & 0 & 0 & \frac{2J}{P}\end{bmatrix} \tag{44}$$

$$A = \begin{bmatrix} -r_s & 0 & 0 & 0 & 0 & 0 \\ 0 & -r_S & 0 & 0 & -1 & 0 \\ 0 & \frac{N_m}{N_a} w_r L_{mS} & -r'_r & \frac{N_m}{N_a} w_r (L'_{lR} + L_{mS}) & 0 & 0 \\ -\frac{N_a}{N_m} w_r L_{ms} & 0 & -\frac{N_a}{N_m} w_r (L'_{lr} + L_{ms}) & -r'_R & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ \frac{p \frac{N_a}{N_m} L_{ms} i'_{dr}}{2} & \frac{-p \frac{N_a}{N_m} L_{ms} i'_{qr}}{2} & 0 & 0 & 0 & -D - k w_r \end{bmatrix} \tag{45}$$

RUN-PHASE

The switch is not connected. Due to this the auxiliary winding of the stator will be disconnected and we have that:

$$\text{ids} = 0 \tag{46}$$

We apply KVL to the three left inner circuits. The dynamic system state variables are stacked in the vector:

$$x = [\text{iqs}, \text{iqr}, \text{idr}, \text{wr}]^T \tag{47}$$

We have four variables, so we need to set four equations: one for each inner circuit and one for Newton´s Second Law applied to the rotor.

- Main winding stator:

$$v_{qs} - i_{qs} \cdot r_s - L_{ls} \cdot \frac{di_{qs}}{dt} - L_{ms}\left(\frac{di_{qs}}{dt} + \frac{di_{qr}}{dt}\right) = 0 \tag{48}$$

$$\frac{di_{qs}}{dt}(L_{ls} + L_{ms}) + \frac{di_{qr}}{dt} L_{ms} = v_{qs} - i_{qs} \cdot r_s \tag{49}$$

- Main winding rotor:

$$L_{ms}\left(\frac{di_{qs}}{dt}+\frac{di_{qr}}{dt}\right)+L'_{lr}\cdot\frac{di_{qr}}{dt}-\frac{N_m}{N_a}\cdot\lambda_{dr}\cdot w_r+i_{qr}\cdot r'_r=0 \tag{50}$$

Since

$$\lambda_{dr}=i_{dr}\cdot L'_{lR}+i_{dr}\cdot L_{mS}=i_{dr}(L'_{lR}+L_{mS}) \tag{51}$$

We have that

$$\frac{di_{qs}}{dt}L_{ms}+\frac{di_{qr}}{dt}(L_{ms}+L'_{lr})=-i_{qr}\cdot r'_r+\frac{N_m}{N_a}w_r\cdot i_{dr}(L'_{lR}+L_{mS}) \tag{52}$$

$$\frac{di_{qs}}{dt}L_{ms}+\frac{di_{qr}}{dt}(L_{ms}+L'_{lr})=-i_{qr}\cdot r'_r+\frac{N_m}{N_a}w_r\cdot i_{dr}(L'_{lR}+L_{mS}) \tag{53}$$

- Auxiliary winding rotor:

$$L_{mS}\cdot\frac{di_{dr}}{dt}+L'_{lR}\cdot\frac{di_{dr}}{dt}-\frac{N_a}{N_m}\cdot\lambda_{qr}\cdot w_r+i_{dr}\cdot r'_R=0 \tag{54}$$

Since

$$\lambda_{qr}=i_{qr}\cdot L'_{lr}+\left(i_{qs}+i_{qr}\right)\cdot L_{ms}=i_{qs}\cdot L_{ms}+i_{qr}(L'_{lr}+L_{ms}) \tag{55}$$

We have that

$$\frac{di_{dr}}{dt}(L_{mS}+L'_{lR})=-i_{dr}\cdot r'_R-\frac{N_a}{N_m}w_r\cdot i_{qs}\cdot L_{ms}-\frac{N_a}{N_m}w_r\cdot i_{qr}(L'_{lr}+L_{ms}) \tag{56}$$

The correspondent matrixes for this phase are:

$$M = \begin{bmatrix} L_{ls} + L_{ms} & L_{ms} & 0 & 0 \\ L_{ms} & L_{ms} + L'_{lr} & 0 & 0 \\ 0 & 0 & L_{mS} + L'_{lR} & 0 \\ 0 & 0 & 0 & \frac{2J}{P} \end{bmatrix} \tag{57}$$

$$A = \begin{bmatrix} -r_s & 0 & 0 & 0 \\ 0 & -r'_r & \frac{N_m}{N_a} w_r (L'_{lR} + L_{mS}) & 0 \\ -\frac{N_a}{N_m} w_r \cdot L_{ms} & -\frac{N_a}{L_m} w_r (L'_{lr} + L_{ms}) & -r'_R & 0 \\ \frac{p \frac{N_a}{N_m} L_{ms} i'_{dr}}{2} & 0 & 0 & D - k w_r \end{bmatrix} \tag{58}$$

Appendix D: Simulation Programs

For the simulation of the CSSPIM model we have design two programs:

1. Motor: In this program we will define all the variables, matrixes…etc that we have found in Appendix D (State-space CSSPIM model). This way we will simulate the main structure and performance of the motor. We have to distinguish between :
   - CSCR (Capacitor Start – Capacitor Run SPIM)
   - CS (Capacitor Start SPIM)

   In our case we will be interested in the second option (CSSPIM), that is the type of motor we will be working with.

2. MyDemo: will perform the simulation. We have the option of choosing some of the variables that will affect the simulation.
   - Motor Type (CS/CSCR)
   - Voltage applied
   - Switching time
   - Torque

We will do it by introducing the values we want in the following function:

stdstates = MotorSimCont(myMotor, myV, tsw, myTm);

Table 9. Parameters to choose in MyDemo program

Motor Type	myMotor
Voltage applied	myV
Switching time	tsw
Torque	myTm

MOTOR

```
 classdef Motor
  %MOTOR Summary of this class goes here
  % Date: 10/24/2014
  % 1. If ParamIdx and Variation are not empty, then the Parameters are
  % changed accordingly
  %   Detailed explanation goes here

  properties
      MotorType
      Params
      Variation
      States
      Tdiscrete
      FuncGenState
  end
  properties (Constant)
      ParamsName =
{'rs','rS','rr_p','Lls','LlS','Llr_p','Lms','n','r1','r2','C1','C2','p','J','D','k'} % Name of
independent parameters
      DepParamsName = {'Lss','LSS','LmS','LlR_p','Lrr_p','LRR_p','rR_p'} % Name of
dependent parameters
  end
  properties (Dependent)
      ParamsVal
      DepParamsVal
      %Ms      % M matrix in start phase
```

```
    %MsInv   % Inverse of M matrix in start phase

    %Bs      % Relates input to states matrix in start phase

    %Mr      % M matrix in run phase

    %MrInv   % Inverse of M matrix in run phase

    %Br      % Relates input to states matrix in run phase


    % The set of discrete parameters

    %AsD

    %BsD

    %ArD

    %BrD


    SymDynamics

end


methods

    % Class constructor

    function myMotor = Motor(arg1, arg2, arg3, arg4)


    %Description

    % If no argument in, put out an error and don’t do anything.

    % 1. If one argument is none zero, arg1 is type of motor (‘CSCR’ or ‘CS’)

    % 2. If two arguments are none zero, arg1 is type of motor, arg2 is Tdiscrete

    % 3. If three arguments are none zero, arg1 is type of motor, arg2 is parameters to
      change, and arg3 is percentage of variation

    % 4.If four arguments are none zero,arg1 is type of motor, arg2 is Tdicrete,  arg3 is
      parameters to change, and arg4 is percentage of vatiation

    % Note:
```

```
%  - Params to vary must be in cell structure, i.e. {'Lms','Llr'}
%  - Variation rate is in percent, i.e. [10, -10] means plus 10%
%    for Lms and -10% for Llr

    if (nargin == 0)
        myMotor.Tdiscrete = 1e-4;
    elseif (nargin == 1)
        myMotor.Tdiscrete = 1e-4;
        myMotor.MotorType = arg1;
    elseif (nargin == 2)
        myMotor.MotorType = arg1;
        myMotor.Tdiscrete = arg2;
    elseif (nargin == 3)
        myMotor.Tdiscrete = 1e-4;
        myMotor.MotorType = arg1;
        myMotor.Params = arg2;
        myMotor.Variation = arg3;
    elseif (nargin == 4)
        myMotor.MotorType = arg1;
        myMotor.Tdiscrete = arg2;
        myMotor.Params = arg3;
        myMotor.Variation = arg4;
    end;
end;

% Calculate the dependent properties
function value = get.ParamsVal(theMotor)
```

```
            value = [2.0200, 7.1400, 4.1200, 0.0074, 0.0085, 0.0056, 0.1772, 1.1800, 9,
3.5030, 1.5422e-05,1.6793e-04, 4, 0.0146, 0, 0];

            if ~isempty(theMotor.Params) && ~isempty(theMotor.Variation)

                if isequal(class(theMotor.Params),'double') % if the input is double, use it as
Idx

                    Idx = theMotor.Params;

                elseif isequal(class(theMotor.Params),'cell') % if the input is cell or char, find
the Idx

                    for k1 = 1:length(theMotor.Params)

                        %disp(length(theMotor.Params))

                        for k2 = 1:length(theMotor.ParamsName)

                            %disp('loop 2 checked')

                            if isequal(theMotor.Params(k1), theMotor.ParamsName(k2))

                                Idx(k1) = k2;

                            end;

                        end;

                    end;

                end;


                value(Idx) = value(Idx).*(1+theMotor.Variation/100);

                %disp('Entered the parameter variation');

            end;

        end; % Varying the parameters as desired

        function value = get.DepParamsVal(theMotor)

            % ParamsName    'rs',  'rS',  'rr_p',  'Lls',  'LlS',  'Llr_p',  'Lms',  'n',  'r1',  'r2',
'C1',  'C2',  'p',  'J',  'D',  'k'

            % index         1      2      3       4       5       6        7       8    9      10     11     12
13    14    15    16
```

```
            value(1) = theMotor.ParamsVal(4) + theMotor.ParamsVal(7);% Lss = Lls +
Lms
            value(3) = theMotor.ParamsVal(8)^2*theMotor.ParamsVal(7);% LmS =
n^2*Lms
            value(2) = theMotor.ParamsVal(5) + value(3);% LSS = LlS + LmS
            value(4) = theMotor.ParamsVal(8)^2*theMotor.ParamsVal(6);% LlR_p =
n^2*Llr_p
            value(5) = theMotor.ParamsVal(6) + theMotor.ParamsVal(7);% Lrr_p = Llr_p +
Lms
            value(6) = value(4) + value(3);% LRR_p = LlR_p + LmS
            value(7) = theMotor.ParamsVal(8)^2*theMotor.ParamsVal(3); % rR_p =
n^2*rr_p;
        end; % Calculating the dependent params
%       function value = get.Ms(theMotor)
%           Lss = theMotor.DepParamsVal(1);
%           Lms = theMotor.ParamsVal(7);
%           LSS = theMotor.DepParamsVal(2);
%           LmS = theMotor.DepParamsVal(3);
%           r1 = theMotor.ParamsVal(9);
%           C1 = theMotor.ParamsVal(11);
%           r2 = theMotor.ParamsVal(10);
%           C2 = theMotor.ParamsVal(12);
%           Lrr_p = theMotor.DepParamsVal(5);
%           LRR_p = theMotor.DepParamsVal(6);
%           J = theMotor.ParamsVal(14);
%           p = theMotor.ParamsVal(13);
%           value = [   Lss     0      Lms     0      0      0      0;...
%                       0      LSS     0      LmS    r1*C1   0      0;...
%                       Lms     0      Lrr_p   0      0      0      0;...
%                  0      LmS     0      LRR_p   0      0      0;...
%                  0      0       0      0       r1*C1  -r2*C2 0;...
%                  0      0       0      0       C1     C2     0;...
%                  0      0       0      0       0      0      J*2/p];
%          end; % Calculating the M matrix for start phase
%          function value = get.MsInv(theMotor)
%             value = inv(theMotor.Ms);
%          end; % Calculating the inver of M matrix for the start phase
%          function value = get.Mr(theMotor)
%             Lss = theMotor.DepParamsVal(1);
%             Lms = theMotor.ParamsVal(7);
%             LSS = theMotor.DepParamsVal(2);
%             LmS = theMotor.DepParamsVal(3);
%             r1 = theMotor.ParamsVal(9);
%             C1 = theMotor.ParamsVal(11);
%             C2 = theMotor.ParamsVal(12);
%             Lrr_p = theMotor.DepParamsVal(5);
%             LRR_p = theMotor.DepParamsVal(6);
%             J = theMotor.ParamsVal(14);
%             p = theMotor.ParamsVal(13);
%             value  = [   Lss     0      Lms     0      0      0      0;...
%                  0      LSS     0      LmS     r1*C1  0      0;...
%                  Lms    0       Lrr_p  0       0      0      0;...
%                  0      LmS     0      LRR_p   0      0      0;...
%                  0      0       0      0       C1     0      0;...
%                  0      0       0      0       0      C2     0;...
%                  0      0       0      0       0      0      J*2/p];
%       end; % Calculating the M matrix for run phase
%       function value = get.MrInv(theMotor)
%           value = inv(theMotor.Mr);
%       end; % Calculating the inverse of M matrix for the run phase
%       function value = get.Bs(theMotor)
%           value = theMotor.MsInv * [1;1;0;0;0;0;0];
%       end; % Calculating the Bs matrix in start phase
%       function value = get.Br(theMotor)
%           value = theMotor.MrInv*[1;1;0;0;0;0;0];
%       end; % Calculating the Br matrix in run phase
%       function value = get.BsD (theMotor)
%           value = theMotor.MsInv * [1;1;0;0;0;0;0] * theMotor.Tdiscrete;
%       end; % Calculating the matrix that relates input to the states for the start phase
%       function value = get.BrD (theMotor)
%           value = theMotor.MrInv * [1;1;0;0;0;0;0] * theMotor.Tdiscrete;
%       end;
%
    % Calculating the matrix that relates input to the states for the start phase
    function value = get.SymDynamics(theMotor)


         if isequal(theMotor.MotorType,'CSCR')
        syms J n P_rated wb p rs Xms rr_p Xls Xlr_p rS XmS rR_p XlS XlR_p
        syms r2 C2 r1 C1 Lms Lls Llr_p LlS LmS LlR_p Lss LSS Lrr_p LRR_p k D
        syms ids iqs iqr_p idr_p V_C1 V_C2 wr
        syms ui Tm


        X = [iqs; ids; iqr_p; idr_p; V_C1; V_C2; wr];
```

```
alpha = [rs; rS; rr_p; Lls; LlS; Llr_p; Lms; n; r1; r2; C1; C2; p; J; D; k];


LmS = Lms*n^2;

rR_p = rr_p*n^2;

LlR_p = Llr_p*n^2;


Lss = Lls + Lms;

LSS = LlS + LmS;

Lrr_p = Llr_p + Lms;

LRR_p = LlR_p + LmS;


value.Ms = [    Lss     0          Lms         0          0             0             0;...

             0      LSS        0          LmS        r1*C1          0             0;...

             Lms     0          Lrr_p       0          0             0             0;...

             0      LmS        0          LRR_p      0             0             0;...

             0      0          0          0          r1*C1         -r2*C2        0;...

             0      0          0          0          C1            C2            0;...

             0      0          0          0          0             0             J*2/p];


value.As = [   -rs            0                 0               0             0       0       0;...

             0             -rS               0               0             -1      0       0;...

             0       X(7)*LmS/n       -rr_p            X(7)*LRR_p/n    0       0       0;...

             -n*X(7)*Lms     0         -n*X(7)*Lrr_p     -rR_p          0       0       0;...

             0             0                 0               0             -1      1       0;...

             0             1                 0               0              0       0       0;...

             p*n*Lms*X(4)/2  -p*n*Lms*X(3)/2    0      0    0     0       -D-k*X(7)];
```

```
value.Mr = [   Lss     0       Lms     0       0       0       0;...
               0       LSS     0       LmS     r1*C1   0       0;...
               Lms     0       Lrr_p   0       0       0       0;...
               0       LmS     0       LRR_p   0       0       0;...
               0       0       0       0       C1      0       0;...
               0       0       0       0       0       C2      0;...
               0       0       0       0       0       0       J*2/p];

value.Ar = [ -rs              0                 0             0           0          0          0;...
               0           -rS               0             0           -1         0          0;...
               0      X(7)*LmS/n         -rr_p          X(7)*LRR_p/n    0    0        0;...
               -n*X(7)*Lms        0       -n*X(7)*Lrr_p   -rR_p      0        0        0;...
               0                1                 0             0           0          0          0;...
               0                0                 0             0           0          0          0;...
               p*n*Lms*X(4)/2     -p*n*Lms*X(3)/2     0    0     0    0    -D-k*X(7)];

 value.Bs = [1;1;0;0;0;0;0];
 value.Ds = [0;0;0;0;0;0;-1];
 value.Br = [1;1;0;0;0;0;0];
 value.Dr = [0;0;0;0;0;0;-1];
 value.Statess = X;
 value.Statesr = X;
 value.Params = alpha;
 value.Vinput = ui;
 value.Tinput = Tm;

elseif isequal(theMotor.MotorType, 'CS')
```

```
syms J n P_rated wb p rs Xms rr_p Xls Xlr_p rS XmS rR_p XlS XlR_p
syms r2 C2 r1 C1 Lms Lls Llr_p LlS LmS LlR_p Lss LSS Lrr_p LRR_p k D
syms ids iqs iqr_p idr_p V_C1 V_C2 wr
syms ui Tm

X_s = [iqs; ids; iqr_p; idr_p; V_C2; wr]; % start phase
alpha = [rs; rS; rr_p; Lls; LlS; Llr_p; Lms; n; r1; r2; C1; C2; p; J; D; k]; % start
phase

X_r = [iqs; iqr_p; idr_p; wr]; % run phase
%alpha_r = [rs; rr_p; Lls; Llr_p; Lms; n;  p; J; D; k]; % run phase

LmS = Lms*n^2;
rR_p = rr_p*n^2;
LlR_p = Llr_p*n^2;

Lss = Lls + Lms;
LSS = LlS + LmS;
Lrr_p = Llr_p + Lms;
LRR_p = LlR_p + LmS;

value.Ms = [    Lss     0          Lms         0          0              0;...
                0       LSS        0          LmS        0              0;...
                Lms     0          Lrr_p      0          0              0;...
                0       LmS        0          LRR_p      0              0;...
                0       0          0          0          C2             0;...
```

```
                 0     0        0        0        0           J*2/p];

value.As = [   -rs            0              0              0            0      0;...
            0            -rS-r2          0              0            -1     0;...
            0      X_s(6)*LmS/n      -rr_p          X_s(6)*LRR_p/n    0      0;...
            -n*X_s(6)*Lms    0        -n*X_s(6)*Lrr_p    -rR_p          0      0;...
            0            1              0              0            0                0;...
          p*n*Lms*X_s(4)/2  -p*n*Lms*X_s(3)/2    0    0     0    -D-k*X_s(6)];

value.Mr = [   Lss     Lms     0      0;...
            Lms     Lrr_p   0      0;...
            0       0      LRR_p   0;...
            0       0      0      J*2/p];

value.Ar = [    -rs               0              0            0;...
             0               -rr_p          X_r(4)*LRR_p/n    0;...
             -n*X_r(4)*Lms      -n*X_r(4)*Lrr_p   -rR_p          0;...
             p*n*Lms*X_r(3)/2      0              0            -D-k*X_r(4)];

 value.Bs = [1;1;0;0;0;0];
 value.Ds = [0;0;0;0;0;-1];
 value.Br = [1;0;0;0];
 value.Dr = [0;0;0;-1];
 value.Statess = X_s;
 value.Statesr = X_r;
 value.Params = alpha;
 value.Vinput = ui;
```

```
         value.Tinput = Tm;
        end;


    end; % Get the dynamics of motor in symbolic representation


    % Simulate the motor's operation
    function states = MotorSimCont(theMotor, Vsource, tsw, Tmsource)
        if isempty(theMotor.FuncGenState)
            FuncGen(theMotor);
        end;


        if isequal(theMotor.MotorType,'CSCR')
        %states(1:n,:) = MotorStartSim(theMotor,tstart)
        [val, swIdx] = min(abs(Vsource.time-tsw));
        X(1:swIdx,:) = MotorSimStart(theMotor, Vsource, swIdx, Tmsource);
        swStates = X(swIdx,:);
        X(swIdx+1:length(Vsource.time),:) = MotorSimRun(theMotor, Vsource, swIdx,
swStates, Tmsource);


        states.iqs = X(:,1);
        states.ids = X(:,2);
        states.iqr_p = X(:,3);
        states.idr_p = X(:,4);
        states.vC1 = X(:,5);
        states.vC2 = X(:,6);
        states.wr = X(:,7);
        elseif isequal(theMotor.MotorType, 'CS')
        [val, swIdx] = min(abs(Vsource.time-tsw));
        X(1:swIdx,:) = MotorSimStart(theMotor, Vsource, swIdx, Tmsource);
        swStates = X(swIdx,[1,3,4,6]);
        X(swIdx+1:length(Vsource.time),[1,3,4,6]) = MotorSimRun(theMotor, Vsource,
swIdx, swStates, Tmsource);


        states.iqs = X(:,1);
        states.ids = X(:,2);
        states.iqr_p = X(:,3);
        states.idr_p = X(:,4);
        states.vC2 = X(:,5);
        states.wr = X(:,6);
        end;
      end;
      function states = MotorSimDis(theMotor, Vsource, tsw, Tmsource)
        if isempty(theMotor.FuncGenState)
          FuncGen(theMotor);
        end;
        [val, swIdx] = min(abs(Vsource.time-tsw));
        X(1:swIdx,:) = MotorSimStartDis(theMotor, Vsource, swIdx, Tmsource);
        swStates = X(swIdx,:);
        X(swIdx+1:length(Vsource.time),:) = MotorSimRunDis(theMotor, Vsource,
swIdx, swStates, Tmsource);
        states.iqs = X(:,1);
        states.ids = X(:,2);
        states.iqr_p = X(:,3);
        states.idr_p = X(:,4);
```

```
    states.vC1 = X(:,5);
    states.vC2 = X(:,6);
    states.wr = X(:,7);
end;

function value = FuncGen(theMotor)
    temp = theMotor.SymDynamics;
    Nstart = length(temp.Bs);
    Nrun = length(temp.Br);

    Abarsc = inv(temp.Ms)*temp.As;
    Bbarsc = inv(temp.Ms)*temp.Bs;
    Dbarsc = inv(temp.Ms)*temp.Ds;

    Abarrc = inv(temp.Mr)*temp.Ar;
    Bbarrc = inv(temp.Mr)*temp.Br;
    Dbarrc = inv(temp.Mr)*temp.Dr;

    dt = theMotor.Tdiscrete;
    Abarsd = eye(Nstart) + inv(temp.Ms)*temp.As*dt;
    Bbarsd = inv(temp.Ms)*temp.Bs*dt;
    Dbarsd = inv(temp.Ms)*temp.Ds*dt;

    Abarrd = eye(Nrun) + inv(temp.Mr)*temp.Ar*dt;
    Bbarrd = inv(temp.Mr)*temp.Br*dt;
    Dbarrd = inv(temp.Mr)*temp.Dr*dt;
```

```
            f1 = matlabFunction(Abarsc, 'File', 'AbarscCal', 'vars', {[temp.Statess;
temp.Params; temp.Vinput; temp.Tinput]});

            f2 = matlabFunction(Bbarsc, 'File', 'BbarscCal', 'vars', {[temp.Statess;
temp.Params; temp.Vinput; temp.Tinput]});

            f3 = matlabFunction(Dbarsc, 'File', 'DbarscCal', 'vars', {[temp.Statess;
temp.Params; temp.Vinput; temp.Tinput]});



            f4 = matlabFunction(Abarrc, 'File', 'AbarrcCal', 'vars', {[temp.Statesr;
temp.Params; temp.Vinput; temp.Tinput]});

            f5 = matlabFunction(Bbarrc, 'File', 'BbarrcCal', 'vars', {[temp.Statesr;
temp.Params; temp.Vinput; temp.Tinput]});

            f6 = matlabFunction(Dbarrc, 'File', 'DbarrcCal', 'vars', {[temp.Statesr;
temp.Params; temp.Vinput; temp.Tinput]});



            f1 = matlabFunction(Abarsd, 'File', 'AbarsdCal', 'vars', {[temp.Statess;
temp.Params; temp.Vinput; temp.Tinput]});

            f2 = matlabFunction(Bbarsd, 'File', 'BbarsdCal', 'vars', {[temp.Statess;
temp.Params; temp.Vinput; temp.Tinput]});

            f3 = matlabFunction(Dbarsd, 'File', 'DbarsdCal', 'vars', {[temp.Statess;
temp.Params; temp.Vinput; temp.Tinput]});



            f4 = matlabFunction(Abarrd, 'File', 'AbarrdCal', 'vars', {[temp.Statesr;
temp.Params; temp.Vinput; temp.Tinput]});

            f5 = matlabFunction(Bbarrd, 'File', 'BbarrdCal', 'vars', {[temp.Statesr;
temp.Params; temp.Vinput; temp.Tinput]});

            f6 = matlabFunction(Dbarrd, 'File', 'DbarrdCal', 'vars', {[temp.Statesr;
temp.Params; temp.Vinput; temp.Tinput]});



            theMotor.FuncGenState = 1;

        end; % Generate functions to calculate dynamical model numerically



    end
```

```
end

%% Supporting function for the class method

% Solve the differential equation for start phase

function states = MotorSimStart(theMotor, Vsource, swIdx, Tmsource)

   Nstart = length(theMotor.SymDynamics.Bs);

   pVal = theMotor.ParamsVal';

   tstart = Vsource.time(1:swIdx);

   Vin = Vsource.vt(1:swIdx);

   tVin = Vsource.time(1:swIdx);

   Tmin = Tmsource.Tm(1:swIdx);

   tTmin = Tmsource.time(1:swIdx);

   Xinit = zeros(Nstart,1);

   options = odeset('RelTol', 1e-6, 'AbsTol', ones(1,Nstart)*1e-4, 'MaxStep', 0.001);

   [T_start, X_start] = ode15s(@(t,X) MotorStartModel(t,X, pVal, tVin, Vin, tTmin,
Tmin), tstart , Xinit, options);

   states = X_start;

end


% Solve the differential equation for run phase

function states = MotorSimRun(theMotor,Vsource, swIdx, Xinit, Tmsource)

   Nrun = length(theMotor.SymDynamics.Br);

   pVal = theMotor.ParamsVal';

   trun = Vsource.time(swIdx+1:length(Vsource.time));

   Vin = Vsource.vt(swIdx+1:length(Vsource.time));

   tVin = trun;

   Tmin = Tmsource.Tm((swIdx+1:length(Vsource.time)));

   tTmin = Tmsource.time(swIdx+1:length(Vsource.time));
```

```
    options = odeset('RelTol', 1e-6, 'AbsTol', ones(1,Nrun)*1e-4, 'MaxStep', 0.001);
    [T_run, X_run] = ode15s(@(t,X) MotorRunModel(t,X, pVal, tVin, Vin, tTmin,
Tmin), trun , Xinit, options);
    states = X_run;
end

% Define the differential equation for start phase
function dX = MotorStartModel(t,X, pVal, tVin, Vin, tTmin, Tmin)
    u = interp1(tVin, Vin, t);
    Tm = interp1(tTmin, Tmin, t);
    Abar = AbarscCal([X;pVal]);
    Bbar = BbarscCal([X;pVal]);
    Dbar = DbarscCal([X;pVal]);
    dX = Abar*X + Bbar * u + Dbar * Tm;
end

% Define the differential equation for the run phase
function dX = MotorRunModel(t,X, pVal, tVin, Vin, tTmin, Tmin)
    u = interp1(tVin, Vin, t);
    Tm = interp1(tTmin, Tmin, t);
    Abar = AbarrcCal([X;pVal]);
    Bbar = BbarrcCal([X;pVal]);
    Dbar = DbarrcCal([X;pVal]);
    dX = Abar*X + Bbar * u + Dbar * Tm;
end

% Simulation of motor start phase operation in discrete time
```

```
function states = MotorSimStartDis(theMotor, Vsource, swIdx, Tmsource)
    tstart = Vsource.time(1:swIdx);
    u = Vsource.vt(1:swIdx);
    Tm = Tmsource.Tm(1:swIdx);
    Xinit = zeros(7,1);
    pVal = theMotor.ParamsVal';
    X = Xinit;
    Bbard = BbarsdCal([X; pVal; 0; 0]);
    Dbard = DbarsdCal([X; pVal; 0; 0]);
    for idx = 2:length(tstart)
        Abard = AbarsdCal([X; pVal; 0; 0]);
        X = Abard * X + Bbard * u(idx-1) + Dbard * Tm(idx-1);
        myX(idx,:) = X';
    end;
    states = myX;
end

% Simulation of motor run phase operation in discrete time
function states = MotorSimRunDis(theMotor, Vsource, swIdx, Xinit, Tmsource)
    trun = Vsource.time(swIdx+1:length(Vsource.time));
    u = Vsource.vt(swIdx:length(Vsource.time)-1);
    Tm = Tmsource.Tm(swIdx:length(Vsource.time)-1);
    X = Xinit';
    pVal = theMotor.ParamsVal';
    Bbar = BbarrdCal([X; pVal; 0; 0]);
    Dbar = DbarrdCal([X; pVal; 0; 0]);
    for idx = 1:length(trun)
```

```
        Abar = AbarrdCal([X; pVal; 0; 0]);

        X = Abar*X + Bbar * u(idx) + Dbar * Tm(idx);

        myX(idx,:) = X';

    end;

    states = myX;

end
```

MyDemo

```
close all;
clear all;
clc;

% add classes to the path
addpath('Class');

% Create voltage source object at 120Vrms, 60Hz, 0 phase angle, last 3
% seconds, and sampling period of 10^-4 s
myV = Vsource(120*sqrt(2), 60, 0, 1e-4, 2);
time = myV.time;
% Create a motor object at standard parameters values
myMotor = Motor('CS');

% Create a load profile
Tmag = 0; % no load acceleration
myTm = Tmsource(Tmag, 1e-4, 2); % magnitude of 0.0 N.m, sampling period
of 10^-4s, and last 3 seconds

% Generate motor transients under voltage given in myV
tsw = 0.2960; % motor switching time
stdstates = MotorSimCont(myMotor, myV, tsw, myTm);

label = {'i_{qs}','i_{ds}','i_{qr}p','i_{dr}p','V_{C1}','V_{C2}','w_r'};
figure(1);
subplot(4,2,1); plot(time, stdstates.iqs); grid on; xlabel('Time [s]'); ylabel('[A]');
legend(label{1}); axis([min(time) max(time) 1.1*min(stdstates.iqs)
1.1*max(stdstates.iqs)]);
```

```
subplot(4,2,2); plot(time, stdstates.ids); grid on; xlabel('Time [s]'); ylabel('[A]');
legend(label{2}); axis([min(time) max(time) 1.1*min(stdstates.ids)
1.1*max(stdstates.ids)]);
subplot(4,2,3); plot(time, stdstates.iqr_p); grid on; xlabel('Time [s]');
ylabel('[A]'); legend(label{3}); axis([min(time) max(time)
1.1*min(stdstates.iqr_p) 1.1*max(stdstates.iqr_p)]);
subplot(4,2,4); plot(time, stdstates.idr_p); grid on; xlabel('Time [s]');
ylabel('[A]'); legend(label{4}); axis([min(time) max(time)
1.1*min(stdstates.idr_p) 1.1*max(stdstates.idr_p)]);
%subplot(4,2,7); plot(time, stdstates.vC1); grid on; xlabel('Time [s]');
ylabel('[V]'); legend(label{5}); axis([min(time) max(time)
1.1*min(stdstates.vC1) 1.1*max(stdstates.vC1)]);
subplot(4,2,5); plot(time, stdstates.vC2); grid on; xlabel('Time [s]'); ylabel('[V]');
legend(label{6}); axis([min(time) max(time) 1.1*min(stdstates.vC2)
1.1*max(stdstates.vC2)]);
subplot(4,2,6); plot(time, stdstates.wr); grid on; xlabel('Time [s]');
ylabel('[rad/s]'); legend(label{7}); axis([min(time) max(time)
1.1*min(stdstates.wr) 1.1*max(stdstates.wr)]);
```

Simulation results

In the following figures are shown a few examples of the plots that we obtained from the programs we developed previously. We have changed the parameters in each case to appreciate the differences and similarities depending on the conditions.

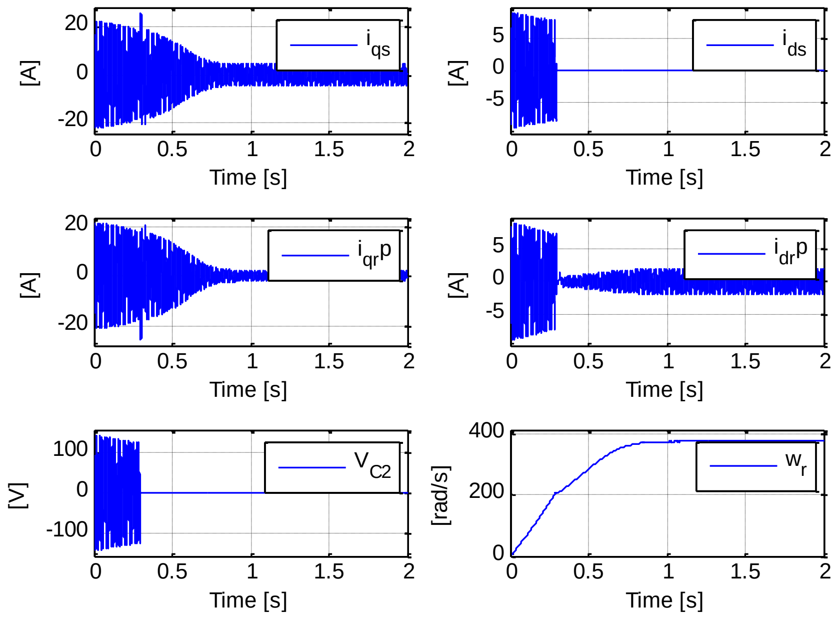


Figure 21 . V=120RMS, 60hZ , torque=0, tsw= 0.2960

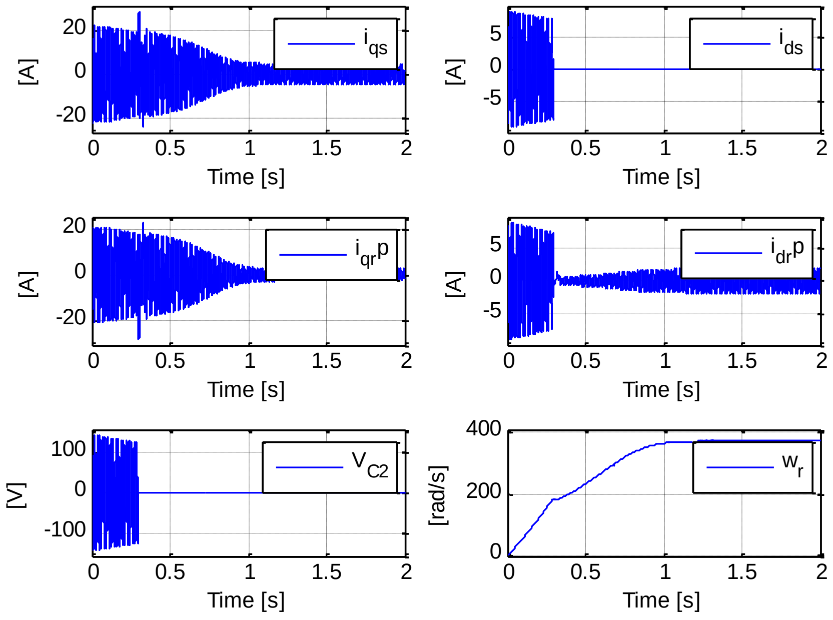


Figure 22 . V=120RMS, 60hZ , torque=0.5, tsw= 0.2960

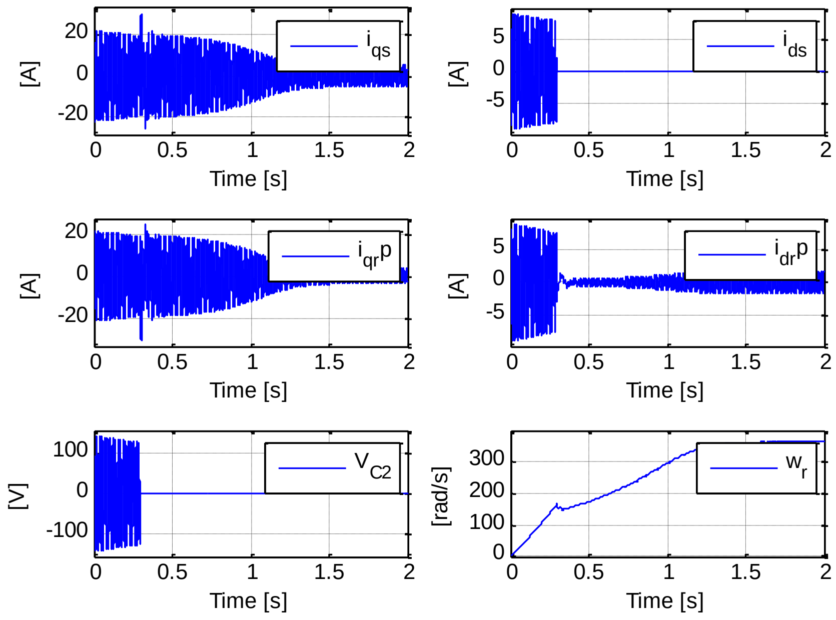


Figure 23 . V=120RMS, 60hZ, torque=1, tsw= 0.2960

Appendix E: SPIM

In this appendix are gathered the basics and operation principles of the induction motors in order to a better understanding of the project.

[6] An induction or asynchronous motor is an AC electric motor in which the electric current in the rotor needed to produce torque is obtained by electromagnetic induction from the magnetic field of the stator winding. An induction motor therefore does not require mechanical commutation, separate-excitation or self-excitation for the energy transferred from stator to rotor.

Like any other electrical motor asynchronous motor also have two main parts namely rotor and stator.

- Stator: As its name indicates stator is a stationary part of induction motor. A single phase ac supply is given to the stator of single phase induction motor.
- Rotor: The rotor is a rotating part of induction motor. The rotor is connected to the mechanical load through the shaft. The rotor in single phase induction motor is of squirrel cage rotor type.

The construction of single phase induction motor is almost similar to the squirrel cage three phase motor except that in case of asynchronous motor the stator have two windings instead of one as compare to the single stator winding in three phase induction motor.

- **Stator of Single Phase Induction Motor**

The stator of the single phase induction motor has laminated stamping to reduce eddy current losses on its periphery. The slots are provided on its stamping to carry stator or main winding. In order to reduce the hysteresis losses, stamping are made up of silicon steel. When the stator winding is given a single phase ac supply, the magnetic field is produced and the motor rotates at a speed slightly less than the synchronous speed N_s which is given by

$$N_s = \frac{120f}{P}$$

The construction of the stator of asynchronous motor is similar to that of three phase induction motor except there are two dissimilarity in the winding part of the single phase induction motor.

1. Firstly the single phase induction motors are mostly provided with concentric coils. As the number of turns per coil can be easily adjusted with the help of concentric coils, the mmf distribution is almost sinusoidal.
2. Except for shaded pole motor, the asynchronous motor has two stator windings namely the main winding and the auxiliary winding. These two windings are placed in space quadrature with respect to each other.

- **Rotor of Single Phase Induction Motor**

The construction of the rotor of the single phase induction motor is similar to the squirrel cage three phase induction motor. The rotor is cylindrical in shape and has slots all over its periphery. The slots are not made parallel to each other but are bit skewed as the skewing prevents magnetic locking of stator and rotor teeth and makes the working of induction motor more smooth and quieter. The squirrel cage rotor consists of aluminium, brass or copper bars. These aluminium or copper bars are called rotor conductors and are placed in the slots on the periphery of the rotor. The rotor conductors are permanently shorted by the copper or aluminium rings called the end rings. In order to provide mechanical strength these rotor conductor are braced to the end ring and hence form a complete closed circuit resembling like a cage and hence got its name as "squirrel cage induction motor". As the bars are permanently shorted by end rings, the rotor electrical resistance is very small and it is not possible to add external resistance as the bars are permanently shorted. The absence of slip ring and brushes make the construction of single phase induction motor very simple and robust.

Working Principle of Single Phase Induction Motor

We know that for the working of any electrical motor whether its ac or dc motor, we require two fluxes as, the interact of these two fluxes produced the required torque, which is desired parameter for any motor to rotate.

When single phase ac supply is given to the stator winding of single phase induction motor, the alternating current starts flowing through the stator or main winding. This alternating current produces an alternating flux called main flux. This main flux also links with the rotor conductors and hence cut the rotor conductors. According to the

Faraday's law of electromagnetic induction, emf gets induced in the rotor. As the rotor circuit is closed one so, the current starts flowing in the rotor. This current is called the rotor current. This rotor current produces its own flux called rotor flux. Since this flux is produced due to induction principle so, the motor working on this principle got its name as induction motor. Now there are two fluxes one is main flux and another is called rotor flux. These two fluxes produce the desired torque which is required by the motor to rotate.

Why Single Phase Induction Motor is not Self Starting?

According to double field revolving theory, any alternating quantity can be resolved into two components, each component have magnitude equal to the half of the maximum magnitude of the alternating quantity and both these component rotates in opposite direction to each other. For example - a flux, φ can be resolved into two components

$$\frac{\phi_m}{2}\ and\ -\frac{\phi_m}{2}$$

Each of these components rotates in opposite direction i. e if one φ_m / 2 is rotating in clockwise direction then the other φ_m / 2 rotates in anticlockwise direction.

When a single phase ac supply is given to the stator winding of single phase induction motor, it produces its flux of magnitude, φ_m. According to the double field revolving theory, this alternating flux, φ_m is divided into two components of magnitude φ_m /2. Each of these components will rotate in opposite direction, with the synchronous speed, N_s. Let us call these two components of flux as forward component of flux, φ_f and backward component of flux, φ_b. The resultant of these two component of flux at any instant of time, gives the value of instantaneous stator flux at that particular instant.

$$i.e. \phi_r = \frac{\phi_m}{2} + \frac{\phi_m}{2}\ or\ \phi_r = \phi_f + \phi_b$$

Now at starting, both the forward and backward components of flux are exactly opposite to each other. Also both of these components of flux are equal in magnitude. So, they cancel each other and hence the net torque experienced by the rotor at starting is zero. So, the single phase induction motors are not self starting motors.

Methods for Making Single Phase Induction as Self Starting Motor

From the above topic we can easily conclude that the single phase induction motors are not self starting because the produced stator flux is alternating in nature and at the starting the two components of this flux cancel each other and hence there is no net torque. The solution to this problem is that if the stator flux is made rotating type, rather than alternating type, which rotates in one particular direction only. Then the induction motor will become self starting. Now for producing this rotating magnetic field we require two alternating flux, having some phase difference angle between them. When these two fluxes interact with each other they will produce a resultant flux. This resultant flux is rotating in nature and rotates in space in one particular direction only. Once the motor starts running, the additional flux can be removed. The motor will continue to run under the influence of the main flux only. Depending upon the methods for making asynchronous motor as Self Starting Motor, there are mainly four types of single phase induction motor namely,

1. Split phase induction motor,
2. Capacitor start inductor motor,
3. Capacitor start capacitor run induction motor,
4. Shaded pole induction motor.

The motor we will be working with in this project is a Capacitor – Start Single Phase Induction Motor.

[7] Capacitor start / induction run motors are similar in construction to split phase motors. The major difference is the use of a capacitor connected in series to start windings to maximize starting torque.

The capacitor is mounted either at the top or side of the motor. A normally closed centrifugal switch is located between the capacitor and the start winding. This switch opens when the motor has reached about 75 percent of its operating speed.

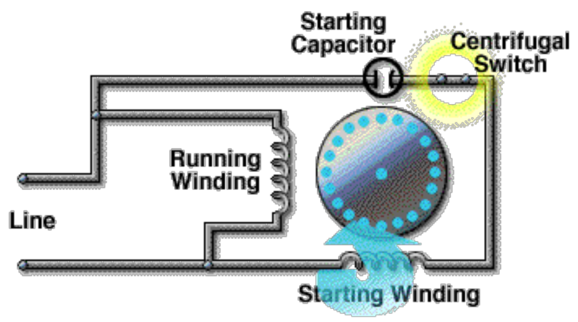


Figure 24. Schematic of CSSPIM

Capacitors in induction run motors enable them to handle heavier start loads by strengthening the magnetic field of the start windings. These loads might include refrigerators, compressors, elevators, and augers. The size of capacitors used in these types of applications ranges from 1/6 to 10 horsepower. High starting torque designs also require high starting currents and high breakdown torque.

Capacitor start / induction run motors typically deliver 250 to 350 percent of full load torque when starting. Motors of this design are used in compressors and other types of industrial, commercial, and farm equipment.

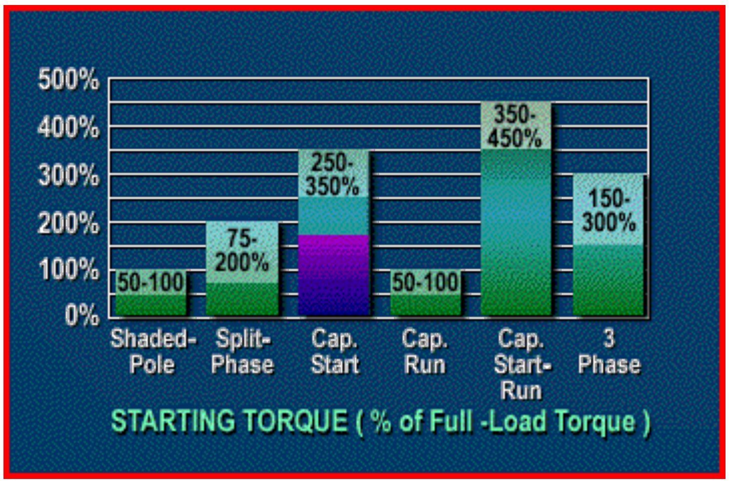


Figure 25. Starting torque graphic for different kind of induction motors

Capacitor start induction run motors of moderate torque values are used on applications that require less than 175 percent of the full load. These are used with lighter loads like fans, blowers, and small pumps.

References

[1] Split Core Current Transformer Data Sheet: *http://www.echun-elc.com/en/ProductView.asp?ID=442*

[2] Buffer amplifier Data Sheet: *http://www.ti.com/lit/ds/slos081h/slos081h.pdf*

[3] Operational amplifier Data Sheet: *http://www.ti.com/lit/ds/slos063b/slos063b.pdf*

[4] *http://www.arduino.cc/en/Main/ArduinoBoardUno*

[5] *Ghial, Saini, Singh - 2014 - Parameter Estimation of Permanent-Split Capacitor-Run Single-Phase Induction Motor Using Computed Complex Voltage Ratio*

[6] *www.electrical4u.com/sinlge-phase-induction-motor/*

[7] *elpaso.epogee.net/md/mfmscsi.asp*